Trust-Region Variational Inference with Gaussian Mixture Models

Oleg Arenz\textsuperscript{1}, Mingjun Zhong\textsuperscript{2}, and Gerhard Neumann\textsuperscript{3,4}

\textsuperscript{1}Intelligent Autonomous Systems, TU Darmstadt
\textsuperscript{2}Machine Learning Group, University of Lincoln
\textsuperscript{3}Bosch Center for Artificial Intelligence, Renningen
\textsuperscript{4}Center for Autonomous Systems, University of Lincoln

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 establishing 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. The use of the lower bound ensures convergence to a local optimum 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, multi-modal 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.

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

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

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 \( p(x) \). 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 \( \tilde{p}(x) \) 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 \( p(x) \) is approximated by a tractable distribution \( q(x; \theta) \) which can be used for exact inference instead of \( p(x) \), or as a more direct alternative to MCMC for drawing samples for (possibly importance weighted) Monte Carlo estimates. The approximation \( q(x; \theta) \) is typically found by minimizing the reverse Kullback-Leibler (KL) divergence

\[
\text{KL} (q(x; \theta)||p(x)) = \int_x q(x; \theta) \log \left( \frac{q(x; \theta)}{p(x)} \right) dx,
\]

with respect to the parameters \( \theta \) of the approximation.

By framing inference as an optimization problem, variational inference can make better use of previous function evaluations of \( \tilde{p}(x) \) than MCMC and is therefore computationally more efficient. However, in order to perform the KL minimization efficiently, \( q(x; \theta) \) 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 multi-modal 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 multi-modal 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 \( \tilde{p}(x) \). The unnormalized target distribution, however, is typically evaluated at locations that have been sampled from the current approximation \( q(x; \theta) \), because these samples are well suited for the optimization, for example for approximating the objective (Equation
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 $\hat{p}(x)$ 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 multi-modal 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 introducing a line search for initializing the covariance matrices of new components 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).

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 \( \theta \) of a model \( q(x; \theta) \) that minimize the KL divergence between \( q(x; \theta) \) and the target distribution \( p(x) \),

\[
\text{KL} (q(x; \theta)||p(x)) = \int_x q(x; \theta) \log \left( \frac{q(x; \theta)}{p(x)} \right) dx = \int_x q(x; \theta) \log \left( \frac{q(x; \theta)}{\hat{p}(x)} \right) dx + \log Z = -L(\theta) + \log Z.
\]

The normalizer \( Z \) does not affect the optimal solution for the parameters \( \theta \) as it enters the objective function as constant offset and can thus be ignored. Hence, the KL divergence can be minimized by maximizing \( L(\theta) \), which is a lower bound on the log normalizer due to the non-negativity of the KL divergence. In Bayesian inference, the target distribution \( p(x) \) corresponds to the posterior, the unnormalized distribution \( \hat{p}(x) \) 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, \( L(\theta) \), 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, \( q(x; \theta) \), over a parameter space \( x \), that maximizes an expected return \( R(x) = \log \hat{p}(x) \) with an additional objective of maximizing its entropy \( H(q(x; \theta)) = -\int_x q(x; \theta) \log q(x; \theta) dx \),

\[
\theta^* = \arg \max_{\theta} L(\theta) = \arg \max_{\theta} \int_x q(x; \theta) \left( \log \hat{p}(x) - \log q(x; \theta) \right) dx = \arg \max_{\theta} \int_x q(x; \theta) R(x) dx + H(q(x; \theta)).
\]

(2)

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 multi-modal 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

In reinforcement learning, areas of high reward are initially not known and have to be discovered based on evaluations of the reward function \( R(x) \) 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 \( q(x; \theta^{(0)}) \) with sufficiently high entropy and to ensure that the update does not erroneously discard high reward regions 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 \cite{Peters2010, Levine2013, Schulman2015, Abdolmaleki2015, Abdolmaleki2017}. 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 \cite{Abdolmaleki2015} additionally limits the entropy loss of the distribution update by solving the following optimization problem at iteration \( i \):

\[
\theta^{(i+1)} = \arg \max_{\theta} \int_x q(x; \theta) R(x) dx, \\
\text{subject to } \text{KL}(q(x; \theta)||q(x; \theta^{(i)})) \leq \epsilon, \\
H(q(x; \theta)) \geq \beta,
\]

where the lower bound \( \beta = H(q(x; \theta^{(i)})) - \gamma \) is computed based on the entropy of the current search distribution. This optimization problem can be solved using Lagrangian multipliers. The updated search distribution \( q(x; \theta^{(i+1)}) \) is given by

\[
q(x; \theta^{(i+1)}) \propto q(x; \theta^{(i)}) \frac{\eta}{\eta + \omega} \exp \left( R(x) \right)^{\frac{1}{\eta + \omega}}, \tag{4}
\]

where \( \eta \) and \( \omega \) are the Lagrangian multipliers of the KL divergence and entropy constraint, respectively.

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

\[
\bar{R}(x) = -\frac{1}{2} x^\top R^{(i)} x + x^\top r^{(i)} + \text{const}, \tag{5}
\]

that is fit using linear regression based on samples from the current approximation. For this choice of variational distribution and reward surrogate, the updated distribution according to Equation \( \ref{eq:var-dist} \) is also Gaussian with natural parameters

\[
Q(\eta, \omega) = \frac{\eta}{\eta + \omega} Q^{(i)} + \frac{1}{\eta + \omega} R^{(i)}, \tag{6}
\]

\[
q(\eta, \omega) = \frac{\eta}{\eta + \omega} q^{(i)} + \frac{1}{\eta + \omega} r^{(i)}, \tag{7}
\]

which directly relate to mean \( \mu = Q^{-1} q \) and covariance matrix \( \Sigma = Q^{-1} \). It can be seen from Equation \( \ref{eq:var-dist} \) and \( \ref{eq:q-dist} \) that \( \eta \) controls the step size, whereas \( \omega \) affects the entropy by scaling the covariance matrix. The optimal parameters \( \eta^* \) and \( \omega^* \) can be learned by minimizing the convex dual objective

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

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

\[
\frac{\partial G(\eta, \omega)}{\partial \eta} = \epsilon - \text{KL}(q_{\eta, \omega}(x) || q(x; \theta^{(i)})), \\
\frac{\partial G(\eta, \omega)}{\partial \omega} = H(q_{\eta, \omega}(x)) - \beta,
\]

where \(q_{\eta, \omega}(x)\) refers to the Gaussian distribution with natural parameters computed according to Equation 6 and Equation 7.

## 2.3 Adapting MORE to Variational Inference

Inspired by policy search methods, we want to establish information-geometric trust regions for variational inference in order to achieve more stable optimization and exploration. Hence, we want to compute each update of the approximation by solving the constrained optimization problem

\[
\theta^{(i+1)} = \arg \max_{\theta} \int_x q(x; \theta) R(x) dx + H(q(x; \theta)),
\]

subject to \(\text{KL}(q(x; \theta) || q(x; \theta^{(i)}) \leq \epsilon).\) (8)

This optimization problem is very similar to optimization problem (3) 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. The dual function is thus given by

\[
G(\eta) = \eta \epsilon + \eta \log Z(Q^{(i)}, q^{(i)}) - (\eta + 1) \log Z(Q(\eta, 1), q(\eta, 1)),
\]

and the gradient by

\[
\frac{dG(\eta)}{d\eta} = \epsilon - \text{KL}(q_{\eta, 1}(x) || q(x; \theta^{(i)})).
\] (10)

Hence, a Gaussian variational approximation can be learned analogously to MORE by iteratively (1) fitting a local, quadratic surrogate \(\tilde{R}(x) \approx \log \tilde{p}(x),\) (2) finding the optimal step size \(\eta\) by convex optimization and (3) updating the approximation based on Equation 6 and Equation 7. 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 multi-modal target distributions. We will now derive VIPS++, a general-purpose method for learning GMM approximations of an unnormalized target distribution \(\tilde{p}(x)\). 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. Section 3.2 discusses several extensions to this procedure that are critical for efficiently learning high quality approximations in practice. Namely, we will discuss dynamically adapting the number of components and reusing function evaluations of previous iterations.
Algorithm 1 Updating a Gaussian variational approximation based on surrogate

Require: coefficients of quadratic surrogate $R, r$ (equation 5)

Require: current mean and covariance matrix $\mu, \Sigma$

Require: KL bound $\epsilon$

1: function $GVA\_update(\mu, \Sigma, R, r, \epsilon)$
2: $Q \leftarrow \Sigma^{-1}$
3: $q \leftarrow \Sigma^{-1}\mu$
4: $\eta \leftarrow$ minimize dual (Equation 9) with gradient (Equation 10)
5: $Q' \leftarrow \frac{\eta}{\eta + 1}Q + \frac{1}{\eta + 1}R$
6: $q' \leftarrow \frac{\eta}{\eta + 1}q + \frac{1}{\eta + 1}r$
7: $\Sigma' \leftarrow Q'^{-1}$
8: $\mu' \leftarrow Q'^{-1}q'$
9: return $\Sigma', \mu'$
10: end function

3.1 Learning a GMM Approximation

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

$$q(x; \theta) = \sum_o q(o; \theta)q(x|o; \theta),$$

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

The approximation is learned by maximizing the ELBO

$$L(\theta) = \sum_o q(o) \int_x q(x|o)(R(x) - \log q(x)) dx$$

$$= \sum_o q(o) \int_x q(x|o)(R(x) - \log q(o) - \log q(x|o) + \log q(o|x)) dx$$

$$= \sum_o q(o) \left[ \int_x q(x|o)(R(x) + \log q(o|x)) dx + H(q(o|x)) \right] + H(q(o)), \quad (11)$$

where we used the identity

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

which can be derived from Bayes’ rule.

3.1.1 Variational Lower Bound

Unfortunately, the occurrence of the log responsibilities, $\log q(o|x)$, in equation 11 prevents us from optimizing each component independently. However, we can derive a lower bound
\( \tilde{L}(\theta, \tilde{q}(o|x)) \) on the objective by adding and subtracting an auxiliary distribution \( \tilde{q}(o|x) \),

\[
\begin{align*}
L(\theta) &= \sum_o q(o) \left[ \int_x q(x|o)(R(x) + \log q(o|x))dx + H(q(x|o)) \right] + H(q(o)) \\
&= \sum_o q(o) \left[ \int_x q(x|o)(R(x) + \log \tilde{q}(o|x) + \log q(o|x) - \log \tilde{q}(o|x))dx + H(q(x|o)) \right] \\
&\quad + H(q(o)) \\
&= \sum_o q(o) \left[ \int_x q(x|o)(R(x) + \log \tilde{q}(o|x))dx + H(q(x|o)) \right] + H(q(o)) \\
&\quad + \int_x q(x)\text{KL}(q(o|x)||\tilde{q}(o|x)) dx.
\end{align*}
\]

Please note, that the last term in Equation 12 corresponds to an expected KL divergence and is therefore non-negative which implies that

\[
\tilde{L}(\theta, \tilde{q}(o|x)) \leq L(\theta).
\]

The decomposition in Equation 12 is similar to the one used in the expectation-maximization algorithm \cite{Bishop2006} for learning GMMs for density estimation. However, whereas EM for density estimation minimizes the forward KL divergence, \( \text{KL}(p(x)||q(x; \theta)) \), our decomposition can be used for minimizing the reverse KL divergence, \( \text{KL}(q(x; \theta)||p(x)) \).

The forward KL divergence can be easier optimized when samples from the target distribution are available while the (unnormalized) target density function \( \tilde{p}(x) \) 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 (locally) maximize the ELBO \( L(\theta) \) by iteratively setting \( \tilde{q}(o|x) = q(o|x) \) (analogously to an E-step) and increasing the lower bound \( \tilde{L}(\theta, \tilde{q}(o|x)) \) (M-step) while keeping the auxiliary distribution fixed. Tightening the lower bound by setting \( \tilde{q}(o|x) = q(o|x) \) does not affect the ELBO since the parameters \( \theta \) are not changed. Unless the current approximation is already locally optimal, maximizing the lower bound increases both the lower bound and the expected KL divergence and thus also increases the ELBO.

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 \( \tilde{L}(\theta, \tilde{q}(o|x)) \), 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 \cite{Akrour2018, Schulman2015}. 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 \( \tilde{L}(\theta, \tilde{q}(o|x)) \) with respect to the mean and covariance matrix \( \theta_o = [\mu_o, \Sigma_o] \) 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 12, that is,

\[
\arg \max_{\theta_o} \int_x q(x|o; \theta_o) \left( R(x) + \log \tilde{q}(o|x) \right) dx + H(q(x|o)),
\]

subject to

\[
\text{KL}(q(x|o; \theta_o)||q(x|o; \theta^{(i)}) \leq \epsilon(o),
\]

(13)

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—based on different samples than those used for the maximization—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 8) with a component specific reward function \( R_o(x) = R(x) + \log \tilde{q}(o|x) \). As the auxiliary distribution \( \tilde{q}(o|x) \) was fixed to the responsibilities \( q(o|x; \theta^{(i)}) \) according to the previous mixture model, the component specific part of \( R_o(x) \) 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 \( \tilde{R}_o(x) \approx R_o(x) \) that approximates the component specific reward \( R_o(x) \) in the vicinity of the respective component \( q(x|o) \). 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 \( R_o(x) \). 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 3.2.1.

After fitting the surrogate, the optimization problem in Equation 13 can be solved efficiently using L-BFGS-B \( \text{[Byrd et al., 1995]} \) to minimize the dual problem (Equation 9) and using the learned step size \( \eta \) to compute the update in closed form as outlined in Section 2.3.

Drawing the connection to reinforcement learning and investigating the reward function \( R_o(x) \) 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 \( \log \tilde{p}(x) \) 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 \( \text{[Arenz et al., 2018]} \), 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

\[
\arg \max_{q(o)} \sum_o q(o) R(o) + H(q(o)) \tag{14}
\]

where the objective for the component update

\[
R(o) = \int_x q(x|o) \big( R(x) + \log q(o|x) \big) dx + H(q(x|o))
\]

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 \( R_o(x) \) used for the component update is a continuous function. It is not beneficial to approximate \( R(o) \) based on a quadratic surrogate of \( R_o(x) \), since we can estimate each element of the vector more efficiently and more accurately using a Monte-Carlo estimate

\[
\hat{R}(o) = \frac{1}{N_o} \sum_{n=1}^{N_o} \left[ R(x_{o,n}) + \log \hat{q}(o|x_{o,n}) \right] + H(q(x|o)), \tag{15}
\]

where \( x_{o,n} \) refers to the \( n \)th of \( N_o \) samples from component \( q(x|o) \). We will discuss in Section 3.2.1 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 \( \hat{R}(o) \), the optimal solution of optimization problem in Equation 14 is given in closed form as

\[
q(o) = \frac{\exp \left( \hat{R}(o) \right)}{\sum_o \exp \left( \hat{R}(o) \right)} \tag{16}.
\]

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 as well as Gaussian lower level policies \( q(x|o) \). 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 \( \text{(Mania et al., 2018; Salimans et al., 2017; Abdolmaleki et al., 2015).} \) HiREPS \( \text{(Daniel et al., 2012)} \) already applied black-box optimization for learning GMM policies based on episodic REPS \( \text{(Peters et al., 2010).} \)

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 according to the previous approximation \( \log \hat{q}(o|x) \) (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 \( \theta \). 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 \cite{Neal1998}.

**Algorithm 2** Variational Inference by Policy Search (Basic Variant)

**Require:** number of components $N_o$

**Require:** initial mixture parameters $\theta = \{q(o), \mu_o, \ldots, N_o; \Sigma_o, \ldots, N_o\}$

**Require:** number of EM-iterations $N_{EM}$

**Require:** number of samples per component $N_s$

1: for $i = 1 \ldots N_{EM}$ do
2:   for $o = 1 \ldots N_o$ do
3:     $X_o \leftarrow \text{SAMPLE\_GAUSSIAN}(\mu_o, \Sigma_o, N_s)$
4:     $\tilde{p}_o \leftarrow \log \tilde{p}(X_o)$ ⇨ EVALUATE TARGET LOG LIKELIHOOD FOR EACH SAMPLE
5:     $q_{o|x} \leftarrow \log q(X_o, o; \theta) - \log q(X_o; \theta)$ ⇨ EVALUATE LOG RESPONSIBILITIES
6:     $y_o \leftarrow \tilde{p}_o + q_o$ ⇨ COMPUTE TARGETS FOR ORDINARY LEAST SQUARES
7:     $R_o, r_o \leftarrow \text{OLS}(X_o, y_o)$ ⇨ LEARN QUADRATIC SURROGATE
8:     $\mu'_o, \Sigma'_o \leftarrow \text{GVA\_UPDATE}(\mu_o, \Sigma_o, R_o, r_o, \epsilon_o)$
9:   end for
10: $\theta \leftarrow \text{UPDATE\_COMPONENTS}(\theta, \mu'_o, \ldots, N_o; \Sigma'_o, \ldots, N_o)$
11: for $o = 1 \ldots N_o$ do
12:     $X_o \leftarrow \text{SAMPLE\_GAUSSIAN}(\mu_o, \Sigma_o, N_s)$
13:     $\tilde{p}_o \leftarrow \log \tilde{p}(X_o)$ ⇨ EVALUATE TARGET LOG LIKELIHOOD FOR EACH SAMPLE
14:     $q_{o|x} \leftarrow \log q(X_o, o; \theta) - \log q(X_o; \theta)$ ⇨ EVALUATE LOG RESPONSIBILITIES
15:     $\tilde{R}_o \leftarrow \text{MC\_ESTIMATE}(q_o, \tilde{p}_o, H(\Sigma_o))$ ⇨ ESTIMATE REWARD (EQUATION 15)
16: end for
17: $\theta' \leftarrow \text{UPDATE\_WEIGHTS}(\theta, q'(o))$
18: end for

**3.2 Algorithmic modifications**

In this section, we introduce two extensions to Algorithm 2 that increase its effectiveness in practice. We will discuss how we use importance weighting to perform the weight update and the component updates on the same set of samples for better sample efficiency, and how we adapt the number of components during training in order to make the quality of the learned approximation less reliant on a good initialization.

**3.2.1 Sample Reuse**

VIPS relies on samples for computing the quadratic surrogates for the component update and for approximating the reward for choosing a given component, $R(o)$. These samples need to be evaluated on the unnormalized target distribution $\tilde{p}(x)$ which can often be very costly. In order to reduce the number of function evaluations we use importance weighting, enabling us to use the same set of samples for the component and weight updates. 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
\[ S = \{(x_0, \log \tilde{p}(x_0), N_{x_0}), \ldots, (x_N, \log \tilde{p}(x_N), N_{x_N})\}, \]
where \( N_x \) refers to the Gaussian distribution that was used for obtaining the sample \( x \). By also storing its respective Gaussian distributions, we can compute for any subset \( \mathcal{X}_C \) of samples in the database, and any current mixture component \( q(x|o) \) the self-normalized importance weights
\[ w_o(x_s) = \frac{1}{Z} \frac{q(x_s|o)}{z_C(x_s)}, \quad Z = \sum_{x_s \in \mathcal{X}_C} \frac{q(x_s|o)}{z_C(x_s)}, \tag{17} \]
where \( z_C(x) \) is a Gaussian mixture model that contains for each sample \( x_s \in \mathcal{X}_C \) the respective Gaussian distribution \( N_{x_s}(x) \), that is,
\[ z_C(x) = \sum_{x_s \in \mathcal{X}_C} \frac{1}{|\mathcal{X}_C|} N_{x_s}(x). \tag{18} \]

We will now show how the importance weights can be used to learn the quadratic surrogates for the component updates and how to approximate the rewards for the weight updates based on the same subset \( \mathcal{X}_C \) of samples. We will further discuss how to select a concise and relevant subset at each learning iteration and how to determine how many new samples should be drawn from each component by taking into account its effective number of samples within the chosen subset.

Instead of fitting the quadratic surrogates using ordinary least squares based on samples from the respective component, we can reuse samples from previous iterations by using weighted least squares where each sample is weighted according to its self-normalized importance weight. Hence, we can compute the parameters of the quadratic surrogate for a given component \( o \) as
\[ \beta_o = (X^\top W_o X + \kappa_o I)^{-1} X^\top W_o y \]
where \( X \) is the design matrix where each row contains the linear and quadratic features for the respective sample \( x_s \in \mathcal{X}_C \) as well as a constant feature, \( W_o \) is a diagonal matrix where each element relates to the respective self-normalized importance weight \( w_o(x_s) \), \( \kappa_o \) is a small coefficient for \( \ell_2 \)-regularization, \( y \) is a vector containing the targets \( y_i = R(x_s) + \log q(o|x_s) \) and \( \beta_o \) is a vector containing the elements of \( r_o \) and \( R_o \) as well as a constant offset that can be discarded. Specifying an appropriate ridge coefficient \( \kappa_o \) 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, and clip it to the interval \([1e^{-14}, 1e^{-6}]\).

For updating the mixture weights, we approximate the reward \( R(o) \) assigned to each component based on the same set of samples that is used for the component updates using importance weighting, that is,
\[ \tilde{R}(o) = \sum_{x_s \in \mathcal{X}_C} w_o(x_s) \left[ R(x_s) + \log \tilde{q}(o|x_s) \right] + H(q(x|o)). \]

\(^1\)The linear features can be zero-centered by subtracting the component’s mean \( \mu_o \).
Using all previous samples in each iteration would be computationally costly and might further lead to overfitting. Instead, we want to select a small set of samples such that it contains for each component a sufficient amount of samples with high importance weight $q(x|o)/z_C(x)$ which is crucial for obtaining good approximations. In our previous work [Arenz et al., 2018], we always chose all samples from the three most recent iterations and added a fixed number of new samples from each component to the set. As the components that were used for the most recent iterations were similar to the current components due to the trust region optimization, the reused samples could usually provide meaningful information about the target distribution in the vicinity of the respective components.

However, we should also make use of samples from much older iterations, if they are useful for the current iteration, which can for example happen, when some of the components have already converged. By considering samples from all iterations, VIPS can be more sample efficient, because modes that have been sufficiently explored do not need to be sampled anymore while the other components can still improve and explore other regions of the search space. It is to note, that we can still update those components of the mixture model that have already converged and can thereby react to changes in the log responsibilities due to the changes of the remaining components.

However, we can not simply cherry-pick those samples with high density $q(x|o)$ for a given component, because constructing the background distribution $z_C(x)$ according to Equation 18 would not be sound if one sample from a given component $N_x$ would be preferred over another sample from that same component. For example, consider a Gaussian distribution of an early iteration that covers a current component but has significantly more entropy. By only adding those samples of the higher entropy distribution that have high density at the current component we would effectively sample from a distribution with lower entropy. The background distribution would assign too little probability mass to these samples and the resulting importance weights would thus be too large.

Instead of selecting relevant individual samples, we identify relevant distributions within the database and add all samples from these distributions, ensuring that the background distribution $z_C(x)$ is consistent with the chosen sample set $X_C$. For each component $o$ of the current mixture model, we select relevant distributions within the database by iteratively sampling (without replacement) a distribution with probability

$$h(i, o) \propto p(\mu_i|o) \exp(-n_i), \quad (19)$$

where $p(\mu_i|o)$ corresponds to the probability density of the current mixture component $o$ evaluated at the mean of component $i$ in the database and $n_i$ keeps track of the number of times the samples of this distribution have been reused. We add all samples from the chosen component to the active set of samples $X_C$ and stop sampling distributions when a desired number of reused samples $n_{\text{reused}}$ is reached. The second term of the probability distribution $h(i, o)$ provides an incentive to vary the chosen distributions, because using the same set of samples on a given mode too often could lead to overfitting. The routine for identifying relevant samples is shown in Algorithm 3.

### 3.2.2 Drawing new samples

After selecting the set $X_C$ of samples to be reused during the current iteration, we need to draw new samples from those components that are not sufficiently covered. We therefore
Algorithm 3 Identifying relevant samples in the database

Require: database $\mathcal{S} = \{(x_0, \log \hat{p}(x_0), N_{x_0}), \ldots, (x_N, \log \hat{p}(x_N), N_{x_N})\}$

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: $\mathcal{X}_C \leftarrow \{\}$
3: for $o = 1 \ldots N_o$ do
4: $n_{\text{added}} \leftarrow 0$
5: $h(\cdot, o) \leftarrow$ compute for each component in the database according to (19)
6: while $n_{\text{added}} < n_{\text{reuse}}$ do
7: $i \sim h(\cdot, o)$ $\triangleright$ choose a distribution by sampling $h(i, o)$
8: $h(\cdot, o) \leftarrow$ remove element $i$ and normalize
9: for each sample $x_j$ of component $N_i$ do
10: $\mathcal{X}_C \leftarrow \mathcal{X}_C \cup x_j$
11: $n_{\text{added}} \leftarrow n_{\text{added}} + 1$
12: if $n_{\text{added}} == n_{\text{reuse}}$ then
13: break
14: end if
15: end for
16: end while
17: end for
18: $\mathcal{X}_C \leftarrow$ REMOVE_DUPLICATES($\mathcal{X}_C$)
19: return $\mathcal{X}_C$
20: end function
compute for each component, the number of effective samples,

\[ n_{\text{eff}}(o) = \left( \sum_{x_s \in \mathcal{X}_C} w_o(x_s)^2 \right)^{-1} \]

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 \( n_{\text{des}} \). These samples are added to the database and to the set of active samples \( \mathcal{X}_C \) as illustrated in Algorithm 4.

### Algorithm 4
Ensure that every component has sufficiently many effective samples.

**Require:** database \( \mathcal{S} = \{(x_0, \log \hat{p}(x_0), \mathcal{N}_{x_0}), \ldots, (x_N, \log \hat{p}(x_N), \mathcal{N}_{x_N})\} \)

**Require:** Set of chosen samples \( \mathcal{X}_C \), respective self-normalized importance weights \( w_o(x) \)

**Require:** desired number of effective samples per component \( n_{\text{des}} \)

1: function SAMPLE\_WHERE\_NEEDED
2:   for \( o = 1 \ldots N_o \) do
3:     \( n_{\text{eff}}(o) \leftarrow \left( \sum_{x_s \in \mathcal{X}_C} w_o(x_s)^2 \right)^{-1} \)
4:     \( n_{\text{new}}(o) \leftarrow n_{\text{des}} - \lfloor n_{\text{eff}}(o) \rfloor \)
5:     \( \mathcal{X}_{\text{new},o} \leftarrow \text{SAMPLE\_GAUSSIAN}(\mu_o, \Sigma_o, n_{\text{new}}(o)) \)
6:     for \( x_s \) in \( \mathcal{X}_{\text{new},o} \) do
7:         \( \mathcal{S} \leftarrow \mathcal{S} \cup \{(x_s, \log \hat{p}(x_s), \mathcal{N}_{x_s})\} \)
8:     end for
9:     \( \mathcal{X}_C \leftarrow \mathcal{X}_C \cup \mathcal{X}_{\text{new},o} \)
10:   end for
11:   return \( \mathcal{X}_C \)
12: end function

3.2.3 Adapting the number of components

The component optimization (Algorithm 1) is a local optimization, because the component will typically converge to a nearby mode (although the trust region constraint may help traversing 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.

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 to add new components at regions where we expect the target distribution to have a relevant mode and where the current approximation is missing probability mass.
We treat every sample that was evaluated on the target distribution as candidate for the mean of the new component. Every $n_{\text{add}}$ iterations, we compute a heuristic score

$$h(x) = \log \tilde{p}(x) - \max \left( \log q(x; \theta), \max_{x_i \in \mathcal{X}_{\text{total}}} \log q(x_i; \theta) - \Delta \right)$$

for each candidate in the set $\mathcal{X}_{\text{total}}$ of previously evaluated samples and create a new component at the location with the highest score. If the outer maximum operation in Equation [20] returns the first operand, the heuristic $h(x)$ corresponds to the amount of missing log probability density. The second operand is returned, when the log density $\log q(x; \theta)$ on a given candidate $x$ is smaller than the highest log density among all candidates by a margin of $\Delta$. By upper bounding the benefit for being far from the current approximation, we ensure that a sample $x_i$ is never preferred over a different sample if its (unnormalized) log target density $\log \tilde{p}(x_i)$ is worse by more than $\Delta$. Without such threshold, the heuristic $h(x)$ could choose a location with insignificant target density, if it had large amounts of missing log density, which can often happen for target distributions with heavy tails. However, the proposed heuristic usually 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 $\Delta$ typically affects the proposed location. For very small thresholds, we will create the component at a location where $\tilde{p}(x)$ 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.

Choosing the new component’s mean based on a large threshold will, in contrary, place the component far from the current mixture model at locations where $\tilde{p}(x)$ 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.

Similar to the KL bounds $\epsilon(o)$, we can use $\Delta$ as parameter to control the exploration-exploitation trade-off. However, whereas the KL bound controls local exploration in the vicinity of the respective component, $\Delta$ controls global exploration by deciding where to add a new component. We want to add both, components that improve representational power at discovered modes, and components that explore hardly covered regions of the search space—with a varying degree of optimism regarding their potential of improving. We therefore specify several different thresholds in an array, e.g., $\Delta = [1000, 500, 200, 100, 50]$ and pick one of these values by cycling through this array.

After deciding for the mean of the new component, $\mu_{\text{new}}$, we need to initialize its covariance matrix. We want to initialize the new component with an entropy that is similar to those of the best components in the current model and therefore start by computing the desired entropy $h_{\text{des}} = \sum_o q(o) \mathbb{H}(q(x|o))$. For deciding on the correlations among the different dimensions, we can consider restarting the local search from scratch.
by choosing an isotropic covariance matrix $\Sigma_{iso} = c_{iso} I$, and making use of the existing components by averaging their covariance matrices, i.e., $\Sigma_{avg} = c_{avg} \sum_o p(o|\mu_{new}) \Sigma_o$, where $c_{iso}$ and $c_{avg}$ are appropriately chosen to obtain the desired entropy $h_{des}$. 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

$$\Sigma_\alpha = \alpha \Sigma_{iso} + (1 - \alpha) \Sigma_{avg}$$

between both candidate covariance matrices with respect to the expected reward

$$R_{new}(\alpha) = \int x N(x|\mu_{new}, \Sigma_\alpha) \log \tilde{p}(x) dx.$$ 

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

$$z(x) = 0.5 N(x|\mu_{new}, \Sigma_{iso}) + 0.5 N(x|\mu_{new}, \Sigma_{avg}).$$ 

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

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 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.

The resulting method is shown in Algorithm 5. By using importance weighting, we can use the same set of samples for the weight updates (line 7-12) and component updates (line 14-21). Each iteration $i$ starts by adapting the number of components (line 2), which also adds those samples to the database $S$ that were drawn for initializing the covariance matrices of the newly added components. For constructing the set of samples for the following iteration, we start by selecting relevant samples from previous iterations (line 3) and add new samples if necessary (line 5). For the resulting set of samples, the log target densities $\tilde{p}_C$ are retrieved from the database and the background distribution is constructed according to Equation 18 and evaluated on the samples (line 6). An open-source implementation is available online.

In comparison to VIPS, VIPS++ makes 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.

\[^2\]https://github.com/OlegArenz/VIPS
Algorithm 5 Variational Inference by Policy Search

Require: (initial) number of components $N_o$
Require: (initial) mixture parameters $\theta = \{q(o), \mu_{o,...,N_o}, \Sigma_{o,...,N_o}\}$
Require: number of learning iterations $N_i$
Require: number of samples to be reused $n_{\text{reused}}$
Require: desired number of effective samples $n_{\text{des}}$

1: for $i = 1 \ldots N_i$ do
2: $S, N_o, \theta \leftarrow \text{ADD\_AND\_DELETE\_COMPONENTS}(S, N_o, \theta)$
3: $X_C \leftarrow \text{SELECT\_SAMPLES}(S, N_o, \theta)$
4: $w_o(x) \leftarrow \text{COMPUTE\_ALL\_IMPORTANCE\_WEIGHTS}(N_o, \theta, S, X_C)$
5: $X_C, S \leftarrow \text{SAMPLE\_WHERE\_NEEDED}(S, N_o, \theta, n_{\text{des}}, w_o(x))$
6: $\hat{p}_C, z_C \leftarrow \text{OBTAIN\_FROM\_DATABASE}(S, X_C)$
7: for $o = 1 \ldots N_o$ do
8: $q_o(x) \leftarrow \log q(X_C, o; \theta) - \log q(X_C; \theta)$ ▷ EVALUATE LOG RESPONSIBILITIES
9: $w_o \leftarrow \text{COMPUTE\_IMPORTANCE\_WEIGHTS}(\mu_o, \Sigma_o, z_C, X_C)$
10: $\hat{R}_o \leftarrow \text{MC\_ESTIMATE}(w_o, q_o(x), \hat{p}_C, H(\Sigma_o))$ ▷ ESTIMATE REWARD (Eq. 15)
11: end for
12: $q'(o) \leftarrow \frac{\exp(\hat{R}_o)}{\sum_o \exp(\hat{R}_o)}$
13: $\theta \leftarrow \text{UPDATE\_WEIGHTS}(q'(o))$
14: for $o = 1 \ldots N_o$ do
15: $\tilde{q}_o(x) \leftarrow \log q(X_C, o; \theta) - \log q(X_C; \theta)$ ▷ EVALUATE LOG RESPONSIBILITIES
16: $y_o \leftarrow p_C + \tilde{q}_o(x)$ ▷ COMPUTE TARGETS FOR WEIGHTED LEAST SQUARES
17: $w_o \leftarrow \text{COMPUTE\_IMPORTANCE\_WEIGHTS}(\mu_o, \Sigma_o, z_C, X_C)$
18: $R_o, r_o \leftarrow \text{WLS}(w_o, X_C, y_o)$ ▷ LEARN QUADRATIC SURROGATE
19: $\mu'_o, \Sigma'_o \leftarrow \text{GVA\_UPDATE}(\mu_o, \Sigma_o, R_o, r_o, \epsilon_o)$
20: end for
21: $\theta \leftarrow \text{UPDATE\_COMPONENTS}(\mu'_o,...,N_o, \Sigma'_o,...,N_o)$
22: end for

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)] and [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) learn mean-field approximations based on Monte-Carlo gradient estimates of Equation 1. The variance of the gradient is reduced by using Rao-Blackwellization as well as control variates.

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, an 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 $p^*$. 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.

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 but 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 \( x \) is only accepted if the new target density is larger than \( y \). In case of rejection, the area where a new \( x \) 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 \( x \) 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.

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 multi-modal 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 Policy Search

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 LDIPS (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.

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, multi-modal target distribution by starting with a single component and iteratively adding more components according to our heuristic. The selected methods for our comparisons are shown in Section 5.4. The results of the quantitative experiments are presented and discussed in Section 5.5.

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, $N(x|0, I)$ 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 $\sigma^2 = 0.2^2$. Please refer to Appendix D 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 multi-modal probability distributions with distant modes, we consider the problem of approximating an unknown GMM comprising of 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 $\Sigma = A^T A + I_D$ where each entry of the $D \times D$-dimensional matrix $A$ is sampled from a normal distribution with mean 0 and standard deviation $0.1^D$. 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.

5.1.6 Planar Robot

In order to test VIPS++ on a multi-modal 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 $1e^{-2}$ 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 E for details on how the target distribution is computed.

Ground-truth samples for both experiments are shown in Figure 1. 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 a large number of 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.

Figure 1: 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 red circles.

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 multi-modal 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 3.2.3, 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 $N(0, \alpha I)$. 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 0. All covariance matrices are initialized as $\Sigma = \alpha I$. The achieved MMDs are shown in Figure 2. 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 F for a plot of the average number of components that are learned by VIPS++ for all experiments in the testbed.

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 3.2.3. We compare the proposed line-search used by VIPS++ with the interpolation used by VIPS as well as an isotropic initialization. Figure 3 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 multi-modal 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.

5.2.3 Sample Reusage

Compared to VIPS, VIPS++ uses a more sophisticated method for reusing samples from previous iteration—as detailed in Section 3.2.1—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 4 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
Figure 2: 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.

Figure 3: 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.
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 5 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 Stein variational gradient descent (SVGD) [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]. Please refer to Appendix A 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. 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 B.

5.5 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
Figure 5: 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 at iteration 330 (left) and the target distribution (right).
For example, we directly compare VIPS++ also used in our previous work (Arenz et al., 2018) and we use some of the previous results. The MMD experiment, as well as the German credit in the legends. The problems in the testbed. 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.

Figure 6 shows plots of the MMD over the number of function evaluations for the different sampling problems in the testbed. VIPS++ 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 embeddings in a reproducible kernel Hilbert space. Please refer to Appendix C on how the MMD and the ground-truth samples are computed.

Figure 6 shows plots of the MMD over the number of function evaluations for the different sampling problems in the testbed. 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.
Figure 7: 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.

to add new components if the current number of components matches a given threshold. This threshold is indicated in the respective legends. Figure 7 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 G.

5.5.1 Discussion

Figure 8: 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 grey 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.

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. VIPS and VIPS++ were also 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 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 $\log q(o|x)$ exactly can become prohibitive. Figure 8 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.

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 multi-modal 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 multi-modal 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.

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

- 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.

B VIPS++ Hyper-Parameters

The hyper-parameters used for all experiments are given in Table 1.
Table 1: 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 $\ell_2$-regularization when fitting the surrogates are automatically adapted within in the provided ranges.

### C Computing the Maximum Mean Discrepancy

We approximate the MMD between two sample sets $X$ and $Y$ as

$$\text{MMD}(X, Y) = \frac{1}{m^2} \sum_{i,j} k(x_i, x_j) + \frac{1}{n^2} \sum_{i,j} k(y_i, y_j) \tag{33}$$

$$- \frac{2}{mn} \sum_i \sum_j k(x_i, y_j).$$

We use a squared exponential kernel given by

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

where $\Sigma$ is a diagonal matrix where each entry is set to the median of squared distances within the ground-truth set and the bandwidth $\alpha$ 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.
D Goodwin Model

The Goodwin model is defined as

\[
\begin{align*}
\frac{dx_1}{dt} &= \frac{a_1}{1 + a_2 x_2^\rho} - \alpha x_1 \\
\frac{dx_2}{dt} &= k_1 x_1 - \alpha x_2 \\
&\vdots \\
\frac{dx_g}{dt} &= k_{g-1} x_{g-1} - \alpha x_g,
\end{align*}
\] (21)

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 \(\kappa_1, \ldots, \kappa_8\) that need to be inferred. We use the prior also to randomly choose their true values. For an initial condition \(x_0 = 0\), we create 81 noisy observations \(o_{1..81}\) 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

\[
p(a_1, a_2, \kappa_1, \ldots, \kappa_8 | o_{40..81}) = \frac{1}{Z} p(a_1)p(a_2) \prod_{i=1}^8 p(\kappa_i) \prod_{t=40}^{81} p_t(o_t | a_1, a_2, \kappa_1, \ldots, \kappa_8),
\] (22)

where \(p_t(o_t | a_1, a_2, \kappa_1, \ldots, \kappa_8)\) is a Gaussian distribution with variance \(\sigma^2 = 0.2\) and a mean which is computed by numerically integrating the ODE (Equation 21).

E Planar Robot Experiment

Figure 9: The plots show the target densities for the planar robot experiments with one goal position (left) and four goal positions (right), when varying the first two dimensions of a ground truth sample.
The x and y coordinate of the end-effector are given by
\[
x(\theta) = \sum_{i=1}^{10} \cos \left( \sum_{j=1}^{i} \theta_j \right), \quad y(\theta) = \sum_{i=1}^{10} \sin \left( \sum_{j=1}^{i} \theta_j \right).
\]

The target distribution is given as the product of two distributions,
\[
p(\theta) = \frac{1}{Z} p_{\text{conf}}(\theta)p_{\text{cart}}(\theta),
\]
where \(p_{\text{conf}}(\theta)\) enforces smooth configurations and \(p_{\text{cart}}(\theta)\) penalizes deviations from the goal position. We model \(p_{\text{conf}}(\theta)\) 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 \(1{e^{-2}}\). 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 \(1{e^{-4}}\) in both directions, namely
\[
p_{\text{cart},1}(\theta) = \mathcal{N} \left( \begin{bmatrix} x(\theta) \\ y(\theta) \end{bmatrix} \mid \begin{bmatrix} 7 \\ 0 \end{bmatrix}, \begin{bmatrix} 1{e^{-4}} & 0 \\ 0 & 1{e^{-4}} \end{bmatrix} \right).
\]
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. Figure 9 visualizes the target densities for both variants, when varying the first dimension of the respective first ground-truth sample.

F Number of Components

The average number of components learned by VIPS++ is shown in Figure 10.

Figure 10: The average number of components learned by VIPS++ is plotted over function evaluations for all experiments in the testbed. When using the faster adding rate, \(n_{\text{add}} = 1\), VIPS++ learns GMMs with approximately 350 components.
Evaluations with Respect to Computational Time

Figure 11 shows the achieved MMDs with respect to time for the experiments that have been omitted in the main document.

Figure 11: 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 testbed.
References

A. Abdolmaleki, R. Lioutikov, N. Lua, L. P. Reis, J. Peters, and G. Neumann. Model-based relative entropy stochastic search. In Advances in Neural Information Processing Systems (NIPS), 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.

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, 19(14):1–25, 2018.

O. Arenz, M. Zhong, and G. Neumann. Efficient gradient-free variational inference using policy search. In International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 234–243. PMLR, 2018. URL http://proceedings.mlr.press/v80/arenz18a.html.

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, 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.

R. 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.

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, 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.
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)*, May 2017. URL http://eprints.lincoln.ac.uk/26737/.

K. Fan, Z. Wang, J. Beck, J. T. Kwok, and K. Heller. Fast second-order stochastic back-propagation for variational inference. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, NIPS’15, pages 1387–1395, 2015.

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

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

A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13:723–773, Mar. 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.

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

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.

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

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

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 29*, pages 2378–2386. Curran Associates, Inc., 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 31*, 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*, 2017.

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, K. K., and M. 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. *CoRR*, abs/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*, 15(1):2087–2112, Jan. 2014.

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.

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

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 30*, pages 2399–2408, 2017.

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.

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

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

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