Journal of Machine Learning Research 21 (2020) 1-60 Submitted 6/19; Revised 8/20; Published 9/20

Trust-Region Variational Inference with Gaussian Mixture Models

Oleg Arenz OLEG@ROBOT-LEARNING.DE
Intelligent Autonomous Systems
Technische Universität Darmstadt
Hochschulstraße 10
64289 Darmstadt, Germany

Mingjun Zhong MINGJUN.ZHONG@ABDN.AC.UK
Department of Computing Science
University of Aberdeen
King's College
Aberdeen, AB24 3FX, Scotland

Gerhard Neumann GERI@ROBOT-LEARNING.DE
Autonomous Learning Robots
Karlsruhe Institute of Technology
Adenauerring 4
76131 Karlsruhe, Germany
and
Bosch Center for Artificial Intelligence
Robert-Bosch-Campus 1
71272 Renningen, Germany

Editor: Qiang Liu

Abstract

Many methods for machine learning rely on approximate inference from intractable probability distributions. Variational inference approximates such distributions by tractable models that can be subsequently used for approximate inference. Learning sufficiently accurate approximations requires a rich model family and careful exploration of the relevant modes of the target distribution. We propose a method for learning accurate GMM approximations of intractable probability distributions based on insights from policy search by using information-geometric trust regions for principled exploration. For efficient improvement of the GMM approximation, we derive a lower bound on the corresponding optimization objective enabling us to update the components independently. Our use of the lower bound ensures convergence to a stationary point of the original objective. The number of components is adapted online by adding new components in promising regions and by deleting components with negligible weight. We demonstrate on several domains that we can learn approximations of complex, multimodal distributions with a quality that is unmet by previous variational inference methods, and that the GMM approximation can be used for drawing samples that are on par with samples created by state-of-the-art MCMC samplers while requiring up to three orders of magnitude less computational resources.

Keywords: approximate inference, variational inference, sampling, policy search, mcmc, markov chain monte carlo

1. Introduction

Inference from a complex distribution $p(\mathbf{x})$ is a huge problem in machine learning that is needed in many applications. Typically, we can evaluate the distribution except for the normalization factor Z, that is, we can only evaluate the unnormalized distribution $\tilde{p}(\mathbf{x})$, where

$$p(\mathbf{x}) = \tilde{p}(\mathbf{x})/Z,$$

with $Z = \int_{\mathbf{x}} \tilde{p}(\mathbf{x}) d\mathbf{x}$. For example, in Bayesian inference $\tilde{p}(\mathbf{x})$ would correspond to the product of prior and likelihood. As exact inference is often intractable, we have to rely on approximate inference.

Markov chain Monte Carlo (MCMC) is arguably the most commonly applied technique for approximate inference. Samples are drawn from the desired distribution by building Markov chains for which the equilibrium distribution matches the desired distribution $p(\mathbf{x})$. Monte Carlo estimates based on these samples are then used for inference. However, MCMC can be very inefficient, because it is difficult to make full use of function evaluations of $\tilde{p}(\mathbf{x})$ without violating the Markov assumption.

Instead, we propose a method based on variational inference, which is another commonly applied technique for approximate inference. In variational inference, the desired distribution $p(\mathbf{x})$ is approximated by a tractable distribution $q(\mathbf{x};\boldsymbol{\theta})$ which can be used for exact inference instead of $p(\mathbf{x})$, or as a more direct alternative to MCMC for drawing samples for (possibly importance weighted) Monte Carlo estimates. The approximation $q(\mathbf{x};\boldsymbol{\theta})$ is typically found by minimizing the reverse Kullback-Leibler (KL) divergence

$$\mathrm{KL}\left(q(\mathbf{x};\boldsymbol{\theta})||p(\mathbf{x})\right) = \int_{\mathbf{x}} q(\mathbf{x};\boldsymbol{\theta}) \log\left(\frac{q(\mathbf{x};\boldsymbol{\theta})}{p(\mathbf{x})}\right) d\mathbf{x}, \tag{1}$$

with respect to the parameters $\boldsymbol{\theta}$ of the approximation.

By framing inference as an optimization problem, variational inference can make better use of previous function evaluations of $\tilde{p}(\mathbf{x})$ than MCMC and is therefore computationally more efficient. However, in order to perform the KL minimization efficiently, $q(\mathbf{x};\boldsymbol{\theta})$ is often restricted to belonging to a simple family of models or is assumed to have non-correlating degrees of freedom (Blei et al., 2017; Peterson and Hartman, 1989), which is known as the mean field approximation. Unfortunately, such restrictions can introduce significant approximation error especially for multimodal target distributions. Comparing MCMC with variational inference, we can conclude that we should use MCMC when we require accuracy (due to its asymptotic guarantee of exactness), whereas we should prefer variational inference when we need computationally efficient solutions (Blei et al., 2017).

Hence, there is a huge interest in finding computationally efficient solutions with high sample quality. Our work aims at learning highly accurate approximations for computationally efficient variational inference methods. We use Gaussian mixture models (GMMs) as model family, because they can be sampled efficiently and are capable of representing any target distribution arbitrarily well if the number of components is sufficiently large. As the required number of components is typically not known a priori, we dynamically add or delete components during optimization.

A major challenge of learning highly accurate approximations of multimodal distributions is to achieve stable and efficient optimization of an intractable objective function. We derive a lower bound on the KL divergence (Equation 1) based on a decomposition that is related to the one used by the expectation-maximization procedure for fitting GMMs for density estimation. We can thus optimize the original objective by iteratively maximizing and tightening this lower bound. Maximizing the lower bound decomposes into independent sub-problems for each Gaussian component that are solved, analogously to the policy search method MORE (Abdolmaleki et al., 2015), based on local quadratic approximations. Due to its strong ties to policy search, we call our method Variational Inference by Policy Search (VIPS).

Another major challenge when striving for high quality approximations is to discover the relevant modes of the target distribution. The areas of high density are initially unknown and have to be discovered during learning based on function evaluations of $\tilde{p}(\mathbf{x})$. The unnormalized target distribution, however, is typically evaluated at locations that have been sampled from the current approximation $q(\mathbf{x};\boldsymbol{\theta})$, because these samples are well suited for the optimization, for example for approximating the objective (Equation 1) or its gradient. The current approximation thus serves as search distribution and needs to be adapted carefully in order to avoid erroneously discarding important regions. The conflicting goals of moving the approximation towards high density regions and evaluating $\tilde{p}(\mathbf{x})$ at unexplored regions can be seen as an instance of the exploration-exploitation dilemma that is well-known in reinforcement learning (Sutton and Barto, 1998) but currently hardly addressed by the variational inference community.

Our proposed method leverages insights from policy search (Deisenroth et al., 2013), a sub-field of reinforcement learning, by bounding the KL divergence between the updated approximation and the current approximation at each learning step. This information-geometric trust region serves the dual-purpose of staying in the validity of the local quadratic models as well as ensuring careful exploration of the search space. By finding the best approximation within such information-geometric trust region, we limit the change in search space while making sufficient progress during each iteration. However, information-geometric trust regions only address local exploration in the vicinity of the components of the current approximation and may in practice still discard regions of the search space prematurely. In order to discover modes that are not covered by the current approximation, we dynamically create new mixture components at interesting regions. Namely, we add additional components at regions where the current approximation has little probability mass although we suspect a mode of the target distribution based on previous function evaluations.

We evaluate VIPS on several domains and compare it to state-of-the-art methods for variational inference and Markov-chain Monte Carlo. We demonstrate that we can learn high quality approximations of several challenging multimodal target distributions that are significantly better than those learned by competing methods for variational inference. Compared to sampling methods, we show that we can achieve similar sample quality while using several orders of magnitude less function evaluations. Samples from the learned approximation can therefore often be used directly for approximate inference without needing importance weighting. Still, knowing the actual generative model can be a further advantage compared to model-free samplers.

This work extends previously published work about VIPS (Arenz et al., 2018) by using more efficient sample reuse, by showcasing and fixing a failure case of the previous initialization of covariance matrices, and by several other improvements such as adaptation of regularization coefficients and KL bounds. These modifications lead to a further reduction of sample complexity by approximately one order of magnitude. We will refer to the improved version as VIPS++. We evaluate VIPS++ on additional, more challenging domains, namely Bayesian Gaussian process regression and Bayesian parameter estimation of ordinary differential equations applied to the Goodwin oscillator (Goodwin, 1965) as well as more challenging variations of the previously published *planar robot* and *Gaussian mixture model* experiments (Arenz et al., 2018). Furthermore, we now also compare to normalizing flows (Kingma et al., 2016) and black-box variational inference (Ranganath et al., 2014).

2. Preliminaries

In this section we formalize the optimization problem and show its connection to policy search. We further discuss the policy search method MORE (Abdolmaleki et al., 2015) and show that a slight variation of it can be used for learning Gaussian variational approximations (GVAs) for variational inference. This variant of MORE is used by VIPS for independent component updates, which will be discussed in Section 3.

2.1 Problem formulation

Variational inference is typically framed as an information projection (I-projection) problem, that is, we want to find the parameters $\boldsymbol{\theta}$ of a model $q(\mathbf{x};\boldsymbol{\theta})$ that minimize the KL divergence between $q(\mathbf{x};\boldsymbol{\theta})$ and the target distribution $p(\mathbf{x})$,

$$\begin{aligned}\mathrm{KL}\left(q(\mathbf{x};\boldsymbol{\theta})||p(\mathbf{x})\right) &= \int_{\mathbf{x}} q(\mathbf{x};\boldsymbol{\theta}) \log\left(\frac{q(\mathbf{x};\boldsymbol{\theta})}{p(\mathbf{x})}\right) d\mathbf{x} \\ &= \int_{\mathbf{x}} q(\mathbf{x};\boldsymbol{\theta}) \log\left(\frac{q(\mathbf{x};\boldsymbol{\theta})}{\tilde{p}(\mathbf{x})}\right) d\mathbf{x} + \log Z \\ &= -L(\boldsymbol{\theta}) + \log Z.\end{aligned}$$

The normalizer Z does not affect the optimal solution for the parameters $\boldsymbol{\theta}$ as it enters the objective function as constant offset and can thus be ignored. Hence, the KL divergence can be minimized by maximizing $L(\boldsymbol{\theta})$, which is a lower bound on the log normalizer due to the non-negativity of the KL divergence. In Bayesian inference, the target distribution $p(\mathbf{x})$ corresponds to the posterior, the unnormalized distribution $\tilde{p}(\mathbf{x})$ corresponds to the product of prior and likelihood, and the normalizer corresponds to the evidence. Minimizing the KL divergence thus corresponds to maximizing a lower bound on the (log) evidence, $L(\boldsymbol{\theta})$, which is therefore commonly referred to as the evidence lower bound objective (ELBO, e.g., Blei et al. 2017).

Although VIPS is not restricted to the Bayesian setting but aims to approximate intractable distributions in general, we also frame our objective as ELBO maximization because this formulation highlights an interesting connection to policy search. We treat information projection as the problem of finding a search distribution, $q(\mathbf{x};\boldsymbol{\theta})$, over a parameter space $\mathbf{x}$, that maximizes an expected return $R(\mathbf{x}) = \log \tilde{p}(\mathbf{x})$ with an additional objective of maximizing its entropy $\mathrm{H}\big(q(\mathbf{x};\boldsymbol{\theta})\big) = -\int_{\mathbf{x}} q(\mathbf{x};\boldsymbol{\theta}) \log q(\mathbf{x};\boldsymbol{\theta}) d\mathbf{x}$, that is, we aim to solve

$$\arg\max_{\boldsymbol{\theta}} \left[L(\boldsymbol{\theta}) = \int_{\mathbf{x}} q(\mathbf{x};\boldsymbol{\theta}) \big(\log \tilde{p}(\mathbf{x}) - \log q(\mathbf{x};\boldsymbol{\theta}) \big) d\mathbf{x} = \int_{\mathbf{x}} q(\mathbf{x};\boldsymbol{\theta}) R(\mathbf{x}) d\mathbf{x} + \mathrm{H}(q(\mathbf{x};\boldsymbol{\theta})) \right].$$

Entropy objectives are also commonly used in policy search for better exploration (Neu et al., 2017; Abdolmaleki et al., 2015). Policy search methods that support such entropy objectives can thus be applied straightforwardly for variational inference. However, many policy search methods are restricted to unimodal distributions (typically Gaussians) and are therefore not suited for learning accurate approximations of multimodal target distributions. We will now review one such policy search method, MORE (Abdolmaleki et al., 2015), and show that it can be adapted straightforwardly for learning Gaussian variational approximations.

2.2 Model-Based Relative Entropy Stochastic Search

Policy search methods start with an initial search distribution $q^{(0)}(\mathbf{x})$ and iteratively update it in order to increase its expected reward.[1] Areas of high reward are initially not known and have to be discovered based on evaluations of the reward function $R(\mathbf{x})$ during learning. Policy search methods, therefore, typically evaluate the reward function on samples from the current search distribution in order to identify regions of high reward, and update the search distribution to increase the likelihood of the search distribution in these areas.

In order to avoid premature convergence to poor local optima, it is crucial to start with an initial search distribution $q^{(0)}$ with sufficiently high entropy and to ensure that high reward regions are not erroneously discarded due to too greedy updates. This trade-off between further exploring the search space and focusing on high reward areas is an instance of the exploration-exploitation dilemma that several policy search methods address using information-geometric trust regions (Peters et al., 2010; Levine and Koltun, 2013; Schulman et al., 2015; Abdolmaleki et al., 2015, 2017). These methods compute each policy update by solving a constrained optimization problem that bounds the KL divergence between the next policy and the current policy.

MORE (Abdolmaleki et al., 2015) additionally limits the entropy loss between subsequent iterations by computing the update as

$$\begin{aligned} q^{(i+1)} = \quad & \arg\max_{q} \quad \int_{\mathbf{x}} q(\mathbf{x}) R(\mathbf{x}) d\mathbf{x}, \\ & \text{s.t.} \quad \mathrm{KL}\Big(q(\mathbf{x}) || q^{(i)}(\mathbf{x})\Big) \leq \epsilon, \qquad \mathrm{H}\Big(q(\mathbf{x})\Big) \geq \beta^{(i)}, \qquad \int_{\mathbf{x}} q(\mathbf{x}) d\mathbf{x} = 1, \end{aligned} \tag{2}$$

where the lower bound on the entropy, $\beta^{(i)} = \mathrm{H}\Big(q^{(i)}(\mathbf{x})\Big) - \gamma$, is computed at each iteration based on a hyper-parameter γ, and ϵ specifies the maximum allowable KL divergence.

1. Here and in the following, we indicate variables and functions at a given iteration by using superscripts that are set in parentheses.

Hence, at each iteration, the entropy of the search distribution may not decrease by more than γ.

Introducing Lagrangian multipliers η, ω and λ, the Lagrangian function corresponding to Optimization Problem 2 is given by

$$\mathcal{L}(q,\eta,\beta,\omega) = \int_{\mathbf{x}} q(\mathbf{x})R(\mathbf{x})d\mathbf{x} + \eta\left(\epsilon - \mathrm{KL}\Big(q(\mathbf{x})||q^{(i)}(\mathbf{x})\Big)\right) \\ + \omega\left(\mathrm{H}\Big(q(\mathbf{x})\Big) - \beta^{(i)}\right) + \lambda\left(1 - \int_{\mathbf{x}} q(\mathbf{x})d\mathbf{x}\right).$$

Maximizing the Lagrangian with respect to the search distribution q allows us to express the optimal search distribution $q^{(i+1)}$ as a function of the Lagrangian multipliers,

$$q^{(i+1)}(\mathbf{x}) \propto q^{(i)}(\mathbf{x})^{\frac{\eta}{\eta+\omega}} \exp\left(R(\mathbf{x})\right)^{\frac{1}{\eta+\omega}}. \tag{3}$$

The update according to Equation 3 can not be computed analytically for general choices of policies q and reward functions $R(\mathbf{x})$. MORE is therefore restricted to Gaussian search distributions $q(\mathbf{x};\boldsymbol{\theta}^{(i)}) = \mathcal{N}\big(\mathbf{x};\boldsymbol{\mu}^{(i)},\boldsymbol{\Sigma}^{(i)}\big)$ and optimizes a local, quadratic reward surrogate

$$\tilde{R}(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^\top\mathbf{R}^{(i)}\mathbf{x} + \mathbf{x}^\top\mathbf{r}^{(i)} + \text{const.} \tag{4}$$

The parameters of the reward surrogate, $\mathbf{R}^{(i)}$ and $\mathbf{r}^{(i)}$, are learned using linear regression based on samples from the current approximation. For this choice of search distribution and reward surrogate, the updated distribution according to Equation 3 is also Gaussian with natural parameters

$$\mathbf{Q}(\eta,\omega) = \frac{\eta}{\eta+\omega}\mathbf{Q}^{(i)} + \frac{1}{\eta+\omega}\mathbf{R}^{(i)}, \tag{5}$$

$$\mathbf{q}(\eta,\omega) = \frac{\eta}{\eta+\omega}\mathbf{q}^{(i)} + \frac{1}{\eta+\omega}\mathbf{r}^{(i)}, \tag{6}$$

which directly relate to mean $\boldsymbol{\mu} = \mathbf{Q}^{-1}\mathbf{q}$ and covariance matrix $\boldsymbol{\Sigma} = \mathbf{Q}^{-1}$. It can be seen from Equation 5 and 6 that η controls the step size, whereas ω affects the entropy by scaling the covariance matrix without affecting the mean. The optimal parameters $\eta^\star$ and $\omega^\star$ can be learned by minimizing the convex dual objective

$$\mathcal{G}(\eta,\omega) = \eta\epsilon - \omega\beta^{(i)} + \eta\log Z(\mathbf{Q}^{(i)},\mathbf{q}^{(i)}) - (\eta+\omega)\log Z(\mathbf{Q}(\eta,\omega),\mathbf{q}(\eta,\omega)),$$

where $\log Z(\mathbf{X},\mathbf{x}) = -\frac{1}{2}(\mathbf{x}^\top\mathbf{X}^{-1}\mathbf{x} + \log|2\pi\mathbf{X}^{-1}|)$ is the log partition function of a Gaussian with natural parameters $\mathbf{X}$ and $\mathbf{x}$. This optimization can be performed very efficiently using the partial derivatives

$$\frac{\partial\mathcal{G}(\eta,\omega)}{\partial\eta} = \epsilon - \mathrm{KL}(q_{\eta,\omega}(\mathbf{x})||q(\mathbf{x};\boldsymbol{\theta}^{(i)})), \qquad \frac{\partial\mathcal{G}(\eta,\omega)}{\partial\omega} = \mathrm{H}(q_{\eta,\omega}(\mathbf{x})) - \beta,$$

where $q_{\eta,\omega}(\mathbf{x})$ refers to the Gaussian distribution with natural parameters computed according to Equation 5 and Equation 6. In the next section we introduce a slight variant of MORE that can be used for variational inference. The derivations of that variant are shown in Appendix A and can be straightforwardly extended to derive the equations shown in this section.

2.3 Adapting MORE to Variational Inference

Inspired by policy search methods, we want to use information-geometric trust regions for variational inference in order to achieve efficient optimization while avoiding premature convergence. Hence, we want to compute each update of the approximation by solving the constrained optimization problem

$$\begin{aligned}\boldsymbol{\theta}^{(i+1)} = \; & \arg\max_{\boldsymbol{\theta}} \quad \int_{\mathbf{x}} q(\mathbf{x};\boldsymbol{\theta})R(\mathbf{x})d\mathbf{x} + \mathrm{H}(q(\mathbf{x};\boldsymbol{\theta})), \\ & \text{subject to } \; \mathrm{KL}\Big(q(\mathbf{x};\boldsymbol{\theta})||q(\mathbf{x};\boldsymbol{\theta}^{(i)})\Big) \leq \epsilon, \\ & \qquad\qquad \int_{\mathbf{x}} q(\mathbf{x};\boldsymbol{\theta})d\mathbf{x} = 1.\end{aligned} \tag{7}$$

Optimization Problem 7 is very similar to Optimization Problem 2 solved by MORE and only differs due to the fact that the entropy of the search distribution does not enter the optimization problem as constraint, but as additional term in the objective. It can be solved analogously to MORE by introducing Lagrangian multipliers and minimizing the dual problem

$$\mathcal{G}(\eta) = \eta\epsilon + \eta \log Z(\mathbf{Q}^{(i)}, \mathbf{q}^{(i)}) - (\eta+1)\log Z(\mathbf{Q}(\eta,1), \mathbf{q}(\eta,1)), \tag{8}$$

using the gradient

$$\frac{d\mathcal{G}(\eta)}{d\eta} = \epsilon - \mathrm{KL}(q_{\eta,1}(\mathbf{x})||q(\mathbf{x};\boldsymbol{\theta}^{(i)})). \tag{9}$$

Here, the natural parameters $\mathbf{Q}(\eta,1)$ and $\mathbf{q}(\eta,1)$ for a given step size η are obtained by substituting $\omega = 1$ in Equation 5 and 6. Please refer to Appendix A for the full derivations.

Hence, a Gaussian variational approximation can be learned analogously to MORE by iteratively (1) fitting a local, quadratic surrogate $\tilde{R}(\mathbf{x}) \approx \log \tilde{p}(\mathbf{x})$, (2) finding the optimal step size η by convex optimization and (3) updating the approximation based on Equation 5 and 6. The update of a Gaussian variational approximation given a quadratic reward surrogate is shown in Algorithm 1.

3. Variational Inference by Policy Search

We showed in Section 2.3 that we can learn Gaussian variational approximations using our variant of MORE (Abdolmaleki et al., 2015). However, Gaussian approximations can lead to high modeling errors, especially for multimodal target distributions. We will now derive VIPS++, a general-purpose method for learning GMM approximations of an unnormalized target distribution $\tilde{p}(\mathbf{x})$. In Section 3.1 we will show that an I-projection to a GMM can be decomposed into independent I-projections for its Gaussian components using a similar decomposition as used by expectation-maximization. In combination with our variant of MORE, this result enables us to learn GMM approximations with a fixed number of components. Sections 3.2, 3.3 and 3.4 discuss several extensions to this procedure that are critical for efficiently learning high quality approximations in practice. Namely, we will discuss reusing function evaluations from previous iterations, selecting relevant samples and dynamically adapting the number of components.

Algorithm 1 Updating a Gaussian variational approximation based on surrogate

```
Require: coefficients of quadratic surrogate $\mathbf{R}, \mathbf{r}$ (equation 4)
Require: current mean and covariance matrix $\boldsymbol{\mu}, \boldsymbol{\Sigma}$
Require: KL bound $\epsilon$
 1: function GVA_UPDATE($\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{R}, \mathbf{r}, \epsilon$)
 2:     Compute natural parameters
 3:     $\mathbf{Q} \leftarrow \boldsymbol{\Sigma}^{-1}, \quad \mathbf{q} \leftarrow \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$
 4:     $\eta \leftarrow$ minimize dual (Equation 8) using the gradient (Equation 9)
 5:     Compute new natural parameters
 6:     $\mathbf{Q}' \leftarrow \frac{\eta}{\eta+1}\mathbf{Q} + \frac{1}{\eta+1}\mathbf{R}, \quad \mathbf{q}' \leftarrow \frac{\eta}{\eta+1}\mathbf{q} + \frac{1}{\eta+1}\mathbf{r}$
 7:     Compute new search distribution
 8:     $\boldsymbol{\Sigma}' \leftarrow \mathbf{Q}'^{-1}, \quad \boldsymbol{\mu}' \leftarrow \mathbf{Q}'^{-1}\mathbf{q}'$
 9:     return $\boldsymbol{\Sigma}', \boldsymbol{\mu}'$
10: end function
```

3.1 Learning a GMM Approximation

In order to represent high quality approximations of multimodal distributions, we want to learn a GMM approximation,

$$q(\mathbf{x};\boldsymbol{\theta}) = \sum_o q(o;\boldsymbol{\theta})q(\mathbf{x}|o;\boldsymbol{\theta}),$$

where o is the index of the mixture component, $q(o;\boldsymbol{\theta})$ are the weights of the components and $q(\mathbf{x}|o;\boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_o, \boldsymbol{\Sigma}_o)$ is a multivariate normal distribution with mean $\boldsymbol{\mu}_o$ and full covariance matrix $\boldsymbol{\Sigma}_o$. The parameters $\boldsymbol{\theta}$ of our variational approximation are thus given by the mixture weights, means and covariance matrices. To improve readability we will often omit the parameter $\boldsymbol{\theta}$ when referring to the distribution q.

The approximation is learned by maximizing the ELBO

$$\begin{aligned} L(\boldsymbol{\theta}) &= \sum_o q(o) \int_{\mathbf{x}} q(\mathbf{x}|o)\big(R(\mathbf{x}) - \log q(\mathbf{x})\big)d\mathbf{x} \\ &= \sum_o q(o) \int_{\mathbf{x}} q(\mathbf{x}|o)\big(R(\mathbf{x}) - \log q(o) - \log q(\mathbf{x}|o) + \log q(o|\mathbf{x})\big)d\mathbf{x} \\ &= \sum_o q(o)\Big[\int_{\mathbf{x}} q(\mathbf{x}|o)\big(R(\mathbf{x}) + \log q(o|\mathbf{x})\big)d\mathbf{x} + \mathrm{H}\big(q(\mathbf{x}|o)\big)\Big] + \mathrm{H}\big(q(o)\big), \end{aligned} \tag{10}$$

where we used the identity

$$\log q(\mathbf{x}) = \log q(o) + \log q(\mathbf{x}|o) - \log q(o|\mathbf{x})$$

which can be derived from Bayes' rule.

3.1.1 Variational Lower Bound

Unfortunately, the occurrence of the *log responsibilities*, $\log q(o|\mathbf{x})$, in Equation 10 prevents us from optimizing each component independently. However, we can derive a lower bound

$\tilde{L}(\boldsymbol{\theta}, \tilde{q}(o|\mathbf{x}))$ on the objective by adding and subtracting an auxiliary distribution $\tilde{q}(o|\mathbf{x})$,

$$\begin{aligned}
L(\boldsymbol{\theta}) =& \sum_o q(o) \Big[\int_{\mathbf{x}} q(\mathbf{x}|o) \big(R(\mathbf{x}) + \log q(o|\mathbf{x}) \big) d\mathbf{x} + \mathrm{H}\big(q(\mathbf{x}|o)\big) \Big] + \mathrm{H}\big(q(o)\big) \\
=& \sum_o q(o) \Big[\int_{\mathbf{x}} q(\mathbf{x}|o) \big(R(\mathbf{x}) + \log \tilde{q}(o|\mathbf{x}) + \log q(o|\mathbf{x}) - \log \tilde{q}(o|\mathbf{x}) \big) d\mathbf{x} + \mathrm{H}\big(q(\mathbf{x}|o)\big) \Big] \\
&+ \mathrm{H}\big(q(o)\big) \\
=& \underbrace{\sum_o q(o) \Big[\int_{\mathbf{x}} q(\mathbf{x}|o) \big(R(\mathbf{x}) + \log \tilde{q}(o|\mathbf{x}) \big) d\mathbf{x} + \mathrm{H}\big(q(\mathbf{x}|o)\big) \Big] + \mathrm{H}\big(q(o)\big)}_{\tilde{L}(\boldsymbol{\theta}, \tilde{q}(o|\mathbf{x}))} \\
&+ \int_{\mathbf{x}} q(\mathbf{x}) \mathrm{KL}\left(q(o|\mathbf{x}) || \tilde{q}(o|\mathbf{x}) \right) d\mathbf{x}.
\end{aligned} \tag{11}$$

Please note, that the last term in Equation 11 corresponds to an expected KL divergence and is therefore non-negative which implies that

$$\tilde{L}(\boldsymbol{\theta}, \tilde{q}(o|\mathbf{x})) \leq L(\boldsymbol{\theta}).$$

The decomposition in Equation 11 has already been previously applied in the broad context of variational inference (Agakov and Barber, 2004; Tran et al., 2016; Ranganath et al., 2016; Maaløe et al., 2016). However, these approaches parameterize the auxiliary distribution and are not well-suited for learning accurate GMM approximations. In contrast, we exploit that the responsibilities $q(o|\mathbf{x})$ can be computed in closed form for Gaussian mixture models, which allows us to exactly tighten the lower bound similar to expectation-maximization (Bishop, 2006). However, whereas EM minimizes the forward KL divergence, $\mathrm{KL}(p(\mathbf{x})||q(\mathbf{x};\boldsymbol{\theta}))$, for density estimation, our approach can be used for minimizing the reverse KL divergence, $\mathrm{KL}(q(\mathbf{x};\boldsymbol{\theta})||p(\mathbf{x}))$, in a variational inference setting. The forward KL divergence can be easier optimized when samples from the target distribution are available while the (unnormalized) target density function $\tilde{p}(\mathbf{x})$ is unavailable and is therefore well suited for density estimation. In contrast, the reverse KL divergence can be more easily optimized based on samples from the model only, when assuming access to the (unnormalized) target density function and is therefore well suited for variational inference.

Following the same reasoning as EM, we can show convergence to a stationary point of the ELBO $L(\boldsymbol{\theta})$ by iteratively setting $\tilde{q}(o|\mathbf{x}) = q(o|\mathbf{x})$ (analogously to an E-step) and increasing the lower bound $\tilde{L}(\boldsymbol{\theta}, \tilde{q}(o|\mathbf{x}))$ (M-step) while keeping the auxiliary distribution fixed. Tightening the lower bound by setting $\tilde{q}(o|\mathbf{x}) = q(o|\mathbf{x})$ does not affect the ELBO since the parameters $\boldsymbol{\theta}$ are not changed. Increasing the lower bound increases both the lower bound and the expected KL divergence and thus also increases the ELBO. Such procedure strictly increases the ELBO until we reach a fixed point of the (hierarchical) lower bound optimization, that is,

$$\boldsymbol{\theta}^{(i)} = \arg\max_{\boldsymbol{\theta}} \tilde{L}\left(\boldsymbol{\theta}, q(\mathbf{x}, \boldsymbol{\theta}^{(i)}) \right).$$

At such fixed point, the gradients of both terms of Equation 11 are zero (since they are both at an extremum) and thus the gradient of the ELBO is also zero.

In order to ensure monotonous improvement of the approximation, we need to ensure that the lower bound indeed increases during the M-Step. The lower bound $\tilde{L}(\boldsymbol{\theta}, \tilde{q}(o|\mathbf{x}))$, however, contains intractable integrals that need to be approximated based on samples. In order to keep the resulting approximation errors low, we need to stay close to the current set of samples. We therefore combine the iterative procedure with trust region optimization by bounding the change of each component during the M-step. For sufficiently small step sizes, such trust region updates ensure monotonous improvement (Akrour et al., 2018; Schulman et al., 2015). Furthermore, such constrained maximization does not affect the theoretical guarantees of the iterative procedure as any increase of the lower bound ensures an increase of the ELBO.

3.1.2 M-Step for Component Updates

Maximizing the lower bound $\tilde{L}(\boldsymbol{\theta}, \tilde{q}(o|\mathbf{x}))$ with respect to the mean and covariance matrix $\boldsymbol{\theta}_o = [\boldsymbol{\mu}_o, \boldsymbol{\Sigma}_o]$ of an individual component is not affected by the mixture coefficients $q(o)$ or the parameters of the remaining components and can be performed independently and in parallel by maximizing the term inside the square brackets of Equation 11, that is,

$$\begin{aligned} &\underset{\boldsymbol{\theta}_o}{\arg\max} \quad \int_{\mathbf{x}} q(\mathbf{x}|o;\boldsymbol{\theta}_o)\big(R(\mathbf{x}) + \log \tilde{q}(o|\mathbf{x})\big)d\mathbf{x} + \mathrm{H}\big(q(\mathbf{x}|o)\big), \\ &\text{subject to } \ \mathrm{KL}\Big(q(\mathbf{x}|o;\boldsymbol{\theta}_o)||q(\mathbf{x}|o;\boldsymbol{\theta}^{(i)})\Big) \leq \epsilon(o), \end{aligned} \tag{12}$$

where we already added the trust region constraint for better exploration and stability. The upper bound on the Kullback-Leibler divergence, $\epsilon(o)$, is adapted during learning. If the Monte-Carlo estimate of the component-specific objective after the component update is smaller than the Monte-Carlo estimate before the update, we decrease $\epsilon(o)$ by multiplying it by 0.8; otherwise we increase it slightly by multiplying it by 1.1. The optimization problem can be solved using our variant of MORE (Equation 7) with a component specific reward function $R_o(\mathbf{x}) = R(\mathbf{x}) + \log \tilde{q}(o|\mathbf{x})$. As the auxiliary distribution $\tilde{q}(o|\mathbf{x})$ was fixed to the responsibilities $q(o|\mathbf{x};\boldsymbol{\theta}^{(i)})$ according to the previous mixture model, the component specific part of $R_o(\mathbf{x})$ penalizes each component for putting probability mass on areas that are already covered by other components.

For applying our variant of MORE, we need to fit a quadratic reward surrogate $\tilde{R}_o(\mathbf{x}) \approx R_o(\mathbf{x})$ that approximates the component specific reward $R_o(\mathbf{x})$ in the vicinity of the respective component $q(\mathbf{x}|o)$. The surrogate can be fit using ordinary least squares, where the independent variables are samples from the respective component and the dependent variables are the corresponding function evaluations of $R_o(\mathbf{x})$. However, because we want to use the same set of samples for all component updates as well as the weight update, we use weighted least squares based on importance weights which will be discussed in greater detail in Section 3.2. After fitting the surrogate, the optimization problem in Equation 12 can be solved efficiently using L-BFGS-B (Byrd et al., 1995) to minimize the dual problem (Equation 8) and using the learned step size η to compute the update in closed form as outlined in Section 2.3.

Drawing the connection to reinforcement learning and investigating the reward function $R_o(\mathbf{x})$ for a given component reveals that the proposed algorithm treats every component update as a reinforcement learning problem, where the reward is computed based on the achieved log-densities $\log \tilde{p}(\mathbf{x})$ with a penalty for sampling in regions that are already covered by other components due to low log responsibilities. Moreover, the components strive for high entropy which prevents them from always choosing the same sample.

3.1.3 M-Step for Weight Updates

After updating the individual components, we can keep the learned means and covariance matrices fixed while updating the mixture coefficients $q(o)$. As shown in previous work (Arenz et al., 2018), we can also enforce an information-geometric trust region for the weight update. However, in subsequent experiments we could not show a significant effect of such constraint and will therefore only consider the unconstrained optimization. The M-step with respect to the mixture coefficients is thus framed as

$$\underset{q(o)}{\arg\max} \quad \sum_o q(o)R(o) + \mathrm{H}\big(q(o)\big), \tag{13}$$

where the objective for the component update,

$$R(o) = \int_{\mathbf{x}} q(\mathbf{x}|o)\big(R(\mathbf{x}) + \log \tilde{q}(o|\mathbf{x})\big)d\mathbf{x} + \mathrm{H}\big(q(\mathbf{x}|o)\big), \tag{14}$$

serves as reward for choosing component o. The reward $R(o)$ contains an intractable integral, and thus it needs to be approximated from samples. It is to note that $R(o)$ corresponds to a discrete function, which can be represented by a vector, whereas the reward function $R_o(\mathbf{x})$ used for the component update is a continuous function. It is not beneficial to approximate $R(o)$ based on a quadratic surrogate of $R_o(\mathbf{x})$, since we can estimate each element of the vector more efficiently and more accurately using a Monte-Carlo estimate

$$\tilde{R}(o) = \frac{1}{N_o}\sum_{n=1}^{N_o}\big[R(\mathbf{x}_{o,n}) + \log \tilde{q}(o|\mathbf{x}_{o,n})\big] + \mathrm{H}(q(\mathbf{x}|o)), \tag{15}$$

where $\mathbf{x}_{o,n}$ refers to the nth of N_o samples from component $q(\mathbf{x}|o)$. We will discuss in Section 3.2 how we use importance weighting to estimate the reward of each component based on the same set of samples that is used for the component update.

Based on the approximated rewards $\tilde{R}(o)$, the optimal solution of optimization problem in Equation 13 is given in closed form as

$$q(o) = \frac{\exp\left(\tilde{R}(o)\right)}{\sum_o \exp\left(\tilde{R}(o)\right)}. \tag{16}$$

The weight optimization can also be treated as a reinforcement-learning problem, where actions correspond to choosing components and the agent gets rewarded for choosing components that sample in important regions, that do not interfere with other components and that have high entropy. The agent itself also strives for high entropy and will thus make use of every component.

The complete optimization can be treated as a method for hierarchical reinforcement learning where we learn both, a higher level policy $q(o)$ over options and Gaussian lower level policies $q(\mathbf{x}|o)$. However, since our approach does not consider time series data, it mainly relates to black-box approaches to reinforcement learning that use stochastic optimizers such as ARS, NES or MORE (Mania et al., 2018; Salimans et al., 2017; Abdolmaleki et al., 2015). HiREPS (Daniel et al., 2012) already applied black-box optimization for learning GMM policies based on episodic REPS (Peters et al., 2010).

The basic variant of our method is shown in Algorithm 2. The individual component updates (line 3-8) are performed by sampling from the respective components (line 3), evaluating the samples on the target distribution (line 4), computing the log responsibilities $\log \tilde{q}(o|\mathbf{x})$ according to the previous approximation (line 5), fitting the reward surrogate (line 6-7) and performing the trust region update (line 8). The components can be updated in parallel since the responsibilities are computed based on the same mixture parameters $\boldsymbol{\theta}$. The weight update (line 11-17) is computed based on Equation 16 (line 17) using the Monte-Carlo estimates of the component rewards (line 15). Updating the parameters of the GMM in between the component updates and the weight update (line 10) is optional and relates to an additional E-Step in EM, which does not affect the theoretical guarantees (Neal and Hinton, 1998).

Algorithm 2 Variational Inference by Policy Search (Basic Variant)

```
Require: number of components $N_o$
Require: initial mixture parameters $\boldsymbol{\theta} = \{q(o), \boldsymbol{\mu}_{o,\dots,N_o}, \boldsymbol{\Sigma}_{o,\dots,N_o}\}$
Require: number of iterations $N_i$
Require: number of samples per component $N_s$
 1: for $i = 1 \dots N_i$ do
 2:     for $o = 1 \dots N_o$ do
 3:         $\mathcal{X}_o \leftarrow$SAMPLE_GAUSSIAN$(\boldsymbol{\mu}_o, \boldsymbol{\Sigma}_o, N_s)$
 4:         $\tilde{\mathbf{p}}_o \leftarrow \log \tilde{p}(\mathcal{X}_\mathbf{o})$   ▷ EVALUATE TARGET LOG LIKELIHOOD FOR EACH SAMPLE
 5:         $\tilde{\mathbf{q}}_{o|x} \leftarrow \log q(\mathcal{X}_o, o; \boldsymbol{\theta}) - \log q(\mathcal{X}_o; \boldsymbol{\theta})$   ▷ EVALUATE LOG RESPONSIBILITIES
 6:         $\mathbf{y}_o \leftarrow \tilde{\mathbf{p}}_o + \tilde{\mathbf{q}}_{o|x}$   ▷ COMPUTE TARGETS FOR ORDINARY LEAST SQUARES (OLS)
 7:         $\mathbf{R}_o$,$\mathbf{r}_o \leftarrow$OLS$(\mathcal{X}_o, \mathbf{y}_o)$   ▷ LEARN QUADRATIC SURROGATE
 8:         $\boldsymbol{\mu}'_o, \boldsymbol{\Sigma}'_o \leftarrow$GVA_UPDATE$(\boldsymbol{\mu}_\mathbf{o}, \boldsymbol{\Sigma}_\mathbf{o}, \mathbf{R}_o, \mathbf{r}_o, \epsilon_o)$   ▷ ALGORITHM 1
 9:     end for
10:     $\boldsymbol{\theta} \leftarrow$ UPDATE_COMPONENTS$(\boldsymbol{\theta}, \boldsymbol{\mu}'_{o,\dots,N_o}, \boldsymbol{\Sigma}'_{o,\dots,N_o})$
11:     for $o = 1 \dots N_o$ do
12:         $\mathcal{X}_o \leftarrow$SAMPLE_GAUSSIAN$(\boldsymbol{\mu}_o, \boldsymbol{\Sigma}_o, N_s)$
13:         $\tilde{\mathbf{p}}_o \leftarrow \log \tilde{p}(\mathcal{X}_\mathbf{o})$   ▷ EVALUATE TARGET LOG LIKELIHOOD FOR EACH SAMPLE
14:         $\tilde{\mathbf{q}}_{o|x} \leftarrow \log q(\mathcal{X}_o, o; \boldsymbol{\theta}) - \log q(\mathcal{X}_o; \boldsymbol{\theta})$   ▷ EVALUATE LOG RESPONSIBILITIES
15:         $\tilde{R}_o \leftarrow {N_s}^{-1}$ SUM$(\tilde{\mathbf{p}}_o + \tilde{\mathbf{q}}_{o|x}) + \mathrm{H}(\boldsymbol{\Sigma}_o)$   ▷ ESTIMATE REWARD (EQUATION 15)
16:     end for
17:     $q'(o) \leftarrow \frac{\exp(\tilde{R}_o)}{\sum_o \exp(\tilde{R}_o)}$
18:     $\boldsymbol{\theta} \leftarrow$ UPDATE_WEIGHTS$(\boldsymbol{\theta}, q'(o))$
19: end for
```

3.2 Sample Reuse by Importance Weighting

VIPS relies on samples for approximating the reward for choosing a given component, $R(o)$, and for computing the quadratic surrogates for the component update. These samples need to be evaluated on the unnormalized target distribution $\tilde{p}(\mathbf{x})$ which may be costly. In order to reduce the number of function evaluations we want to also make use of samples from previous iterations, which can be achieved by using importance weighting. We will now show how importance weights can be used to approximate the rewards for the weight updates and how to learn the quadratic surrogates for the component update based on the same subset $\boldsymbol{\mathcal{X}}_\subset$ of samples.

3.2.1 Importance Weighting for Updating the Mixture Weights

Importance sampling is a technique for estimating the expected value $E_q[f(\mathbf{x})]$ of a given function $f(\mathbf{x})$ with respect to a distribution $q(\mathbf{x})$ while using samples from a different distribution $z(\mathbf{x}) \neq q(\mathbf{x})$. Assuming that the support of $z(\mathbf{x})$ covers the support of $q(\mathbf{x})$, we can express the desired expectation as

$$E_q[f(\mathbf{x})] = \int_{\mathbf{x}} q(\mathbf{x}) f(\mathbf{x}) d\mathbf{x} = \int_{\mathbf{x}} z(\mathbf{x}) \frac{q(\mathbf{x})}{z(\mathbf{x})} f(\mathbf{x}) d\mathbf{x} = E_z[w(\mathbf{x}) f(\mathbf{x})],$$

using *importance weights* $w(\mathbf{x}) = \frac{q(\mathbf{x})}{z(\mathbf{x})}$. Hence, the desired expectation can be approximated by using a Monte-Carlo estimate based on N_z samples from the sampling distribution $z(\mathbf{x})$,

$$E_q[f(\mathbf{x})] \approx \sum_{i=1}^{N_z} \frac{1}{N_z} w(\mathbf{x}_i) f(\mathbf{x}_i). \tag{17}$$

Instead of using the estimator given by Equation 17, it is also common to use *self-normalized importance sampling*

$$E_q[f(\mathbf{x})] \approx \sum_{i=1}^{N_z} \bar{w}(\mathbf{x}_i) f(\mathbf{x}_i), \qquad \bar{w}(\mathbf{x}_i) = \left(\sum_{i=1}^{N_z} \frac{q(\mathbf{x}_i)}{z(\mathbf{x}_i)} \right)^{-1} \frac{q(\mathbf{x}_i)}{z(\mathbf{x}_i)}.$$

Self-normalized importance sampling introduces a bias that is asymptotically zero since $\lim_{N_z \to \infty} \sum_{i=1}^{N_z} \frac{q(\mathbf{x}_i)}{z(\mathbf{x}_i)} = N_z$, but it has the advantages that it is consistent for different constant offsets on the function $f(\mathbf{x})$ and that it is also applicable if the target distribution is not normalized.

An important consideration for choosing the sampling distribution is the variance of the estimator. In general, the estimator's variance can be significantly worse than standard Monte-Carlo (Hesterberg, 1988). When using samples from the desired distribution, that is, $z(\mathbf{x}) = q(\mathbf{x})$, the importance weighted estimate and the self-normalized estimate are both equivalent to standard Monte-Carlo. However, it is also possible to obtain lower variance than standard Monte-Carlo, for example, when using the optimal sampling distribution

$$z(\mathbf{x}) = \frac{1}{C} q(\mathbf{x}) |f(\mathbf{x}) - c|, \tag{18}$$

where C is a normalizing constant and $c = 0$ for importance sampling and $c = E_q[f(\mathbf{x})]$ for self-normalized importance sampling (Hesterberg, 1988). If the function $f(\mathbf{x})$ is positive everywhere, the former estimate has even zero variance since

$$w(\mathbf{x}_i)f(\mathbf{x}_i) = \frac{q(\mathbf{x}_i)}{z(\mathbf{x}_i)}f(\mathbf{x}_i) = C\frac{q(\mathbf{x}_i)}{q(\mathbf{x}_i)f(\mathbf{x}_i)}f(\mathbf{x}_i) = C = \int_{\mathbf{x}} q(\mathbf{x})f(\mathbf{x})d\mathbf{x} = E_q[f(\mathbf{x})].$$

Although the optimal sampling distributions according to Equation 18 are intractable as they depend on the expectation $E_q[f(\mathbf{x})]$, which is the value of interest, they can be useful for designing appropriate sampling distributions.

In order to estimate the expected reward $R(o)$ using a subset $\mathcal{X}_\subset$ of the samples from previous iterations $\mathcal{X}$ we need to evaluate the respective sampling distribution $z_\subset(\mathbf{x})$ for computing the importance weights. For that purpose, we store all samples together with the respective unnormalized target densities and the parameters of the component from which it was sampled in a database

$$\mathcal{S} = \{(\mathbf{x}_0, \log\tilde{p}(\mathbf{x_0}), \mathcal{N}_{\mathbf{x}_0}), \dots, (\mathbf{x}_N, \log\tilde{p}(\mathbf{x_N}), \mathcal{N}_{\mathbf{x}_N})\},$$

where $N_{\mathbf{x}}$ refers to the Gaussian distribution that was used for obtaining the sample $\mathbf{x}$. By also storing its respective Gaussian distributions, we can represent the sampling distribution as a Gaussian mixture model $z^\subset(\mathbf{x})$ that contains for each sample $\mathbf{x}_s \in \boldsymbol{\mathcal{X}}_\subset$ the respective Gaussian distribution $\mathcal{N}_{\mathbf{x}_s}(\mathbf{x})$, that is,

$$z_\subset(\mathbf{x}) = \sum_{\mathbf{x}_s \in \boldsymbol{\mathcal{X}}_\subset} \frac{1}{|\boldsymbol{\mathcal{X}}_\subset|}\mathcal{N}_{\mathbf{x}_s}(\mathbf{x}).$$

Please note, that in practice, we represent the GMM $z_\subset(\mathbf{x})$ more concisely by exploiting that usually several samples were drawn from the same Gaussian distribution. We estimate the reward $R_o(\mathbf{x})$ for each component using self-normalized importance sampling, that is,

$$\tilde{R}(o) = \sum_{\mathbf{x}_s \in \boldsymbol{\mathcal{X}}_\subset} \bar{w}_o(\mathbf{x}_s)\big[R(\mathbf{x}_s) + \log\tilde{q}(o|\mathbf{x}_s)\big] + \mathrm{H}(q(\mathbf{x}|o)).$$

where the self-normalized importance weights for component o are given by

$$\bar{w}_o(\mathbf{x}_s) = \frac{1}{Z}\frac{q(\mathbf{x}_s|o)}{z_\subset(\mathbf{x}_s)}, \qquad Z = \sum_{\mathbf{x}_s \in \boldsymbol{\mathcal{X}}_\subset} \frac{q(\mathbf{x}_s|o)}{z_\subset(\mathbf{x}_s)}.$$

We could choose different subsets, depending on the component for which we want to estimate the reward $R(o)$. However, because we need to evaluate each sample on any component anyway in order to compute the responsibilities $q(o|\mathbf{x})$, we use the same subset $\boldsymbol{\mathcal{X}}_\subset$ for estimating all component rewards as well as the surrogate models.

3.2.2 Importance Weighting for Fitting the Quadratic Surrogates

For updating the individual components we need to learn local quadratic surrogates $\tilde{R}_o(\mathbf{x})$ in the vicinity of the respective components. MORE achieves locality by using samples from the respective component $q(\mathbf{x}|o)$ as independent variables for ordinary least-squares.

Learning the surrogate based on samples from a different distribution $z_{\subset}(\mathbf{x})$ introduces covariate shift, that is, the distribution of the training data $z_{\subset}(\mathbf{x})$ does not match the distribution of the test data $q(\mathbf{x}|o)$. The covariate shift can be accommodated by minimizing a weighted least-squares problem (Chen et al., 2016),

$$\arg\min_{\boldsymbol{\beta}_o} E_{z_{\subset}}\left[\bar{w}_o(\mathbf{x}_s)\left(R_o(\mathbf{x}) - \tilde{R}_o(\mathbf{x};\boldsymbol{\beta}_o)\right)^2\right],$$

where the quadratic surrogate $\tilde{R}_o(\mathbf{x};\beta_o)$ is linear in the parameters $\boldsymbol{\beta}_o$. In practice, we also perform ℓ_2-regularization with *ridge coefficient* κ_o. The optimal parameters are thus given by

$$\boldsymbol{\beta}_o = (\mathbf{X}^\top\mathbf{W}_o\mathbf{X} + \kappa_o\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{W}_o\mathbf{y},$$

where $\mathbf{X}$ is the design matrix where each row contains the linear and quadratic features for the respective sample $\mathbf{x}_s \in \boldsymbol{\mathcal{X}}_{\subset}$ as well as a constant feature, $\mathbf{W}_o$ is a diagonal matrix where each element relates to the respective self-normalized importance weight $w_o(\mathbf{x}_s)$, $\mathbf{y}$ is a vector containing the targets $y_s = R(\mathbf{x}_s) + \log q(o|\mathbf{x}_s)$ and $\boldsymbol{\beta}_o$ is a vector containing the elements of $\mathbf{r}_o$ and $\mathbf{R}_o$ as well as a constant offset that can be discarded. Specifying an appropriate ridge coefficient κ_o can be difficult as different components may require different amounts of regularization. We therefore adapt the coefficient during optimization by multiplying it by 10 if the matrix inversion failed and by dividing it by 2 if it succeeded.

Although we use importance weights for learning the surrogates, we do not aim to estimate an expected value. The minimum-variance sampling distributions given by Equation 18 are in general not useful for learning accurate surrogate models as they focus on bringing the weighted function evaluation $w(\mathbf{x})f(\mathbf{x})$ close to the expected value, rather than aiming to accurately represent the function's landscape. Instead, we aim to construct a sampling distribution $z_{\subset}(\mathbf{x})$ that covers all components of the current approximation well. Such sampling distribution ensures that the importance weighted estimates are not much worse than Monte-Carlo estimations, both, for estimating the expected rewards $\tilde{R}(o)$ and for learning locally valid surrogate models $\tilde{R}_o(\mathbf{x})$. In the next section, we will discuss a heuristic for constructing such sampling distribution.

3.3 Sample Selection

Using all previous samples in each iteration would be computationally costly. Instead, we want to select a small set of samples such that we can get good approximations of all surrogate models and component rewards while requiring only a small number of new samples from each component. A common technique that was used in CMA-ES (Shirakawa et al., 2015), MORE (Abdolmaleki et al., 2015) and VIPS (Arenz et al., 2018) is to reuse all samples from the k latest iterations, where k is a hyper-parameter to balance between sample efficiency and computational efficiency. As the components that were used for the most recent iterations were similar to the current components, the reused samples can usually provide meaningful information about the target distribution in the vicinity of the respective components. However, we noticed that such procedure can be wasteful when optimizing large GMMs if some component have already converged and others still need to improve. For example, we typically have enough samples in the database to estimate the reward and local surrogate for components that did not significantly change during several iterations even without requiring any new samples; yet, when only using the latest k samples we need to continuously sample from each component during the whole optimization in order to maintain stability.

In order to avoid discarding old samples, we could sub-sample uniformly among the sample database. However, such procedure can result in a large number of irrelevant samples and, furthermore, does not ensure that the relevant samples are evenly distributed among the components of the current approximation. A more sophisticated method was presented by Uchibe (2018) in the context of policy search. Instead of sub-sampling uniformly, they treat all components in the database as components of a mixture model, $q^{\alpha}_{\text{sampling}}(\mathbf{x})$, and optimize the corresponding mixture coefficients α such that the model is close to the optimal sampling distribution given by Equation 18. However, the resulting sampling distribution might not be suited for learning the surrogate models, and, furthermore, such approach would be computational intractable because, by optimizing a GMM, VIPS may add up to several hundreds of components to the database in each iteration and would also need to identify an optimal sampling distribution for each of the respective components.

Furthermore, it is hard to make use of function evaluations such as $\tilde{p}(\mathbf{x}_i)$ or $q(\mathbf{x}_i|o)$ for deciding whether to reuse a given sample $\mathbf{x}_i$ without introducing additional bias in the importance sampling estimate. When such function evaluations influence our decision to use a given sample $\mathbf{x}_i$ for importance weighting, we can no longer consider it as an unbiased draw from $\mathcal{N}_{\mathbf{x}_i}(\mathbf{x})$ and computing the importance weights based on the background distribution $z_{\subset}(\mathbf{x})$ would, thus, not be admissible.

Instead, we propose to identify for each component $q(\mathbf{x}|o)$ of the current approximation those components in the database $\mathcal{N}_{\mathbf{x}_i}(\mathbf{x})$ that are close according to a given dissimilarity measure $d\big(q(\mathbf{x}|o), \mathcal{N}_{\mathbf{x}_i}(\mathbf{x})\big)$ that is independent of the actual samples drawn from $\mathcal{N}_{\mathbf{x}_i}(\mathbf{x})$. In order to reduce the risk of selecting the same samples in each iteration, which may result in overfitting, we iteratively sample (without replacement) components from our database according to

$$h(i, o) \propto \exp\left(-d\big(q(\mathbf{x}|o), \mathcal{N}_{\mathbf{x}_i}(\mathbf{x})\big) - n_i\right), \tag{19}$$

where n_i keeps track of the number of times the samples of distribution $\mathcal{N}_{\mathbf{x}_i}(\mathbf{x})$ have been reused. We add all samples from the chosen component to the active set of samples $\mathcal{X}_{\subset}$ and stop sampling distributions when a desired number of reused samples n_{reused} is reached. This process is performed for each component $q(\mathbf{x}|o)$ of the current approximation.

A natural choice for the dissimilarity is to use the Kullback-Leibler divergence,

$$d_{\text{KL}}\big(q(\mathbf{x}|o), \mathcal{N}_{\mathbf{x}_i}(\mathbf{x})\big) = \text{KL}\big(q(\mathbf{x}|o)||\mathcal{N}_{\mathbf{x}_i}(\mathbf{x})\big),$$

which favors sampling distributions $\mathcal{N}_{\mathbf{x}_i}(\mathbf{x})$ that cover the respective mixture component $q(\mathbf{x}|o)$ well. However, even though the KL divergence between two Gaussian distributions can be computed in closed form, computing it for every component in the current approximation with respect to every component in the database can quickly become the computational bottleneck of the whole optimization.

Instead, VIPS++ computes the dissimilarity as the negative Mahalanobis distance of the mean μ_i of the sampling distribution $\mathcal{N}_{\mathbf{x}_i}(\mathbf{x})$ with respect to the given component $q(\mathbf{x}|o)$, that is,

$$d_{\text{Mahalanobis}}\big(q(\mathbf{x}|o), \mathcal{N}_{\mathbf{x}_i}(\mathbf{x})\big) = -\log p(\boldsymbol{\mu}_i|o).$$

While neglecting the covariance matrix of the sampling distribution may appear too crude, we argue that it is necessary to stay within a reasonable computational budget for selecting relevant samples. We demonstrate in Section 5.2 that the proposed selection strategy is able to identify relevant samples for each component $q(\mathbf{x}|o)$ among all previous samples without adding significant computational overhead. We also compare the Mahalanobis distance to different dissimilarity measures, namely, forward and reverse KL, as well as uniform selection in Appendix B. Pseudo-code for identifying relevant samples is shown in Appendix C.

3.3.1 Drawing new samples

After selecting the set $\boldsymbol{\mathcal{X}}_{\subset}$ of samples to be reused during the current iteration, we need to draw new samples from those components that are not sufficiently covered. A useful diagnostic for monitoring the quality of the chosen sampling distribution is the *effective sample size*

$$n_{\text{eff}}(o) = \Big(\sum_{\mathbf{x}_s \in \boldsymbol{\mathcal{X}}_{\subset}} \bar{w}_o(\mathbf{x}_s)^2\Big)^{-1},$$

which approximates the number of samples that standard Monte-Carlo would require to achieve the same variance as the importance sampling estimate (Kong et al., 1994; Djuric et al., 2003).

Hence, we compute for each component the number of effective samples, and draw $n_{\text{new}}(o) = n_{\text{des}} - \lfloor n_{\text{eff}}(o) \rfloor$ new samples, such that its effective sample size should approximately match a specified desired number of effective samples n_{des}. These samples are added to the database and to the set of active samples $\boldsymbol{\mathcal{X}}_{\subset}$ as illustrated in Algorithm 3.

3.4 Adapting the Number of Components

The component optimization (Algorithm 1) is a local optimization, and the component will typically converge to a nearby mode (although the trust region constraint may help to traverse several poor optima). The quality of the learned approximation thus depends crucially on the initialization of the mixture model. However, the modes of the target distribution are often not known a priori and have to be discovered during optimization. We therefore adapt the number of components dynamically by adding new components in promising regions and by deleting components with very low weight. The number of components is adapted at the beginning of each learning iteration before obtaining new samples. By always assigning low weight to newly added components and by only deleting components that have low weight, the effect on the approximation is negligible and the stability of the optimization is thus not affected.

3.4.1 Deleting Bad Components

Components that have been initialized at poor locations may converge to irrelevant modes of the target distribution and get very low weights such that they do not affect the approximation in practice. As keeping such components would add unnecessary computational

Algorithm 3 Ensure that every component has sufficiently many effective samples.

```
Require: database $\mathcal{S} = \{(\mathbf{x}_0, \log \tilde{p}(\mathbf{x_0}), \mathcal{N}_{\mathbf{x_0}}), \dots, (\mathbf{x}_N, \log \tilde{p}(\mathbf{x_N}), \mathcal{N}_{\mathbf{x_N}})\}$
Require: Set of chosen samples $\boldsymbol{\mathcal{X}}_\subset$, respective self-normalized importance weights $w_o(\mathbf{x})$
Require: desired number of effective samples per component $n_\text{des}$
 1: function SAMPLE_WHERE_NEEDED
 2:     for $o = 1 \dots N_o$ do
 3:         $n_\text{eff}(o) \leftarrow \left(\sum_{\mathbf{x}_s \in \boldsymbol{\mathcal{X}}_\subset} w_o(\mathbf{x}_s)^2\right)^{-1}$
 4:         $n_\text{new}(o) \leftarrow n_\text{des} - \lfloor n_\text{eff}(o) \rfloor$
 5:         $\boldsymbol{\mathcal{X}}_{\text{new},o} \leftarrow$ SAMPLE_GAUSSIAN$(\boldsymbol{\mu}_o, \boldsymbol{\Sigma}_{\boldsymbol{o}}, n_\text{new}(o))$
 6:         for $\mathbf{x}_s$ in $\boldsymbol{\mathcal{X}}_{\text{NEW},o}$ do
 7:             $\mathcal{S} \leftarrow \mathcal{S} \cup \{(\mathbf{x}_s, \log \tilde{p}(\mathbf{x}_s), \mathcal{N}_{\mathbf{x}_s})\}$
 8:         end for
 9:         $\boldsymbol{\mathcal{X}}_\subset \leftarrow \boldsymbol{\mathcal{X}}_\subset \cup \boldsymbol{\mathcal{X}}_{\text{new},o}$
10:     end for
11:     return $\boldsymbol{\mathcal{X}}_\subset$
12: end function
```

overhead, we delete any component that had low weight for a given number of iterations, n_{del}, and that further did not increase its expected reward $\tilde{R}(o)$ during that period.

3.4.2 Initializing the Mean of New Components

By adding components to the mixture model, we can increase the representational power and thus improve the quality of the approximation. Furthermore, adding components affects the search distribution and can thus be used for exploration. In either case, we want the new components to eventually contribute to the approximation and hence achieve high weight $q(o) \propto \exp(R(o))$ and thus high reward $R(o)$. We treat every sample $\mathbf{x}_s$ in the database as candidate for the initial mean of the new component and then select the most promising candidate according to an estimate of its initial reward. As we will discuss in Section 3.4.3, we will decide on the initial entropy irrespective of the initial mean, but we will choose the exact initial covariance only after deciding for an initial mean. Hence, in the following we will derive an estimate of the initial reward that depends on the initial mean and initial entropy H_init, but not on the covariance matrix.

Let $q_{\mathbf{x}_s}(\mathbf{x}|o_n)$ denote the new component o_n assuming that its mean was initialized at location $\boldsymbol{\mu}_n = \mathbf{x}_s$ and let $q_{\mathbf{x}_s}(\mathbf{x}) = (1 - q(o_n))q(\mathbf{x}) + q(o_n)q_{\mathbf{x}_s}(\mathbf{x}|o_n)$ denote the GMM approximation after adding the new component with initial weight $q(o_n)$. According to

Equation 14 the initial reward of the new component $R_{\mathbf{x}_s}(o_n)$ would be given by

$$
\begin{aligned}
R_{\mathbf{x}_s}(o_n) &= \int_{\mathbf{x}} q_{\mathbf{x}_s}(\mathbf{x}|o_n)\big(R(\mathbf{x}) + \log q_{\mathbf{x}_s}(o_n|\mathbf{x})\big)d\mathbf{x} + \mathrm{H}\big(q_{\mathbf{x}_s}(\mathbf{x}|o_n)\big) \qquad (20)\\
&= \int_{\mathbf{x}} q_{\mathbf{x}_s}(\mathbf{x}|o_n)\big(R(\mathbf{x}) + \log q(o_n) + \log q_{\mathbf{x}_s}(\mathbf{x}|o_n) - \log q_{\mathbf{x}_s}(\mathbf{x})\big)d\mathbf{x} + \mathrm{H}\big(q_{\mathbf{x}_s}(\mathbf{x}|o_n)\big)d\mathbf{x}\\
&= \log q(o_n) + \int_{\mathbf{x}} q_{\mathbf{x}_s}(\mathbf{x}|o_n)\big(R(\mathbf{x}) - \log q_{\mathbf{x}_s}(\mathbf{x})\big)d\mathbf{x}\\
&= \log q(o_n) + \int_{\mathbf{x}} q_{\mathbf{x}_s}(\mathbf{x}|o_n)\Big(R(\mathbf{x}) - \log\big((1-q(o_n))q(\mathbf{x}) + q(o_n)q_{\mathbf{x}_s}(\mathbf{x}|o_n)\big)\Big)d\mathbf{x}. \qquad (21)
\end{aligned}
$$

Based on Equation 21 we can estimate the initial reward depending on the initial weight of the new component $q(o_n)$, the *current* mixture model $q(\mathbf{x})$, the target distribution $R(\mathbf{x})$, and the new component $q_{\mathbf{x}_s}(\mathbf{x}|o_n)$. The first term can be ignored because we choose the initial weight of the new component irrespective of its mean and a constant offset does not affect which initial mean achieves the maximum initial reward. The integral is intractable but can be approximated based on the sample $\mathbf{x}_s = \boldsymbol{\mu}_n$ as

$$
\tilde{R}_{\mathbf{x}_s}(o_\mathrm{n}) = R(\mathbf{x}_s) - \log\left((1-q(o_n))q(\mathbf{x}_s) + q(o_n)\exp\left(\frac{1}{2}D - \mathrm{H}_\mathrm{init}\right)\right) \qquad (22)
$$

where we exploit that the Gaussian density at its mean can be computed based on its entropy H_init and the number of dimensions D, that is, $\log q_{\mathbf{x}_s}(\mathbf{x}_s|o_n) = \frac{1}{2}D - \mathrm{H}_\mathrm{init}$. As the function evaluations $R(\mathbf{x}_s)$ of the target distribution are stored in the database, we only need to evaluate the current mixture model $q(\mathbf{x})$ on all candidate samples $\mathbf{x}_s$ to estimate the initial reward for these locations.

To investigate the approximated reward in Equation 22 we note that the second term corresponds to a log-sum-exp (LSE), that is,

$$
\begin{aligned}
\tilde{R}_{\mathbf{x}_s}(o_\mathrm{n}) &= R(\mathbf{x}_s) - \log\big((1-q(o_n))q(\mathbf{x}_s) + q(o_n)q_{\mathbf{x}_s}(\mathbf{x}_s|o_n)\big)\\
&= R(\mathbf{x}_s) - \mathrm{LSE}\big(\log(1-q(o_n)) + \log q(\mathbf{x}_s), \log q(o_n) + \log q_{\mathbf{x}_s}(\mathbf{x}_s|o_n)\big)\\
&\approx R(\mathbf{x}_s) - \max\left(\log q(\mathbf{x}_s), \log q(o_n) + \log q_{\mathbf{x}_s}(\mathbf{x}_s|o_n)\right), \qquad (23)
\end{aligned}
$$

where we exploit that $\mathrm{LSE}(a_1, a_2, \ldots, a_n) = \log\sum_{i=1}^{n}\exp(a_i)$ behaves similar to a maximum and that $(1-q(o_n))q(\mathbf{x}) \approx q(\mathbf{x})$, since we initialize the new component with negligible weight, $q(o_n) \approx 0$. Although the effect of the initial weight on the first operand of the log-sum-exp is negligible, it may have considerable effect on the second operand because the logarithm of small values is a large negative value. Hence, the initial weight that we choose for a new component may affect its approximated reward, which can be explained by its effect on the responsibilities $q_{\mathbf{x}_s}(o_n|\mathbf{x}_s)$ in Eq. 20.

If we would add the new components with an initial weight of zero, the maximum-operator would always return the first operand and the proposed estimate (which ignores the constant offset $\log q(o_n) = -\infty$ in Eq. 22) of the initial reward would return the amount of missing log probability-density, $\tilde{R}_{\mathbf{x}_s}(o_\mathrm{n}|q(o_n) = 0) = R(\mathbf{x}_s) - \log q(\mathbf{x}_s)$. Adding a new component at the location where our current approximation misses most log probability density seems sensible. However, the problem of such heuristic becomes evident when considering target distributions with heavy tails. In such cases, the amount of missing log probability density increases the farther we move away from the current approximation. The new component might, thus, be added in a region where the target distribution has low probability density, since the current approximation might have even lower probability density.

This failure case is a direct consequence of ignoring the effect of the new component on the mixture model. When considering non-zero weights, the log-responsibilities of the new component are finite and tend to increase the farther we move away from the current approximation. Yet, they saturate at $\log q_{\mathbf{x}_s}(o_n|\mathbf{x}_s) \approx 0$ for every candidate location $\mathbf{x}_s$ that is sufficiently far from the current approximation, that is, where $q(\mathbf{x}_s) \approx 0$. This behavior is reflected by the log-sum-exp in Equation 22, which provides additional reward based on the negative log probability density $-\log q(\mathbf{x}_s)$ of the current approximation but never much more than $-(\log q(o_n) + \log q_{\mathbf{x}_s}(\mathbf{x}_s|o_n))$.

The proposed heuristic has different effects depending on the choice of the initial weight, which upper-bounds the benefit of adding a component far from the current approximation to $-(\log q(o_n) + \log q_{\mathbf{x}_s}(\mathbf{x}_s|o_n))$ (Eq. 23). Small initial weights increase this threshold and, thus, the proposed heuristic becomes more explorative by tending to initialize new components far from the current approximation. However, a benefit of the proposed heuristic is that it often does not rely on a specific threshold to propose useful candidate locations. For example, when a candidate is very close to a mode of the target distribution that is currently not covered by the approximation, the heuristic will often choose it for a large range of different thresholds that might vary across several orders or magnitude. If there is no clear winner, the choice of $\log q(o_n)$ typically affects the proposed location. For relatively large initial weights, we will create the component at a location where $R(\mathbf{x}_s)$ is close to the best values that we have discovered and therefore often close to an existing component. Such component will improve our approximation with high probability by allowing the mixture model to approximate the mode more accurately, but is not likely to discover a new mode. Estimating the initial reward based on a small initial weight, in contrary, is more likely to place the component far from the current mixture model at locations where $R(\mathbf{x}_s)$ may be significantly worse than the best discovered values. Such component might converge to an irrelevant mode, that is, a local maximum of the target distribution that is still significantly worse than the best mode. The component will then get a very low weight, such that its effect on the approximation is negligible and the computational time (e.g., function evaluations) that was spent for improving this component was mainly wasted. If, however, such component discovers a new relevant mode, it will turn out much more valuable than a component that was added close to an existing mode.

In our experiments, we always add component with an initial weight of 1e−29 which results in $\log q(o_n) \approx -66.77$. However, this value is quite arbitrary because adding a new component with initial weight of 1e−300 would result in essentially the same mixture model and $\log q(o_n) \approx -690.78$. Hence, we do not estimate the initial reward based on the actual initial weight, but instead choose a value in place of $\log q(o_n)$ and vary it in the range of $[-1000, -50]$. By varying the (assumed) initial weight we can maintain exploration and avoid only adding components at irrelevant locations. Please refer to Appendix D for a sensitivity analysis and for details on how the initial reward in Equation 22 is approximated.

3.4.3 Initializing the Covariance Matrix of New Components

The initialization of the covariance matrix of the new component is performed in two steps. In the first step, we decide on the initial entropy; in the second step, we decide on the initial correlations.

A possible option for choosing the initial entropy is to use the same entropy that was used when initializing the mixture at the beginning of the optimization, which would typically be relatively large in accordance with an uninformed prior. Such an initialization has the benefit of maintaining broad exploration during the whole optimization, and is not very sensitive to the initialization of the mean. However, initializing new components with high entropy can also be very wasteful as it will typically take a long time until they can contribute to the approximation. Furthermore, smaller initial entropies in combination with our heuristic for initializing the mean will result in a more directed exploration of promising regions. Hence, we initialize the new component with an entropy that is similar to those of the best components in the current model, namely we choose $\mathrm{H}_{init} = \sum_o q(o)\mathrm{H}(q(\mathbf{x}|o))$ as initial entropy. The entropy of the best components will typically decrease during optimization until it reaches a problem specific level. Hence, the exploration of new components will also become more local, without falling below a reasonable level.

For deciding on the correlations among the different dimensions, we can consider restarting the local search from scratch by choosing an isotropic covariance matrix $\mathbf{\Sigma}_{iso} = c_{iso}\boldsymbol{I}$, and making use of the existing components by averaging their covariance matrices, that is, $\mathbf{\Sigma}_{avg} = c_{avg}\sum_o p(o|\boldsymbol{\mu}_{new})\mathbf{\Sigma}_o$, where c_{iso} and c_{avg} are appropriately chosen to obtain the desired entropy H_{init} as shown in Appendix E. In VIPS we always averaged the covariance matrices, which can be sensible when adding components close to existing ones, or when similar correlations occur at different locations. However, we noticed that such initialization can impair exploration and, thus, degrade performance in one of our new experiments as shown in Section 5.2.1. As it is often difficult to predict, whether the curvature at the most responsible components is similar to the curvature at the new component, we perform a line search over a step size $\alpha \in [0, 1]$ to find the best interpolation

$$\mathbf{\Sigma}_\alpha = \alpha\mathbf{\Sigma}_{iso} + (1-\alpha)\mathbf{\Sigma}_{avg}$$

between both candidate covariance matrices with respect to the expected reward

$$R_{\text{new}}(\alpha) = \int_{\mathbf{x}} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{new}, \mathbf{\Sigma}_\alpha)\log\tilde{p}(\mathbf{x})d\mathbf{x}.$$

The expected reward can be approximated using an importance weighted Monte Carlo estimate based on samples from the mixture

$$z(\mathbf{x}) = 0.5\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{new}, \mathbf{\Sigma}_{iso}) + 0.5\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{new}, \mathbf{\Sigma}_{avg}).$$

These samples and the respective function evaluations are also stored in the database $\mathcal{S}$ and can thus be reused during subsequent learning iterations.

Flow charts for the basic variant and the modified version are shown in Figure 1. An open-source implementation is available online.[2] In comparison to VIPS, VIPS++ makes

2. The implementation can be found at `https://github.com/OlegArenz/VIPS`.

better use of previous function evaluations and initializes new components based on a line search. Furthermore, VIPS++ uses fewer hyper-parameters by automatically adapting the bounds on the KL-divergences for the individual component updates and the regularization coefficients for fitting the reward surrogates. The number of hyper-parameters was further reduced by simplifications of the algorithms; namely, by performing an unconstrained optimization for the weight updates and by performing a single EM-like iteration on a given set of samples.

4. Related Work

We will now discuss related work in the fields of variational inference, sampling and policy search.

4.1 Variational Inference

Traditionally, variational inference was applied for learning coarse approximations of high dimensional distributions, typically by assuming that the individual dimensions of the random variable are uncorrelated—the so-called mean-field assumption—and by choosing the variational distribution based on the target distribution. For example, Saul et al. (1996) approximated the hidden nodes of sigmoid belief networks with Bernoulli distributions, enabling them to maximize a lower bound on the ELBO in closed form. An iterative procedure was used for improving this lower bound. As such approach can only model unimodal distributions, it was later extended to mixtures of mean field distributions (Jaakkola and Jordan, 1998; Bishop et al., 1998).

However, relying on a variational distribution that can be fitted in closed form can be restrictive and the necessary derivations can be a major burden when applying such variational inference approaches to different models. Hence, Gershman et al. (2012) introduced non-parametric variational inference (NPVI), a black-box approach to variational inference that can be applied to any twice-differentiable target distribution. NPVI is restricted to GMMs with uniform weights and isotropic components that are iteratively optimized using first-order and second-order Taylor approximations. Although such variational approximation can in principle approximate any target distribution arbitrarily well, NPVI is in general not suited for learning highly accurate approximations with a reasonable number of components as shown in our comparisons.

Similar to VIPS, several black box approaches to variational inference rely on function evaluations of the target distributions that are chosen by sampling the variational approximation. Ranganath et al. (2014) apply the log-derivative trick, which is well-known in reinforcement learning (Williams, 1992), to variational inference in order to estimate the gradient of the ELBO with respect to the policy parameters. The gradient estimation does not require the gradient of the reward $\log \tilde{p}(\mathbf{x})$ but typically suffers from high variance. Ranganath et al. (2014), thus, suggest control-variates and Rao-Blackwellization (for which they assume a mean-field approximation) for variance reduction. If the target distribution is differentiable and the variational approximation is reparameterizable, it is usually preferable to estimate the gradient with the reparameterization trick (Kingma and Welling, 2014; Rezende et al., 2014) which typically has much lower variance. Such approach can, for example, be used to train normalizing flows (Dinh et al., 2014). Normalizing flows are

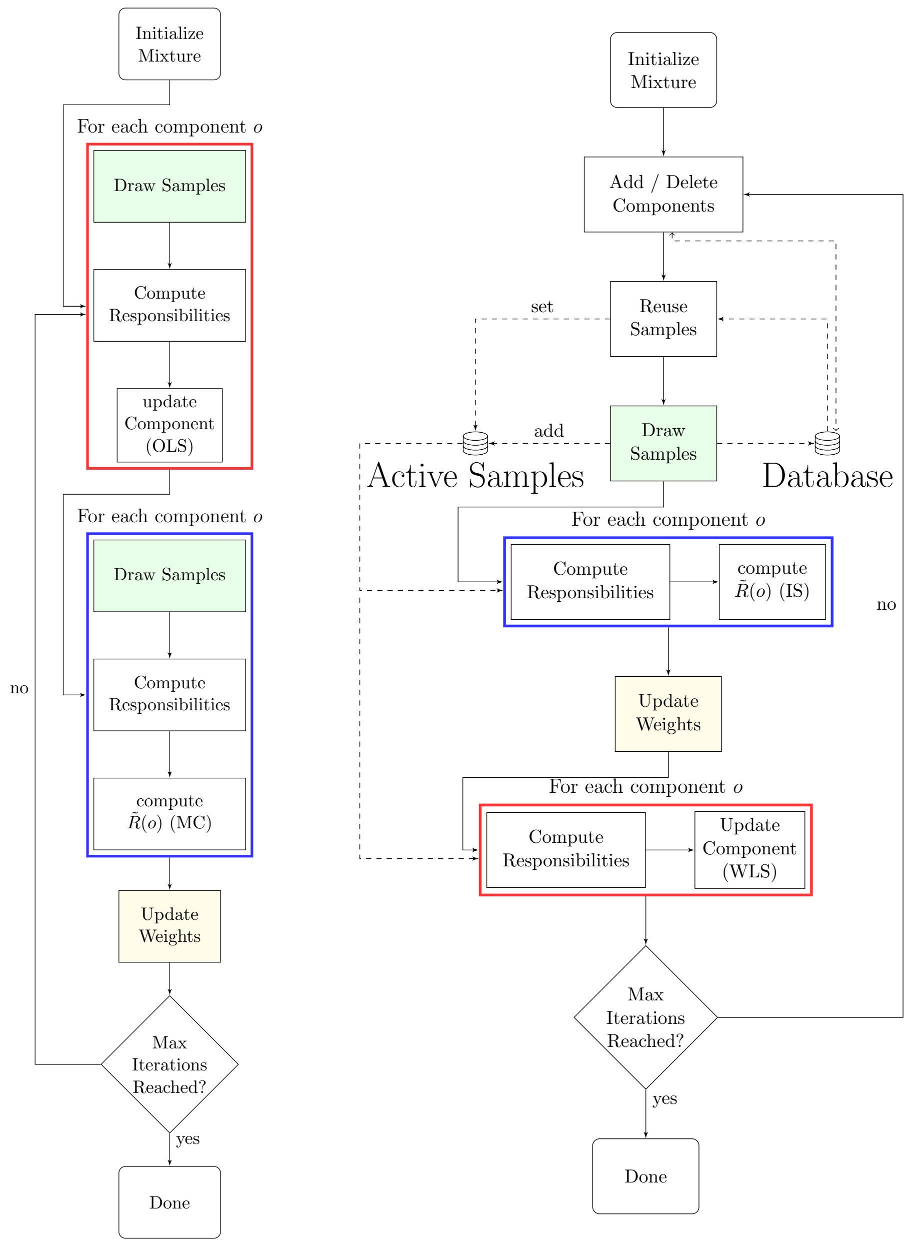


Figure 1: We show flow charts for the basic variant (left) and VIPS++ (right). The basic variant updates the individual components by learning surrogates using ordinary least-squares (OLS) and uses Monte-Carlo (MC) for estimating the component's reward $\tilde{R}(o)$. VIPS++ adapts the number of component and uses the same set of samples for computing the components' reward using importance sampling (IS) and for updating the individual components using weighted least squares. The order of the weight and component updates has been swapped on the right flow chart to match the actual implementation.

likelihood-based models that transform a simple distribution through one or several non-linear mappings. The probability density of the transformed distribution can be evaluated using the change-of-variables formula, which requires that the transformations are invertible and that the (log-)determinants of their Jacobians can be efficiently computed. Rezende and Mohamed (2015) proposed transformations that contract the density with respect to a learned hyperplane or to a point. The expressiveness of these planar and radial flows is rather limited and thus many flows has to be stacked to obtain rich approximations. However, several more expressive flows have been recently proposed (Kingma et al., 2016; Kingma and Dhariwal, 2018; Dinh et al., 2016; Papamakarios et al., 2017; Huang et al., 2018; Grathwohl et al., 2019). Most of these flows make use of autoregressive transformations. For example, *inverse autoregressive flows* (IAF, Kingma et al., 2016) shift and scale each dimension of an input, x_i, by quantities that are computed based on the previous input dimensions $x_{j<i}$. As the resulting Jacobian matrices are triangular, the log determinants can be efficiently computed based on the diagonal elements. Rich approximations can be learned by stacking several such flows and shuffling the dimensions in-between based on fixed random or learned (Kingma and Dhariwal, 2018) permutations, which can also be seen as normalizing flows. In order to ensure the autoregressive property, IAFs use a technique that was previously used for autoregressive auto-encoders (Germain et al., 2015). Namely, a mask is applied to a fully connected neural network in order to cut weights such that each output y_i is only connected to inputs x_j if $j < i$. Although such flows are invertible by construction, computing the inverse can be expensive because the different dimensions have to be inverted sequentially. Hence, evaluating the probability density of a sample that was produced by different distribution can be inefficient. *Masked autoregressive flows* (Papamakarios et al., 2017), thus, parameterize the inverse transformation (compared to IAFs) which makes them more efficient for density estimation at the cost of less efficient sampling. In general, normalizing flows are very popular nowadays, because they scale to high dimensions, allow for rich representations and are reparameterizable whenever the initial distribution is reparameterizable. However, we argue that such purely gradient-based optimization is not suited for learning accurate approximations of multimodal target distributions due to insufficient exploration. In our experiments, we compare against IAFs, which are well-suited for variational inference because we only need to evaluate the density of samples that were drawn from the normalizing flow.

Hessian-free stochastic Gaussian variational inference (HFSGVI, Fan et al. 2015) and TrustVI (Regier et al., 2017) can be used for learning Gaussian variational approximations. HFSGVI (Fan et al., 2015) learns GVAs with full covariance matrices using fast second order optimization. This idea has been extended by Regier et al. (2017) to trust region optimization. However, in difference to our approach, a euclidean trust region is used in parameter space of the variational distribution. Such approach requires the computation of the Hessian of the objective which is only tractable for mean-field approximations of single Gaussian distributions. In contrast, we use the trust regions directly on the change of the distributions instead of the change of the parameters of the distribution. The information geometric trust regions in this paper allow for efficient estimation of GMMs with full covariance matrices without requiring gradient information from $\tilde{p}(\mathbf{x})$.

Information geometric trust regions and related methods such as certain proximal point methods as well as methods based on natural gradient descent have already been applied to variational inference. Salimans and Knowles (2013) derive a fixed point update of the natural parameters of a distribution from the exponential family that corresponds to a Monte-Carlo estimate of the gradient of Equation 1 preconditioned by the inverse of their empirical covariance. By making structural assumptions on the target distribution, they extend their method to mixture models and show its applicability to bivariate GMMs. Hoffman et al. (2013) consider mean-field variational inference and assume a certain structure on the target distribution. Namely, they consider models that consist of a product of conditionally independent distributions parameterized by local parameters that are correlated through global parameters. Furthermore, all distributions are assumed to belong to the exponential family and the distribution of the global parameters is assumed to be conjugate for computational reasons. They show that the natural gradient of the corresponding mean-field approximation can be efficiently computed, and approximated from mini-batches. Theis and Hoffman (2015) extended their approach by enforcing a trust-region based on the KL-divergence for better exploration. Khan et al. (2015) consider slightly more general models where optimizing the ELBO can be computationally expensive. They propose to apply the proximal point method by adding a penalty to the ELBO based on the reverse Kullback-Leibler divergence to the current iterate. They decompose the ELBO into easy and difficult parts and linearize the difficult parts. The derivations where extended by Khan et al. (2016) to other divergences and to stochastic gradients making the approach applicable to posterior approximations based on mini-batches. Altosaar et al. (2018) propose a slightly more general framework that can penalize derivations from a moving average instead of derivations from the last iterate, which can further help in avoiding bad local optima.

Several methods use the same hierarchical bound as VIPS in the broad context of variational inference. The first usage seems to date back to 2004, where Agakov and Barber (2004) proposed the bound for learning an optimal weighting between several mean-field approximations. Ranganath et al. (2016) proposed Hierarchical variational methods (HVM) where the lower-level distributions $q(\mathbf{x}|\mathbf{o})$ where again mean-field distributions. In their setting, the latent variable $\mathbf{o}$ corresponds to a parameter vector that fully specifies the mean-field distribution. They learned complex priors $q(\mathbf{o})$ over these parameters, namely GMMs and normalizing flows, in order to allow for rich variational approximations. However, in contrast to the responsibilities in VIPS the conditional $q(\mathbf{o}|\mathbf{x})$ is not tractable and thus has to be approximated and learned along the variational distribution. Our EM-inspired approach based on exact tightening of the hierarchical lower bound would thus not be applicable in their setting. Although Ranganath et al. (2016) learned Gaussian mixture models to model the upper-level distribution $q(\mathbf{o})$, they did not apply the hierarchical bound for this, but optimized the parameters directly using stochastic gradient descent. As we will show in our experiments, such black-box approach is not suited for learning variational GMM approximations. Tran et al. (2016) consider a similar setup for their *variational Gaussian process*. For the mean-field factors $p(x_n|o_n)$ of their lower-level components they consider degenerated point masses specified by their scalar parameter value o_n. As Ranganath et al. (2016), they optimize the hierarchical lower bound with respect to the prior distribution $q(\mathbf{o})$ and the conditional $q(\mathbf{o}|\mathbf{x})$. Their main contribution is the representation of the prior distribution. Each parameter value o_n is sampled by evaluating a Gaussian process (GP, Rasmussen and Williams, 2006) on an input that was sampled from a fixed distribution. The parameters of the prior are given by the kernel hyper-parameters of the GP as well as the *variational data* that is interpolated by the GP. Whereas all these methods only consider mean-field distributions for the lower-level components that are fully specified by the latent variable, Maaløe et al. (2016) represent them using inference networks, that is, neural networks that take a data point as input and output the parameters of a (typically diagonal) Gaussian distribution. They consider variational autoencoders VAE and aim to learn more expressive approximations of the latent code z. They also introduce an additional latent variable representing class labels in order to train a classifier end-to-end while optimizing the variational autoencoder in semi-supervised fashion. In contrast to these previous applications of the hierarchical lower bound, VIPS shows that it can also be used to learn accurate variational approximations without having to approximate the inverse model $p(o|\mathbf{x})$. This enables us to optimize the ELBO by alternately maximizing and (exactly) tightening the hierarchical lower bound.

Closely related to our work are two recent approaches for variational inference that concurrently explored the idea of applying boosting to make the training of GMM approximations tractable (Miller et al., 2017; Guo et al., 2016). These methods start by minimizing the ELBO objective for a single component and then successively add and optimize new components and learn an optimal weighting between the previous mixture and the newly added component. However, because these methods can not adapt previously added components or their relative weighting, they can require an unnecessary large number of components to learn accurate approximations. Furthermore, they do not use information-geometric trust regions to efficiently explore the sample space and therefore have problems finding all the modes as well as accurate estimates of the covariance matrices. GMMs are also used by Zobay (2014) where an approximation of the GMM entropy is used to make the optimization tractable. The optimization is gradient-based and does not consider exploration of the sample space. It is therefore limited to rather low dimensional problems.

The work of Weber et al. (2015) already explored the use of reinforcement learning for VI by formalizing VI as sequential decision problem. However, only simple policy gradient methods have been proposed in this context which are unsuitable for learning GMMs.

4.2 Sampling

Although MCMC samplers can not directly be used for approximating distributions, they are for many applications the main alternative to VI. Especially, when applying VIPS as a model-based sampler, that is, if we do not have direct interest in learning a GMM approximation, it should be compared to other zero-order sampling methods that do not need gradient information from the target density. The most prominent methods to use here are MCMC methods such as slice sampling (Neal, 2003), elliptical slice sampling (Murray et al., 2010) or generalized elliptical slice sampling (Nishihara et al., 2014). MCMC methods define a Markov chain for the sampling process, that is, the current sample defines the state of the chain, and we define a conditional distribution how to generate new samples from the current state.

Slice sampling introduces an auxiliary variable y to define this conditional distribution. The variable y is always sampled between 0 and the unnormalized target density of the current sample. The random variable $\mathbf{x}$ is only accepted if the new target density is larger than y. In case of rejection, the area where a new $\mathbf{x}$ sample is generated is reduced to limit the number of rejections. However, the sampling process is still very inefficient for higher dimensional random variables. Elliptical slice sampling (Murray et al., 2010) is a special case of slice sampling and defines the slice by an ellipse defined by the current state $\mathbf{x}$ and a random sample from a Gaussian prior (with origin 0). Such ellipse allows for more efficient sampling and rejection in high dimensional spaces but relies on a strong Gaussian prior.

If the gradient of the target distribution is available, Hamiltonian MCMC (Duane et al., 1987) and the Metropolis-adjusted Langevin algorithm (Roberts and Stramer, 2002) are also popular choices. The No-U-Turn sampler (NUTS) (Hoffman and Gelman, 2014) is a notable variant of Hamiltonian MCMC that is appealing for not requiring hyper-parameter tuning.

While many of these MCMC methods have problems with multimodal distributions in terms of mixing time, other methods use multiple chains and can therefore better explore multimodal sample spaces (Earl and Deem, 2005; Neal, 1996; Nishihara et al., 2014; Calderhead, 2014). Parallel tempering MCMC (Earl and Deem, 2005) runs multiple chains, where each chain samples the target distribution at a different temperature. Each step consists either of updating each chain independently, or swapping the state between two neighboring chains which allows for more efficient mixing between isolated modes. However, because only one chain samples the target distribution at the correct temperature, PTMCMC can be inefficient if the number of chains and their respective temperatures are not adequately tuned for the sampling problem. Generalized elliptical slice sampling (Nishihara et al., 2014) uses multiple Markov chains simultaneously using massive parallel computing. The current state of the Markov chains is used to learn a more efficient proposal distribution, where either Student-t distributions or Gaussian mixture models can be used. Yet, learning such distributions in high dimensional spaces using maximum likelihood is prone to overfitting and the GMM approach has not been evaluated on practical examples. Moreover, the approach requires a massive amount of sample evaluations. In this paper, we want to minimize the amount of sample evaluations.

Rainforth et al. (2018) explicitly consider the exploration-exploitation trade-off. They use a method similar to Monte-Carlo tree search (Coulom, 2006) to build a tree for partitioning the search space. By covering regions where the target distribution has high density more finely, the resulting *inference trees* (IT) are well-suited for inference on multimodal distributions, for example, in combination with sequential Monte-Carlo (Doucet et al., 2001).

Stein variational gradient descent (SVGD) (Liu and Wang, 2016) is a sampling method that closely relates to variational inference. However, instead of optimizing the parameters of a model, SVGD directly optimizes an initial set of particles. By framing sampling as optimization problem, SVGD inherits the computational advantages of variational inference and because it is non-parametric, it is capable of approximating multimodal distributions. However, this method requires to construct the Gram matrix of the particles and is thus not suitable for drawing large number of samples. Furthermore, defining appropriate kernels can be challenging for high-dimensional problems.

4.3 Reinforcement Learning

Our algorithm shares a lot of ideas with information-geometric policy search algorithms such as REPS (Peters et al., 2010), HiREPS (Daniel et al., 2016) and MORE (Abdolmaleki et al., 2015). In difference to policy search, where we want to maximize an average reward objective, we want to minimize the KL-divergence to a target distribution. REPS introduces the first time information-geometric policy updates, while the MORE algorithm introduces closed form updates for single Gaussians using compatible function approximation and additional entropy regularization terms that yields an optimization problem similar to KL minimization.

The HiREPS (Daniel et al., 2016) and LaDiPS (End et al., 2017) algorithms extended the REPS and MORE ideas to mixture distributions such that multiple modes can be represented. However, the used updates were based on approximations or heuristics and can not optimize the entropy of the complete mixture model.

Storing all samples and the corresponding rewards in a database in order to reuse them during later iterations is known in reinforcement learning as experience replay (Lin, 1993; Mnih et al., 2013). Rather than uniformly sub-sampling from such *replay buffer*, we prioritize those samples that are more useful for the current updates, which can be seen as an instance of prioritized experience replay (Schaul et al., 2016). However, whereas Schaul et al. (2016) prioritize samples with large temporal-difference errors, which assumes time-series data, we prioritize samples that are close to the current components.

5. Experiments

In this section we will evaluate VIPS++ with respect to the quality of the learned approximation and relate it to a variety of state-of-the-art methods in variational inference and Markov chain Monte Carlo. We start with a description of the considered sampling problems in Section 5.1. The effects of the most important hyper-parameters and algorithmic choices are examined in Section 5.2. Section 5.3 contains an illustrative experiment to show how VIPS++ approximates a two-dimensional, multimodal target distribution by starting with a single component and iteratively adding more components according to our heuristic. The selected methods for our comparisons, and the selection of their hyper-parameters are discussed in Section 5.4 and Section 5.5. The results of the quantitative experiments are presented and discussed in Section 5.6.

5.1 Sampling Problems

We will evaluate VIPS++ on typical sampling problems such as Bayesian logistic regression, Bayesian Gaussian process regression and posterior sampling of a multi-level Poisson generalized linear model. We further approximate the posterior distribution over the parameters of a system of ordinary differential equations known as the Goodwin model, which can be used for modeling oscillating gene-protein interaction. As these problems tend to have concentrated modes, we devised several more challenging problems that require careful exploration of the sampling space. Namely, we consider sampling from unknown GMMs with distant modes and sampling the joint configurations of a planar robot such that it reaches given goal positions.

5.1.1 Bayesian Logistic Regression

We perform two experiments for binary classification that have been taken from Nishihara et al. (2014) using the *German credit* and *breast cancer* data sets (Lichman, 2013). The *German credit* data set has twenty-five parameters and 1000 data points, whereas the *breast cancer* data set is thirty-one dimensional and contains 569 data points. We standardize both data sets and perform linear logistic regression where we put zero-mean Gaussian priors with variance 100 on all parameters.

5.1.2 Multi-Level Poisson GLM

We also took an experiment from the related work VBOOST (Miller et al., 2017). For this experiment we want to sample the posterior of a hierarchical Poisson GLM on the 37-dimensional *stop-and-frisk* data set, where we refer to Miller et al. (2017) for the description of the hierarchical model.

5.1.3 GP Regression

We perform Bayesian Gaussian process regression on the ionosphere data set (Lichman, 2013) as described by Nishihara et al. (2014). Namely, we use 100 data points and want to sample the hyper-parameters of a squared exponential kernel where we put a gamma prior with shape 1 and rate 0.1 on the 34 length-scale hyper-parameters. We initialize VIPS with a single Gaussian component, $\mathcal{N}(\mathbf{x}|\mathbf{0}, \mathbf{I})$ and sample in log-space to ensure positive values for the hyper-parameters.

5.1.4 Goodwin Model

Similar to Calderhead and Girolami (2009), we want to sample the posterior over the parameters of a Goodwin oscillator (Goodwin, 1965) based on noisy observations. The Goodwin oscillator is a system of nonlinear ordinary differential equations (ODE) that models the oscillatory behavior between protein expression and mRNA transcription in enzymatic control processes. We consider a Goodwin oscillator with ten unknown parameters and put a Gamma prior with shape 2 and rate 1 on each of these. The likelihood of 41 observations is computed by numerically integrating the ODE and assuming Gaussian observation noise with zero mean and variance $\sigma^2 = 0.2$. Please refer to Appendix F for more details on the ODE and the experimental setup.

5.1.5 Gaussian Mixture Model

In order to evaluate how VIPS++ can explore and approximate multimodal probability distributions with distant modes, we consider the problem of approximating an unknown GMM comprising 10 components. We consider different number of dimensions, namely $D = 20$, $D = 40$ and $D = 60$. For each component, we draw each dimension of the mean uniformly in the interval $[-50, 50]$. The covariance matrices are given by $\mathbf{\Sigma} = \mathbf{A}^\top \mathbf{A} + \mathbf{I}_D$ where each entry of the $D \times D$-dimensional matrix $\mathbf{A}$ is sampled from a normal distribution with mean 0 and standard deviation $0.1D$. Note that each component of the target distribution can have a highly correlated covariance matrix, which is even a problem for the tested MCMC methods.

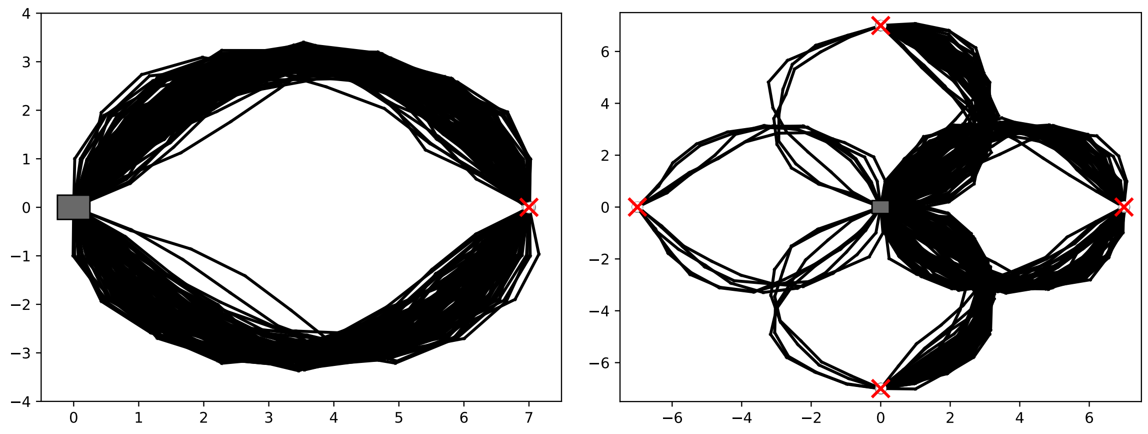


Figure 2: The plots show 200 ground-truth samples for both *planar robot* experiments that have been generated using generalized elliptical slice sampling. The base of the planar robot is shown as a gray box and the end-effector positions are shown as circles.

5.1.6 Planar Robot

In order to test VIPS++ on a multimodal problem with non-Gaussian modes we devised a challenging toy task where we want to sample the joint configurations of a planar robot with 10 links of length 1 such that it reaches desired goal positions. The robot base is at position $(0, 0)$ and the joint configuration describes the angles of the links in radian. In order to induce smooth configurations, we put a zero mean Gaussian prior on the joint configurations where we use a variance of 1 for the first joint and a variance of 4e$-$2 for the remaining joints. Deviations from the nearest goal position are penalized based on a likelihood that is given by a Gaussian distribution in the Cartesian end-effector space, with a variance of 1e$-$4 in both directions. We consider two experiments that differ in the number of goal positions. For the first experiment, we want to reach a single goal-position at position $x = 7$ and $y = 0$. For the second experiment we want to reach four goal positions at positions $(7, 0)$, $(0, 7)$, $(-7, 0)$ and $(0, -7)$. Please refer to Appendix G for details on how the target distribution is computed.

Ground-truth samples for both experiments are shown in Figure 2. Each goal position can be reached from two different sides, either up and down, or left and right. Other configurations that would reach the goal position, for example some zig-zag configurations, are not relevant due to the smoothness prior and can create poor local optima. Although there are only two relevant ways for reaching each goal position, closely approximating these modes can require many mixture components, because the small variance of the Cartesian likelihood term enforces components with small variance. We therefore also evaluate slightly different hyper-parameters for VIPS++, where we add a new component at every iteration.

5.2 Ablations

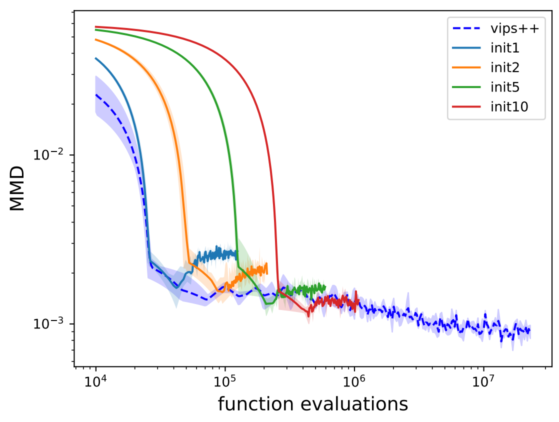
In this subsection we will evaluate the effects of some algorithmic choices. Namely, we will show that adapting the number of components can be crucial for discovering relevant modes of multimodal target distributions, that the previously proposed initializing of covariance matrices can have detrimental effects, and that the sample reusage of VIPS++ can significantly increase sample efficiency.

5.2.1 Adapting the number of components

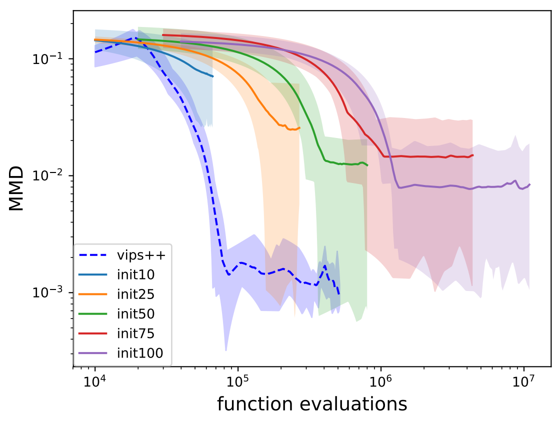
As discussed in Section 3.4, VIPS automatically adapts the number of components during learning for better exploration, which enables it to improve on local optimal solutions. We evaluate the effect of this adaptation by comparing VIPS++ with a variant that keeps the number of components fixed on the *breast cancer* experiment and the 20-dimensional *GMM* experiment. We initialize the non-adaptive variant with different numbers of initial components, where each mean is drawn from an isotropic Gaussian $\mathcal{N}(\mathbf{0}, \alpha \boldsymbol{I})$. We use $\alpha = 100$ for the *breast cancer* experiment and $\alpha = 1000$ for the *GMM* experiments. For VIPS++ we start with a single component with mean $\mathbf{0}$. All covariance matrices are initialized as $\Sigma = \alpha \boldsymbol{I}$. The achieved MMDs are shown in Figure 3. The non-adaptive variant converges to better approximations when increasing the number of components on the *breast cancer* experiment. However, the required number of function evaluations until convergence scales approximately linearly with the number of components. VIPS++ can learn good approximations with few function evaluations and further improves while increasing the size of the mixture model. On the *GMM* experiment, all tested variants would in principle be able to model the target distribution exactly. However, depending on the initialization, several components may converge to the same mode which results in bad local optima. We therefore needed at least 25 initial components for occasionally learning good approximations during this experiment and even when initializing with 100 components the non-adaptive variant would sometimes fail to discover all true modes. In contrast, by adaptively adding new components at interesting regions VIPS++ reliably discovers all ten modes. Please refer to Appendix H for a plot of the average number of components that are learned by VIPS++ for all experiments in the test bed.

5.2.2 Initializing the covariance matrices

We also evaluate the different strategies for initializing the covariance matrix of a newly added component, which were discussed in Section 3.4. We compare the proposed line search used by VIPS++ with the interpolation used by VIPS as well as an isotropic initialization. Figure 4 compares the different strategies on the *Goodwin* experiment and the *planar robot* experiment (with four goal positions). The *planar robot* experiment shows, that interpolating based on the responsibilities can seriously impair the performance on multimodal problems. We believe that interpolating based on the responsibilities can lead to highly anisotropic initial covariance matrices that do not sufficiently explore along relevant directions which would explain the detrimental effects. Although we could not show a benefit of the line search compared to the isotropic initialization, we opted for the line search for the quantitative experiments, because it seems sensible and did not perform significantly worse in our experiments.

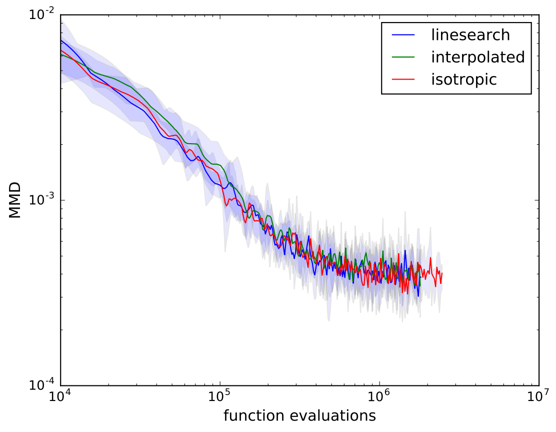


(a) breast cancer

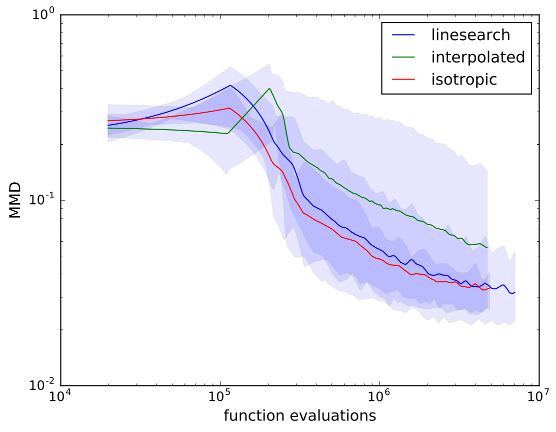


(b) GMM

Figure 3: We compare VIPS++ with a variant that does not add or delete components. On the *breast cancer* experiment, VIPS++ converges to a good approximation as fast as the variant that learns a single component, but it refines the approximation by adding more components. When not adapting the number of components on the *GMM* experiment, the quality of the approximation strongly depends on the initialization and even 100 initial components would sometimes fail to detect all modes.

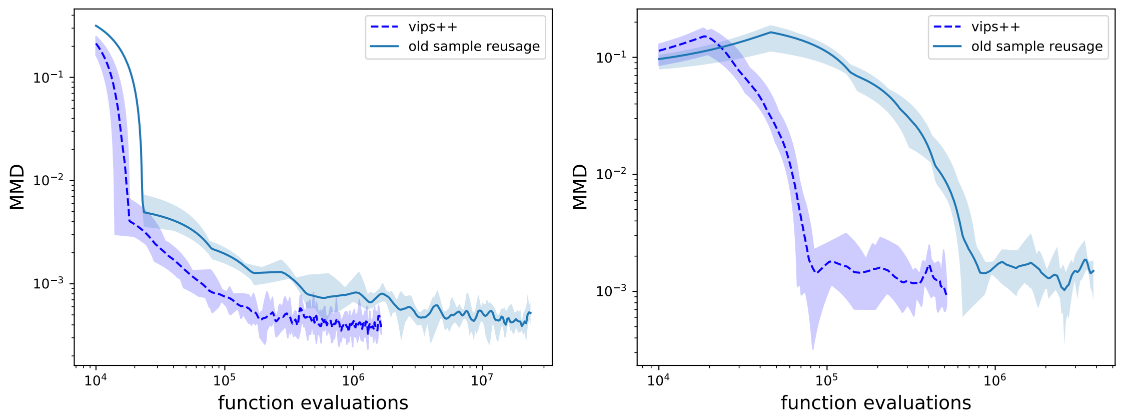


(a) Goodwin model



(b) planar robot

Figure 4: We compare different strategies for initializing the covariance matrices of newly added components. Interpolating the covariance matrices of the current model based on the responsibilities can have detrimental effects as shown in the planar robot experiment.



(a) Goodwin model (b) 20-dimensional GMM

Figure 5: The sample reusage of VIPS++ is approximately one order of magnitude more efficient than the sample reusage of VIPS.

5.2.3 Sample Reusage

Compared to VIPS, VIPS++ uses a more sophisticated method for reusing samples from previous iteration—as detailed in Section 3.2 and 3.3—by identifying relevant samples among all previous function evaluations and by controlling the number of new samples from each component based on its number of effective samples. We compare the new sample strategy with the previously employed method of always using the samples of the three most recent iterations. Figure 5 evaluates the different strategies on the *Goodwin* experiment and the 20-dimensional *GMM* experiment. The proposed strategy of VIPS++ significantly outperforms the previous method by reducing the sample complexity by approximately one order of magnitude.

5.3 Illustrative Experiment

We start with a qualitative two-dimensional experiment to illustrate the sample reusage and the adaptation of the number of components. The target distribution is given by a Gaussian mixture model with ten components similar to the higher-dimensional GMM experiments. We use the same hyper-parameters as in the remaining experiments and start with a single component. Figure 6 shows the target distribution as well as the learned approximation directly after adding each new component. The new components are often added close to missing modes and components are typically not sampled after they have converged. The learned model closely approximates the target distribution.

5.4 Considered Competitors

We compare VIPS++ to the closely related methods variational boosting (VBOOST, Miller et al., 2017) and non-parametric variational inference (NPVI, Gershman et al., 2012) as well as state-of-the-art methods in variational inference and MCMC, namely inverse autoregressive flows (IAF, Kingma et al., 2016), Stein variational gradient descent (SVGD,

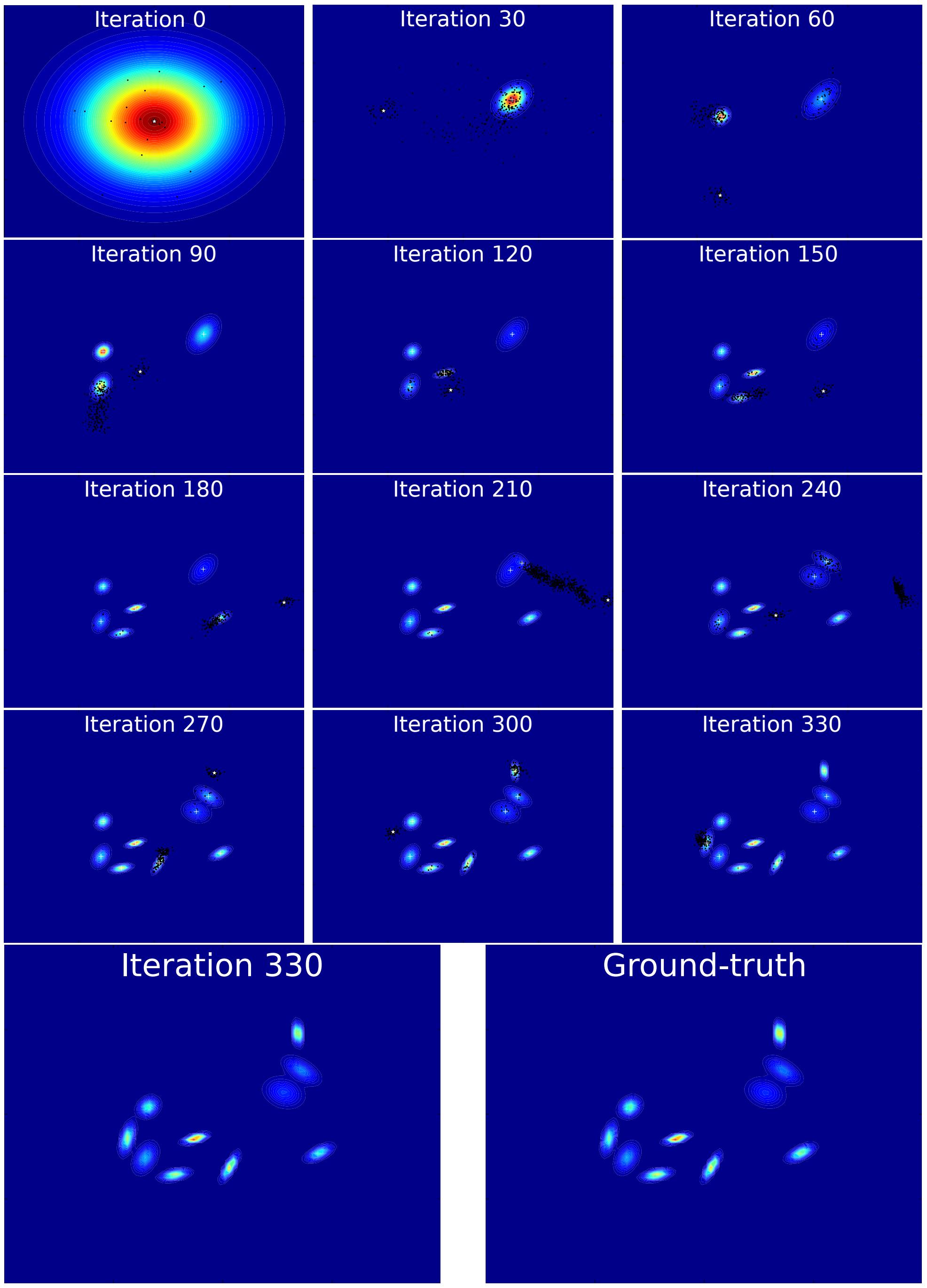


Figure 6: The first 12 plots show the learned approximation for the illustrative experiment every 30 iterations, directly after adding a new component. The means of the Gaussian mixture model are indicated with a white plus except for the newest component which is marked by a star. Black dots indicate all samples that have been drawn except for those that have already been shown at previous plots. The last two plots compare the learned approximation and the target distribution.

Liu and Wang, 2016), Hamiltonian Monte Carlo (HMC, Duane et al., 1987), elliptical slice sampling (ESS, Murray et al., 2010), parallel tempering MCMC (PTMCMC, Earl and Deem, 2005) and slice sampling (Neal, 2003). We also compare to naive gradient based optimization of a Gaussian mixture model (with fixed but tuned number of components). As GMMs are not exactly reparameterizable, we compute their stochastic gradients using black-box variational inference (BBVI). Please refer to Appendix I for details on the specific implementations. Due to the high computational demands, we do not compare to every method on each experiment but rather select promising candidates based on the sampling problem or on the preliminary experiments that we had to conduct for hyper-parameter tuning. We present our justification for each omitted experiment in Appendix J, where we also present a table that shows the competitors we compared against on each test problem.

Instead of using a variant of MORE (which we denote as VIPS1), it would also be possible to update the individual components using the reparameterization trick (Kingma and Welling, 2014; Rezende et al., 2014)—which assumes that the target distribution is differentiable—or black-box variational inference (Ranganath et al., 2014). We evaluated these options by comparing VIPS1, black-box variational inference and the reparameterization trick for learning Gaussian variational approximations on the *breast cancer* experiment and the *planar robot* experiment. The results are presented in Appendix K and show that VIPS1 is not only more efficient than black-box variational inference, but also one to two orders of magnitude more efficient than the reparameterization trick.

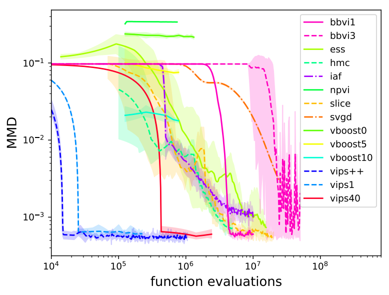
5.5 Hyper-Parameters

For the competing methods, we tuned the hyper-parameters independently for each test problem. We typically tuned the hyper-parameters based on our test metric, the maximum mean discrepancy (MMD). However, in all our experiments black-box variational inference and inverse autoregressive flows collapsed to single modes on multimodal test problems which increased the MMD. In these cases, we tuned the hyper-parameters with respect to the ELBO, rather than setting the learning rate to zero which would perform better on our test metric. For VIPS++, we use the same set of hyper-parameters on all experiments. However, for the planar robot experiment which can profit from large GMMs with several hundred components, we add a new component at every iteration. Learning such large mixture models for simpler, unimodal problems would be wasteful, and we thus use a slower adding rate $n_{\text{add}} = 30$ for the remaining experiments. The remaining hyper-parameters are shown in Appendix L.

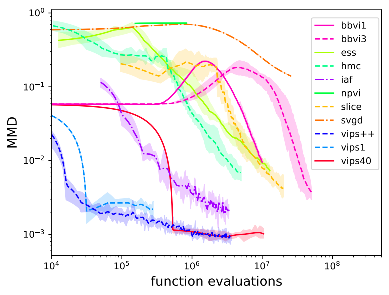
5.6 Results

We compare the different methods in terms of efficiency, regarding both, the number of function evaluations and wall clock time, and in terms of sample quality which we assess by computing the maximum mean discrepancy (MMD, Gretton et al., 2012) with respect to ground-truth samples. The MMD is a nonparametric divergence between mean embeddings in a reproducible kernel Hilbert space. Please refer to Appendix M on how the MMD and the ground-truth samples are computed.

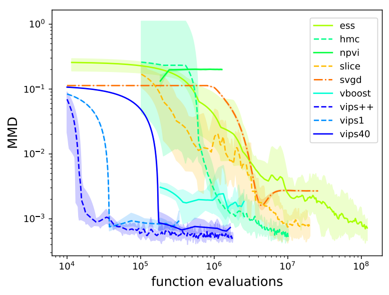
Figure 7 shows plots of the MMD over the number of function evaluations for the different sampling problems in the test bed. We perform five runs for each method and



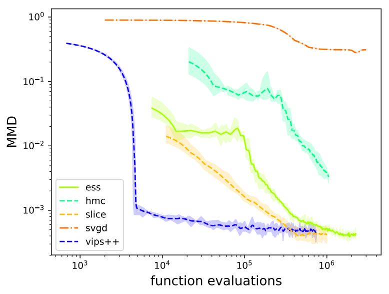
German credit (Log. Reg.)



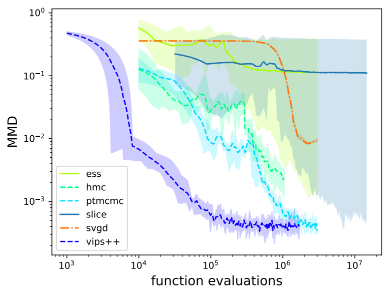
breast cancer (Log. Reg.)



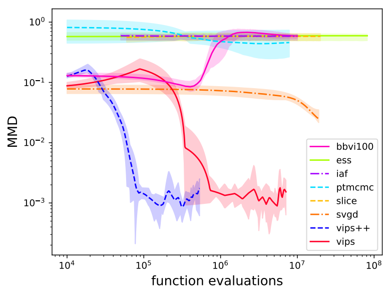
stop-and-frisk (Poisson GLM)



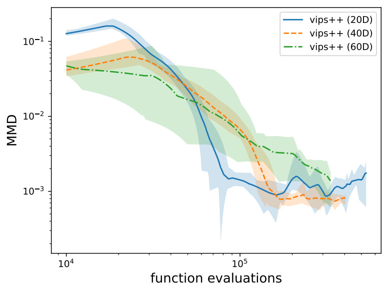
ionosphere (GP Regression)



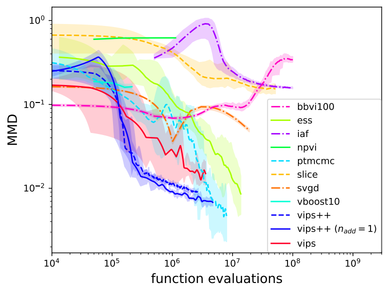
Goodwin model (ODE)



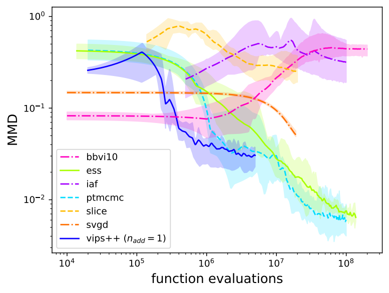
GMM (20 dimensions)



GMM (different dimensions)

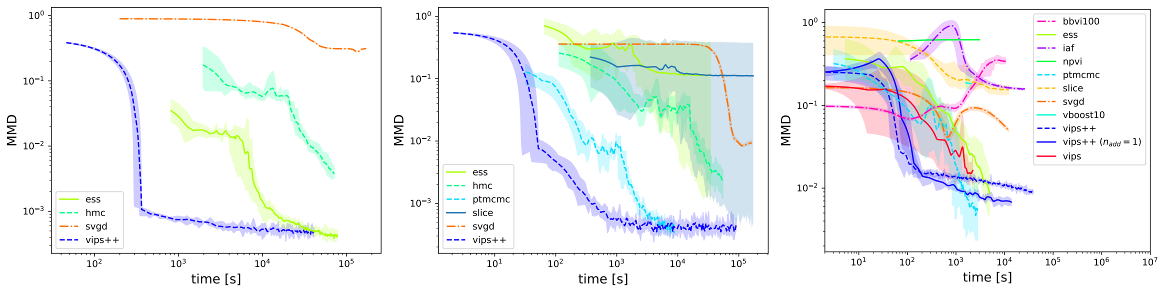


planar robot (1 goal)



planar robot (4 goals)

Figure 7: The maximum mean discrepancy with respect to ground-truth samples is plotted over the number of function evaluations on log-log plots for the different sampling problems in the test bed. VIPS++ achieves in most cases a sample quality that is on par with the best MCMC sampler while requiring up to three orders of magnitude fewer function evaluations.



ionosphere (GP Regression) Goodwin model (ODE) planar robot (1 goal)

Figure 8: Evaluating the methods with respect to computational time yields comparable results as evaluating with respect to the number of function evaluations. These results show that VIPS++ can also be competitive to MCMC in terms of computational time.

linearly interpolate the MMD values to produce continuous curves. The plots show the mean of these curves, as well as the smallest and largest value as shaded area. The tested methods are apparent from the legends. VBOOST can make use of low-rank approximations for learning the covariance matrices and we indicate the chosen ranks in the legends. The *German credit*, *breast cancer*, *stop-and-frisk* and the 20-dimensional *GMM* experiment, as well as the *planar robot* experiment with a single goal position were also used in our previous work (Arenz et al., 2018) and we use some of the previous results. For example, we directly compare VIPS++ with the previously published results of VIPS. Unlike VIPS++, VIPS bounds the maximum number of components by stopping to add new components if the current number of components matches a given threshold. This threshold is indicated in the respective legends. Figure 8 presents the results with respect to computational time for the *ionosphere* and *Goodwin model* experiment as well as the *planar robot* experiment with a single goal position. As the results are similar compared to the evaluations with respect to the number of function evaluations, we show the remaining plots in Appendix N.

Furthermore, for our comparisons with the variational inference methods BBVI and IAF we also present learning curves regarding the ELBO in Appendix O.

5.6.1 Discussion

The sample quality achieved by VIPS is unmatched by any variational inference method on all considered experiments and in most cases on par with the best MCMC sampler. VIPS requires significantly fewer function evaluations and computational resources for producing such high quality samples. VIPS++ is approximately one order of magnitude more efficient than VIPS and two to three orders of magnitude more efficient than the remaining methods. Among the considered methods, VIPS and VIPS++ were the only methods that could produce good results on the 20-dimensional *GMM* experiment, where they were able to reliably discover and approximate all ten modes of the target distribution. We therefore only evaluated VIPS++ on the higher-dimensional *GMM* experiments where it also approximated the target distribution with high accuracy. However, on the *planar robot*

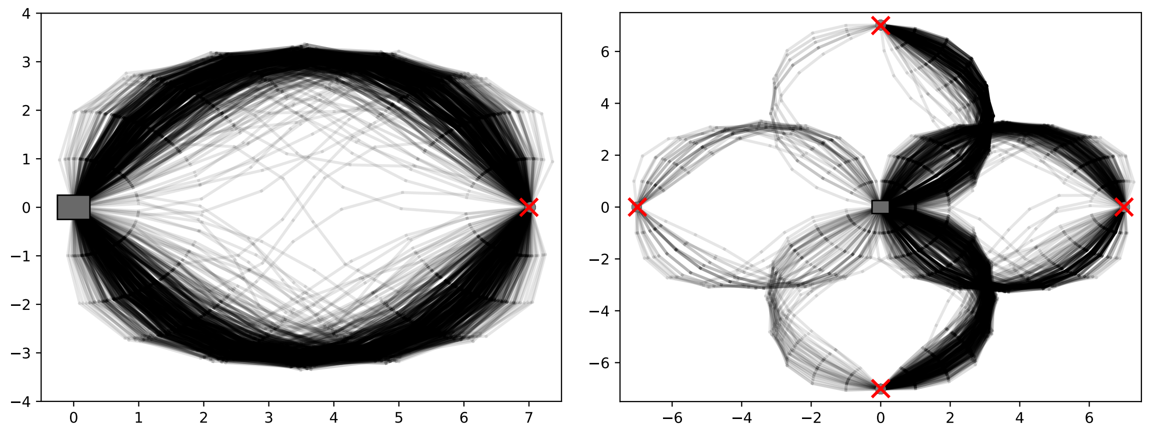


Figure 9: The plots visualize the weights and means of the mixture models learned by VIPS++ for each of the planar robot experiments when adding new components with adding rate $n_{\text{add}} = 1$. The gray box indicates the base of the robot; the red crosses indicate the goal positions. Components with larger weight are drawn darker. The visualized mixture models comprise of 333 and 360 components for the experiments with one goal position (left) and four goal positions (right), respectively.

experiment with four goal positions ESS and PTMCMC could produce significantly better samples than VIPS++. We believe that learning highly accurate GMM approximations would require a very large number of components for this experiment. Already on the *planar robot* experiment with a single goal position, we could slightly improve the learned approximations by adding new components more frequently. Compared to the default adding rate, which learned GMMs with approximately 150 components, the faster adding rate resulted in GMMs with approximately 350 components. We believe that VIPS++ would require significantly more components to achieve comparable sample quality to the MCMC samplers on the more challenging *planar robot* experiment. However, learning very large mixture models can become infeasible, because computing the (log-)responsibilities $\log q(o|\mathbf{x})$ exactly can become prohibitive. Figure 9 visualizes the weights and means of the learned approximation of the first run for both *planar robot* experiments when adding new components at every iteration. We can see that the learned components are still of very good quality. Samples from the learned models are shown in Appendix P and compared to those obtained by BBVI, IAF, PTMCMC.

6. Conclusion and Future Work

We proposed VIPS++, a method for learning GMM approximations of intractable probability distributions that exploits the connection between variational inference and policy search. We introduced a variant of MORE (Abdolmaleki et al., 2015) that can be efficiently used for learning Gaussian variational approximations. We further derived a lower bound on the I-projection to latent variable models that can be used for learning a local optimum of the true objective, similar to expectation-maximization. By applying this decomposition to Gaussian mixture models, the I-projection can be performed independently for each component, allowing us to improve the GMM approximation by independently updating the components using our variant of MORE. We argue that a good trade-off between exploration and exploitation is essential for efficiently learning accurate multimodal approximations. We tackle the exploration-exploitation dilemma locally for each component by updating them using information-geometric trust regions. For global exploration, we dynamically add new components at interesting regions.

For target distributions that can be well approximated with a small number of components, VIPS does not only outperform existing methods for variational inference, but is also several orders of magnitude more efficient than Markov chain Monte Carlo at drawing samples. We also showed that VIPS can learn large mixture models comprising several hundred components. However, learning very large GMMs is computationally expensive and MCMC methods can be more efficient at drawing samples.

Learning Gaussian components with full covariance matrices can become intractable for high dimensional problems, and we thus applied VIPS only for medium-scaled problems with up to 60 dimensions. For significantly higher-dimensional problems, learning low-rank approximations and using gradient information for the component updates are interesting routes of future work. It is also interesting to further investigate the strong ties between variational inference and policy search. Using our decomposition we can learn GMMs of policy parameters for the black-box reinforcement learning setting where time-series data is not assumed and exploited. In order to apply VIPS for multimodal reinforcement learning with time-series data, we aim to contextualize the GMM parameterization on the state of an MDP to directly learn GMM policies. Furthermore, it is interesting to investigate how our decomposition can be applied to different problems such as clustering or density estimation, or to other latent variable models.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645582 (RoMaNS). Calculations for this research were conducted on the Lichtenberg high performance computer of the TU Darmstadt.

Appendix A. VIPS1 Derivations

For each update we wish to solve the optimization problem

$$\begin{aligned}
\max_{q(\mathbf{x})} \quad & \int_{\mathbf{x}} q(\mathbf{x})\tilde{R}(\mathbf{x})d\mathbf{x} + \mathrm{H}(q(\mathbf{x})),\\
\text{subject to } \ & \mathrm{KL}\Big(q(\mathbf{x})||q^{(i)}(\mathbf{x})\Big) \leq \epsilon,\\
& \int_{\mathbf{x}} q(\mathbf{x})d\mathbf{x} = 1,
\end{aligned}$$

where we recall that the the reward surrogate $\tilde{R}(\mathbf{x})$ is a quadratic function and the variational approximation of the previous iteration, $q^{(i)}(\mathbf{x})$, is Gaussian. We formulate the optimization for general distributions $q(\mathbf{x})$, but we will see that the optimal solution is also Gaussian. Using the definition of the Shannon entropy and Kullback-Leibler divergence and introducing the Lagrangian multipliers η and λ, the Lagrangian function is given by

$$\begin{aligned}
\mathcal{L}(q,\eta,\lambda) =& \int_{\mathbf{x}} q(\mathbf{x})\big(\tilde{R}(\mathbf{x}) - \log q(\mathbf{x})\big)d\mathbf{x} + \eta\left(\epsilon - \int_{\mathbf{x}} q(\mathbf{x})\big(\log q(\mathbf{x}) - \log q^{(i)}(\mathbf{x})\big)d\mathbf{x}\right)\\
&+ \lambda\big(1 - \int_{\mathbf{x}} q(\mathbf{x})d\mathbf{x}\big)\\
=& \int_{\mathbf{x}} q(\mathbf{x})\big(\tilde{R}(\mathbf{x}) - (1+\eta)\log q(\mathbf{x}) + \eta\log q^{(i)}(\mathbf{x}) - \lambda\big)d\mathbf{x} + \eta\epsilon + \lambda.
\end{aligned}$$

The optimum $q^{\star}(\mathbf{x})$ occurs where the partial derivative $\frac{\partial\mathcal{L}(q,\eta,\lambda)}{\partial q(\mathbf{x})}$ is equal to zero, that is,

$$\begin{aligned}
\frac{\partial\mathcal{L}(q^{\star},\eta,\lambda)}{\partial q(\mathbf{x})} &= \tilde{R}(\mathbf{x}) - (1+\eta)\log q^{\star}(\mathbf{x};\eta,\lambda) - (1+\eta) + \eta\log q^{(i)}(\mathbf{x}) - \lambda \overset{!}{=} 0\\
&\Rightarrow q^{\star}(\mathbf{x};\eta,\lambda) = \exp\Big(-\frac{\lambda+1+\eta}{1+\eta}\Big)\exp\Big(\frac{\tilde{R}(\mathbf{x}) + \eta\log q^{(i)}(\mathbf{x})}{1+\eta}\Big). \qquad (24)
\end{aligned}$$

The Lagrange dual function is, thus, given by

$$\begin{aligned}
\mathcal{G}(\eta,\lambda) =& \mathcal{L}(q^{\star},\eta,\lambda)\\
=& \int_{\mathbf{x}} q^{\star}(\mathbf{x};\eta,\lambda)\big(\tilde{R}(\mathbf{x}) - \big(-\lambda - 1 - \eta + \tilde{R}(\mathbf{x}) + \eta\log q^{(i)}\big) + \eta\log q^{(i)}(\mathbf{x}) - \lambda\big)d\mathbf{x}\\
&+ \eta\epsilon + \lambda\\
=& (1+\eta)\int_{\mathbf{x}} q^{\star}(\mathbf{x};\eta,\lambda)d\mathbf{x} + \eta\epsilon + \lambda.
\end{aligned}$$

As strong duality holds due to Slater's condition (Boyd and Vandenberghe, 2004), we can find the optimal distribution $q^\star(\mathbf{x};\eta,\lambda)$ by minimizing the dual function with respect to η and λ and then using the optimal step size $\eta^\star$ and Lagrangian multiplier $\lambda^\star$ to compute $q^\star(\mathbf{x};\eta^\star,\lambda^\star)$ according to Equation 24. The partial derivatives are given by

$$\begin{aligned}\frac{\partial\mathcal{G}(\eta,\lambda)}{\partial\eta} &= \epsilon + \int_{\mathbf{x}} q^\star(\mathbf{x};\eta,\lambda)d\mathbf{x} + (1+\eta)\int_{\mathbf{x}} q^\star(\mathbf{x};\eta,\lambda)\Big(\frac{\log q^{(i)}(\mathbf{x})-1}{1+\eta} - \frac{\log q^\star(\mathbf{x};\eta,\lambda)}{(1+\eta)}\Big)d\mathbf{x}\\ &= \epsilon - \int_{\mathbf{x}} q^\star(\mathbf{x};\eta,\lambda)\big(\log q^\star(\mathbf{x};\eta,\lambda) - \log q^{(i)}\big)d\mathbf{x}\end{aligned}$$

and

$$\frac{\partial\mathcal{G}(\eta,\lambda)}{\partial\lambda} = -\int_{\mathbf{x}} q^\star(\mathbf{x};\eta,\lambda)d\mathbf{x} + 1,$$

where the optimal Lagrangian multiplier $\lambda^\star(\eta)$ for a given η normalizes $q^\star(\mathbf{x};\eta,\lambda^\star)$, that is,

$$\frac{\partial\mathcal{G}(\eta,\lambda^\star)}{\partial\lambda} = 0 \Leftrightarrow \int_{\mathbf{x}} q^\star(\mathbf{x};\eta,\lambda^\star)d\mathbf{x} = 1.$$

Hence, we can perform coordinate descent by alternately updating η along its partial derivative and computing the optimal λ. Such procedure corresponds to optimizing the dual

$$\mathcal{G}(\eta) = (1+\eta)\int_{\mathbf{x}} q^\star(\mathbf{x};\eta,\lambda^\star)d\mathbf{x} + \eta\epsilon + \lambda^\star(\eta) = 1+\eta+\eta\epsilon+\lambda^\star(\eta) \tag{25}$$

based on the gradient

$$\frac{\partial\mathcal{G}(\eta)}{\partial\eta} = \epsilon - \mathrm{KL}(q^\star(\mathbf{x};\eta,\lambda^\star)||q^{(i)}(\mathbf{x})).$$

We will now express the approximation of the previous iteration $q^{(i)}(\mathbf{x})$ in terms of its natural parameters $\mathbf{Q}^{(i)}$ and $\mathbf{q}^{(i)}$ and the reward surrogate as

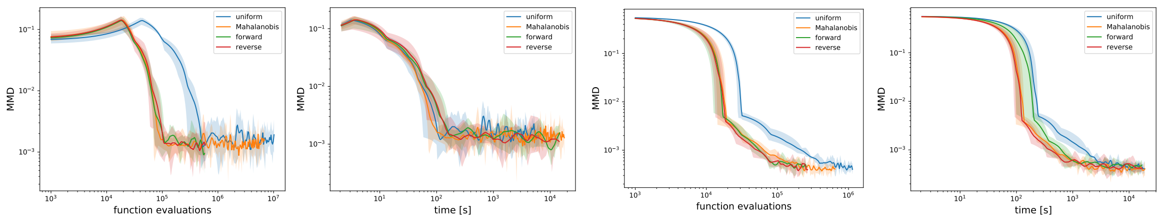
$$\tilde{R}(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^\top\mathbf{R}^{(i)}\mathbf{x} + \mathbf{x}^\top\mathbf{r}^{(i)}.$$

Then, according to Equation 24, the optimal distribution

$$\begin{aligned}q^\star(\mathbf{x};\eta) =& \exp\big(\frac{\eta\log Z(\mathbf{Q}^{(i)},\mathbf{q}^{(i)}) - \lambda^\star(\eta) - 1 - \eta}{1+\eta}\big)\\ &\cdot\exp\big(-\frac{1}{2}\mathbf{x}^\top\frac{\mathbf{R}^{(i)}+\eta\mathbf{Q}^{(i)})}{1+\eta}\mathbf{x} + \mathbf{x}^\top\frac{\mathbf{r}^{(i)}+\eta\mathbf{q}^{(i)})}{1+\eta}\big)\end{aligned} \tag{26}$$

is Gaussian with natural parameters

$$\mathbf{Q}(\eta) = \frac{\eta}{\eta+1}\mathbf{Q}^{(i)} + \frac{1}{\eta+1}\mathbf{R}^{(i)}, \qquad \mathbf{q}(\eta) = \frac{\eta}{\eta+1}\mathbf{q}^{(i)} + \frac{1}{\eta+1}\mathbf{r}^{(i)}.$$



(a) GMM (20D) (b) GMM (20D) (c) Goodwin (d) Goodwin

Figure 10: Using the Mahalanobis distance as dissimilarity measure results in similar sample efficiency compared to using the KL divergence while adding less computational overhead.

Further, we can see from Equation 26 and the optimality condition $\int_{\mathbf{x}} q(\mathbf{x};\eta,\lambda^\star)=1$, that

$$\begin{aligned}\lambda^\star(\eta) =& -(1+\eta)\log\int_{\mathbf{x}}\exp\Big(-\frac{1}{2}\mathbf{x}^\top\frac{\mathbf{R}^{(i)}+\eta\mathbf{Q}^{(i)})}{1+\eta}\mathbf{x}+\mathbf{x}^\top\frac{\mathbf{r}^{(i)}+\eta\mathbf{q}^{(i)})}{1+\eta}\Big)d\mathbf{x}-1-\eta\\ &+\eta\log Z(\mathbf{Q}^{(i)},\mathbf{q}^{(i)})\\ =&\eta\log Z(\mathbf{Q}^{(i)},\mathbf{q}^{(i)})-(1+\eta)\log Z(\mathbf{Q}(\eta),\mathbf{q}(\eta))-1-\eta. \end{aligned} \tag{27}$$

Using Equation 27 and Equation 25, the dual function can be expressed as

$$\mathcal{G}(\eta)=\eta\epsilon+\eta\log Z(\mathbf{Q}^{(i)},\mathbf{q}^{(i)})-(1+\eta)\log Z(\mathbf{Q}(\eta),\mathbf{q}(\eta)). \tag{28}$$

Appendix B. Effects of Different Dissimilarity Measures for Sample Selection

VIPS++ uses the Mahalanobis distance to the mean of the distributions in the sample database as dissimilarity measure for sample selection according to Equation 19. We compared this choice to different dissimilarity measures, namely $KL\left(q(\mathbf{x}|o)||\mathcal{N}_{\mathbf{x}_i}(\mathbf{x})\right)$ (denoted as reverse KL) and $KL\left(\mathcal{N}_{\mathbf{x}_i}(\mathbf{x}||q(\mathbf{x}|o))\right)$ (denoted as forward) and against using a uniform distribution instead of Equation 19. The results are shown in Figure 10.

Appendix C. Pseudo-Code for Sample Selection

The procedure for selecting relevant samples from the database is shown in Algorithm 4.

Appendix D. Approximating the Initial Reward and Sensitivity Regarding its Hyper-parameter

We approximate the initial reward of a new component based on the approximation given by Equation 23 because it is simpler and more efficient and unlikely to affect the selected candidate. Please note that the difference between the log-sum-exp and the maximum is numerically zero unless for candidates where the density of the current mixture model is close to the threshold. In such case the log-sum-exp can be larger by at most log(2).

Algorithm 4 Identifying relevant samples in the database

```
Require: database $\mathcal{S} = \{(\mathbf{x}_0, \log \tilde{p}(\mathbf{x_0}), \mathcal{N}_{\mathbf{x_0}}), \dots, (\mathbf{x}_N, \log \tilde{p}(\mathbf{x_N}), \mathcal{N}_{\mathbf{x_N}})\}$
Require: number of components in the approximation, $N_o$
Require: desired number of samples that should be reused per component, $n_{\text{reuse}}$
 1: function SELECT_SAMPLES
 2:     $\boldsymbol{\mathcal{X}}_{\subset} \leftarrow \{\}$
 3:     for $o = 1 \dots N_o$ do
 4:         $n_{\text{added}} \leftarrow 0$
 5:         $h(\cdot, o) \leftarrow$ compute for each distinct component in the database according to (19)
 6:         while $n_{\text{added}} < n_{\text{reuse}}$ do
 7:             $i \sim h(\cdot, o)$                    ▷ choose a distribution by sampling $h(i, o)$
 8:             $h(\cdot, o) \leftarrow$ remove element $i$ and normalize
 9:             for each sample $\mathbf{x_j}$ of component $\mathcal{N}_i$ do
10:                 if $\mathbf{x}_j \notin \boldsymbol{\mathcal{X}}_{\subset}$ then
11:                     $\boldsymbol{\mathcal{X}}_{\subset} \leftarrow \boldsymbol{\mathcal{X}}_{\subset} \cup \mathbf{x}_j$
12:                 end if
13:                 $n_{\text{added}} \leftarrow n_{\text{added}} + 1$          ▷ also count $\mathbf{x}_j$ if it was already added
14:                 if $n_{\text{added}} == n_{\text{reuse}}$ then
15:                     break
16:                 end if
17:             end for
18:         end while
19:     end for
20:     return $\boldsymbol{\mathcal{X}}_{\subset}$
21: end function
```

Furthermore, during our experiments we did not exploit that the initial entropy of the new component can already be computed before deciding on the mean of the new component. Hence, we estimated the density at its mean as

$$q_{\mathbf{x}_s}(\mathbf{x}_s|o_n) \approx \max_{\mathbf{x}_i \in \mathcal{X}_{\text{total}}} \log q(\mathbf{x}_i), \tag{29}$$

since the current approximation needs to be evaluated anyway on each candidate for the first operand of the maximum operator in Equation 23. In the main document we presented the more principled, exact computation of $q_{\mathbf{x}_s}(\mathbf{x}_s|o_n)$ to improve clarity. However, in practice the difference between the described heuristic and the implemented heuristic is negligible because the errors that are introduced by the approximation are small compared to the variations of the assumed log weight $\log q(o_n)$.

For varying the (negated) assumed initial weights we specify several different values in an array $\mathbf{\Delta} = [1000, 500, 200, 100, 50]$ and pick one of these values $\mathbf{\Delta}_j$ by cycling through this array. The adding heuristic is thus computed as

$$\tilde{R}_{\mathbf{x}_s}(o_\text{n}) = R(\mathbf{x}_s) - \max\left(\log q(\mathbf{x}_s), \max_{\mathbf{x}_i \in \mathcal{X}_{\text{total}}} \log q(\mathbf{x}_i) - \mathbf{\Delta}_j\right). \tag{30}$$

Instead of pre-specifying the possible values for the assumed initial weights, it would also be possible to sample continuous values from a given distribution. However, for small changes in the assumed initial weight the heuristic would typically select qualitatively similar candidates and, thus, it is simpler to specify a few values that relate to different levels of exploration than to specify a distribution. It would also be possible to specify a single value Δ, however, this would add a hyper-parameter that has to be tuned depending on the experiment. Furthermore, switching between different levels of exploration can be more efficient because we do not only want to add components close to missing modes, but also close to modes that are already covered in order to approximate them better. Figure 11 shows learning curves for different values of Δ on the *planar robot* experiment with four goals, which features several disconnected non-Gaussian modes. Here, varying the values performed better than any fixed assumed value for the initial weight. Figure 12 shows the different initial means that would have been chosen depending on the assumed initial weight. The selected candidates are sensible for a large range of Δ.

Appendix E. Scaling a Gaussian to Obtain a Desired Entropy

We want to find the scaling factor c to obtain a desired entropy H_init for a Gaussian distribution with given covariance matrix $\mathbf{\Sigma}$ of order $n \times n$.

$$\begin{aligned}\text{H}(\mathbf{\Sigma}; c) &= \frac{1}{2}\log|2\pi e c\mathbf{\Sigma}| \\ &= \frac{1}{2}n\log(c) + \frac{1}{2}\log|2\pi e\mathbf{\Sigma}| \overset{!}{=} \text{H}_\text{init} \Rightarrow c = \exp\left(\frac{1}{n}\left(2\text{H}_\text{init} - \log|2e\pi\mathbf{\Sigma}|\right)\right)\end{aligned}$$

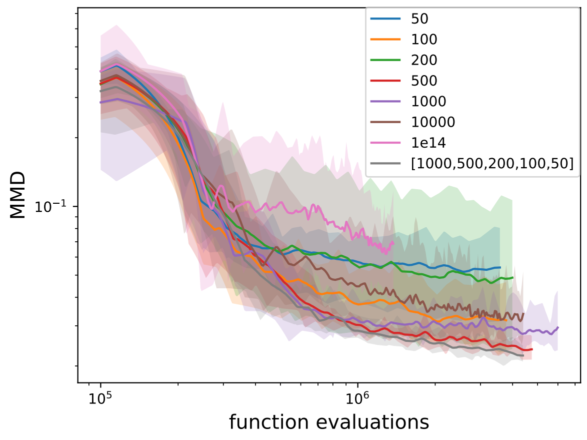


Figure 11: We evaluated the MMD for the *planar robot* experiment with four goals for different fixed values Δ of the assumed initial log-weight (negated) as well as for varying values. Varying the value (VIPS++) performed better than any fixed value that we tested. However, the experiment with $\Delta = 500$ indicates that tuning a fixed value may also perform well.

Appendix F. Goodwin Model

The Goodwin model is defined as

$$\begin{aligned}
\frac{dx_1}{dt} &= \frac{a_1}{1 + a_2 x_g^{\rho}} - \alpha x_1 \\
\frac{dx_2}{dt} &= k_1 x_1 - \alpha x_2 \\
&\vdots \\
\frac{dx_g}{dt} &= k_{g-1} x_{g-1} - \alpha x_g,
\end{aligned} \tag{31}$$

where x_1 represents the concentration of mRNA for a target gene, x_2 represents the corresponding protein product of the gene, and x_3 to x_g are intermediate protein species that ultimately lead to a negative feedback, via x_g, on the rate at which mRNA is transcribed. We consider $g = 9$ intermediate species and assume that the parameters $\rho = 10$ and $\alpha = 0.53$ are known. We put a Gamma prior with shape 2 and rate 1 on the remaining 10 parameters a_1, a_2 and $\kappa_1 \dots \kappa_8$ that need to be inferred. We use the prior also to randomly choose their true values. For an initial condition $\mathbf{x}_0 = \mathbf{0}$, we create 81 noisy observations $\mathbf{o}_{1\dots81}$ of x_1 and x_2 using steps of $dt = 1$. We assume Gaussian observation noise with zero mean and variance $\sigma^2 = 0.2$ and discard the first 40 observations. The posterior distribution is given by

$$p(a_1, a_2, \kappa_1, \dots, \kappa_8 | \mathbf{o}_{40\dots81}) = \frac{1}{Z} p(a_1) p(a_2) \prod_{i=1}^{8} p(\kappa_i) \prod_{t=40}^{81} p_t(\mathbf{o}_t | a_1, a_2, \kappa_1, \dots, \kappa_8), \tag{32}$$

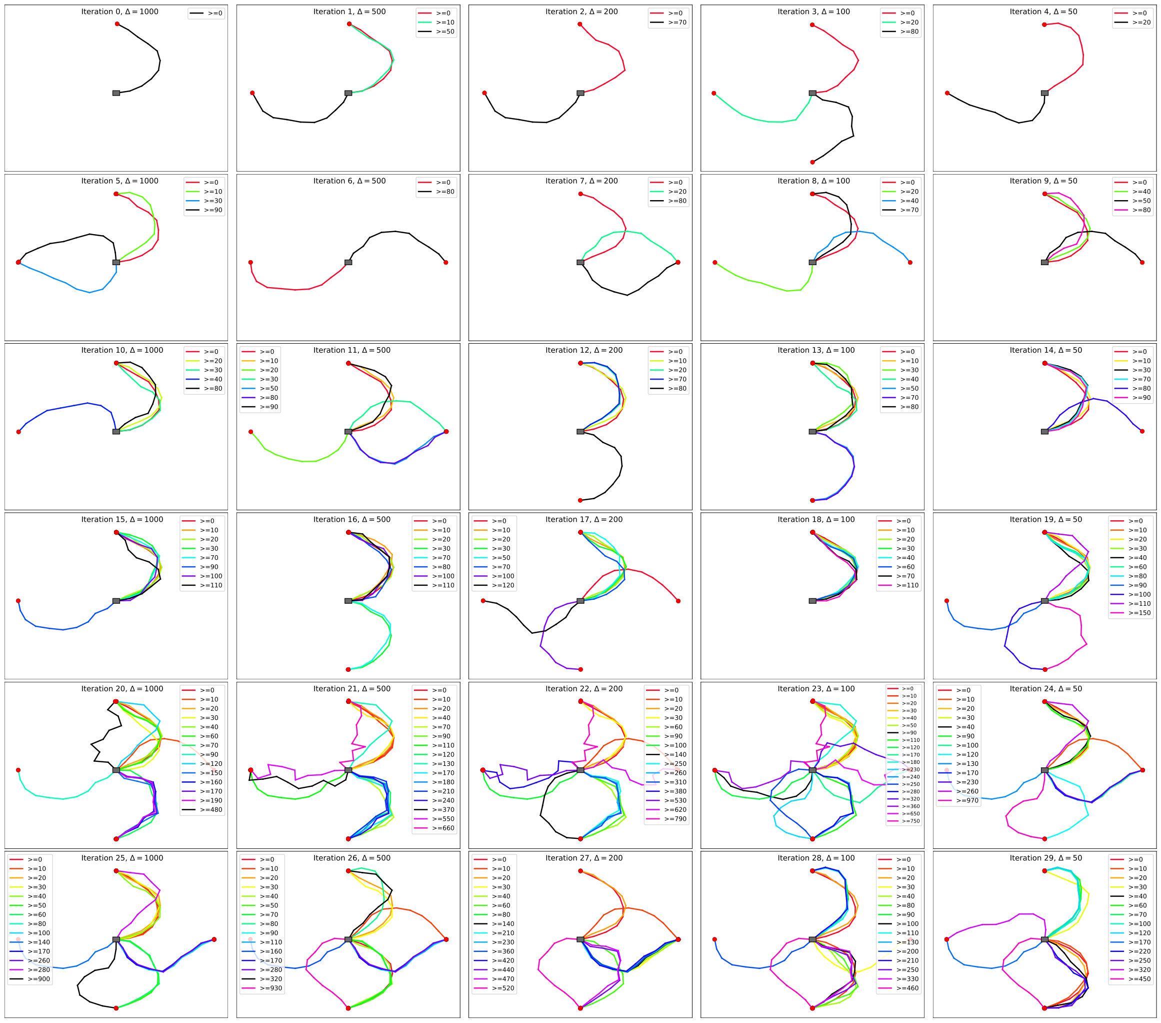


Figure 12: The plots show the candidates selected by the heuristic (Equation 30) for values of Δ from 0 to 1000 in steps of 10 on the *planar robot* experiment with 4 goal positions for the first 30 iterations. At each iteration a new component is added based on the value shown in the title. These components are colored in black. Often the same candidate is selected for large ranges of Δ. All selected candidates seem reasonable. However, although all candidates reach one of the desired goal positions, the configurations can be less smooth (resulting in low likelihood due to the prior) for large values of Δ, which can be seen especially at iterations 20-23. While optimizing such components may require more iterations and samples, they are also more likely to discover a new mode. For example, the component added at iteration 20 is the first component that reaches the top goal position from the left side.

where $p_t(\mathbf{o}_t|a_1, a_2, \kappa_1, \dots, \kappa_8)$ is a Gaussian distribution with variance $\sigma^2 = 0.2$ and a mean which is computed by numerically integrating the ODE (Equation 31).

Appendix G. Planar Robot Experiment

The x and y coordinate of the end-effector are given by

$$x(\boldsymbol{\theta}) = \sum_{i=1}^{10} \cos\left(\sum_{j=1}^{i} \theta_j\right), \qquad y(\boldsymbol{\theta}) = \sum_{i=1}^{10} \sin\left(\sum_{j=1}^{i} \theta_j\right).$$

The target distribution is given as the product of two distributions,

$$p(\boldsymbol{\theta}) = \frac{1}{Z} p_{\text{conf}}(\boldsymbol{\theta}) p_{\text{cart}}(\boldsymbol{\theta}),$$

where $p_{\text{conf}}(\boldsymbol{\theta})$ enforces smooth configurations and $p_{\text{cart}}(\boldsymbol{\theta})$ penalizes deviations from the goal position. We model $p_{\text{conf}}(\boldsymbol{\theta})$ as zero mean Gaussian distribution with diagonal covariance matrix, where the angle of the first joint has a variance of 1 and the remaining joints have a variance of 4e−2. We consider two experiments that differ in the choice of goal positions. For the first experiment we specify a single goal position at position $(7, 0)$ modeled by a Gaussian distribution in Cartesian space with variance 1e−4 in both directions, namely

$$p_{\text{cart},1}(\boldsymbol{\theta}) = \mathcal{N}\left(\begin{bmatrix} x(\boldsymbol{\theta}) \\ y(\boldsymbol{\theta}) \end{bmatrix} \mid \begin{bmatrix} 7 \\ 0 \end{bmatrix}, \begin{bmatrix} 1\text{e}{-4} & 0 \\ 0 & 1\text{e}{-4} \end{bmatrix}\right).$$

For the second experiment we specify four goal positions at positions $(7, 0)$, $(0, 7)$, $(-7, 0)$ and $(0, -7)$. The likelihood $p_{\text{cart},2}$ is given by the maximum over the four respective Gaussian distributions.

Appendix H. Number of Components

The average number of components learned by VIPS++ is shown in Figure 13.

Appendix I. Implementations

For our comparisons we relied on open-source implementations, preferably by the original authors.

- For PTMCMC, we use an implementation by Ellis and van Haasteren (2017) that uses adaptive proposal distributions for the individual chains. We roughly tuned the number of chains for each experiment. As we could not run this implementation on our cluster, we ran the experiments on a fast quad-core laptop and made use of multi-threading. We therefore report four times the actual wall-clock time.

- For ESS, we use a Python implementation by Bovy (2013) that is based on the Matlab implementation by Iain Murray. If the target distribution decomposes into a product of a Gaussian prior and an arbitrary likelihood term, we directly provide this decomposition to the algorithm. If the target distribution does not use a Gaussian prior,

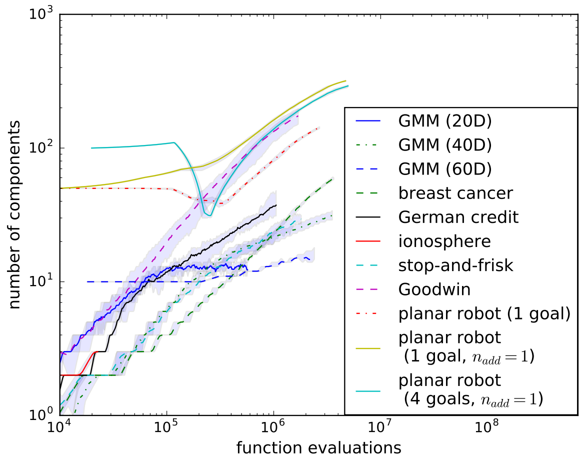


Figure 13: The average number of components learned by VIPS++ is plotted over function evaluations for all experiments in the test bed. When using the faster adding rate, $n_{\text{add}} = 1$, VIPS++ learns GMMs with approximately 350 components.

we choose an appropriate Gaussian distribution $p_{\text{prior}}(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mathbf{0}, \alpha\mathbf{I})$ as prior and provide it along with the resulting likelihood $\log p_{\text{likelihood}}(\mathbf{x}) = \log \tilde{p}(\mathbf{x}) - \log p_{\text{prior}}(\mathbf{x})$, as described by Nishihara et al. (2014).

- Our comparisons with HMC are based on PYHMC (Nabney et al., 2018). We tuned the step size and trajectory length for each experiment based on preliminary experiments. We also performed some experiments with NUTS (Hoffman and Gelman, 2014), however, HMC with tuned parameters always outperformed the automatically tuned parameters of NUTS.

- For slice sampling, we use a Python adaptation (Slavitt, 2013) of a Matlab implementation by Iain Murray and tuned the step size based on preliminary experiments.

- For SVGD, we use the implementation of the original authors (Liu and Wang, 2016) and tune the step size based on preliminary experiments.

- For Variational Boosting, we use the implementation of the original authors (Miller et al., 2017). However, this implementation is not optimized with respect to the number of function evaluations and often uses an unnecessary large number of samples. We therefore modified the implementation slightly. We also use their implementation of NPVI for our experiments.

- For black-box variational inference and inverse autoregressive flows we used our own implementation based on tensorflow (Abadi et al., 2015). The code for conducting these experiments is available online.[3] For black-box variational inference, we tuned the learning rate as well as the number of samples per iteration (batch size). For

3. The implementation can be found at `https://github.com/OlegArenz/tensorflow_VI`.

inverse autoregressive flows, we tuned the learning rate, the batch size, the number of flows and the (common) width of the two hidden layers of the autoregressive networks for each flow.

Appendix J. Considered Algorithms and Experiments

Table 1 provides an overview about which algorithms have been evaluated on which experiments.

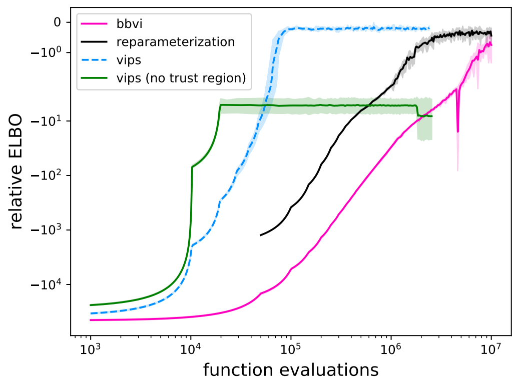
- Our implementations of IAF and BBVI use a different code base (based on Tensorflow (Abadi et al., 2015)) for which we only implemented a subset of the experiments. However, we ensured that the test bed includes simple, unimodal experiments (*German credit* and *breast cancer*) as well as the most challenging, multimodal experiments that we considered (*planar robot* and *GMM*).
- We did not evaluate PTMCMC on the simple test problems where parallel Markov chains would be wasteful.
- We did not evaluate HMC on the experiments with disconnected modes because we do not expect it to mix efficiently on such problems.
- We tried to evaluate VBOOST and NPVI on all test problems. However, we could not always obtain reliable results due to numerical problems that we could not fix without major changes to the implementation.
- We only evaluated VIPS++ on the higher-dimensional GMM experiments because it was the only method to solve the twenty-dimensional variant.

Appendix K. Alternatives for Learning Gaussian Variational Approximations

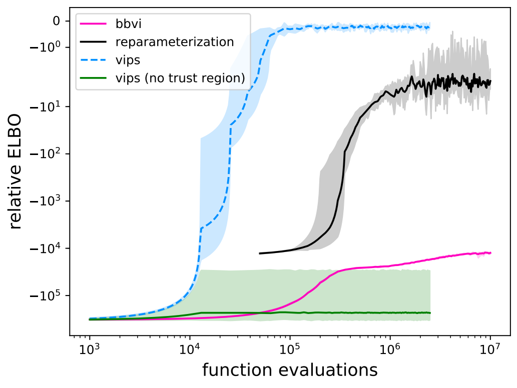
VIPS++ uses a variant of MORE (which we denote as VIPS1) for learning Gaussian variational approximations. However, it would also be possible to update the individual components using black-box variational inference (Ranganath et al., 2014) or the reparameterization trick, which assumes that the target distribution is differentiable. We compared against these alternatives on *breast cancer* experiment as well as on the *planar robot* experiment with a single goal position. The learning curves of the ELBO are shown in Figure 14. For each experiment, we subtracted a constant offset from the ELBO such that the highest (approximated) ELBO on each plot equals zero. Such relative ELBO ensures high resolution in the vicinity of the best ELBO on each of the plots. Please note that we use the symmetric logarithm to scale the y-Axis. Remarkably, VIPS1 is significantly more efficient than the reparameterization trick even though we do not require the gradient of the target distribution. We also compared against a variant of VIPS1 that does not constrain the KL divergence between updates. Such optimization is unstable as it exploits model errors caused by the local surrogate.

	VIPS	SVGD	ESS	HMC	NPVI	VBOOST	PTMCMC	SLICE	IAF	BBVI
German Credit	X	X	X	X	X	X	-	**X**	**X**	**X**
Breast Cancer	X	X	X	**X**	X	-	-	**X**	**X**	**X**
Frisk	X	X	**X**	X	X	X	-	**X**	-	-
GMM	X	X	X	-	-	-	X	**X**	**X**	**X**
Planar (1 goal)	X	X	X	-	X	X	X	X	**X**	**X**
Ionosphere	**X**	**X**	**X**	**X**	-	-	-	**X**	-	-
Goodwin	**X**	**X**	**X**	**X**	-	-	**X**	**X**	-	-
Planar (4 goal)	**X**	**X**	**X**	-	-	-	**X**	**X**	**X**	**X**
GMM (Higher Dim.)	**X**	-	-	-	-	-	-	-	-	-

Table 1: The table shows which algorithms were applied to each test problem. New experiment compared to our previous work (Arenz et al., 2018) are marked in bold.



(a) breast cancer



(b) planar robot (1 goal)

Figure 14: The proposed variant of MORE is significantly more efficient at optimizing Gaussian variational approximations compared to stochastic gradients using the reparameterization trick or black-box variational inference. By using locale surrogate objectives, we require trust regions to ensure stable optimization.

DESCRIPTION	VALUE
KL BOUND FOR COMPONENTS	$1\text{e}{-2} \leq \epsilon(o) \leq 5$
NUMBER OF DESIRED SAMPLES (PER DIMENSION AND COMPONENT)	20
NUMBER OF REUSED SAMPLES (PER DIMENSION AND COMPONENT)	40
ADDING RATE FOR COMPONENTS	30 OR 1
DELETION RATE FOR COMPONENTS	10
MINIMUM WEIGHT	1e−6
INITIAL WEIGHT	1e−29
$\boldsymbol{\Delta}$ FOR ADDING-HEURISTIC	[1000, 500, 200, 100, 50]
ℓ_2-REGULARIZATION FOR WLS	$1\text{e}{-14} \leq \kappa \leq 1\text{e}{-6}$

Table 2: The table shows the hyper-parameters of VIPS++ as well as their values used during the experiments. The bound on the KL-divergence and the coefficient for ℓ_2-regularization when fitting the surrogates are automatically adapted within in the provided ranges.

Appendix L. VIPS++ Hyper-Parameters

The hyper-parameters used for all experiments are given in Table 2.

Appendix M. Computing the Maximum Mean Discrepancy

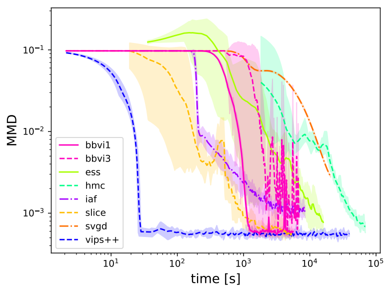
We approximate the MMD between two sample sets $\mathbf{X}$ and $\mathbf{Y}$ as

$$\begin{aligned}\text{MMD}(\mathbf{X},\mathbf{Y}) = &\frac{1}{m^2}\sum_{i,j}^{m} k(\mathbf{x}_i,\mathbf{x}_j) + \frac{1}{n^2}\sum_{i,j}^{n} k(\mathbf{y}_i,\mathbf{y}_j)\\ &- \frac{2}{mn}\sum_{i}^{m}\sum_{j}^{n} k(\mathbf{x}_i,\mathbf{y}_i).\end{aligned}$$

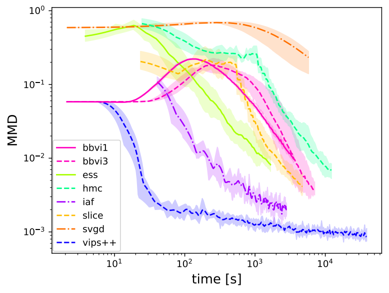
We use a squared exponential kernel given by

$$k(\mathbf{x},\mathbf{y}) = \exp\left(-\frac{1}{\alpha}(\mathbf{x}-\mathbf{y})^\top \boldsymbol{\Sigma}(\mathbf{x}-\mathbf{y})\right),$$

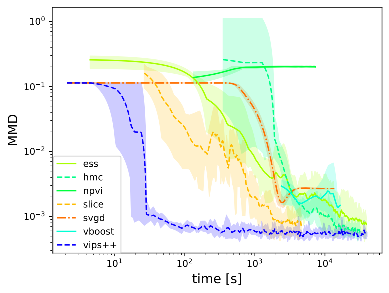
where $\boldsymbol{\Sigma}$ is a diagonal matrix where each entry is set to the median of squared distances within the ground-truth set and the bandwidth α is chosen depending on the problem. As true ground-truth samples are only available for the GMM experiment, we apply generalized elliptical slice sampling (Nishihara et al., 2014) with large values for burn-in, thinning and chain lengths to produce baseline samples that are regarded as ground-truth for the remaining experiments. Note that obtaining these ground-truth samples is computationally very expensive, taking up to two days of computation time on 128 CPU cores. We estimate the MMD based on ten thousand ground-truth samples and two thousand samples from the given sampling method. For MCMC methods, we choose the two thousand most promising samples by applying a sufficient amount of burn-in and using the largest thinning that keeps at least two thousand samples in the set.



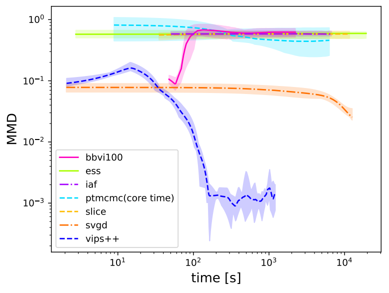
German credit (Log. Reg.)



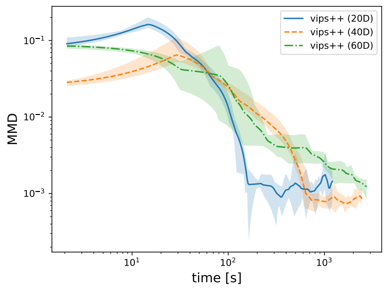
breast cancer (Log. Reg.)



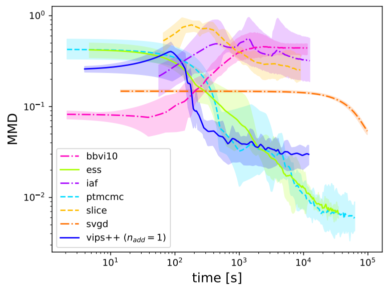
stop-and-frisk (Poisson GLM)



GMM (20 dimensions)



GMM (different dimensions)



planar robot (4 goals)

Figure 15: The maximum mean discrepancy with respect to baseline samples is plotted over computational time on log-log plots for the different sampling problems in the test bed.

Appendix N. Evaluations with Respect to Computational Time

Figure 15 shows the achieved MMDs with respect to time for the experiments that have been omitted in the main document.

Appendix O. Evaluations with respect to ELBO

We also compared the achieved ELBO $L(\theta)$ between VIPS++, inverse autoregressive flows (IAF) and black-box variational inference (BBVI). We approximate the ELBO based on 2000 samples from the learned approximation. The respective learning curves are shown in Figure 16 where we subtracted a constant offset as described in Appendix K.

Appendix P. Visualization of Samples for planar robot experiments

Samples obtained by BBVI, IAF, PTMCMC and VIPS++ for the planar robot experiment with one goal and four goals are shown in Figure 17 and Figure 18, respectively.

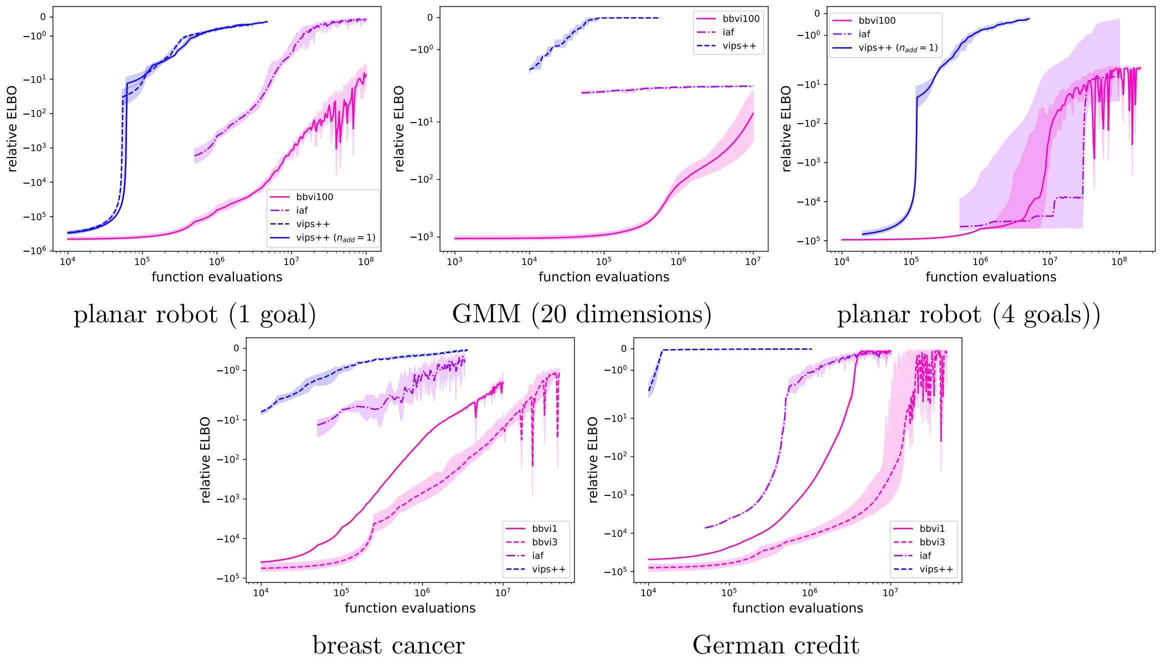


Figure 16: In contrast to the evaluation with respect to the MMD, all methods improve on the ELBO during learning, which is expected as the respective optimization problems aim to maximize the ELBO. Interestingly, IAF achieves a similar ELBO on the simpler planar robot experiment as VIPS++, although it performed significantly worse on the MMD. We verified that IAF achieves a similar approximated entropy as VIPS++, which is surprising since the learned approximation only sampled from one of the two main configurations (see Figure 17). We hypothesize that even the large GMMs learned by VIPS++ are not able to cover the modes as well as the normalizing flows.

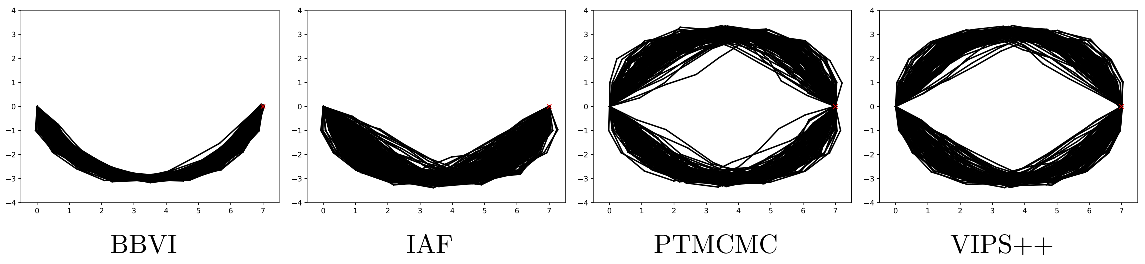


Figure 17: 200 sampled configurations are shown for the first training run for the planar robot experiment with a single goal. For the variational inference methods BBVI, IAF and VIPS++, the plots show samples of the final learned model. For PTMCMC, the plots show the 200 most promising samples, which are obtained by applying a sufficient amount of burn-in and using the largest thinning that keeps at least 200 samples in the set.

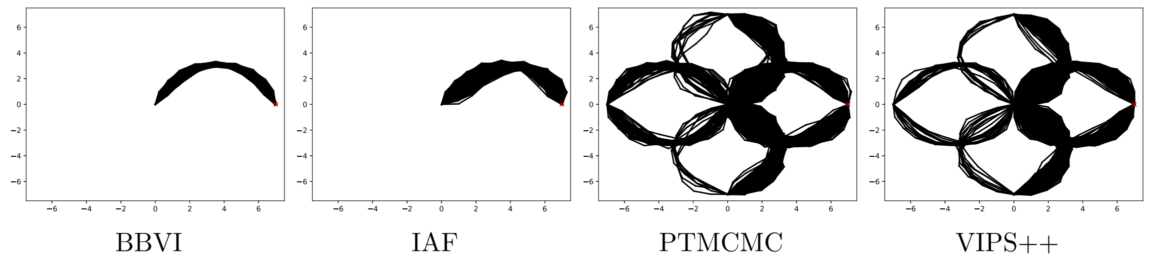


Figure 18: A thousand sampled configurations are shown for the first training run for the planar robot experiment with four goals. For the variational inference methods BBVI, IAF and VIPS++, the plots show samples of the final learned model. For PTMCMC, the plots show the thousand most promising samples, which are obtained by applying a sufficient amount of burn-in and using the largest thinning that keeps at least thousand samples in the set.

References

M. Abadi, M. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

A. Abdolmaleki, R. Lioutikov, N. Lua, L. Paulo Reis, J. Peters, and G. Neumann. Model-based relative entropy stochastic search. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 153–154, 2015.

A. Abdolmaleki, B. Price, N. Lau, L. P. Reis, and G. Neumann. Deriving and improving cma-es with information geometric trust regions. In *The Genetic and Evolutionary Computation Conference (GECCO 2017)*, July 2017.

F. V. Agakov and D. Barber. An auxiliary variational method. In *International Conference on Neural Information Processing*, pages 561–566. Springer, 2004.

R. Akrour, A. Abdolmaleki, H. Abdulsamad, J. Peters, and G. Neumann. Model-free trajectory-based policy optimization with monotonic improvement. *Journal of Machine Learning Research (JMLR)*, 19(14), 2018.

J. Altosaar, R. Ranganath, and D. M. Blei. Proximity variational inference. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.

O. Arenz, M. Zhong, and G. Neumann. Efficient gradient-free variational inference using policy search. In *International Conference on Machine Learning (ICML)*, 2018.

C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, 2006.

C. M. Bishop, N. D. Lawrence, T. Jaakkola, and M. I. Jordan. Approximating posterior distributions in belief networks using mixtures. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 416–422, 1998.

D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 2017.

J. Bovy. Python implementation of elliptical slice sampling, 2013. URL `https://github.com/jobovy/bovy_mcmc`.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

R.H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

B. Calderhead. A general construction for parallelizing Metropolis-Hastings algorithms. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, Nov 2014.

B. Calderhead and M. Girolami. Estimating Bayes factors via thermodynamic integration and population mcmc. *Computational Statistics & Data Analysis*, 53(12):4028–4045, 2009.

X. Chen, M. Monfort, A. Liu, and B. D. Ziebart. Robust covariate shift regression. In *International Conference on Artificial Intelligence and Statistics*, pages 1270–1279, 2016.

R. Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer, 2006.

C. Daniel, G. Neumann, and J. Peters. Hierarchical relative entropy policy search. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012.

C. Daniel, G. Neumann, O. Kroemer, and J. Peters. Hierarchical relative entropy policy search. *Journal of Machine Learning Research (JMLR)*, 17:1–50, June 2016. URL `http://eprints.lincoln.ac.uk/25743/`.

M. P. Deisenroth, G. Neumann, and J. Peters. A survey on policy search for robotics. *Foundations and Trends in Robotics*, pages 388–403, 2013.

L. Dinh, D. Krueger, and Y. Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

P. M. Djuric, J. H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. F. Bugallo, and J. Miguez. Particle filtering. *IEEE signal processing magazine*, 20(5):19–38, 2003.

A. Doucet, N. De Freitas, and N. Gordon. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pages 3–14. Springer, 2001.

S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, 1987.

D. J. Earl and M. W. Deem. Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics*, 7(23):3910–3916, 2005.

J. Ellis and R. van Haasteren. jellis18/ptmcmcsampler: Official release, 2017. URL `https://doi.org/10.5281/zenodo.1037579`.

F. End, R. Akrour, J. Peters, and G. Neumann. Layered direct policy search for learning hierarchical skills. In *International Conference on Robotics and Automation (ICRA)*, 2017.

K. Fan, Z. Wang, J. Beck, J. T. Kwok, and K. Heller. Fast second-order stochastic backpropagation for variational inference. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1387–1395, 2015.

M. Germain, K. Gregor, I. Murray, and H. Larochelle. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning (ICML)*, pages 881–889, 2015.

S. J. Gershman, M. D. Hoffman, and D. M. Blei. Nonparametric variational inference. In *International Conference on Machine Learning (ICML)*, 235–242, 2012.

B.C. Goodwin. Oscillatory behavior in enzymatic control processes. *Advances in Enzyme Regulation*, 3:425–437, 1965.

W. Grathwohl, R. T. Q. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud. FFJORD: free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations (ICLR)*, 2019.

A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research (JMLR)*, 13:723–773, March 2012. ISSN 1532-4435.

F. Guo, X. Wang, K. Fan, T. Broderick, and D. B. Dunson. Boosting variational inference. *arXiv:1611.05559v2 [stat.ML]*, 2016.

T. C. Hesterberg. *Advances in Importance Sampling*. PhD thesis, Stanford University, 1988.

M. D. Hoffman and A. Gelman. The no-u-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research (JMLR)*, 15(1): 1593–1623, 2014.

M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *Journal of Machine Learning Research (JMLR)*, 14(4):1303–1347, 2013.

C. Huang, D. Krueger, A. Lacoste, and A. Courville. Neural autoregressive flows. In *International Conference on Machine Learning (ICML)*, pages 2078–2087, 2018.

T. S. Jaakkola and M. I. Jordan. Improving the mean field approximation via the use of mixture distributions. *Learning in Graphical Models*, 89:163–174, 1998.

M. E. Khan, R. Babanezhad, W. Lin, M. Schmidt, and M. Sugiyama. Faster stochastic variational inference using proximal-gradient methods with general divergence functions. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 319–328, 2016.

M. E. E. Khan, P. Baqué, F. Fleuret, and P. Fua. Kullback-leibler proximal variational inference. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3402–3410, 2015.

D. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 10215–10224, 2018.

D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4743–4751, 2016.

A. Kong, J. S. Liu, and W. H. Wong. Sequential imputations and bayesian missing data problems. *Journal of the American Statistical Association*, 89(425):278–288, 1994.

S. Levine and V. Koltun. Guided policy search. In *International Conference on Machine Learning (ICML)*, pages 1–9, 2013.

M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

L. Lin. Reinforcement learning for robots using neural networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 1993.

Q. Liu and D. Wang. Stein variational gradient descent: A general purpose Bayesian inference algorithm. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2378–2386. Curran Associates, Inc., 2016.

L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther. Auxiliary deep generative models. In *International Conference on Machine Learning (ICML)*, pages 1445–1454, 2016.

H. Mania, A. Guy, and B. Recht. Simple random search of static linear policies is competitive for reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1803–1812. Curran Associates, Inc., 2018.

A. C. Miller, N. J. Foti, A. D'Amour, and R. P. Adams. Variational boosting: Iteratively refining posterior approximations. In *International Conference on Machine Learning (ICML)*, 2017.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

I. Murray, R. Adams, and D. MacKay. Elliptical slice sampling. In *International Conference on Artificial Intelligence and Statistics*, pages 541–548, 2010.

I. T. Nabney, A. Vehtari, Koepsell K., and McGibbon R. T. pyhmc: Hamiltonian Monte Carlo in python, 2018. URL `https://github.com/rmcgibbo/pyhmc`.

R. Neal and G. E. Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers, 1998.

R. M. Neal. Sampling from multimodal distributions using tempered transitions. *Statistics and Computing*, 6(4):353–366, Dec 1996.

R. M. Neal. Slice sampling. *The Annals of Statistics*, 31(3):705–767, 06 2003. doi: 10.1214/aos/1056562461.

G. Neu, A. Jonsson, and V. Gómez. A unified view of entropy-regularized Markov decision processes. *arXiv preprint arXiv: 1705.07798*, 2017. URL `http://arxiv.org/abs/1705.07798`.

R. Nishihara, I. Murray, and R. P. Adams. Parallel mcmc with generalized elliptical slice sampling. *Journal of Machine Learning Research (JMLR)*, 15(1):2087–2112, January 2014.

G. Papamakarios, T. Pavlakou, and I. Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2338–2347, 2017.

J. Peters, K. Muelling, and Y. Altun. Relative entropy policy search. In *AAAI Conference on Artificial Intelligence*, 2010.

C. Peterson and E. Hartman. Explorations of the mean field theory learning algorithm. *Neural Networks*, 2(6):457–494, 1989.

T. Rainforth, Y. Zhou, X. Lu, Y. W. Teh, F. Wood, H. Yang, and J. van de Meent. Inference trees: Adaptive inference with exploration. *arXiv preprint arXiv:1806.09550*, 2018.

R. Ranganath, S. Gerrish, and D. Blei. Black box variational inference. *Artificial Intelligence and Statistics*, pages 814–822, 2014.

R. Ranganath, D. Tran, and D. Blei. Hierarchical variational models. In *International Conference on Machine Learning (ICML)*, pages 324–333, 2016.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning.* The MIT Press, 2006.

J. Regier, M. I. Jordan, and J. McAuliffe. Fast black-box variational inference through stochastic trust-region optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2399–2408, 2017.

D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning (ICML)*, pages 1530–1538, 2015.

D. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning (ICML)*, pages 1278–1286, 2014.

G. O. Roberts and O. Stramer. Langevin diffusions and Metropolis-Hastings algorithms. *Methodology and Computing in Applied Probability*, 4(4):337–357, 2002.

T. Salimans and D. A. Knowles. Fixed-form variational posterior approximation through stochastic linear regression. *Bayesian Analysis*, 8(4):837–882, 2013.

T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

L. K. Saul, T. Jaakkola, and M. I. Jordan. Mean field theory for sigmoid belief networks. *Journal of Artificial Intelligence Research*, 4:61–76, 1996.

T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. *International Conference on Learning Representations (ICLR)*, 2016.

J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, 2015.

S. Shirakawa, Y. Akimoto, K. Ouchi, and K. Ohara. Sample reuse in the covariance matrix adaptation evolution strategy based on importance sampling. In *Annual Conference on Genetic and Evolutionary Computation*, pages 305–312, 2015.

I. Slavitt. Python implementation of slice sampling, 2013. URL `https://isaacslavitt.com/2013/12/30/metropolis-hastings-and-slice-sampling/`.

R. S. Sutton and A. G. Barto. *Reinforcement Learning*. MIT Press, Boston, MA, 1998.

L. Theis and M. Hoffman. A trust-region method for stochastic variational inference with applications to streaming data. In *International Conference on Machine Learning (ICML)*, pages 2503–2511. PMLR, 2015.

D. Tran, R. Ranganath, and D. M. Blei. The variational gaussian process. In *International Conference on Learning Representations (ICLR)*, 2016.

E. Uchibe. Efficient sample reuse in policy search by multiple importance sampling. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 545–552, 2018.

T. Weber, N. Heess, A. Eslami, J. Schulman, D. Wingate, and D. Silver. Reinforced variational inference. In *Advances in Neural Information Processing Systems (NeurIPS) Workshops*, 2015.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

O. Zobay. Variational Bayesian inference with Gaussian-mixture approximations. *Electronic Journal of Statistics*, 8(1):355–389, 2014. doi: 10.1214/14-EJS887.