Karlsruhe Institute of Technology

Efficient Gradient-Based Variational Inference with GMMs

Master's Thesis
of

Zihan Ye

KIT Department of Informatics
Institute for Anthropomatics and Robotics (IAR)
Autonomous Learning Robots (ALR)

Referees:	Prof. Dr. Techn. Gerhard Neumann
	Prof. Dr. Ing. Tamim Asfour
Advisors:	Dr. Ing. Oleg Arenz
	Prof. Dr. Techn. Gerhard Neumann

Duration: April 1st, 2021 — October 1st, 2021

Erklärung

Ich versichere hiermit, dass ich die Arbeit selbstständig verfasst habe, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht habe und die Satzung des Karlsruher Instituts für Technologie zur Sicherung guter wissenschaftlicher Praxis beachtet habe.

Karlsruhe, den 1. October 2021

Zihan Ye

Zusammenfassung

Die Inferenz aus unbekannten Verteilungen ist ein schwieriges Problem beim maschinellen Lernen. Wenn kein approximiertes Modell benötigt wird, ist die Markov-Chain-Monte-Carlo-Methode (MCMC) die am häufigsten verwendete Methode. Sie erreicht eine hohe Genauigkeit auf Kosten der Berechnungszeit. Die Variationsinferenz hingegen ist sehr effizient, aber ungenau. Arenz et al. (2020) haben vor kurzem Variational Inference by Policy Search (VIPS) vorgeschlagen, das Gaussian Mixture Models (GMMs) für die Inferenz verwendet und in der Lage ist, die komplizierte Zielverteilung genau zu approximieren. VIPS allerdings ignoriert die Gradienteninformation, die normalerweise bei Variational Inference Problemen verfügbar ist. In dieser Arbeit werden wir VIPS erweitern, um Gradienteninformationen zu berücksichtigen. Genauer gesagt zeigen wir, dass der quadrierte Fehler zwischen dem Modellgradienten und dem Zielgradienten in der ordinary least squares verwendet werden kann, um Surrogatmodellparameter zu erhalten. Wir vergleichen gradientVIPS mit VIPS, das bereits eine Reihe von State-of-the-Art-Methoden übertroffen hat, und zeigen, dass gradientVIPS die Stichprobeneffizienz um eine Größenordnung erhöht und dabei die hohe Genauigkeit von VIPS beibehält. Gleichzeitig ist gradientVIPS schneller und kann auf höhere Dimensionen skalieren als VIPS. Wir haben außerdem einen weiteren neuen Ansatz für das Lernen hochdimensionaler Verteilungen entwickelt. Anstatt eine vollständige Gauß-Kovarianz zu lernen, verwenden wir ein faktorisiertes GMM-Modell, um eine Annäherung an die vollständige Gauß-Kovarianz mit niedrigem Rang zu lernen, was die Geschwindigkeit und Skalierbarkeit deutlich erhöhen sollte.

Abstract

Inference from unknown complex distributions is a challenging problem in machine learning. When the approximated model is not required, Markov-chain Monte-Carlo (MCMC) is the most commonly used method. It reaches a high precision at the cost of computation time. Variational inference on the other hand, is highly efficient but inaccurate. Arenz et al. (2020) recently proposed Variational inference by policy search (VIPS), which uses Gaussian mixture models (GMMs) and is able to accurately approximate the intractable target distribution. However, VIPS only requires unnormalized target density and ignores the gradient information, which is usually available in variational inference settings. In this work we will extend VIPS to take gradient information. More specifically we will show that the squared error between model gradient and target gradient can be used in ordinary least squares to get surrogate model parameters. We compare gradientVIPS with VIPS, which has already outperformed a variety of state-of-the-art methods and show that gradientVIPS increase the sample efficiency around one magnitude while maintaining the high accuracy of VIPS. At the mean time, gradientVIPS is faster and can scale to higher dimensions than VIPS. We have further derived another novel approach for learning high-dimensional distributions. Instead of learning a full Gaussian covariance, we employ a factorized GMM model to learn a low rank approximation of the full Gaussian in this approach, which should considerably increase the speed and scalability.

Acknowledgements

First of all, I would like to thank my supervisor Oleg Arenz deeply from the bottom of my heart. Before I did my master thesis, I knew nothing about machine learning but you were willing to accept me to do this thesis, lead me to step on the way of robotics and machine learning, keep encouraging me and always be there whenever I meet a problem. You spent countless hours of teaching me how to debug, how to derive a new theory and how to do scientific research, which had an important impact on me.

I also want to thank my supervisor Gerhard Neumann. Thank you for offering me the chance to do the thesis with you and offering me the official help whenever I need. During your machine leaning lecture I have learned quite a lot and you were always willing to answer my questions I had arose during the lecture, no matter how easy they were. Thank you for sparking me the interest of machine learning and robotics in the excellent machine learning lecture.

The next person I want to thank is my girlfriend Ziyuan, you keep encouraging me through all the difficult times, giving me strength when I almost gave up. I also want to thank my parents for your unconditional love, although during the Master study period we are 8000km away from each other, your tolerance, encourage, support and love are always by my side.

At last, to memorize my grandpa Huixian Liu, who tragically passed away during the time I was doing the Master thesis. I feel so heart-broken that because of the corona restriction and quarantine policy I couldn’t fly back and keep accompany with him at the end his life. I will remember his love on me and his spirit "Never give up".

Table of Contents

Chapter 1.

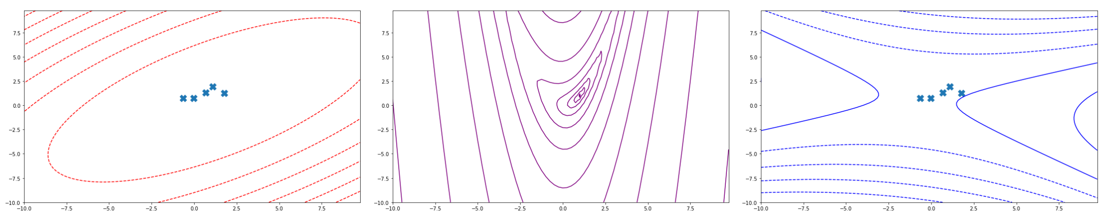
Introduction

Inference from unknown complex distribution is a challenging problem in machine learning. Many methods for machine learning rely on approximate inference from intractable probability distributions. When models of the target distribution is not required, Markov chain Monte-Carlo (MCMC) is the most commonly used technique for approximation inference, it allows user to trade computation time for preciseness. However, MCMC is restricted to run iterative, we can not use our previous function evaluations without violating Markov assumption. Another commonly used method is variational inference, which can bring us computational convenience (Blei et al., 2017). In variational inference settings the problem is formed as:

$$D_{\mathrm{KL}}(p(\boldsymbol{x})||q(\boldsymbol{x}|\boldsymbol{Z})) = \int_{\boldsymbol{Z}} p(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x}|\boldsymbol{Z})} d\boldsymbol{Z},$$

where $\boldsymbol{Z}$ denotes the data sets, $p(\boldsymbol{x})$ denotes the model and $q(\boldsymbol{x}|\boldsymbol{Z})$ stands for the target. Thus, inference is changed into an optimization problem. We need to find the optimal inference parameters to minimize the Kullback-Leibler (KL) divergence. A widely used variational inference algorithm is mean-field variational inference. Mean-field variational inference assumes each dimension is uncorrelated, hence it brings us huge computation convenience, but due to the assumption it can introduce huge impreciseness. Accurately inference requires rich and expressive model families. Inspired by VIPS (Arenz et al., 2020, 2018), which has shown Gaussian mixture models (GMMs) can learn accurate approximation in medium-scale problems, we will also use GMMs to do approximation in our proposed approaches.

Due to the good performance of VIPS in medium-scale (below 60 dimensions) problems, we want to extend VIPS to higher dimensions. VIPS only evaluates the target information without



(a) Model learned with gradient (b) Target Rosenbrock function (c) Model learned without gradient

Figure 1.1.: The left side is the model learned with gradient, based on five samples drawn from the target Rosenbrock function. In the middle is the plot of target Rosenbrock function. In the right is the model learned without gradient, based on the same five samples. As we can see from the plots, by including the gradient information we can get a more reasonable model.

considering the gradient information, which is usually available in a variational inference setting. It is a huge waste considering about the amount of the information gradient can give compared with target information. Hence we consider to include gradient information in VIPS, which should help us to improve the sample efficiency and increase the speed. Besides using GMMs, an expressive model, another guarantee of learning accurate approximation is that, in VIPS we learn a full Gaussian covariance which is feasible in medium-scale problems but can require huge amount of computation in high dimensional problems. Thus, the second point we consider is to do a low rank approximation of the full Gaussian covariance, which can considerably decrease the number of parameters that needed to be learned.

Based on VIPS and the two ideas introduced in the last paragraph, we introduce three approaches to scale VIPS to higher space in this work. The first approach we derived is *gradientVIPS*. In *gradientVIPS* we follow the same scheme as in VIPS: We tighten the lower bound at first and then alternate between weight update and component update. When we update components, we regress the squared error of target density and gradient between model and target distribution to get the model parameter.

The second approach we proposed is embedded VIPS, a novel approach of learning a high dimensional complex distribution. In this approach, instead of learning a full Gaussian covariance, we employ a factorized Gaussian mixture models (GMMs) to learn a low rank approximation of the full Gaussian. Using the same idea as in VIPS, we firstly introduce an auxiliary distribution to derive a new variational lower bound. We tighten the lower bound before we update components and weights like in Expectation-Maximization algorithm (Bishop, 2006). The component update of embedded VIPS can be divided into update the latent components and update the decoder components. We use VIPS to update the latent components, where we get the latent rewards based on Monte-Carlo estimation, and use the modified Model-free trajectory-based policy optimization with monotonic improvement (MOTO, Akrour et al. 2018) algorithm to update the decoder components. For the third approach, we want to combine the benefits of including gradient information and using the low rank approximation. We will use *gradientVIPS* in the latent component update. where the rewards of latent samples are calculated based on re-parameterization trick (Kingma and Welling, 2013) and use the modified MOTO to update decoder.

To evaluate our proposed approaches, for *gradientVIPS* we did GMM, logistic regression and planar robot experiments. The experiments results show that *gradientVIPS* improves the computational efficiency by around one order of magnitude and remains the accuracy of VIPS at the same time, which is state-of-the-art in the considered problem domain. For high dimensional VIPS and embedded VIPS, due to time reason, we evaluated the identity mapping and fixed decoder.

Chapter 2.

Preliminaries

In this chapter we firstly formalize the optimization problem based on information projection and introduce the model we use for inference. Then we discuss the stochastic search algorithm Model-based relative entropy stochastic search (MORE, Abdolmaleki et al. 2015) and its extension Variational inference by Policy Search (VIPS, Arenz et al. 2020). Since our approach embedded-VIPS is based on Model-free trajectory-based policy optimization with monotonic improvement (MOTO, Akrour et al. 2018), we further discuss about MOTO. In embedded VIPS, due to the factorized structure we will use Probablistic principle component analysis (PPCA, Bishop 2006) to initialize the mapping matrix, so we briefly talk about PPCA in this chapter. To reduce the number of function evaluations, we also use the idea as in VIPS (Arenz et al., 2020) to employ importance weighting to do sample reuse. We therefore briefly review the idea of sample reuse by importance weighting. At last, we talk about data whitening.

2.1. Problem Statement

Variational inference is usually framed as an information projection (I-projection) problem. In Variational Inference we want to find the best parameters θ that can minimize the KL divergence between model distribution $q(\mathbf{x};\theta)$ and the unnormalized target distribution $p(\mathbf{x})$. Computing the KL divergence between model and unnormalized target distribution will introduce a constant term, which will not affect the minimization. By subtracting the constant term of both sides we can get the evidence lower bound (ELBO), thus, the optimization problem of minimizing the KL divergence turns into maximizing the ELBO (Jordan et al., 1999; Bishop, 2006),

$$\begin{aligned}
&\arg\min_{q(\boldsymbol{x}|\theta)} D_{\mathrm{KL}}(q(\boldsymbol{x}|\theta)||p(\boldsymbol{x})) = \arg\min_{q(\boldsymbol{x}|\theta)} \int_{\boldsymbol{x}} q(\boldsymbol{x}|\theta) \log \frac{q(\boldsymbol{x}|\theta)}{p(\boldsymbol{x})} d\boldsymbol{x} \\
&= \arg\min_{q(\boldsymbol{x}|\theta)} \int_{\boldsymbol{x}} q(\boldsymbol{x}|\theta) \log \frac{q(\boldsymbol{x}|\theta)}{\tilde{p}(\boldsymbol{x})} d\boldsymbol{x} + const = \arg\max_{q(\boldsymbol{x}|\theta)} \underbrace{\int_{\boldsymbol{x}} q(\boldsymbol{x}|\theta) \log \tilde{p}(\boldsymbol{x}) d\boldsymbol{x} + H(q(\boldsymbol{x}|\theta))}_{ELBO}.
\end{aligned}$$

Same as the previous work VIPS, we will use Gaussian mixture models (GMMs) and we also want to learn the full Gaussian. However, since learning the full Gaussian is computationally heavy, VIPS reaches its limit when dealing with high dimensional problem (over 60 dimensions). In this work we want to extend VIPS to higher dimensional problems.

2.2. Gaussian Mixture Models

Learning sufficiently accurate approximations requires a rich and expressive model family. In our approaches we use Gaussian mixture models (GMMs). When sufficient number of components are given, GMMs can represent arbitrary distributions very well. GMMs has the form

$$q(\mathbf{x}) = \sum_{o} q(o) q(\mathbf{x}|o),$$

where $q(o)$ is the weight and $q(\mathbf{x}|o)$ is the corresponding Gaussian component.

2.3. Information and Moment Projection.

Kullback-Leibler (KL) divergence was introduced by Kullback and Leibler (1951) and defined as

$$D_{\mathrm{KL}}(p(\boldsymbol{x})||q(\boldsymbol{x})) = \int_{\boldsymbol{x}} p(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} d\boldsymbol{x}.$$

KL divergence is a similarity measurement between two distributions. Due to the asymmetry of the KL divergence, we have two formulations of the KL divergence (moment projection and information projection).

The KL formulation: Moment-projection, is defined as

$$\underbrace{\arg\min_{q(\boldsymbol{x})} \quad D_{\mathrm{KL}}(p(\boldsymbol{x})||q(\boldsymbol{x}))}_{Moment-projection}.$$

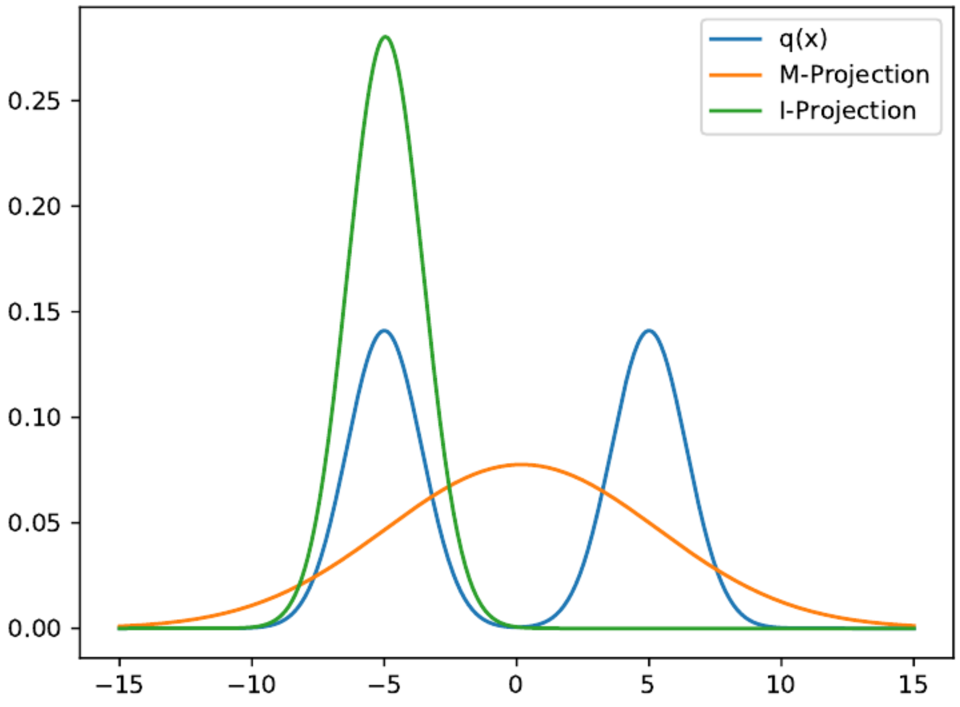


Figure 2.1.: Example of different projections. M-projection averages the two modes while I-projection concentrates on one mode. Picture is taken from Arenz (2021).

The reversed KL formulation: Information-projection, is defined as

$$\underbrace{\arg\min_{q(\boldsymbol{x})} \quad D_{\mathrm{KL}}(q(\boldsymbol{x})||p(\boldsymbol{x}))}_{Information-projection},$$

where we assume $p(\boldsymbol{x})$ to be the target distribution and $q(\boldsymbol{x})$ to be the model distribution. Moment projection forces the approximate distribution $q(\boldsymbol{x})$ to have high probability where $p(\boldsymbol{x})$ has high probability, which means the M-projection tries to average over the modes. Instead, information projection forces the approximate distribution $q(\boldsymbol{x})$ to be zero where $p(\boldsymbol{x})$ is zero. When using one Gaussian to learn a multi-modal distribution, the information projection typically concentrates on a single mode of the target distribution while moment projection tends to average all the modes (Neumann et al., 2011; Deisenroth et al., 2013). Therefore, information projection is usually suitable for mode finding and model learning. However, the computational demand of information projection is heavy and for most of the distributions the information projection are unable to be obtained in closed form. In our approaches, we use Gaussian mixture models (GMMs). For Gaussians we can get a closed form solution of calculating the information projection. Figure 2.1 shows the moment projection, the information projection and target distribution.

2.4. Model-based Relative Entropy Stochastic Search

Model-based relative entropy stochastic search (MORE) is a stochastic optimization algorithm introduced by Abdolmaleki et al. (2015). MORE gives us a closed form update of Gaussian search distribution which doesn't require the gradient information. To prevent model from getting stuck in a local optimum, MORE introduce an entropy bound to guarantee sufficient exploration. To ensure the stability and monotonic improvement, MORE introduces a KL term of the search distribution to bound the update of Gaussian search distribution from stepping too far away. The problem settings of MORE can be framed as

$$\begin{aligned} &\underset{q(\mathbf{x})}{maximize} \int_{\mathbf{x}} q(\mathbf{x}) R(\boldsymbol{x}) d\boldsymbol{x} \\ &s.t\ D_{\mathrm{KL}}(q(\mathbf{x}))||q^i(\mathbf{x})) \leq \varepsilon, H(q(\mathbf{x})) > \beta, \int q(\mathbf{x}) d\boldsymbol{x} = 1, \end{aligned}$$

where $R(\boldsymbol{x})$ is a locally constructed quadratic surrogate model in the vicinity of the current distribution and has the form $R(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top R^i \boldsymbol{x} + \boldsymbol{x}^\top \boldsymbol{r}^i + r_0$. We assume the quadratic term $\tilde{R}$ to be symmetric. The quadratic model can be constructed by using samples drawn from the current distribution. The parameters of the model can be obtained by using ordinary least squares, which we will talk in detail in Chapter 4. By introducing the Lagrange multipliers η, ω, λ, the Lagrange function of the constraint optimization problem can be framed as

$$\begin{aligned} L(q(\boldsymbol{x}), \eta, \omega, \lambda) = &\int_{\mathbf{x}} q(\mathbf{x}) R(\boldsymbol{x}) d\boldsymbol{x} + \eta(\varepsilon - D_{\mathrm{KL}}(q(\mathbf{x})||q^i(\mathbf{x})) \\ &+ \omega(H(q(\mathbf{x})) - \beta) + \lambda(1 - \int q(\mathbf{x}) d\boldsymbol{x}). \end{aligned}$$

The optimal search distribution can be obtained by setting the partial derivative of $L(q(\boldsymbol{x}), \eta, \omega, \lambda)$ with respect to $q(\boldsymbol{x})$ to be zero. By setting the partial derivative to be zero, we can get a closed form solution of the optimal distribution

$$q(\boldsymbol{x}) \propto q^i(\boldsymbol{x})^{(\eta^*/(\eta^*+\omega^*))} exp(\frac{R(\boldsymbol{x})}{\eta^* + \omega^*}).$$

From the optimal distribution we can get

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

the new mean and the new covariance of the updated distribution are $\boldsymbol{\mu} = \mathbf{Q}^{-1}\boldsymbol{q}$, $\boldsymbol{\Sigma} = \mathbf{Q}^{-1}$. The convex dual function can be obtained by integrating out λ in the Lagrange function and the optimal parameters η, ω can be obtained efficiently using L-BFGS-B (Byrd et al., 1995) optimization with respect to the dual function

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

where $\log Z(\mathbf{X},\mathbf{x}) = -\frac{1}{2}\left(\mathbf{x}^\top\mathbf{X}^{-1}\mathbf{x}+\log\left|2\pi\mathbf{X}^{-1}\right|\right)$ is the function of a Gaussian with natural parameters $\mathbf{X}$ and $\mathbf{x}$. The partial derivatives are

$$\frac{\partial G(\eta,\omega)}{\partial\omega} = H(q(\boldsymbol{x}))-\beta, \qquad \frac{\partial G(\eta,\omega)}{\partial\eta} = \varepsilon - D_{\mathrm{KL}}(q(\boldsymbol{x})||q^i(\boldsymbol{x})).$$

MORE can be used for variational inference by substituting ω to 1.

2.5. Variational Inference by Policy Search

Variational inference by policy search (VIPS) was introduced by Arenz et al. (2020) for learning multi-modal distribution. We want to approximate an unnormalized target distribution $\tilde{p}(\mathbf{x})$ using a Gaussian mixture model

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

based on information projection, by introducing an auxiliary distribution. The evidence lower bound (ELBO) can be decomposed into a variational lower bound and the expectation of a KL term.

$$\begin{aligned}\mathscr{L}_o(q(o),q(\mathbf{x}|o),\tilde{q}(o|\mathbf{x})) =& \sum_o q(o)\int_{\mathbf{x}} q(\mathbf{x}|o)\Big(\log\tilde{p}(\mathbf{x})-\log\frac{q(o)q(\mathbf{x}|o)\tilde{q}(o|\mathbf{x})}{q(o|\mathbf{x})\tilde{q}(o|\mathbf{x})}\Big)d\mathbf{x}\\ =& \sum_o q(o)\Big(\int_{\mathbf{x}} q(\mathbf{x}|o)\Big(\log\tilde{p}(\mathbf{x})+\log\tilde{q}(o|\mathbf{x})\Big)d\mathbf{x}+H(q(\mathbf{x}|o))\Big)+H(q(o))\\ &+\mathbb{E}_{q(\mathbf{x})}D_{\mathrm{KL}}\Big(q(o|\mathbf{x})||\tilde{q}(o|\mathbf{x}))\Big)\\ =&\tilde{\mathscr{L}_o}(q(o),q(\mathbf{x}|o),\tilde{q}(o|\mathbf{x}))+\mathbb{E}_{q(\mathbf{x})}D_{\mathrm{KL}}\Big(q(o|\mathbf{x})||\tilde{q}(o|\mathbf{x}))\Big).\end{aligned}$$

We can maximize the ELBO by alternating between updating the lower bound with respect to the weights $q(o)$ and the components $q(\mathbf{x}|o)$, where we set $\tilde{q}(o|\mathbf{x}) = q(o|\mathbf{x})$ before each update to tighten the bound. We will talk about component update and weights update in detail in section 2.5.1 and 2.5.2.

2.5.1. Update Components

Since the variational lower bound is a sum function of independent terms, it can be divided into sub-maximization problem for each component. The objective of each component can be framed as

$$\underset{q(\mathbf{x}|o))}{maximize} \int_{\mathbf{x}} q(\mathbf{x}|o)\Big(\log \tilde{p}(\mathbf{x}) + \log \tilde{q}(o|\mathbf{x})\Big) d\mathbf{x} + H(q(\mathbf{x}|o))\Big)$$
$$s.t \quad D_{\mathrm{KL}}(q(\mathbf{x}|o))||q^{i}(\mathbf{x}|o)) \leq \varepsilon(o),$$

where $q^i(\mathbf{x}|o)$ denotes the $q(\mathbf{x}|o)$ learned from the last iteration and we add a KL bound for stability and ensure the monotonic improvement (Akrour et al., 2018; Schulman et al., 2015). The entropy part enters the objective, which is in several algorithm used for better exploration (Neu et al., 2017). The $\varepsilon(o)$ is a KL upper bound we adapt during iteration. If a component is improved, we increase the KL bound in the next iteration by multiplying it with 1.2. If a component is not improved we decrease the KL bound in the next iteration by multiplying it with 0.8. We construct a local quadratic surrogate model R($\mathbf{x}$) to approximate the component specific reward $\log \tilde{p}(\mathbf{x}) + \log \tilde{q}(o|\mathbf{x})$ in the vicinity of the respective component $q(\mathbf{x}|o)$. The parameters of the quadratic model can be got by using ordinary least squares

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

where $\boldsymbol{X}$ denotes features of the samples, $\boldsymbol{y}$ denotes the stacked target and $\boldsymbol{\beta}_o$ denotes the parameters of the quadratic model, also we need a regularization term κ_o in practice. The samples used for constructing the quadratic model is drawn based on the current distribution $q(\mathbf{x}|o)$. The covariance and mean value of the new distribution can be got using modified MORE

$$\boldsymbol{\Sigma} = \left(\frac{\eta}{\eta+1}\boldsymbol{\Sigma}_i^{-1} + \frac{1}{\eta+1}\boldsymbol{R}_i\right)^{-1}, \boldsymbol{\mu} = \boldsymbol{\Sigma}\left(\frac{\eta}{\eta+1}\boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i + \frac{1}{\eta+1}\boldsymbol{r}_i\right).$$

In practice, to reuse previous samples, we do sample reuse by using importance weighting. The parameters of our model can be get using weighted least squares (Chen et al., 2016),

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

where $\boldsymbol{W}$ is the diagonal weight matrix calculated by samples and assign each sample its weight. We will talk about importance weights in section 2.7 in detail.

2.5.2. Update Weight

The objective of the weight update is framed as

$$\sum_o q(o)R(o) + H(q(o)),$$

where $R(o)$ denotes the objective of the component update,

$$R(o) = \int_{\mathbf{x}} q(\mathbf{x}|o)\Big(\log \tilde{p}(\mathbf{x}) + \log \tilde{q}(o|\mathbf{x})\Big) d\mathbf{x} + H(q(\mathbf{x}|o)).$$

Since the objective is discrete and has an intractable integral part in the reward. It is not beneficial to approximate the reward using a quadratic model. Instead, we want to use Monte-Carlo estimate to approximate the reward $R(o)$

$$\tilde{R}(o) = \frac{1}{N_o} \sum_{n=1}^{N_o} \Big[R\left(\mathbf{x}_{o,n}\right) + \log \tilde{q}(o \mid \mathbf{x}_{o,n})\Big] + \mathrm{H}(q(\mathbf{x}|o)).$$

Based on the approximated reward, the optimal solution can be got in closed form as

$$q(o) = \frac{exp(\tilde{R}(o))}{\sum_o exp(\tilde{R}(o))}.$$

2.5.3. Add and Delete Heuristic

In order to learn a multi modal distribution, we need to dynamically add and delete components. Components which are not improved for a while and also have low weights will be deleted. For adding new components, every sample in the database will be treated as candidate for the initial mean and then we select the most promising candidate according to an estimate of its initial reward. The weight of the newly added component will be set to a very small value. The initial covariance will be set to isotropic. For more details please refer to the reference VIPS (Arenz et al., 2020).

2.6. Model-free Trajectory-based Policy Optimization with Monotonic Improvement

Model-free trajectory-based policy optimization with monotonic improvement (MOTO) is an algorithm introduced by Akrour et al. (2018). It demonstrates on highly non-linear control tasks the improvement in performance of their algorithm in comparison to approaches that linearize the system dynamics. The followings is the objective function and constraints of MOTO

$$\begin{aligned} \underset{\pi}{maximize} & \int\int \rho_t^i(s)\pi(a|s)Q^i(s,a)dads \\ subject\ to \quad & \mathbb{E}_{s\sim\rho_t^i(s)}\left[D_{\mathrm{KL}}(\pi(.|s)||\pi_t^i(.|s))\right] \leq \varepsilon \\ & \mathbb{E}_{s\sim\rho_t^i(s)}\left[H(\pi(.|s))\right] \geq \beta \\ & \int \pi(.|s)da = 1, \end{aligned}$$

where Q denotes the learned quadratic function of the action space, ρ denotes the state space distribution. For a good understanding, the parameters will be omitted and we will use π_1 to denote the π from last iteration. π_1 has the form of $\mathcal{N}(x \mid F_iL_iz + F_if_i, \Sigma_i)$. After using the Lagrange multiplier to take the constraints into account and using the optimal parameters the dual function can be rewritten as

$$\begin{aligned} G(\eta,\omega) &= (\eta+\omega)\int\rho + \eta\varepsilon - \omega\beta - (\eta+\omega)\int\rho + (\eta+\omega)\int\rho(s)log\int\pi_1^{\frac{\eta}{\eta+\omega}}exp(\frac{Q}{\eta+\omega}) \\ &= \eta\varepsilon - \omega\beta + (\eta+\omega)\int\rho(s)\log\int\pi_1^{\frac{\eta}{\eta+\omega}}exp(\frac{Q}{\eta+\omega}). \end{aligned}$$

The optimal policy distribution is

$$\pi \propto \pi^i(\mathbf{x}|\mathbf{z},o)^{(\eta^*/(\eta^*+\omega))}exp(\frac{Q(x,z)}{\eta^*+\omega}),$$

using the partial derivatives with respect to ω and η

$$\frac{\partial G(\eta,\omega)}{\partial\omega} = \mathbb{E}\left[H(\pi^*)\right] - \beta, \qquad \frac{\partial G(\eta,\omega)}{\partial\eta} = \varepsilon - \mathbb{E}\left[D_{\mathrm{KL}}(\pi^*||\pi_1)\right].$$

We can get optimal $\eta^*$and ω^* efficiently using L-BFGS with respect to the dual function. With optimal $\eta^*$and ω^* the parameters of new policy distribution can be got as

$$F = (\eta^*\Sigma_i^{-1} - Q_{xx})^{-1}, \qquad L = \eta^*\Sigma_i^{-1}F_iL_i + Q_{xz}, \qquad f = \eta^*\Sigma_i^{-1}F_if_i + r_x.$$

for detailed derivation of MOTO please refer to the appendix A. For computing the expectation of the KL divergence between two conditional Gaussian distributions please refer to the appendix B.

2.7. Probabilistic Principle Component Analysis

In this subsection we will discuss Probabilistic principle component analysis (PPCA, Bishop 2006) and how it relates to our work. We can formulate probabilistic PCA by first introducing an explicit latent variable z corresponding to the principal-component subspace. Next we define a

Gaussian prior distribution $p(\boldsymbol{z})$ over the latent variable, together with a Gaussian conditional distribution $p(\boldsymbol{x}|\boldsymbol{z})$ for the observed variable $\boldsymbol{x}$ conditioned on the value of the latent variable. Specifically, the prior distribution over z is given by a zero-mean unit-covariance Gaussian

$$p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|0,\boldsymbol{I}).$$

Similarly, the conditional distribution of the observed variable $\boldsymbol{x}$, conditioned on the value of the latent variable $\boldsymbol{z}$, is again Gaussian, of the form

$$p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{W}\boldsymbol{z}+\boldsymbol{\mu},\sigma^2\boldsymbol{I}),$$

in which the mean of $\boldsymbol{x}$ is a general linear function of $\boldsymbol{z}$ governed by the $D*M$ matrix $\boldsymbol{W}$ and the D-dimensional vector μ. In the aspects of generative viewpoint we can get the relation

$$\boldsymbol{x} = \boldsymbol{W}\boldsymbol{z}+\boldsymbol{\mu}+\varepsilon,$$

where ε is a Gaussian distributed noise with the mean value equals zero and covariance equals $\sigma^2\boldsymbol{I}$. If we want to determine the parameters $\boldsymbol{W},\varepsilon,\sigma^2$ by using maximum likelihood, we need firstly write the likelihood function of $p(\boldsymbol{x})$,which is

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})d\boldsymbol{z},$$

since it corresponds to a linear Gaussian model, the marginal distribution is also a Gausssian and given by

$$p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{C}), \quad \boldsymbol{C} = \boldsymbol{W}\boldsymbol{W}^\top+\sigma^2\boldsymbol{I}$$

which is a high dimensional matrix, we can also use the Gaussian identities to get the posterior distribution $p(\boldsymbol{z}|\boldsymbol{x})$, if we want to get the posterior distribution we need firstly invert $\boldsymbol{C}$, which is computationally very heavy. Instead we will use wood-bury identity to evaluate the inversion. The inversion of $\boldsymbol{C}$ is

$$\boldsymbol{C}^{-1} = \sigma^{-1}\boldsymbol{I}-\sigma^{-2}\boldsymbol{W}\boldsymbol{M}^{-1}\boldsymbol{W}^\top, \quad \boldsymbol{M} = \boldsymbol{W}^\top\boldsymbol{W}+\sigma^2\boldsymbol{I}$$

for linear Gaussian model, the posterior can be obtained as

$$p(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{M}^{-1}\boldsymbol{W}^\top(\boldsymbol{x}-\boldsymbol{\mu}),\sigma^{-2}\boldsymbol{M})$$

if we want to directly do the likelihood maximization the computation demand is too heavy, instead, we will use EM algorithm to get the optimal parameters. Firstly we need to get the log-likelihood of complete data. The log-likelihood of the complete data is

$$\ln p\left(\mathbf{X},\mathbf{Z};\boldsymbol{\mu},\mathbf{W},\sigma^2\right) = \sum_{n=1}^{N}\left\{\ln p\left(\mathbf{x}_n \mid \mathbf{z}_n\right)+\ln p\left(\mathbf{z}_n\right)\right\}.$$

Using maximum likelihood estimation we can easily get the optimal parameter of μ is the average x of all the data points. By subtracting μ we can get the complete data log likelihood is

$$\begin{aligned}\mathbb{E}\left[\ln p\left(\mathbf{X},\mathbf{Z};\boldsymbol{\mu},\mathbf{W},\sigma^2\right)\right] = -\sum_{n=1}^{N}\Bigg\{&\frac{D}{2}\ln\left(2\pi\sigma^2\right)+\frac{1}{2}\operatorname{Tr}\left(\mathbb{E}\left[\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}\right]\right)\\ &+\frac{1}{2\sigma^2}\left\|\mathbf{x}_n-\boldsymbol{\mu}\right\|^2-\frac{1}{\sigma^2}\mathbb{E}\left[\mathbf{z}_n\right]^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}\left(\mathbf{x}_n-\boldsymbol{\mu}\right)\\ &+\frac{1}{2\sigma^2}\operatorname{Tr}\left(\mathbb{E}\left[\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}\right]\mathbf{W}^{\mathrm{T}}\mathbf{W}\right)\Bigg\}\end{aligned}$$

In the Expectation step,we use old parameter to calculate

$$\mathbb{E}\left[\mathbf{z}_n\right]=\mathbf{M}^{-1}\mathbf{W}^{\mathrm{T}}\left(\mathbf{x}_n-\overline{\mathbf{x}}\right), \qquad \mathbb{E}\left[\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}\right]=\sigma^2\mathbf{M}^{-1}+\mathbb{E}\left[\mathbf{z}_n\right]\mathbb{E}\left[\mathbf{z}_n\right]^{\mathrm{T}}.$$

In the Maximization step, we will update the corresponding parameters

$$\begin{aligned}\mathbf{W}_{\text{new}} &= \left[\sum_{n=1}^{N}\left(\mathbf{x}_n-\overline{\mathbf{x}}\right)\mathbb{E}\left[\mathbf{z}_n\right]^{\mathrm{T}}\right]\left[\sum_{n=1}^{N}\mathbb{E}\left[\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}\right]\right]^{-1}\\ \sigma_{\text{new}}^2 &= \frac{1}{ND}\sum_{n=1}^{N}\left\{\left\|\mathbf{x}_n-\overline{\mathbf{x}}\right\|^2-2\mathbb{E}\left[\mathbf{z}_n\right]^{\mathrm{T}}\mathbf{W}_{\text{new}}^{\mathrm{T}}\left(\mathbf{x}_n-\overline{\mathbf{x}}\right)\right.\\ &\left.\quad+\operatorname{Tr}\left(\mathbb{E}\left[\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}\right]\mathbf{W}_{\text{new}}^{\mathrm{T}}\mathbf{W}_{\text{new}}\right)\right\}.\end{aligned}$$

Probabilistic PCA is computational more efficient compared with traditional PCA, especially when dealing with high dimensional data, where in the traditional PCA we need to get a high dimensional covariance matrix and calculate the eigenvector of the high dimensional matrix. In embedded VIPS, where we assume to deal with high dimensional problem, we use PPCA to initialize the gain matrix. First, we will sample the target, and then use PPCA to get $\mathbf{W}_{\text{new}}$ and σ_{new}^2, where $\mathbf{W}_{\text{new}}$ correspond to the initial gain matrix and σ_{new}^2 correspond to the scaling factor of the decoder covariance.

2.8. Sample Reuse by Importance Weighting

Arenz et al. (2020) proposed a procedure to reuse previous function evaluations. We store all previous samples together with its unnormalized target density and also the parameters of the distributions that is used to draw these samples. We use our previous samples to estimate the expected value of current distribution of a given function $\mathbb{E}_{q(\mathbf{x})}[f(\mathbf{x})]$. In gradientVIPS we will use the same idea. To do so, we will rely on a technique called importance sampling. Assuming that the support of $z(\mathbf{x})$ covers the support of $q(\mathbf{x})$, we can express the desired expectation as

$$\mathbb{E}_q[f(\mathbf{x})]=\int_{\mathbf{x}}q(\mathbf{x})f(\mathbf{x})d\mathbf{x}=\int_{\mathbf{x}}z(\mathbf{x})\frac{q(\mathbf{x})}{z(\mathbf{x})}f(\mathbf{x})d\mathbf{x}=\mathbb{E}_z[w(\mathbf{x})f(\mathbf{x})],$$

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

$$\mathbb{E}_q[f(\mathbf{x})] \approx \sum_{i=1}^{N_z} \frac{1}{N_z} w(\mathbf{x}_i) f(\mathbf{x}_i).$$

In our approaches, instead of directly using importance sampling, we use self-normalized importance sampling (Hesterberg, 1988). Self-normalized importance sampling has the advantage that it is applicable for unnormalized distribution.

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

However, just using self-normalized importance sampling is not enough, we also need to guarantee the quality of the samples. We use the idea of effective sample size to monitor the quality of the chosen sampling distribution

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

which approximates the number of samples that standard Monte-Carlo would require to achieve the same variance as the importance sampling estimate (Kong et al., 1994; Djuric et al., 2003; Arenz et al., 2020). The work flow is listed in algorithm 1. We firstly select the set of samples to be reused. Then calculate the self-normalized importance weights. After that we calculate the effective samples size. If the effective sample size is smaller than the desired sample size, we will draw new samples from the current distribution. After drawing new samples, we need to recalculate the self-normalized importance weights and then use the newly calculated self-normalized importance weights to do model fitting.

Algorithm 1 Sample reuse for VIPS

```
Require: desired number of samples $N_{des}$
Require: sample sets $\mathbf{X}_s$
Require: current distribution $q(\mathbf{x}|o)$
1: calculate self-normalize importance weighting based sample sets $\mathbf{X}_s$
2: calculate effective number $N_{eff}$ of samples by using self-normalize importance weights
3: if $N_{eff} < N_{des}$ then
4:     draw $N_{des} - N_{eff}$ new samples based on current distribution $q(\mathbf{x}|o)$
5:     recompute self-normalized importance weights
6: end if
```

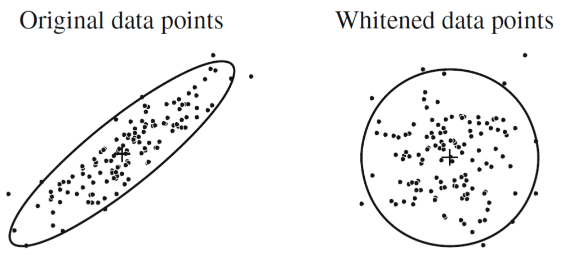


Figure 2.2.: Examples of the original data and the data after whitening. Picture is taken from Mukundan et al. (2017)

2.9. Data Whitening

Whitening is a commonly used technique of preprocessing data. Whitening is a linear transformation that converts the mean and covariance of a data set into zero mean and unit covariance, which makes data uncorrelated with each other (Koivunen and Kostinski, 1999; Bishop, 2006; Kessy et al., 2018). Figure 2.2 shows us a comparison of the original data points and data points after whitening.

Suppose we have a normal distributed data set $\boldsymbol{X}$, the mean value and covariance of the samples are $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. The matrix $\boldsymbol{C}$ is used for whitening, it is the Cholesky decomposition matrix of the $\boldsymbol{\Sigma}$, which has the correlation $\boldsymbol{\Sigma} = \boldsymbol{C}\boldsymbol{C}^\top$. Equation 2.1 shows the general procedure of doing whitening by subtracting the mean and multiply the inversion of the Cholesky decomposition matrix of the distribution covariance,

$$\boldsymbol{z} = \boldsymbol{C}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}). \tag{2.1}$$

Now we talk about how to get the correct gradient after whitening. Assume we have a quadratic surrogate model and a sample $\boldsymbol{x}$, then the target value y should be

$$y = \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} + \boldsymbol{x}^\top \boldsymbol{b} + c.$$

If we do whitening and use the corresponding sample $\boldsymbol{z}$. We will get

$$y = \boldsymbol{z}^\top \boldsymbol{A}_z \boldsymbol{z} + \boldsymbol{z}^\top \boldsymbol{b}_z + c_z.$$

The original gradient is $\frac{\partial y}{\partial \boldsymbol{x}}$, the gradient after whitening is $\frac{\partial y}{\partial \boldsymbol{z}}$. According to the chain rule, the gradient after whitening should be

$$\frac{\partial y}{\partial \boldsymbol{z}} = \frac{\partial y}{\partial \boldsymbol{x}} \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}} = \boldsymbol{C}\frac{\partial y}{\partial \boldsymbol{x}}.$$

It means if we have the gradient from the original data and want to use the original gradient in the whitened data, we need to firstly multiply the original gradient with Cholesky covariance of the whitening to make it compatible to the data after whitening.

Chapter 3.

Related Work

We will discuss the related work with respect to variational inference and reinforcement learning in detail.

3.1. Variational Inference

Traditionally, by applying the mean-field assumption which assumes each dimension of the target distribution is uncorrelated, mean-field variational inference (Anderson and Peterson, 1987; Saul et al., 1996) was applied for learning approximations of high dimensional distributions. Although the mean-field assumption can bring us computational benefits, it is unable to give an accurate approximation. Meanwhile mean-field approximations usually find a uni-modal approximation which is inappropriate for multi modal distribution. It was later extended to mixtures of mean field distributions (Jaakkola and Jordan, 1998; Bishop et al., 1997), which models the posterior as a mixture of mean-field approximations.

Gershman et al. (2012) introduced non-parametric variational inference (NPVI), a black-box approach to variational inference that only requires the first and second derivatives of the log joint probability to be computable. NPVI is restricted to a simple approximating family, which uses GMMs with uniform weights and isotropic components and didn't learn a full Gaussian covariance. Theoretical when given unbounded number of components, NPVI can learn arbitrary distribution well, however, in general, NPVI is not suited for learning highly accurate approximations with a reasonable number of components as shown by Arenz et al. (2020).

Similar to our approaches, black box variational inference (BBVI, Ranganath et al. 2014) relies on function evaluations of the target distributions by sampling the variational posterior. In BBVI the ELBO is optimized based on a stochastic optimization where the noisy gradient is computed from Monte Carlo samples from the variational posterior. However, essential to its success are model-free variance reductions to reduce the variance of the the noisy gradients. It suggests to control the variance in two ways: Rao-Blackwellization and control-variate. In another black box variational inference method, Variational boosting, Miller et al. (2017) also used sample-based gradient to optimize the ELBO, the components used for approximation will be added iteratively and the unbiased estimates of the gradient can be obtained based on re-parameterization trick (Kingma and Welling, 2013), which we will also use for our embedded VIPS.

Nowadays the framework of normalizing flows are very popular. It was first discussed in Rezende and Mohamed (2015) in the context of stochastic variational inference. In Rezende and Mohamed (2015) two specific types of flows are introduced: planar flows and radial flows. These flows are shown to be effective to problems relatively with low-dimensional latent space. Based on normalizing flows, Inverse Autogressive Flows (IAF) was introduced by Kingma et al. (2016). In IAF, functions take as input a variable with some specified ordering such as multidimensional tensors, and output a mean and standard deviation for each element of the input variable conditioned on the previous elements. In the inverse autogressive transformation they have observed that this inverse transformation can be parallelized, and this inverse autoregressive operation has a simple Jacobian determinant, which enables IAF to scale to a very high dimensions. However, due to the gradient-based optimization, it may lacks sufficient exploration. Variational online-Newton method was introduced by Khan et al. (2018), which uses the Hessian matrix to do the Gaussian variational approximation, however the maximum likelihood estimation objective is non-convex, the Hessian matrix can be negative which can make the algorithm break down. Khan et al. (2018) use the Generalized-Gauss-Newton (GGN, Graves 2011; Martens 2014) approximation to avoid negative variance in VON update. The resulting algorithm is variational online Gauss-Newton (VOGN). Based on VOGN Mishkin et al. (2018) proposed the stochastic low-rank approximate natural-gradient (SLANG) method, which uses low rank plus diagonal matrix to approximate the precision matrix.

In practice, Markov-chain Monte-Carlo (MCMC, Andrieu et al. 2003) can always serve as an alternative of variational inference, especially when approximated model is not required. Based on MCMC, Neal (2003) introduced slice sampling, which can be easily implemented for univariate distributions and can be used to sample from a multivariate distribution by updating each variable in turn. However, such sampling process is very inefficient for higher dimensional random variables. To increase the efficiency of slice sampling, Murray et al. (2010) defined the slice by an ellipse and introduced the elliptical slice sampling method. However, Most of the MCMC methods lack the ability of exploring multimodal distribution. Some of the methods use multiple chain to better explore multimodal distribution (Neal, 1996; Earl and Deem, 2005; Nishihara et al., 2014). As shown by Arenz et al. (2020), the most promising among them is parallel tempering MCMC (PTMCMC Earl and Deem 2005). However, if the hyper-parameter of the PTMCMC is not well tuned (such as number of chains), PTMCMC can be inefficient. Stein variational gradient descent (SVGD) is a sampling method introduced by Liu and Wang (2016), which can be applied whenever gradient descent can be applied, however, VIPS has already outperformed SVGD as shown by Arenz et al. (2020).

Arenz et al. (2020) introduced VIPS for leaning full Gaussian covariance and showed in experiments that VIPS is suitable for learning highly accurate approximations of multi modal intractable distribution and Ewerton et al. (2020) applied it to telerobotics. However, learn the full Gaussian is computationally very heavy, hence VIPS is suitable for medium scale problem (below 60 dimensions). Becker et al. (2020) introduced in expectation information maximization (EIM) a novel approach of using the log ratio to learn the distribution which we can also use in our approaches but in EIM doesn't include adding components and reuse previous samples. To evaluate high dimensional log probability, it is not convenient to store the high dimensional covariance, Richardson and Weiss (2018) and Bishop (2006) showed an approach to evaluate the high dimensional sample log probability by using low dimensional data, which we can make use of in our approach embedded VIPS.

3.2. Reinforcement Learning

Our approaches share a lot of ideas with many policy search algorithms. In policy search we need to maximize the reward of a policy, while in variational settings we try to minimize the KL-divergence between model and target distribution. Both problems can be treated as optimization problem. REPS (Peters et al., 2010) was the first algorithm to introduce information-geometric policy update. Daniel et al. (2012) extend REPS to HiRPES to mutimodal problem, but HiREPS is unable to correctly optimize the entropy of mixture models.

Abdolmaleki et al. (2015) proposed Model-based relative entropy policy search (MORE). MORE uses the same idea to bound the policy update by introducing a KL constraint of the search distribution update but additionally add an entropy to prevent the search distribution from prematurely convergence. MORE uses a locally learned surrogate model to search the optimum and it can give us a closed form solution of Gaussian approximation. However, since in policy search the reward gradient information is usually difficult to get, MORE doesn't explore the gradient information, which we can make use of in variational inference. Akrour et al. (2018, 2016) extend MORE to Model-free Trajectory-based Policy Optimization with Monotonic Improvement (MOTO) to solve high dimensional continuous state and action space control problem. The state space is assume to be Gaussian distributed and action space is assumed to be conditional Gaussian distributed. MOTO shows a closed form solution of a conditional Gaussian update with guaranteed monotonic improvement. We can use the same idea as showed in MOTO to update the conditional Gaussian part in the factorized model.

To stored the samples and reuse them is also known as experience replay in reinforcement learning (Lin, 1992; Mnih et al., 2013), we will reuse samples based on sample base by using importance weighting.

Chapter 4.

VIPS for High Dimensional Cases

In this chapter we want to show in Section 4.1 how we can extend VIPS by incorporating the gradient information. Section 4.2 discusses how we can do the low rank approximation with a factorized GMM model and Section 4.3 discusses how we combine gradient information and low rank approximation.

4.1. Gradient based Variational Infernce by Policy Search

Same as VIPS, we want to approximate an unnormalized target distribution $\tilde{p}(\mathbf{x})$ using a Gaussian mixture model

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

Following VIPS (Arenz et al., 2020), we introduce an auxiliary distribution $\tilde{q}(o|\mathbf{x})$ to get a new lower bound $\tilde{\mathscr{L}}(q(o), q(\mathbf{z}|o), q(\mathbf{x}|\mathbf{z}, o), \tilde{q}(o|\mathbf{x}))$ on the ELBO,

$$\mathscr{L}_o(q(o), q(\mathbf{x}|o), \tilde{q}(o|\mathbf{x})) = \tilde{\mathscr{L}_o}(q(o), q(\mathbf{x}|o), \tilde{q}(o|\mathbf{x})) + \mathbb{E}_{q(\mathbf{x})} D_{\mathrm{KL}}\Big(q(o|\mathbf{x})||\tilde{q}(o|\mathbf{x}))\Big).$$

As in VIPS, we can maximize the ELBO by alternating between updating the lower bound with respect to weights $q(o)$ and the components $q(\mathbf{x}|o)$, where we set $\tilde{q}(o|\mathbf{x}) = q(o|\mathbf{x})$ before each update to tighten the bound. Following the same idea, we also alternate between component update and weight update in gradientVIPS. Section 4.1.1 shows us how we can include gradient information when we update the component. Section 4.1.2 and 4.1.3 show we can use the same idea as in VIPS to do the weight update, add and delete components in *gradientVIPS*. Algorithm 2 describes generally the work flow of *gradientVIPS*.

Algorithm 2 *gradientVIPS*

```
Require: number of components $N_0$
Require: number of iterations $N_i$
 1: for $i = 1, \cdots, N_i$ do
 2:     for $i = 1, \cdots, N_o$ do
 3:         sample from current distribution
 4:         rearrange the orders of target and gradient features and values
 5:         calculate model parameters using ordinary least square
 6:     end for
 7:     update component based on gradientMORE
 8:     for $i = 1, \cdots, N_o$ do
 9:         sample from current distribution
10:         estimate reward for each sample
11:     end for
12:     update weights using the closed form solution
13: end for
```

4.1.1. Update Components

As VIPS shows, for each component the objective we want to optimize is

$$\begin{aligned}
&\underset{q(\mathbf{x}|o))}{maximize} \int_{\mathbf{x}} q(\mathbf{x}|o)\Big(\log\tilde{p}(\mathbf{x}) + \log\tilde{q}(o|\mathbf{x})\Big)d\mathbf{x} + H(q(\mathbf{x}|o))\Big) \\
&s.t \qquad D_{\mathrm{KL}}(q(\mathbf{x}|o))||q^i(\mathbf{x}|o)) \leq \varepsilon,
\end{aligned}$$

where $q^i(\mathbf{x}|o)$ denotes the $q(\mathbf{x}|o)$ learned from last iteration. We want to construct a local quadratic model $R(\mathbf{x})$ to approximate the component specific reward $\log\tilde{p}(\mathbf{x}) + \log\tilde{q}(o|\mathbf{x})$ in the vicinity of the respective component $q(\mathbf{x}|o)$.

Surrogate Model Parameters of MORE

In this subsection we will review how to construct a locally quadratic model for MORE, which can be used for updating the search distribution for VIPS.

In MORE, we use the target information with the help of ordinary least squares to get the model parameters. We want to minimize the sum of squared errors of target and model based on the samples. We can write the loss function $L(\omega)$ in Matrix-vector form

$$\begin{aligned}\underset{\omega}{minimize} \quad L(\omega) &= (Y-\Phi\omega)^\top (Y-\Phi\omega) \\ &= Y^\top Y - 2\omega^\top \Phi^\top Y + \omega^\top \Phi^\top \Phi\omega,\end{aligned}$$

where ω denotes the model parameter, Φ denotes the feature matrix and Y denotes the stacked target value. By setting the partial derivative of loss function $L(\omega)$ with respect to parameter ω to zero

$$\frac{\partial L(\omega)}{\partial \omega} = -2\Phi^\top Y + 2\Phi^\top \Phi\omega = 0,$$

we can get the optimal parameter

$$\omega = (\Phi^\top \Phi)^{-1}\Phi^\top Y.$$

Surrogate Model Parameters of gradientMORE

In gradientMORE, we want to incorporate the gradient information together with target information in model fitting. We also use the ordinary least squares to get the optimal parameters. Now the loss function has an additional part which contains the squared error of the gradient part. When taking the regularization part into account, the loss function can be framed as

$$\begin{aligned}\underset{\omega}{minimize} \quad L(\omega) &= \underbrace{(Y-\Phi\omega)^\top (Y-\Phi\omega)}_{Target\ part} + \underbrace{(Y^{'}-\Phi^{'}\omega)^\top (Y^{'}-\Phi^{'}\omega)}_{Gradient\ part} + \underbrace{\lambda\omega^\top\omega}_{Regularization\ part} \\ &= Y^\top Y - 2\omega^\top \Phi^\top Y + \omega^\top \Phi^\top \Phi\omega + Y^{'\top} Y^{'} - 2\omega^\top \Phi^{'\top} Y^{'} + \omega^\top \Phi^{'\top}\Phi^{'}\omega + \lambda\omega^\top\omega\end{aligned}$$

we can get the optimal parameters by using the same idea as before by setting the partial derivative to zero

$$\frac{\partial L(\omega)}{\partial \omega} = -2\Phi^\top Y + 2\Phi^\top \Phi\omega - 2\Phi^{'} Y^{'\top} + 2\Phi^{'\top}\Phi^{'}\omega + 2\lambda\omega = 0.$$

From these equations we can get

$$\begin{aligned}\Phi^\top\Phi\omega + \Phi^{'\top}\Phi^{'}\omega &= \Phi^\top Y + \Phi^{'\top} Y^{'} \\ (\Phi^\top\Phi + \Phi^{'\top}\Phi^{'} + \lambda \boldsymbol{I})\omega &= \Phi^\top Y + \Phi^{'\top} Y^{'}.\end{aligned}$$

We stack Φ and $\Phi^{'}$ and denote $\begin{pmatrix}\Phi \\ \Phi^{'}\end{pmatrix}$ as X and $\begin{pmatrix}Y \\ Y^{'}\end{pmatrix}$ as Y, we can get

$$\left(\begin{pmatrix}\Phi^\top & \Phi^{'\top}\end{pmatrix}\begin{pmatrix}\Phi \\ \Phi^{'}\end{pmatrix} + \lambda \boldsymbol{I}\right)\omega = \begin{pmatrix}\Phi^\top & \Phi^{'\top}\end{pmatrix}\begin{pmatrix}Y \\ Y^{'}\end{pmatrix}$$

which is

$$(X^\top X + \lambda \boldsymbol{I})\omega = XY.$$

Since $X^\top X$ is quadratic, we can get the optimal parameter $\omega = (X^\top X + \lambda \boldsymbol{I})^{-1} X^\top Y.$ where X stands for the stacked features and Y stands for the stacked target and gradient value. In practice, we need importance weighting to do sample reuse in order to reduce the number of function evaluations. For samples with importance weights the objective can be written as

$$\begin{aligned}\underset{\omega}{minimize}\, L &= (Y - \Phi\omega)^\top \mathbf{W}_o (Y - \Phi\omega) + (Y^{'} - \Phi^{'}\omega)^\top \mathbf{W}_1 (Y^{'} - \Phi^{'}\omega) + \lambda\omega^\top\omega \\ &= Y^\top \mathbf{W}_o Y - 2\omega^\top\Phi^\top \mathbf{W}_o Y + \omega^\top\Phi^\top \mathbf{W}_o \Phi\omega + Y^{'\top}\mathbf{W}_1 Y^{'} - 2\omega^\top\Phi^{'\top}\mathbf{W}_1 Y^{'} \\ &+ \omega^\top\Phi^{'\top}\mathbf{W}_1\Phi^{'}\omega + \lambda\omega^\top\omega.\end{aligned}$$

Different from before, now we have a diagonal weight matrix W_o for the target value and a diagonal weight matrix W_1 for the gradient value. For a three dimensional problem, the weights matrix of gradient has the form

$$\mathbf{W}_1 = \begin{pmatrix}\mathbf{W}_o & 0 & 0 \\ 0 & \mathbf{W}_o & 0 \\ 0 & 0 & \mathbf{W}_o\end{pmatrix}.$$

Same as before, by setting the partial derivative to zero, we can get the equations

$$\frac{\partial L}{\partial \omega} = -2\Phi^\top \mathbf{W}_o Y + 2\Phi^\top \mathbf{W}_o \Phi\omega - 2\Phi^{'\top}\mathbf{W}_1 Y^{'} + 2\Phi^{'\top}\mathbf{W}_1\Phi^{'}\omega + 2\lambda\omega = 0,$$

which leads to a equation that can be written as

$$(\Phi^\top \mathbf{W}_o \Phi + \Phi^{'\top}\mathbf{W}_1\Phi^{'} + \lambda \boldsymbol{I})\omega = \Phi^\top \mathbf{W}_o Y + \Phi^{'\top}\mathbf{W}_1 Y^{'}.$$

Same as before, to get a closed form solution of ω , we need to stack the features and denote $\begin{pmatrix}\Phi \\ \Phi^{'}\end{pmatrix}$ as X. Weights are stacked as $\begin{pmatrix}\mathbf{W}_o & 0 \\ 0 & \mathbf{W}_1\end{pmatrix}$ and we denote the weight matrix as W. Values are stacked as $\begin{pmatrix}Y \\ Y^{'}\end{pmatrix}$ and we denote the values as Y. The equation can be rewritten as

$$\left(\begin{pmatrix}\Phi^\top & \Phi^{'\top}\end{pmatrix}\begin{pmatrix}\mathbf{W}_o & 0 \\ 0 & \mathbf{W}_1\end{pmatrix}\begin{pmatrix}\Phi \\ \Phi^{'}\end{pmatrix} + \lambda \boldsymbol{I}\right)\omega = \begin{pmatrix}\Phi^\top & \Phi^{'\top}\end{pmatrix}\begin{pmatrix}\mathbf{W}_o & 0 \\ 0 & \mathbf{W}_1\end{pmatrix}\begin{pmatrix}Y \\ Y^{'}\end{pmatrix}$$

which is equivalent to

$$(X^\top WX+\lambda \boldsymbol{I})\omega = X^\top WY.$$

We can get the optimal parameter of $\omega = (X^\top \mathbf{W}X + \lambda \boldsymbol{I})^{-1} X^\top \mathbf{W}\boldsymbol{Y}$, where X stands for the stacked features. Y stands for stacked target and gradient value. $\mathbf{W}$ stands for the weights for target and gradients. Please note that we have rearranged the orders of the gradients which we will talk about in detail in the next subsection.

Feature constructing for gradientMORE

The quadratic model can be written as $R(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{x}^\top \mathbf{b} + c$. We assume $\mathbf{A}$ is a symmetric matrix, so the number of parameters we need to learn is $\left(\frac{n*(n+1)}{2}+n+1\right)$ for a n-dimensional problem. The partial derivative of $R(\mathbf{x})$ with respect to $\mathbf{x}$ is

$$\frac{\partial R(\mathbf{x})}{\partial \mathbf{x}} = -\mathbf{A}^\top \mathbf{x} + \mathbf{b}.$$

For a three dimensional problem, the gradient of model can be written in detail as

$$\begin{aligned}
\frac{\partial R(\mathbf{x})}{\partial \mathbf{x}} &= -\mathbf{A}^\top \mathbf{x} + \mathbf{b} \\
&= \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{12} & A_{22} & A_{23} \\ A_{13} & A_{23} & A_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} \\
&= \begin{pmatrix} A_{11}x_1 + A_{12}x_2 + A_{13}x_3 + b_1 \\ A_{12}x_1 + A_{22}x_2 + A_{23}x_3 + b_2 \\ A_{13}x_1 + A_{23}x_2 + A_{33}x_3 + b_3 \end{pmatrix}.
\end{aligned}$$

The target value of one sample can be rewritten in matrix multiplication form as

$$\begin{aligned}
y &\approx -\frac{1}{2}\mathbf{A}_{11}x_1^2 - \mathbf{A}_{12}x_1x_2 - \mathbf{A}_{13}x_1x_3 - \frac{1}{2}\mathbf{A}_{22}x_2^2 - \mathbf{A}_{23}x_2x_3 - \frac{1}{2}\mathbf{A}_{33}x_3^2 + x_1b_1 + x_2b_2 + x_3b_3 + c \\
&= \underbrace{\begin{pmatrix} -\frac{1}{2}x_1^2 & -x_1x_2 & -x_1x_3 & -\frac{1}{2}x_2^2 & -x_2x_3 & -\frac{1}{2}x_3^2 & x_1 & x_2 & x_3 & 1 \end{pmatrix}}_{target\ features} \begin{pmatrix} A_{11} \\ A_{12} \\ A_{13} \\ A_{22} \\ A_{23} \\ A_{33} \\ b_1 \\ b_2 \\ b_3 \\ c \end{pmatrix}.
\end{aligned}$$

The gradient of each sample is

$$\begin{pmatrix} \frac{\partial y}{\partial x_1} \\ \frac{\partial y}{\partial x_2} \\ \frac{\partial y}{\partial x_3} \end{pmatrix} = \frac{\partial y}{\partial \mathbf{x}} \approx \frac{\partial R(\mathbf{x})}{\partial \mathbf{x}} = \begin{pmatrix} A_{11}x_1 + A_{12}x_2 + A_{13}x_3 + b_1 \\ A_{12}x_1 + A_{22}x_2 + A_{23}x_3 + b_2 \\ A_{13}x_1 + A_{23}x_2 + A_{33}x_3 + b_3 \end{pmatrix}.$$

Since we want to use target value together with gradient value to do the ordinary least squares, we use the same features as target, fill the part which has no entry corresponds to the model parameters with zeros. The gradient equations can be reformed as the feature multiply model parameters,

$$\begin{pmatrix} \frac{\partial y}{\partial x_1} \\ \frac{\partial y}{\partial x_2} \\ \frac{\partial y}{\partial x_3} \end{pmatrix} \approx \underbrace{\begin{pmatrix} x_1 & x_2 & x_3 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & x_1 & 0 & x_2 & x_3 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & x_1 & 0 & x_2 & x_3 & 0 & 0 & 1 & 0 \end{pmatrix}}_{gradient\ features} \begin{pmatrix} A_{11} \\ A_{12} \\ A_{13} \\ A_{22} \\ A_{23} \\ A_{33} \\ b_1 \\ b_2 \\ b_3 \\ c \end{pmatrix}.$$

Now, we want to show how to stack the features. For example, if we have two samples, the features should be stacked as the first two rows are target features, the next two rows are features of the first dimension gradient features then the second dimension gradient features and so on. After stacking all the features and target features, the ordinary least squares equations can be written as

$$\begin{pmatrix} y^a \\ y^b \\ \frac{\partial y^a}{\partial x_1} \\ \frac{\partial y^b}{\partial x_1} \\ \frac{\partial y^a}{\partial x_2} \\ \frac{\partial y^b}{\partial x_2} \\ \frac{\partial y^a}{\partial x_3} \\ \frac{\partial y^b}{\partial x_3} \end{pmatrix} \approx \underbrace{\begin{pmatrix} -\frac{1}{2}(x_1^a)^2 & -x_1^a x_2^a & -x_1^a x_3^a & -\frac{1}{2}(x_2^a)^2 & -x_2^a x_3^a & -\frac{1}{2}(x_3^a)^2 & x_1^a & x_2^a & x_3^a & 1 \\ -\frac{1}{2}(x_1^b)^2 & -x_1^b x_2^b & -x_1^b x_3^b & -\frac{1}{2}(x_2^b)^2 & -x_2^b x_3^b & -\frac{1}{2}(x_3^b)^2 & x_1^b & x_2^b & x_3^b & 1 \\ x_1^a & x_2^a & x_3^a & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ x_1^b & x_2^b & x_3^b & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & x_1^a & 0 & x_2^a & x_3^a & 0 & 0 & 1 & 0 & 0 \\ 0 & x_1^b & 0 & x_2^b & x_3^b & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & x_1^a & 0 & x_2^a & x_3^a & 0 & 0 & 1 & 0 \\ 0 & 0 & x_1^b & 0 & x_2^b & x_3^b & 0 & 0 & 1 & 0 \end{pmatrix}}_{feature\ matrix} \begin{pmatrix} A_{11} \\ A_{12} \\ A_{13} \\ A_{22} \\ A_{23} \\ A_{33} \\ b_1 \\ b_2 \\ b_3 \\ c \end{pmatrix},$$

where $(x_1^a, x_2^a, x_3^a)^\top$ and $(x_1^b, x_2^b, x_3^b)^\top$ denote the two samples, y^a and y^b denote the corresponding target values. Please note that, the target values and gradient values should be rearranged according to its corresponding features. Figure 4.1 shows us the way to stack features and values for higher dimensional problem. In practice, we always preprocess data by using whitening

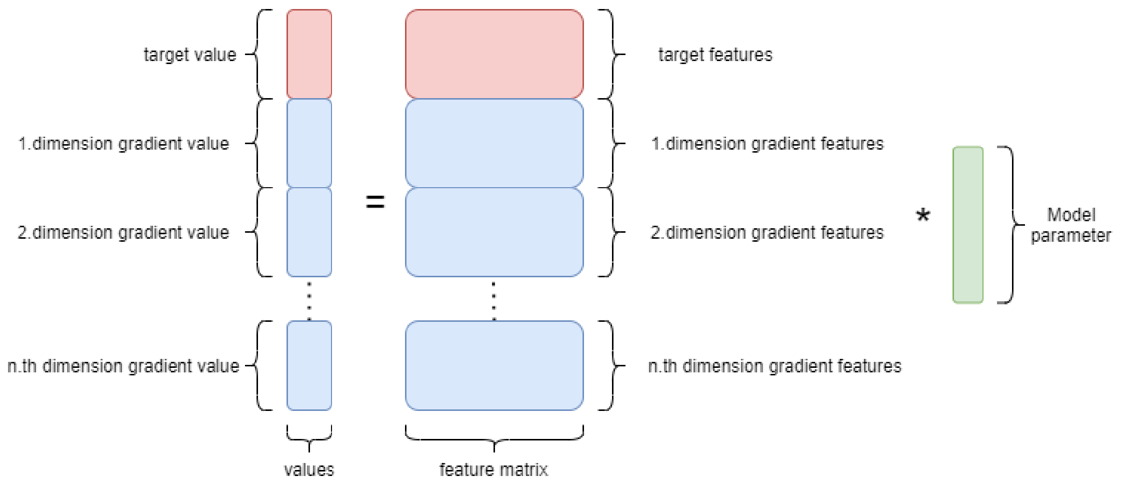

target value
1.dimension gradient value
2.dimension gradient value
n.th dimension gradient value
values
=
feature matrix
target features
1.dimension gradient features
2.dimension gradient features
n.th dimension gradient features
*
Model parameter

Figure 4.1.: Examples on how to stack features and values for higher dimensional case.

technique. When whitening is applied to data, we should also change the gradient values to make it compatible to the data after whitening (as discussed in Section 2.9). We will then use ordinary least squares to get the parameters of the quadratic model as equation shows

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

After getting the surrogate model, we can use the modified MORE to update the component. The updated Gaussian with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ are given by

$$\boldsymbol{\Sigma} = \left(\frac{\eta}{\eta+1}\boldsymbol{\Sigma}_i^{-1} + \frac{1}{\eta+1}\boldsymbol{A}_i\right)^{-1}, \quad \boldsymbol{\mu} = \boldsymbol{\Sigma}\left(\frac{\eta}{\eta+1}\boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i + \frac{1}{\eta+1}\boldsymbol{b}_i\right),$$

where η is the optimal Lagrangian multipliers corresponding to KL constraint, which can be found efficiently by minimizing the convex Lagrangian dual function.

4.1.2. Update Weights

We will use the same idea as used in VIPS to update the weights. With the objective of weight update,

$$\sum_o q(o)R(o) + H(q(o)),$$

where $R(o)$ denotes the objective of the component update,

$$R(o) = \int_{\mathbf{x}} q(\mathbf{x}|o)\Big(\log \tilde{p}(\mathbf{x}) + \log \tilde{q}(o|\mathbf{x})\Big) d\mathbf{x} + H(q(\mathbf{x}|o)).$$

Instead of constructing a quadratic model, we use Monte-Carlo estimation to approximate the reward $R(o)$,

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

The optimal solution can be obtained in closed form as

$$q(o) = \frac{exp(\tilde{R}(o))}{\sum_o exp(\tilde{R}(o))}.$$

4.1.3. Add and Delete Heuristic

For gradient VIPS, we will use the same idea as in VIPS showed to add and delete components. We will add components in promising area and delete components which have very small weights and are unable to improve. The covariance of the components will be set to isotropic and the weight of the newly added component will be always set to a very small value. For details please refer to our reference VIPS (Arenz et al., 2020).

4.2. Embedded VIPS

In this approach, we do a low rank approximation of the full Gaussian by using a factorized Gaussian mixture model to reduce the number of parameters. We want to approximate an unnormalized target distribution $\tilde{p}(\mathbf{x})$ using a Gaussian mixture model

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

where each component $q(\mathbf{x}|o)$ factors into a lower-dimensional Gaussian latent distribution

$$q(\mathbf{z}|o) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_o, \boldsymbol{\Sigma}_o)$$

and a conditional Gaussian decoder (with isotropic or diagonal covariance)

$$q(\mathbf{x}|\mathbf{z}, o) = \mathcal{N}(\mathbf{x}|F_o\mathbf{z} + c_o, \boldsymbol{\Sigma}_{\mathbf{x}|o,\mathbf{z}}),$$

that is

$$q(\mathbf{x}) = \sum_o q(o) \int_{\mathbf{z}} q(\mathbf{z}|o) q(\mathbf{x}|\mathbf{z}, o) d\mathbf{z}.$$

We want to minimize the Kullback-Leibler (KL) divergence to the target distribution $\tilde{p}(\mathbf{x})$ by maximizing the ELBO

$$\mathscr{L}(q(o), q(\mathbf{z}|o), q(\mathbf{x}|\mathbf{z}, o)) = \int_{\mathbf{x}} q(\mathbf{x}) \Big(\log \tilde{p}(\mathbf{x}) - \log q(\mathbf{x}) \Big) d\mathbf{x},$$

with respect to the weights $q(o)$, latent components $q(\mathbf{z}|o)$ and decoder $q(\mathbf{x}|\mathbf{z}, o)$. Following VIPS (Arenz et al., 2020), we introduce an auxiliary distribution $\tilde{q}(o|\mathbf{x})$ to derive a lower bound $\tilde{\mathscr{L}}(q(o), q(\mathbf{z}|o), q(\mathbf{x}|\mathbf{z}, o), \tilde{q}(o|\mathbf{x}))$ on the ELBO,

$$\begin{aligned}
\mathscr{L}(q(o), q(\mathbf{z}|o), q(\mathbf{x}|\mathbf{z}, o)) =& \sum_o q(o) \int_{\mathbf{z}} q(\mathbf{z}|o) \int_{\mathbf{x}} q(\mathbf{x}|\mathbf{z}, o) \Big(\log \tilde{p}(\mathbf{x}) - \log \frac{q(o)q(\mathbf{x}|o)\tilde{q}(o|\mathbf{x})}{q(o|\mathbf{x})\tilde{q}(o|\mathbf{x})} \Big) d\mathbf{x} \\
=& \sum_o q(o) \Big(\int_{\mathbf{z}} q(\mathbf{z}|o) \int_{\mathbf{x}} q(\mathbf{x}|\mathbf{z}, o) \Big(\log \tilde{p}(\mathbf{x}) + \log \tilde{q}(o|\mathbf{x}) \Big) d\mathbf{x} \\
& + H(q(\mathbf{x}|o)) \Big) + H(q(o)) + \mathbb{E}_{q(\mathbf{x})} D_{\mathrm{KL}} \Big(q(o|\mathbf{x}) || \tilde{q}(o|\mathbf{x})) \Big) \\
=& \tilde{\mathscr{L}}(q(o), q(\mathbf{z}|o), q(\mathbf{x}|\mathbf{z}, o), \tilde{q}(o|\mathbf{x})) + \mathbb{E}_{q(\mathbf{x})} D_{\mathrm{KL}} \Big(q(o|\mathbf{x}) || \tilde{q}(o|\mathbf{x})) \Big).
\end{aligned}$$

As in VIPS we can maximize the ELBO by alternating between updating the lower bound with respect to weights the $q(o)$ and the components $q(\mathbf{x}|o)$, where we set $\tilde{q}(o|\mathbf{x}) = q(o|\mathbf{x})$ before each update to tighten the bound.

4.2.1. Update Components

The components $q(\mathbf{x}|o)$ can be updated independently from each other as shown by Arenz et al. (2020). For each component we need to update the latent component $q(\mathbf{z}|o)$ and the decoder $q(\mathbf{x}|\mathbf{z}, o)$ to improve on the objective

$$\mathscr{L}_o(q(\mathbf{z}|o), q(\mathbf{x}|\mathbf{z}, o), \tilde{q}(o|\mathbf{x})) = \int_{\mathbf{z}} q(\mathbf{z}|o) \int_{\mathbf{x}} q(\mathbf{x}|\mathbf{z}, o) \Big(r_o(\mathbf{x}) - \log q(\mathbf{x}|o) \Big),$$

where $r_o = \log \tilde{p}(\mathbf{x}) + \log \tilde{q}(o|\mathbf{x})$ is a component-specific reward function. We can use the same trick as before and introduce an auxiliary distribution $\tilde{q}(z|\mathbf{x}, o)$ to derive a lower bound

$$\begin{aligned}
\tilde{\mathscr{L}}_o(q(\mathbf{z}|o), q(\mathbf{x}|\mathbf{z}, o), \tilde{q}(o|\mathbf{x}), \tilde{q}(z|\mathbf{x}, o)) =& \sum_o q(o) \Big(\int_{\mathbf{z}} q(\mathbf{z}|o) \Big[\int_{\mathbf{x}} q(\mathbf{x}|\mathbf{z}, o) \Big(r_o(\mathbf{x}) + \log \tilde{q}(z|\mathbf{x}, o) \Big) d\mathbf{x} \\
& + H(q(\mathbf{x}|\mathbf{z}, o)) \Big] d\mathbf{z} \ + H(q(\mathbf{z}|o)) \Big) + H(q(o)).
\end{aligned}$$

Following the same scheme as in VIPS, we first maximize the lower bound with respect to the latent component $q(\mathbf{z}|o)$, recompute the responsibilites and finally maximize with respect to the decoder $q(\mathbf{x}|\mathbf{z},o)$.

Update Latent Component

Updating the latent components can be performed like in VIPS by applying MORE based on a quadratic surrogate of

$$r_o(\mathbf{z}) = \int_{\mathbf{x}} q(\mathbf{x}|\mathbf{z},o)\Big(\tilde{r}_o(\mathbf{x}) + \log \tilde{q}(z|\mathbf{x},o)\Big) d\mathbf{x}.$$

We can not learn a model $\tilde{r}_o(\mathbf{x})$ to obtain a quadratic approximation of $\tilde{r}_o(\mathbf{z})$ as this would beat the purpose of the approach. Instead, we obtain unbiased estimates of $r_o(\mathbf{z})$ with respect to $\mathbf{z}$ based on samples from the decoder $q(\mathbf{x}|\mathbf{z},o)$. For each latent sample z, the reward can be obtained based on Monte-Carlo estimation

$$\tilde{R}(o) = \frac{1}{N_o} \sum_{n=1}^{N_o} q(\mathbf{x}|\mathbf{z},o)\Big(\tilde{r}_o(\mathbf{x}) + \log \tilde{q}(z|\mathbf{x},o)\Big).$$

Mean and covariance matrix of $\tilde{q}(z|\mathbf{x},o)$ are

$$\widetilde{\boldsymbol{\mu}_{\mathbf{z}|\mathbf{x},o}} = \boldsymbol{\mu}_{\mathbf{z}|o} + \boldsymbol{\Sigma}_{\mathbf{z}|o}\mathbf{F}_o^\top \boldsymbol{\Sigma}_{\mathbf{x}|o}^{-1}\Big(\mathbf{x} - c_o - F_o \boldsymbol{\mu}_{z|o}\Big)$$

and

$$\widetilde{\boldsymbol{\Sigma}_{\mathbf{z}|\mathbf{x},o}} = \Big(\boldsymbol{\Sigma}_{\mathbf{z}|o}^{-1} + \mathbf{F}_o^\top \boldsymbol{\Sigma}_{\mathbf{x}|\mathbf{z},o}^{-1}\mathbf{F}_o\Big)^{-1}.$$

Update Decoder Components

For updating the decoder $q(\mathbf{x}|\mathbf{z},o)$, we need to maximize the objective

$$\tilde{\mathscr{L}}_o(q(\mathbf{x}|\mathbf{z},o)) = \int_{\mathbf{x}} q(\mathbf{x}|\mathbf{z},o)\Big(r_o(\mathbf{x}) + \log \tilde{q}(z|\mathbf{x},o)\Big) d\mathbf{x} \; + H(q(\mathbf{x}|\mathbf{z},o)).$$

Since the decoder component is conditional Gaussian. We will use the idea of MOTO (Akrour et al., 2018) to update the conditional Gaussian component. In the next subsection, we will talk about how to adapt MOTO to embedded VIPS.

Adapt MOTO to Embedded VIPS.

In our setting, latent corresponds to the state distribution and decoder corresponds to policy distribution. The decoder component update of the embedded VIPS is based on MOTO. We firstly construct a joint quadratic function of **x** and **z**. After constructing the quadratic model, the following is the derivation of decoder update of embedded VIPS

$$\begin{aligned}\underset{q(\mathbf{x}|\mathbf{z},o)}{maximize} \int q(\mathbf{z}|o)\Big(\int q(\mathbf{x}|\mathbf{z},o)R^i(x,z)dx + H(q(\mathbf{x}|\mathbf{z},o))\Big)dz \\ s.t \qquad \mathbb{E}_{q(\mathbf{z}|o)}\left[D_{\mathrm{KL}}(q(\mathbf{x}|\mathbf{z},o)||q(\mathbf{x}|\mathbf{z},o)^i)\right] \le \varepsilon \\ \int q(\mathbf{x}|\mathbf{z},o)dx = 1.\end{aligned}$$

In embedded VIPS, the $q(\mathbf{z}|o)$ corresponds to the $\rho(s)$ and $q(\mathbf{x}|\mathbf{z},o)$ corresponds to the $\pi(s,a)$in MOTO. The constraint of the entropy part enters the objective so we can substitute ω to 1 in embedded VIPS. The dual function is

$$\begin{aligned}L(q(\mathbf{x}|\mathbf{z},o),\eta,\lambda) &= \int q(\mathbf{z}|o)\Big(\int q(\mathbf{x}|\mathbf{z},o)Rdx + H(q(\mathbf{x}|\mathbf{z},o))\Big)dz \\ &+ \eta\left[\varepsilon - \int q(\mathbf{z}|o)D_{\mathrm{KL}}(q(\mathbf{x}|\mathbf{z},o)||q(\mathbf{x}|\mathbf{z},o)^i)\right] + \int \lambda(\mathbf{z})\left[\int q(\mathbf{x}|\mathbf{z},o) - 1\right].\end{aligned}$$

For good understanding, the parameters of the distributions will be omitted and use q to denote $q(\mathbf{z}|o)$ and we use p to denote $q(\mathbf{x}|\mathbf{z},o)$ and use p_1 to denote the learned $q(\mathbf{x}|\mathbf{z},o)$ from the last iteration. The Lagrange function is

$$\begin{aligned}L(p,\eta,\lambda) &= \int q\Big(\int pRdx + H(p)\Big)dz + \eta\left[\varepsilon - \int qD_{\mathrm{KL}}(p||p_1)\right] + \int \lambda\left[\int p - 1\right] \\ &= \int q\int p\Big(R - (\eta+1)\log p + \eta\log p_1\Big) + \eta\varepsilon + \int\lambda\int p - \int\lambda.\end{aligned}$$

with optimal $q(\mathbf{x}|\mathbf{z},o)$ (which is p in our notation) the partial derivative with respect to p should be zero.

$$\frac{\partial L}{\partial p} = qR - (\eta+1)[q\log p + q] + \eta q\log p_1 + \lambda = 0.$$

The optimal solution of the p^* can be obtained in closed form

$$q(\mathbf{x}|\mathbf{z},o) \propto q^i(\mathbf{x}|\mathbf{z},o)^{(\eta^*/(\eta^*+1))}exp(\frac{R(x,z)}{\eta^*+1}).$$

The partial derivative of dual function with respect to η can be got

$$\frac{\partial G}{\partial \eta} = \varepsilon - \mathbb{E}\left[D_{\mathrm{KL}}(p^*||p_1)\right].$$

We can get the optimal η^* efficiently using L-BFGS with respect to the dual function. By using the former learned quadratic function

$$R(x,z) = \frac{1}{2}\mathbf{x}^{\mathrm{T}} R_{xx}\mathbf{x} + \mathbf{x}^{\mathrm{T}} R_{xz}\mathbf{z} + \mathbf{x}^{\mathrm{T}} r_x + r(\mathbf{z}).$$

We can get the newly learned decoder

$$q(\mathbf{x}|\mathbf{z},o) = N(\mathbf{x}|FL\mathbf{z} + Ff, F(\eta^* + 1)),$$

the parameters are listed as

$$F = (\eta^*\Sigma_i^{-1} - R_{xx})^{-1}, \qquad L = \eta^*\Sigma_i^{-1}F_iL_i + R_{xz}, \qquad f = \eta^*\Sigma_i^{-1}F_if_i + r_x,$$

where F_i, L_i, f_i and Σ_i^{-1} denotes the parameters of $q(\mathbf{x}|\mathbf{z},o)$ learned from last iteration.

4.2.2. Update Weights

The weight update can be performed in the same way as in VIPS. The responsibilities are computed based on the last weights and the mean and covariance matrix for component o, which are given by $\boldsymbol{\mu}_{x|o} = F_o\boldsymbol{\mu}_{z|o} + c_o$ and $\boldsymbol{\Sigma}_{x|o} = \boldsymbol{\Sigma}_{\mathbf{x}|o,\mathbf{z}} + F_o\boldsymbol{\Sigma}_{\mathbf{z}|o}F_o^{\top}$.

4.2.3. Add and Delete Heuristic

For embedded VIPS, it is also important to dynamically add and delete components since we are learning a multi-modal unknown distribution. The add heuristic in embedded VIPS is more sophisticate than VIPS. In VIPS we treat every sample in the sample base as candidate for the initial mean of the new component. In embedded VIPS we treat every sample in the sample base as candidate for the initial mean of the decoder component. However, since in embedded VIPS we have the factorized structure, which means we need not only initialize the mean of the decoder, but also the gain matrix and the latent component mean. For the latent initialization, we will use the initial gain matrix to project the mean to latent space. For the gain matrix we will use the idea of probabilistic principle component analysis (PPCA) to do gain matrix initialization.

Probabilistic PCA for Gain Matrix Initialization

We firstly sample isotropiclly in the target space, get the target samples and use PPCA to get the gain matrix. In the E step, we use old parameter to calculate

$$\mathbb{E}[\mathbf{z}_n] = \mathbf{M}^{-1}\mathbf{W}^{\mathrm{T}}(\mathbf{x}_n - \overline{\mathbf{x}}), \qquad \mathbb{E}\left[\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}\right] = \sigma^2\mathbf{M}^{-1} + \mathbb{E}[\mathbf{z}_n]\,\mathbb{E}[\mathbf{z}_n]^{\mathrm{T}}.$$

In the M step, we will update the corresponding parameters

$$\begin{aligned}\mathbf{W}_{\text{new}} &= \left[\sum_{n=1}^{N} (\mathbf{x}_n - \overline{\mathbf{x}}) \mathbb{E}[\mathbf{z}_n]^{\mathrm{T}}\right] \left[\sum_{n=1}^{N} \mathbb{E}\left[\mathbf{z}_n \mathbf{z}_n^{\mathrm{T}}\right]\right]^{-1} \\ \sigma_{\text{new}}^2 &= \frac{1}{ND} \sum_{n=1}^{N} \left\{ \|\mathbf{x}_n - \overline{\mathbf{x}}\|^2 - 2\mathbb{E}[\mathbf{z}_n]^{\mathrm{T}} \mathbf{W}_{\text{new}}^{\mathrm{T}} (\mathbf{x}_n - \overline{\mathbf{x}}) \right. \\ &\quad \left. + \operatorname{Tr}\left(\mathbb{E}\left[\mathbf{z}_n \mathbf{z}_n^{\mathrm{T}}\right] \mathbf{W}_{\text{new}}^{\mathrm{T}} \mathbf{W}_{\text{new}}\right) \right\}.\end{aligned}$$

4.3. Gradient based VIPS for High Dimensional Space

In this section we want to combine the benefits of including the gradient information and doing the low rank approximation of a full Gaussian covariance.

4.3.1. Update Latent Components

Different from embedded VIPS, in this approach we will include the gradient information in the latent update, we can not directly stack the gradient information because now the sample space has a different dimension compared to the latent space. Instead, we will employ the reparameterization trick (Kingma and Welling, 2013) to turn reward

$$r_o(\mathbf{z}) = \int_{\mathbf{x}} q(\mathbf{x}|\mathbf{z}, o) \Big(\tilde{r}_o(\mathbf{x}) + \log \tilde{q}(z|\mathbf{x}, o) \Big) d\mathbf{x},$$

into

$$r_o(\mathbf{z}) = \sum_{\varepsilon} \Big(r_o(\mathbf{x}, \varepsilon) + \log q(z \mid \varepsilon, o) \Big),$$

with

$$\varepsilon \sim \mathcal{N}(0, 1).$$

After getting the reward and gradient value of latent samples we will use gradientVIPS to update latent component.

4.3.2. Update Decoder Components

For decoder update we will include the gradient information, the decoder update is also based on modified MOTO, but now since we assume the covariance of decoder is diagonal, each dimension of the decoder can be updated independently.

Chapter 5.

Experiments

In this chapter we will evaluate gradientVIPS in several experiments and compare gradientVIPS with a variety of state-of-the-art variational inference methods and Markov-chain Monte-Carlo method. We will discuss the gradientVIPS experiments in detail in section 5.1.1. Section 5.1.2 is the ablation of gradientVIPS, where we check some choices that can affect the performance of the algorithm. The results and discussion of gradientVIPS can be found in section 5.1.3 and 5.1.4. For Embedded VIPS, due to the time reason we evaluate the identity mapping and a fixed embedding which can be found in section 5.2.

5.1. GradientVIPS

For gradientVIPS we have done three groups of experiments, namely logistic regression, GMM, and planar robot. In the three groups of experiments we compare different methods with respect to sample efficiency (number of function evaluations) and sample quality which we access by computing the Maximum Mean Discrepancy (MMD) between samples and ground-truth samples. We use two criteria to evaluate the quality of the learned model. The first criterion is the Maximum Mean Discrepency (MMD, Gretton et al. 2012). The MMD is a non-parametric divergence between mean embeddings in a reproducible kernel Hilbert space (Gretton et al., 2012). When two distributions are similar, the MMD value calculated by the sample sets from these two distributions should be very small. The second criterion we use is evidence lower bound (ELBO), which is formulated as

$$\begin{aligned}&\arg\min_{q(\boldsymbol{x})} D_{\mathrm{KL}}(q(\boldsymbol{x})||p(\boldsymbol{x})) = \arg\min_{q(\boldsymbol{x})} \int_{\boldsymbol{x}} q(\boldsymbol{x}) \log \frac{q(\boldsymbol{x})}{p(\boldsymbol{x})} d\boldsymbol{x} \\ &= \arg\min_{q(\boldsymbol{x})} \int_{\boldsymbol{x}} q(\boldsymbol{x}) \log \frac{q(\boldsymbol{x})}{\tilde{p}(\boldsymbol{x})} d\boldsymbol{x} + const = \arg\max_{q(\boldsymbol{x})} \underbrace{\int_{\boldsymbol{x}} q(\boldsymbol{x}) \log \tilde{p}(\boldsymbol{x}) d\boldsymbol{x} + H(q(\boldsymbol{x}))}_{ELBO}.\end{aligned}$$

For normalized target densities (e.g. in the GMM experiments), if all modes are found and accurately approximated, the ELBO value can increase to zero or a very small value. For a non-Gaussian distribution, even if all the modes are found, the ELBO can not increase to zero. Therefore, for non-Gaussian distributed target distribution expriments (logistic regression and planar robot), we will compare the relative ELBO between different methods.

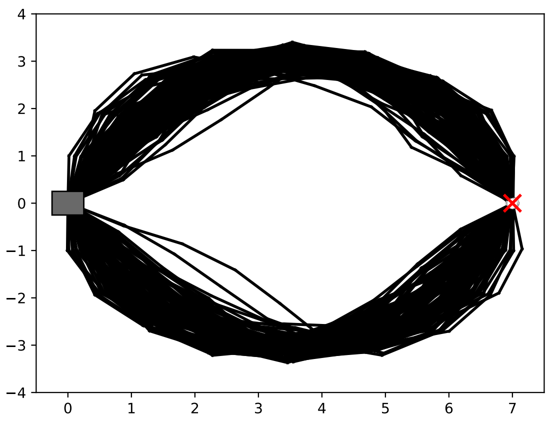
5.1.1. Experiments

Logistic Regression

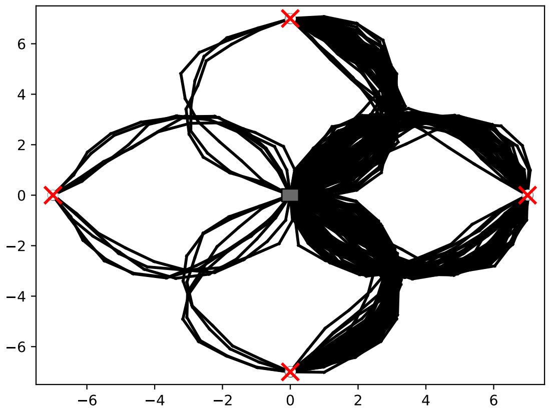
In logistic regression, we perform two experiments namely German credit and Breast cancer that have been taken from Arenz et al. (2020). In these two experiments we want to approximate the posterior for a classification problem. The breast cancer data set (Lichman et al., 2013) is 31 dimensional and has 569 data points. The German credit data set is 25 dimensional and has 1000 data points. Under the same conditions as in VIPS experiments, we standardize both data sets and perform linear logistic regression where we put zero-mean Gaussian priors with variance 100 on all parameters (Arenz et al., 2020).

Gaussian Mixture Model

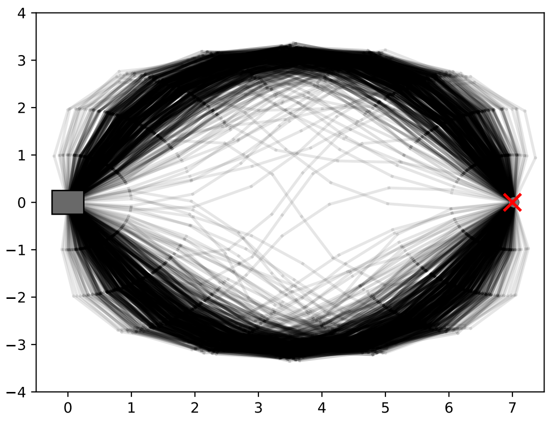
In this experiment we want to test the ability of gradientVIPS for exploring and learning the unknown multi-modal distribution. We will use a Gaussian mixture model with 10 components as unknown target distribution. We compare gradientVIPS with VIPS in GMM experiments at different dimensions namely D=20, D=40 and D=60 (where VIPS already reaches its limit). Furthermore, we test gradientVIPS at D=80, D=100 and D=120. To make a clear comparison between VIPS and gradientVIPS, we just show the results of D=20, D=40 and D=60. The conditions are set to be the same as in VIPS. Please refer to the reference for detailed conditions (Arenz et al., 2020).



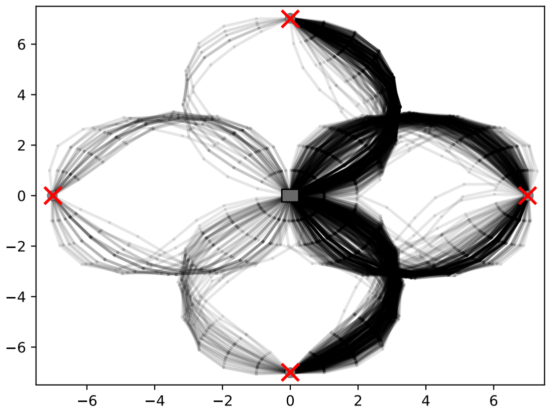
(a) Planar robot 1 goal ground truth



(b) Planar robot 4 goals ground truth



(c) Planar robot 1 goal 333 components



(d) Planar robot 4 goals 360 components

Figure 5.1.: The upper two plots show 200 ground-truth samples for both planar robot experiments. The lower two plots visualize the weights and means of the mixture models learned by VIPS for each of the planar robot experiments. The ground-truth samples are generated using generalized elliptical slice sampling. The gray box indicates the base of the robot, the red crosses indicate the goal positions. Each line represents a component, components with larger weight are drawn darker. The planar robot 1 goal plot shows 333 components learned by VIPS and the planar robot 4 goals plot shows 360 components learned by VIPS. Pictures are taken from Arenz et al. (2020) .

Planar Robot

We test gradientVIPS on planar robot 1 goal and 4 goals experiments. They are more challenging than GMM experiments because in the planar robot experiments the unknown target distribution is multi-modal non-Gaussian distribution. In the experiments, we need to sample from the joint configuration of a 10-link planar robot. The length of each joint is 1 and the joint configuration describes the angles of the links in radian. We perform 2 experiments planar robot 1 goal and planar robot 4 goals. The robot base is fixed at (0,0). In the 1 goal experiment, the planar robot tries to reach the single target at the position of (7,0) and in the 4 goals experiment, the planar robot tries to reach the targets which have the positions (7,0), (0,7), (-7,0) and (0,-7). For each goal, the planar robot has two ways to reach the desire position (through upper and through bottom). Following the same experiment setting as in VIPS (Arenz et al., 2020), we put a zero mean Gaussian prior on the joint configurations where we use a variance of 1 for the first joint and a variance of 4e-2 for the remaining joints to induce smooth configurations. Figure 5.1 shows us plots of planar robot 1 goal and 4 goals from the ground truth samples and a visualization of the means and weights learned by VIPS.

5.1.2. Ablations

In this section we test some of the choice that can affect our method. Firstly, we remove the gradient information in the model fitting part of gradientVIPS to re-implement VIPS. We want to show that using gradient information is crucial to improve sample efficiency. Secondly, we also compare gradientVIPS and target-free gradientVIPS, which solely use the gradient information to do model fitting and update components. We want to show that, the gradient has already contained the information we need to get a good approximation, which indicates that the gradient information is more important than target unnormalized information.

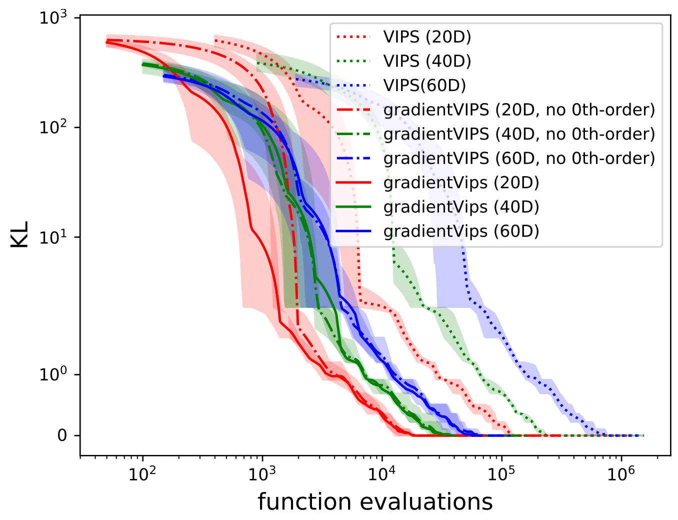
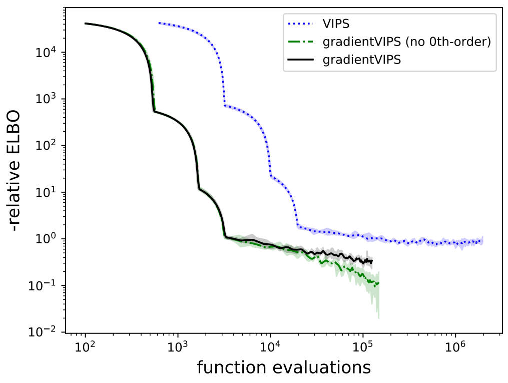
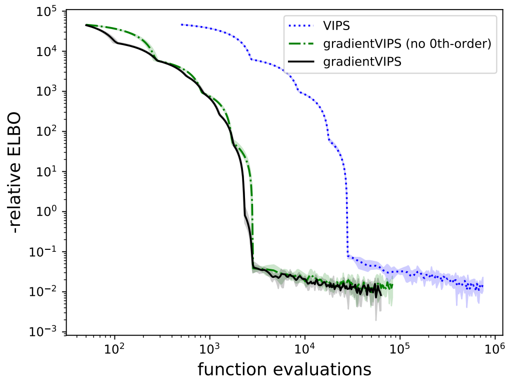


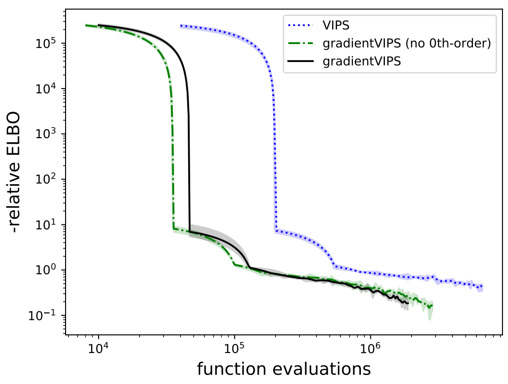
Figure 5.2.: Relative ELBO comparison of gradientVIPS, gradientVIPS without target and re-implemented VIPS in GMM experiments with different dimensions. Comparisons of the same dimensions are marked with the same color. The line-form differs from methods. All the experiments are averaged on five runs.



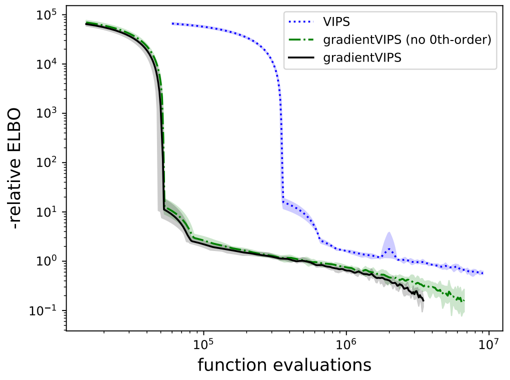
(a) Breast cancer (Logistic regression)



(b) German credit (Logistic regression)



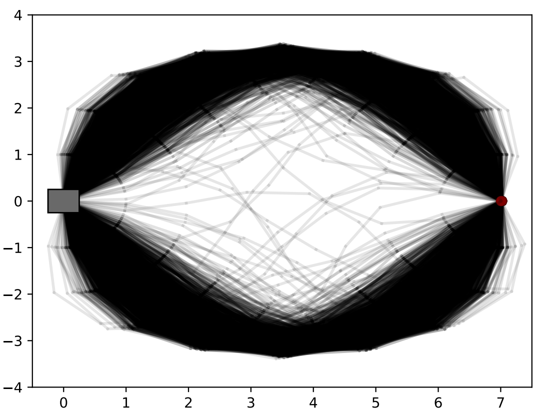
(c) Planar robot 1 goal

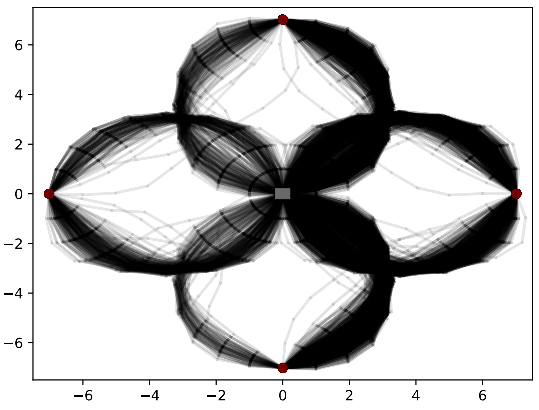


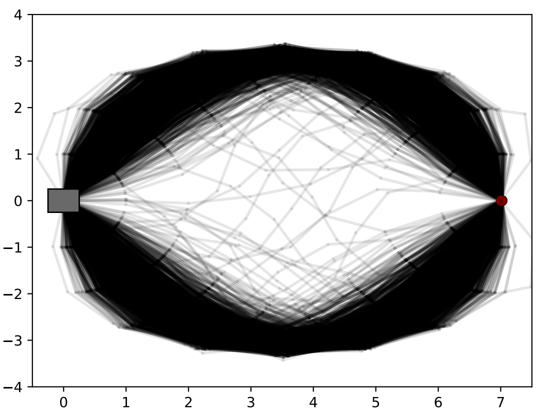
(d) Planar robot 4 goals

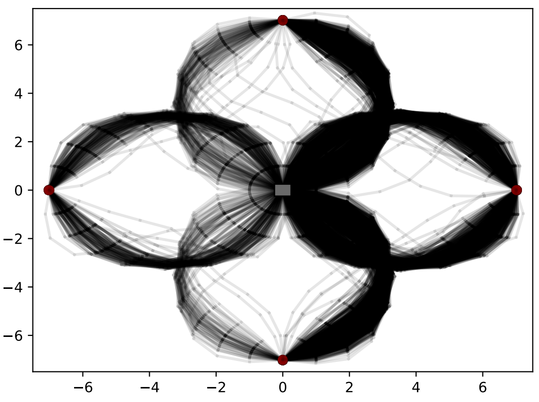
Figure 5.3.: Relative ELBO comparison of gradientVIPS, gradientVIPS without target and re-implemented VIPS in logistic regression and planar robot experiments. All experiments are averaged on five runs. Each method has its own color and differs in the line form.

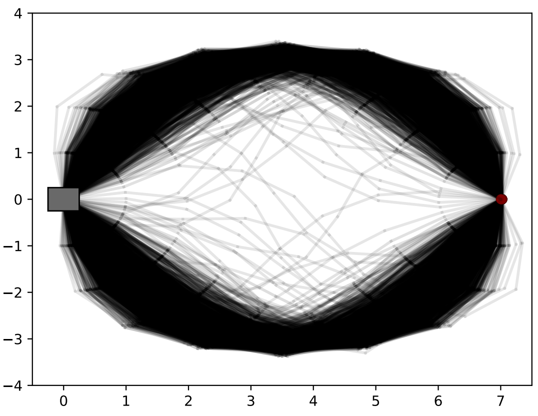
The ablation results of GMM experiments are shown in figure 5.2, the results of logistic regression and planar robot experiments are shown in figure 5.3. We can see from the plots that gradientVIPS outperforms re-implemented VIPS with an improved sample efficiency around one magnitude in all experiments while keeping the accuracy of VIPS or even better, which indicates that gradient information is crucial for improving sample efficiency. Furthermore, when comparing gradientVIPS and gradientVIPS without target, we can find both methods achieve almost the same ELBO by using almost the same number of function evaluations in the all experiments. In breast cancer experiment, gradientVIPS without target even outperforms gradientVIPS. From this comparison we can see gradient information solely contains all the information that is needed for approximating. Figure 5.4 visualized the learned mean and weights of the planar robot 1 goal and


(a) Planar robot 1 goal 1290 components


(b) Planar robot 4 goals 1269 components


(c) Planar robot 1 goal 869 components


(d) Planar robot 4 goals 939 components


(e) Planar robot 1 goal 1363 components

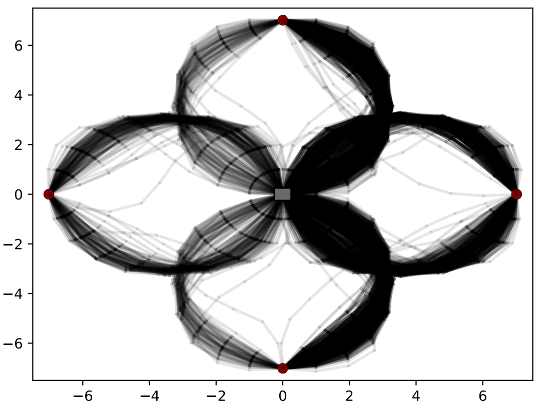

(f) Planar robot 4 goals 999 components

Figure 5.4.: Visualization of learned means and weights in the planar robot one goal and four goals experiments. From top to bottom the plots are arranged as *gradientVIPS* (top), re-implement *VIPS* (middle) and *gradientVIPS without target value* (bottom). The number of components differs from each other due to computational time and the limit of memory. Grey box indicates the robot base and red circle indicates the position of the end-effector. Each line represents a component, components with larger weight are drawn darker.

4 goals using re-implemented VIPS, gradientVIPS and gradientVIPS without target. These plots show that the three methods can more or less achieve the same quality.

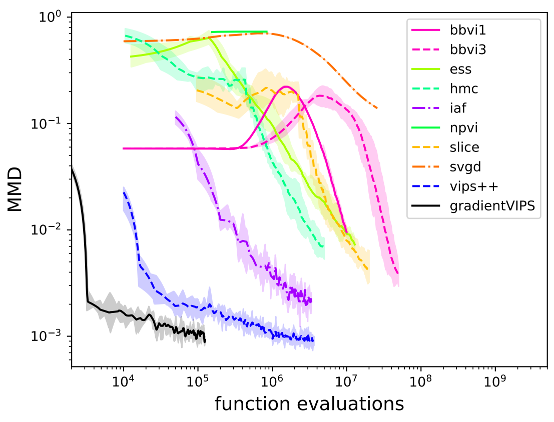
5.1.3. Results

We compare different methods with respect to sample efficiency and sample equality. For efficiency we compare the number of function evaluations, for sample quality we compare the maximum mean discrepancy (MMD).

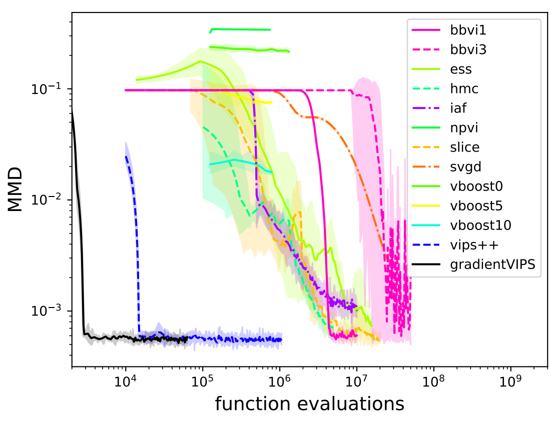
Figure 5.5 shows plots of the maximum mean discrepancy (MMD) with respect to ground truth samples over the number of function evaluations on log-log plots. In these plots, MMD is linear interpolated to show continuously line. For all the plots We can easily tell gradientVIPS and the other methods apart. All the experiments are averaged on five runs, the maximum and minimum value are plotted as the upper bound and lower bound of the shaded area. The line in the middle of the shaded area represent the averaged value of the five runs. We use our previous experiment data in logistic regression, 20-dimensional, 40-dimensional, 60-dimensional GMM, and planar robot experiments. We run gradientVIPS for all the experiments and compare it directly with results from our previous work (Arenz et al., 2020). Figure 5.6 shows the relative ELBO comparison of different methods in the logistic regression and planar robot experiments. For Gaussian distributed target, we can directly compare the ELBO which is equivalent to KL divergence in the GMM experiment. The relative ELBO curves show the same trend as the MMD curves. However, in the Breast cancer experiment, gradientVIPS performs slightly worse than VIPS, we will talk about this in detail in section 5.1.4.

5.1.4. Discussion

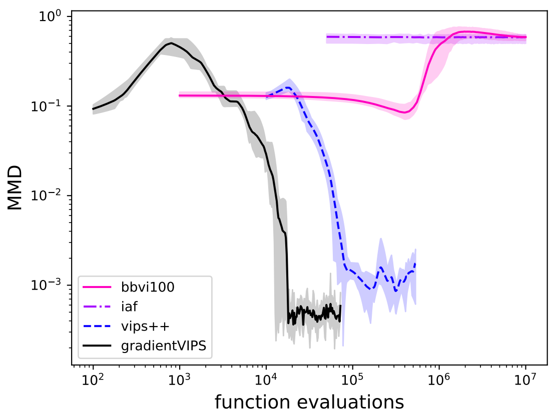
We have compared all methods with respect to sample efficiency and sample quality. For the sample efficiency, gradientVIPS outperforms all the other competitors in all the experiments. GradientVIPS improves the sample efficiency around one magnitude compared with VIPS. In the aspect of sample quality, gradientVIPS beats all the state-of-the-art variational inference methods and can achieve the same quality with the best MCMC sampler by using 3 or 4 magnitude fewer function evaluations. In the planar robot 4 goals experiments we can see the PTMCMC method can reach a lower MMD. We believe, When learning sufficient number of components, gradientVIPS can also achieve the same or even better results as PTMCMC. However, the computation of a Gaussian mixture model with many components will be infeasible and storing all the components will require a lot of memory. We have also noticed that in the breast cancer experiments the relative ELBO of gradientVIPS is slightly worse than VIPS. The reason is, in the result section, we used our previous result from VIPS, which is in a different implementation environment (C++). In the ablation section we have already compared gradientVIPS with re-



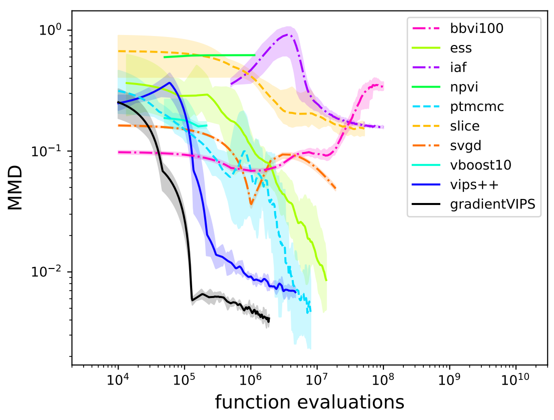
(a) Breast cancer



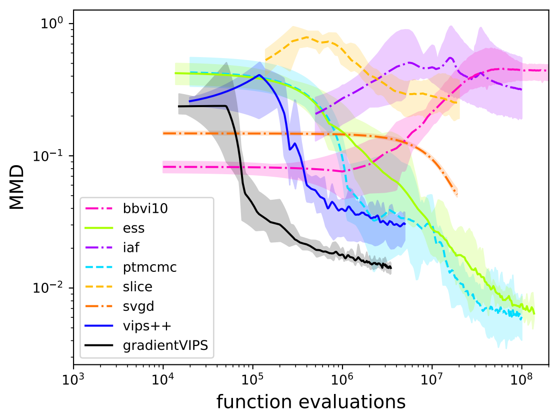
(b) German credit



(c) GMM 20d

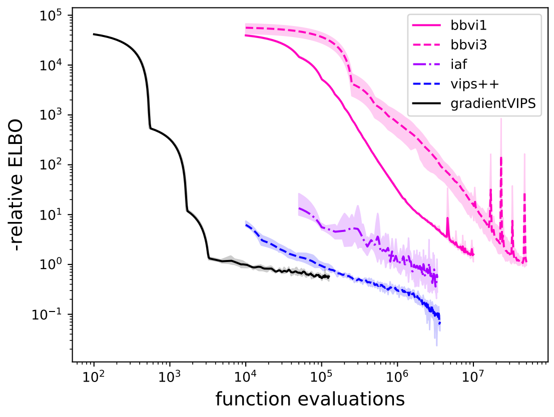


(d) Planar robot 1 goal

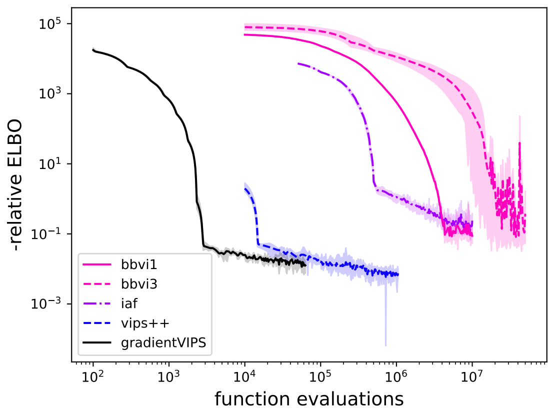


(e) Planar robot 4 goals

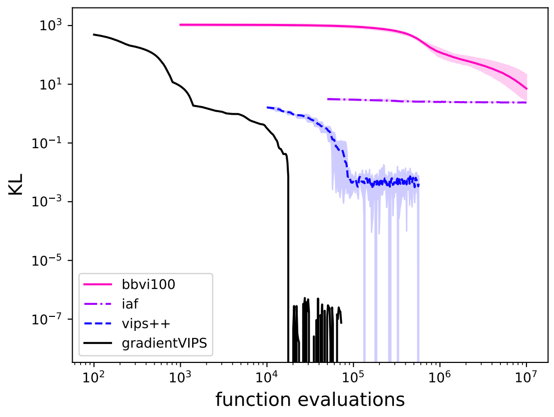
Figure 5.5.: MMD comparisons of different methods.



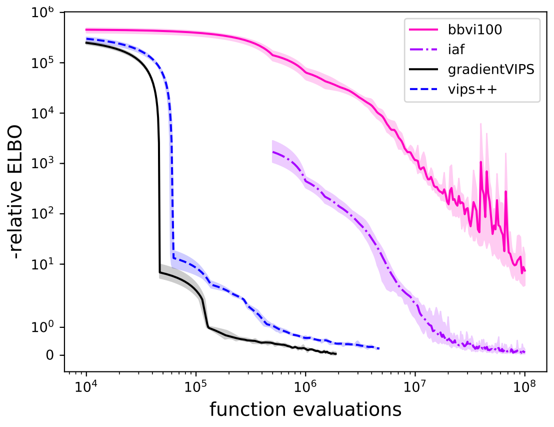
(a) Breast cancer



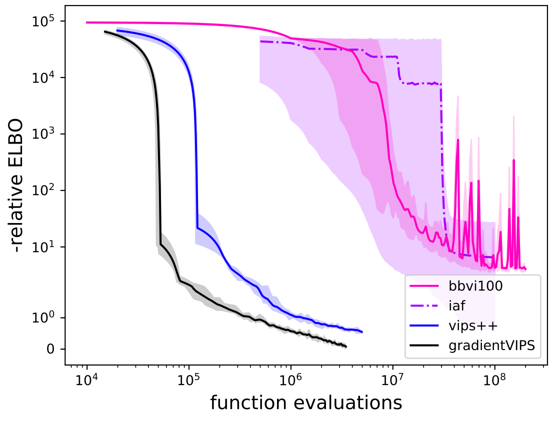
(b) German credit



(c) GMM 20d



(d) Planar robot 1 goal



(e) Planar robot 4 goals

Figure 5.6.: Relative ELBO comparisons of different Methods.

implemented VIPS in Python and showed gradientVIPS outperforms VIPS in all aspects in the same implementation environment.

5.2. Embedded VIPS

In embedded VIPS, due to time reason we have done two experiments which are identity mapping and fixed decoder component. Identity mapping means the model has the factorized structure, the

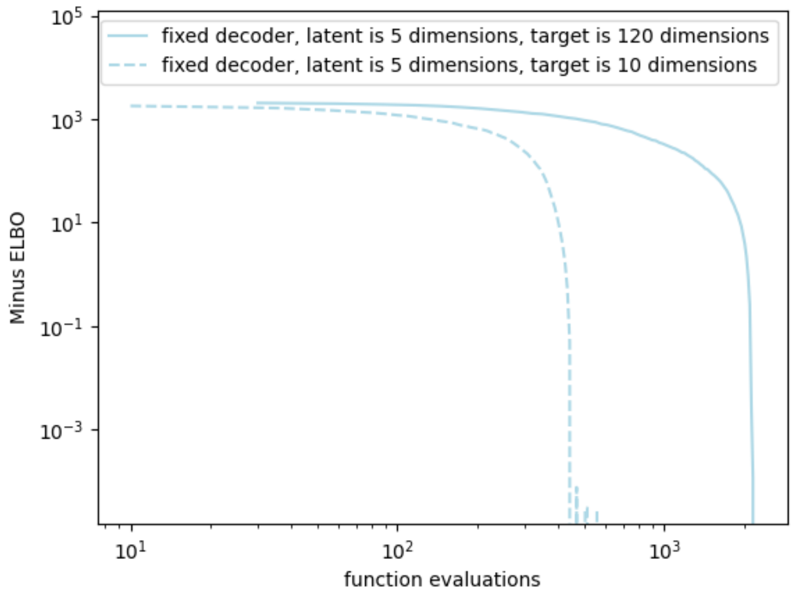


Figure 5.7.: GMM experiments with different dimensions.

latent component has the same dimensions as the target, the decoder is fixed and equals identity but the target distribution doesn't have the factorized structure. Please note that, if we do identity mapping, the way to get the reward of latent is based on Monte-Carlo estimation.

The second experiment we have done for embedded VIPS is the fixed decoder. In this setting, the model has the factorized structure and the decoder is assumed to be known and fixed. Meanwhile for the target distribution, it also has the factorized structure. We can see from the plot the result of learning one unknown target distribution. If we compare the results with gradientVIPS we will find the number of samples we need to use is far less than before, but we should remember that the decoder update is not working now and the problem is equal to learn a 5 dimensional problem. However, figure 5.7 shows the number of samples used for learning a 10 dimensional problem is around five times more than the number of samples used for learning a 120 dimensional problem.

This problem is caused by the hyperparameter the number of samples used per iteration, which can be seen from the start point of the two lines.

Chapter 6.

Conclusion and Future Work

The goal of this thesis is to extend VIPS to take gradient information. We talk about the conclusion of the developed methods and possibilities of the future work in this chapter.

6.1. Conclusion

In this work, we proposed three methods namely gradientVIPS, embedded VIPS and high-d VIPS for learning GMM approximations of intractable probability distributions. The first approach gradientVIPS is an extension of VIPS. We modified the model fitting part in VIPS to take the gradient information. The code was extended to have a sample base and add functions to reuse samples. In all experiments, gradientVIPS outperforms VIPS in aspects of speed and scalability. GradientVIPS achieves comparable accuracy compared with VIPS with one magnitude improved sample efficiency. Since in gradientVIPS we also learn a full Gaussian covariance, gradientVIPS reaches its limit when dealing with high dimensional problems (over 120 dimensions). To scale to a higher dimensional problem, in embedded VIPS and high-d VIPS we employ a factorized model to learn a low rank approximation of the unknown full Gaussian. Following the same scheme as in VIPS we derived a variational lower bound on the I-projection and updated each component separately. The latent component update was based on VIPS or gradientVIPS by employing the reparameterization trick, where we assume the covariance is full Gaussian. Decoder component was updated based on modified MOTO, where we assume the covariance is diagonal. We derived the theory of embedded and high-d VIPS, but due to time reasons we were unable to fix some open questions such as how to use weighted samples in PPCA to initialize the gain matrix, or we can find a better way to initialize the gain matrix.

To evaluate the developed methods, we did many experiments for evaluation. The experiments ranged from non-Gaussian with only one mode to Gaussian with multi modes and non-Gaussian with multi modes. The results of gradientVIPS showed it has already improved the sample efficiency around one magnitude while keeping the high accuracy of VIPS and it can also scale to a higher space than VIPS.

6.2. Future Work

The first work is to fix the open questions of embedded VIPS either theoretically or practically. After fixing these problems we will do more experiments to evaluate embedded VIPS and high-d VIPS. One more interesting point is how to find promising area more efficiently. We have found in practice sometimes two components will try to find the same mode, which is inefficient in terms of function evaluations and calculation. How to efficiently find a area which has a potential to be the promising area is an interesting point to look into.

Bibliography

A. Abdolmaleki, R. Lioutikov, J. R. Peters, N. Lau, L. Pualo Reis, and G. Neumann. Model-based relative entropy stochastic search. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL `https://proceedings.neurips.cc/paper/2015/file/36ac8e558ac7690b6f44e2cb5ef93322-Paper.pdf`.

R. Akrour, G. Neumann, H. Abdulsamad, and A. Abdolmaleki. Model-free trajectory optimization for reinforcement learning. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2961–2970, New York, New York, USA, 20–22 Jun 2016. PMLR. URL `https://proceedings.mlr.press/v48/akrour16.html`.

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*, 19:565–589, 2018.

J. R. Anderson and C. Peterson. A mean field theory learning algorithm for neural networks. *Complex Systems*, 1:995–1019, 1987.

C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan. An introduction to mcmc for machine learning. *Machine learning*, 50(1):5–43, 2003.

J. O. Arenz. Sample-efficient i-projections for robot learning. 2021.

O. Arenz, G. Neumann, and M. Zhong. Efficient gradient-free variational inference using policy search. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 234–243.

PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/arenz18a.html.

O. Arenz, M. Zhong, and G. Neumann. Trust-region variational inference with Gaussian mixture models. *Journal of Machine Learning Research*, 21(163):1–60, 2020. URL http://jmlr.org/papers/v21/19-524.html.

P. Becker, O. Arenz, and G. Neumann. Expected information maximization: Using the i-projection for mixture density estimation. *arXiv preprint arXiv:2001.08682*, 2020.

C. Bishop, N. Lawrence, T. Jaakkola, and M. Jordan. Approximating posterior distributions in belief networks using mixtures. *Advances in neural information processing systems*, 10: 416–422, 1997.

C. M. Bishop. Pattern recognition. *Machine learning*, 128(9), 2006.

D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

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

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

C. Daniel, G. Neumann, and J. Peters. Hierarchical relative entropy policy search. In *Artificial Intelligence and Statistics*, pages 273–281. PMLR, 2012.

M. P. Deisenroth, G. Neumann, and J. Peters. *A survey on policy search for robotics*. now publishers, 2013.

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

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

M. Ewerton, O. Arenz, and J. Peters. Assisted teleoperation in changing environments with a mixture of virtual guides. *Advanced Robotics*, 34(18):1157–1170, 2020.

S. Gershman, M. Hoffman, and D. Blei. Nonparametric variational inference. *arXiv preprint arXiv:1206.4665*, 2012.

A. Graves. Practical variational inference for neural networks. *Advances in neural information processing systems*, 24, 2011.

A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

T. C. Hesterberg. *Advances in importance sampling*. PhD thesis, Stanford University, 1988.

T. S. Jaakkola and M. I. Jordan. Improving the mean field approximation via the use of mixture distributions. In *Learning in graphical models*, pages 163–173. Springer, 1998.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

A. Kessy, A. Lewin, and K. Strimmer. Optimal whitening and decorrelation. *The American Statistician*, 72(4):309–314, 2018.

M. Khan, D. Nielsen, V. Tangkaratt, W. Lin, Y. Gal, and A. Srivastava. Fast and scalable bayesian deep learning by weight-perturbation in adam. In *International Conference on Machine Learning*, pages 2611–2620. PMLR, 2018.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29:4743–4751, 2016.

A. Koivunen and A. Kostinski. The feasibility of data whitening to improve performance of weather radar. *Journal of Applied Meteorology*, 38(6):741–749, 1999.

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

S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

M. Lichman et al. Uci machine learning repository, 2013. *URL http://archive. ics. uci. edu/ml*, 40, 2013.

L.-J. Lin. *Reinforcement learning for robots using neural networks*. Carnegie Mellon University, 1992.

Q. Liu and D. Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. *arXiv preprint arXiv:1608.04471*, 2016.

J. Martens. New insights and perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193*, 2014.

A. C. Miller, N. J. Foti, and R. P. Adams. Variational boosting: Iteratively refining posterior approximations. In *International Conference on Machine Learning*, pages 2420–2429. PMLR, 2017.

A. Mishkin, F. Kunstner, D. Nielsen, M. Schmidt, and M. E. Khan. Slang: Fast structured covariance approximations for bayesian deep learning with natural gradient. *arXiv preprint arXiv:1811.04504*, 2018.

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

A. Mukundan, G. Tolias, and O. Chum. Robust data whitening as an iteratively re-weighted least squares problem. In P. Sharma and F. M. Bianchi, editors, *Image Analysis*, pages 234–247, Cham, 2017. Springer International Publishing. ISBN 978-3-319-59126-1.

I. Murray, R. Adams, and D. MacKay. Elliptical slice sampling. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 541–548. JMLR Workshop and Conference Proceedings, 2010.

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

R. M. Neal. Slice sampling. *The annals of statistics*, 31(3):705–767, 2003.

G. Neu, A. Jonsson, and V. Gómez. A unified view of entropy-regularized markov decision processes. *arXiv preprint arXiv:1705.07798*, 2017.

G. Neumann et al. Variational inference for policy search in changing situations. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pages 817–824, 2011.

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

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

R. Ranganath, S. Gerrish, and D. Blei. Black box variational inference. In *Artificial intelligence and statistics*, pages 814–822. PMLR, 2014.

D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

E. Richardson and Y. Weiss. On gans and gmms. *arXiv preprint arXiv:1805.12462*, 2018.

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. Jordan, and P. Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

Appendix A.

MOTO Derivation

The optimization problem of MOTO can be formed as

$$\begin{aligned} \underset{\pi}{maximize} &\int\int \rho_t^i(s)\pi(a|s)Q_t^i(s,a)dads \\ subject\ to \qquad &\mathbb{E}_{s\sim\rho_t^i(s)}\left[KL(\pi(.|s)||\pi_t^i(.|s))\right] \leq \varepsilon \\ &\mathbb{E}_{s\sim\rho_t^i(s)}\left[H(\pi(.|s))\right] \geq \beta \\ &\int \pi(.|s)da = 1. \end{aligned}$$

For a good understanding, the parameters will be omitted and we will use π_1 to denote the π from last iteration. Using the Lagrange multipliers we can get the Lagrange function like follows

$$\begin{aligned} L(\pi,\eta,\omega,\lambda) &= \int\int \rho\pi Q dads + \eta\left[\varepsilon - \int \rho KL(\pi||\pi_1)\right] + \omega\left[\int \rho H(\pi) - \beta\right] + \int\lambda\left[\int \pi - 1\right] \\ &= \int \rho\left[\int \pi Q - \eta\pi\log\frac{\pi}{\pi_1} - \omega\pi\log\pi\right] + \eta\varepsilon - \omega\beta - \int\lambda + \int\lambda\int\pi, \end{aligned}$$

please note that here $\lambda = \lambda(s)$. For optimal π the partial derivative of π should be 0, according to this we can get the optimal π

$$\begin{aligned}\frac{\partial L}{\partial \pi} &= \rho\,[Q - \eta(\log\pi + 1 - \log\pi_1) - \omega(\log\pi + 1)] + \lambda = 0\\ \frac{1}{\eta+\omega}Q - 1 + \frac{\eta}{\eta+\omega}\log\pi_1 + \frac{1}{\rho(\eta+\omega)}\lambda &= \log\pi\\ \pi^* &= \pi_1^{\frac{\eta}{\eta+\omega}} e^{\frac{Q}{\eta+\omega}} e^{\frac{\lambda}{\rho(\eta+\omega)}-1}.\end{aligned}$$

After getting the optimal π^* we can get the dual function

$$\begin{aligned}g(\eta,\omega,\lambda) &= \int\rho\left[\int \pi^* Q - \eta\int\pi^*\log\pi^* + \eta\int\pi^*\log\pi_1 - \omega\int\pi^*\log\pi^*\right]\\ &+\int\lambda\int\pi^* + \eta\varepsilon - \omega\beta - \int\lambda\\ &= \int\rho\int\pi^*[\eta+\omega] + \eta\varepsilon - \omega\beta - \int\lambda\\ &= (\eta+\omega)\int\rho\int\pi^*(\eta,\omega,\lambda) + \eta\varepsilon - \omega\beta - \int\lambda\\ g(\eta,\omega,\lambda) &= (\eta+\omega)\int\rho\int\pi^* + \eta\varepsilon - \omega\beta - \int\lambda.\end{aligned}$$

Using the dual function we can get the partial derivative of η

$$\frac{\partial g(\eta,\omega,\lambda)}{\partial\eta} = \int\rho\int\pi^* + (\eta+\omega)\int\rho\int\frac{\partial\pi^*}{\partial\eta} + \varepsilon,$$

using the former results of the optimal π^*

$$\pi^* = \pi_1^{\frac{\eta}{\eta+\omega}} e^{-1} exp^{\frac{Q}{\eta+\omega}} e^{\frac{\lambda}{\rho(\eta+\omega)}},$$

and the side calculation

$$\frac{\partial\pi^*}{\partial\eta} = \pi^*\left(\frac{\omega}{(\eta+\omega)^2}\ln\pi_1 - \frac{Q+\lambda/\rho}{(\eta+\omega)^2}\right).$$

The partial derivative is

$$\begin{aligned}\frac{\partial g(\eta,\omega,\lambda)}{\partial\eta} &= \int\int\rho\pi^*\left(1 + \frac{\omega}{(\eta+\omega)}\ln\pi_1 - \frac{Q+\lambda/\rho}{(\eta+\omega)}\right) + \varepsilon\\ &= \varepsilon - \mathbb{E}\left[KL(\pi^*||\pi_1)\right].\end{aligned}$$

Similarly with the dual function we can get the partial derivative of ω

$$g(\eta,\omega,\lambda) = (\eta+\omega)\int\rho\int\pi^* + \eta\varepsilon - \omega\beta - \int\lambda,$$

$$\frac{\partial g(\eta,\omega,\lambda)}{\partial \omega} = \int \rho \int \pi^* + (\eta+\omega)\int \rho \int \frac{\partial \pi^*}{\partial \omega} - \beta.$$

Using the optimal π^*

$$\pi^* = \pi_1^{\frac{\eta}{\eta+\omega}} e^{-1} exp^{\frac{Q}{\eta+\omega}} e^{\frac{\lambda}{\eta+\omega}},$$

and the side calculation

$$\frac{\partial \pi^*}{\partial \omega} = \pi^*(\frac{-\eta}{(\eta+\omega)^2}\ln \pi_1 - \frac{Q+\lambda/\rho}{(\eta+\omega)^2}),$$

the partial derivative of ω is

$$\begin{aligned}\frac{\partial g(\eta,\omega,\lambda)}{\partial \omega} &= \int\int \rho\pi^*(1+\frac{-\eta}{(\eta+\omega)}\ln \pi_1 - \frac{Q+\lambda/\rho}{(\eta+\omega)}) + \beta \\ &= \mathbb{E}[H(\pi^*)] - \beta.\end{aligned}$$

The partial derivative of λ should also be 0

$$\begin{aligned}\frac{\partial g}{\partial \lambda} &= (\eta+\omega)\int \rho \frac{\partial \pi^*}{\partial \lambda} - 1 = 0 \\ &= \int \pi^* - 1 = 0.\end{aligned}$$

We can get the dual function w.r.t η and ω

$$g(\eta,\omega) = (\eta+\omega)\int \rho + \eta\varepsilon - \omega\beta - \int \lambda^*(\eta,\omega).$$

By using the third constraints we can get

$$\int \pi^* - 1 = 0,$$

which leads to

$$\int \pi_1^{\frac{\eta}{\eta+\omega}} e^{-1} exp(\frac{Q}{\eta+\omega}) e^{\frac{\lambda}{\rho(\eta+\omega)}} = 1.$$

After log both sides we can get

$$\lambda = \rho(\eta+\omega) - \rho(\eta+\omega)\log\int \pi_1^{\frac{\eta}{\eta+\omega}} exp(\frac{Q}{\eta+\omega}).$$

Integrating both sides we can get

$$\int \lambda = (\eta+\omega)\int \rho - (\eta+\omega)\int \rho \log \int \pi_1^{\frac{\eta}{\eta+\omega}} exp(\frac{Q}{\eta+\omega}),$$

the dual function

$$g(\eta,\omega,\lambda) = (\eta+\omega)\int \rho + \eta\varepsilon - \omega\beta - \int \lambda.$$

can be rewritten as

$$\begin{aligned} g(\eta,\omega) &= (\eta+\omega)\int \rho + \eta\varepsilon - \omega\beta - (\eta+\omega)\int \rho + (\eta+\omega)\int \rho(s)\log \int \pi_1^{\frac{\eta}{\eta+\omega}} exp(\frac{Q}{\eta+\omega}) \\ &= \eta\varepsilon - \omega\beta + (\eta+\omega)\int \rho(s)\log \int \pi_1^{\frac{\eta}{\eta+\omega}} exp(\frac{Q}{\eta+\omega}) \end{aligned}$$

Appendix B.

Expectation of KL Divergence

Calculate the expected KL divergence between embedding updates

$$q(x \mid z,o) = N\left(x \mid FLz + F_t, F\left(\eta^* + 1\right)\right).$$
$$q^i(x \mid z,o) = N\left(x \mid F_iL_iz + F_if_i, \Sigma_i\right).$$

for simplicity we denote here the covariance and mean of $q(x \mid z,o)$ as A and a we denote the covariance and mean of $q^i(x \mid z,o)$ as B and b,

$$\begin{aligned}2D_{\mathrm{KL}}(q(x \mid z,o) \mid q^i(x \mid z,o)) &= \log\frac{|B|}{(A\,|} + \operatorname{tr}\left(B^{-1}A\right) + (b-a)^\top B^{-1}(b-a) - n.\\ &= \log|B| - \log|A| + \operatorname{tr}(B^{-1}A) + z^\top \underbrace{\left(F_iL_i - FL\right)^\top B^{-1}\left(F_iL_i - FL\right)}_{R} z\\ &+ 2\underbrace{\left(F_if_i - Ff\right)^\top B^{-1}\left(F_iL_i - FL\right)}_{F} z + \left(F_if_i - Ff\right)^\top B^{-1}\left(F_if_i - Ff\right) - n\end{aligned}$$

just the parts with under brackets have relations with z, if we calculate the expectation w.r.t z, then we can just evaluate the under brackets part, other parts are constant w.r.t.z.

$$\begin{aligned}
&\mathbb{E}_{q(\mathbf{z}|o)}\left[D_{\mathrm{KL}}(q(\mathbf{x}|\mathbf{z},o)||q(\mathbf{x}|\mathbf{z},o)^{i})\right] = 1/2\Big(\log\frac{|B|}{|A\,|} + \mathrm{tr}\left(B^{-1}A\right) + (b-a)^{\top}B^{-1}(b-a) - n\Big)\\
&= 1/2\Big(\log|B| - \log|A| + \mathrm{tr}(B^{-1}A) + (F_i f_i - Ff)^{\top}B^{-1}(F_i f_i - Ff)\\
&-n + \int q(\mathbf{z}|o) z^{\top}\underbrace{(F_i L_i - FL)^{\top}B^{-1}(F_i L_i - FL)}_{R} z\\
&+2\int q(\mathbf{z}|o)\underbrace{(F_i f_i - Ff)^{\top}B^{-1}(F_i L_i - FL)}_{F} z\Big).
\end{aligned}$$

According to Gaussian identity 33. the integral with under bracket marked with R equals

$$\int q(\mathbf{z}|o) z^{\top}\underbrace{(F_i L_i - FL)^{\top}B^{-1}(F_i L_i - FL)}_{R} z = \mu_z{}^{\top}R\mu_z + \mathrm{tr}\left(R\Sigma_{z|o}\right)$$

According to Gaussian identity 31. the integral marked with under bracket F equals

$$\int q(\mathbf{z}|o)\underbrace{(F_i f_i - Ff)^{\top}B^{-1}(F_i L_i - FL)}_{F} z = F\mu_z$$