Parallel Tempering VIPS

Paralleles Tempering zur Variationsinferenz durch Policy Search
Bachelor thesis by Samuel Lokadjaja
Date of submission: November 29, 2022

1. Review: Dr. Ing. Oleg Arenz
2. Review: Prof. Jan Peters, Ph.D.
Darmstadt

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Erklärung zur Abschlussarbeit gemäß § 22 Abs. 7 und § 23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Samuel Lokadjaja, die vorliegende Bachelorarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Fall eines Plagiats (§ 38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß § 23 Abs. 7 APB überein.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, November 29, 2022

S. Lokadjaja

Abstract

The VIPS algorithm introduced by Arenz et al. [1] is an algorithm to approximate intractable multimodal probability distributions using Gaussian mixture models, which leverages methods from policy search. A drawback with this algorithm is that modes that are not found at an early stage will also not be found later, as the samples used to construct an approximation are only drawn from the current approximation. Therefore, in this paper, we investigate how the existing VIPS algorithm could be extended to improve exploration of the search space to use more than solely samples from the current distribution. We use an approach inspired by parallel tempering, by using multiple Gaussian mixture models with different target distributions $\frac{1}{\beta_1}R(\mathbf{x}), \ldots, \frac{1}{\beta_n}R(\mathbf{x})$ for each model that are scaled by temperature coefficients $\beta_1, \ldots, \beta_n$ with $\beta_1 > \cdots > \beta_n = 1$. This allows the models with lower temperature to access samples from a larger subset of the search space, thus making it less likely to get stuck in local optima. Furthermore, we also construct a tree structure using the components of the models, where one component in a GMM could have one or more "children" components on the next lower level. This structure is enforced by applying soft constraints on the KL divergence of each pair of components. We also experiment using the softplus function as a reward term, which will only penalize a component if it is far away from its parent component, but will not contribute to the optimization when a component is close to its parent component. We found that using parallel tempering for VIPS generally helps exploration, but applying soft constraints on the KL divergence between a component and its parent harms the capacity to approximate the target distribution.

Acronyms

ELBO Evidence Lower Bound.

GMM Gaussian Mixture Model.

KL Kullback-Leibler.

MCMC Markov chain Monte Carlo.

MCTS Monte Carlo Tree Search.

MORE Model-Based Relative Entropy Stochastic Search.

VIPS Variational Inference by Policy Search.

Contents

1. Introduction **2**
1.1. Related Work 4

2. Foundations **5**
2.1. Variational Inference 5
2.2. Variational Inference by Policy Search 6
2.3. Parallel Tempering 9

3. Method **12**
3.1. Parallel Tempering 12
3.2. Tree Structure 13

4. Experiments and Results **16**
4.1. Experiments 16
4.2. Results 18

5. Discussion **30**

6. Conclusion **31**
6.1. Future Work 31

A. MORE Derivation **33**

1. Introduction

One problem in machine learning is working with complex probability distributions that cannot be computed in closed form, for example when computing the posterior predictive distribution of a new data point $\tilde{x}$ given a dataset $\mathcal{D} = \{x_1, \ldots, x_n\}$ and model parameter θ, which is given by:

$$p(\tilde{x}|\mathcal{D}) = \mathbb{E}_{p(\theta|\mathcal{D})}[p(\tilde{x}|\theta, \mathcal{D})] = \int p(\tilde{x}|\theta, \mathcal{D})p(\theta|\mathcal{D})\, d\theta,$$

which requires the posterior distribution of the model parameters given the dataset

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

This posterior distribution is often impossible to compute in closed form because of the integral in the denominator. More generally, these complex distributions typically have the form

$$p(\mathbf{x}) = \frac{\tilde{p}(\mathbf{x})}{\int \tilde{p}(\mathbf{x})\, dx} = \frac{\tilde{p}(\mathbf{x})}{Z}. \tag{1.1}$$

To solve this problem, there are two approaches that are commonly used. Using Markov chain Monte Carlo (MCMC) techniques, expectations with respect to the posterior $p(\theta|\mathcal{D})$ could be estimated by drawing samples from it. Another approach is variational inference, where the posterior is approximated by a parameterized distribution $q(\theta)$, e.g. a Gaussian, which could be found by minimizing the Kullback-Leibler (KL) divergence between $p(\theta|\mathcal{D})$ and $q(\theta)$.

MCMC methods tend to be more computationally intensive than variational inference, but is more accurate, whereas variational inference only produces a distribution that is close to the target distribution, but tends to be more computationally efficient [2].

Arenz et al. [1] introduced VIPS, an iterative method for learning Gaussian mixture model (GMM) approximations for intractable multimodal distributions using variational inference, which will be explained in detail in section 2.2. This method has been shown to be able to outperform existing methods for variational inference in terms of accuracy while also being able to draw samples much more efficiently than MCMC for target distributions that could be adequately approximated by a small number of components.

However, with this method, modes that have not been discovered at the early stage will also likely not be found at a later stage, as samples used for constructing an approximation are only drawn from the current approximation. Evaluating additional samples from a prior with high entropy for further exploration would also likely not help find additional modes, particularly if we are working with a high-dimensional space, as it would just provide samples randomly from the entire search space.

Therefore, this work aims to investigate how VIPS could be extended to improve exploration of the search space. We use an approach inspired by parallel tempering, a global optimization strategy which has been shown to be very efficient [3, 4], by using multiple Gaussian mixture models with different target distributions $\frac{1}{\beta_1}R(\mathbf{x}), \ldots, \frac{1}{\beta_n}R(\mathbf{x})$ for each model that are scaled by temperature coefficients $\beta_1, \ldots, \beta_n$ with $\beta_1 > \cdots > \beta_n = 1$. As the different models share a single sample database, each of the models have access to a larger set of samples coming from every model that are trained using replicas of the target distribution with higher entropy. These models resemble the target distribution while being more uniform than the target, therefore the modes tend to overlap more and they also cover a larger subset of the search space. The samples from the models with higher temperature therefore guides the models with lower temperatures by supplying samples from them and makes it less likely to get stuck in local optima.

Furthermore, we construct a tree structure using the components of the different GMMs, where one component in a GMM could have one or more "children" components on the next lower level. We enforce this structure using a penalty term on the KL divergence of each pair of components. This could allow for more efficient exploration and optimization, as the penalty term encourages the components on lower levels to stay close to their "parent" components and prevents different components from overlapping. We also experiment using the softplus function as a reward term, which will only penalize a component if it is far enough from its parent component, but will not contribute to the optimization when a component is close to its parent component.

In the next section, we will discuss further about variational inference and VIPS, as well as parallel tempering. Then, we will present our method in applying parallel tempering and

constructing the tree structure for VIPS, and finally present the experiments we conducted and the results of these experiments.

1.1. Related Work

Similar to the tree structure proposed in this paper, Rainforth et al. [5] introduced inference trees (ITs) based on MCTS, which are used for balancing exploration and exploitation in Monte Carlo methods. In Monte Carlo methods, the choice of proposal distribution has a large impact on their performance, but it is often difficult to determine a good proposal in advance. As a solution, many methods use past samples to continuously adapt the proposal in future iterations. This could be problematic, as it assumes that these samples already represent the true posterior well, which defeats the purpose of adaptation in the first place. ITs solve this problem by hierarchically partitioning the parameter space into disjoint sets in an online manner, which allows them to identify regions of high posterior mass, as well as track regions where significant posterior mass could have been missed. This algorithm is thus able to investigate whether the current proposal could be improved, for example by searching for missing modes, rather than greedily exploiting the best proposal learned so far.

Huang et al. [6] discuss alpha-annealing, an annealing technique for variational inference which is used to encourage exploration of significant modes and is designed for learning a good inference model. Here, the objective function to be optimized is

$$\mathbb{E}_q[\log q(x) - \alpha \log p(x)],$$

where $q(x)$ is the approximate distribution, $p(x)$ is the target and $\alpha = \frac{1}{T}$. The temperature T is initially set to be high, and is gradually annealed to 1. Huang et al. argue that alpha-annealing is suitable for one-time inference, for example for inferring the posterior distribution of a Bayesian model, but not so much for latent variable models or hierarchical Bayesian models where multiple inferences are needed. In these cases, this will introduce too much noise into the gradient estimates of the model and will harm training. Our approach differs from alpha-annealing as we use multiple models with different temperature coefficients instead of using one model and continuously decreasing the annealing coefficient.

2. Foundations

2.1. Variational Inference

The purpose of variational inference is to approximate a probability distribution $p(\mathbf{x})$ using another distribution $q(\mathbf{x})$ from a tractable family of distributions, such as Gaussians. The most suitable approximate distribution could be found by minimizing the KL divergence between $q(\mathbf{x})$ and $p(\mathbf{x})$, which could be considered as a distance function between two distributions. For parameterized distributions $q(\mathbf{x}|\theta)$, this KL divergence is given by

$$\mathrm{KL}(q(\mathbf{x}|\theta) \,||\, p(\mathbf{x})) = \int q(\mathbf{x}|\theta) \log\left(\frac{q(\mathbf{x}|\theta)}{p(\mathbf{x})}\right) d\mathbf{x} = \underbrace{\int q(\mathbf{x}|\theta) \log\left(\frac{q(\mathbf{x}|\theta)}{\tilde{p}(\mathbf{x})}\right) d\mathbf{x}}_{-L(\theta)} + \log Z, \tag{2.1}$$

where $\tilde{p}(\mathbf{x})$ and Z are as defined in equation 1.1.

The term $\log Z$ does not depend on θ and can thus be ignored. Therefore, minimizing the KL divergence is equivalent to maximizing $L(\theta)$, which is commonly called the evidence lower bound (ELBO), as it is a lower bound on $\log Z$ due to the non-negativity of the KL divergence, and Z being commonly used to denote the evidence in the case of Bayesian inference. We can reformulate $L(\theta)$ to get

$$\begin{aligned} L(\theta) &= -\int q(\mathbf{x}|\theta) \log\left(\frac{q(\mathbf{x}|\theta)}{\tilde{p}(\mathbf{x})}\right) d\mathbf{x} \\ &= \int q(\mathbf{x}|\theta)(\log \tilde{p}(\mathbf{x}) - \log q(\mathbf{x}|\theta))\, d\mathbf{x} \\ &= \int q(\mathbf{x}|\theta) \log \tilde{p}(\mathbf{x})\, d\mathbf{x} + H(q(\mathbf{x}|\theta)) \\ &= \mathbb{E}_{q(\mathbf{x}|\theta)}[\log \tilde{p}(\mathbf{x})] + H(q(\mathbf{x}|\theta)), \end{aligned}$$

which turns our optimization problem into

$$\theta^\star = \text{argmax}_\theta \ \left[\mathbb{E}_{q(\mathbf{x}|\theta)}[R(\mathbf{x})] + H(q(\mathbf{x}|\theta))\right].$$

where we set $R(\mathbf{x}) = \log \tilde{p}(\mathbf{x})$.

This way of expressing the ELBO allows us to interpret its maximization as policy search, which is a problem in reinforcement learning that aims to find the optimal policy $q(\mathbf{x}|\theta)$ that maximizes the expected reward $R(\mathbf{x})$, while also maximizing the entropy of the policy in order to improve exploration and prevent premature convergence [7].

We will now discuss VIPS introduced by Arenz et al. [1], an iterative algorithm which solves this optimization problem in a variational inference setting by using a GMM as the approximate distribution.

2.2. Variational Inference by Policy Search

In VIPS, we want to approximate an intractable probability distribution $p(\mathbf{x})$ using a GMM

$$q(\mathbf{x}|\theta) = \sum_{i=1}^{n} \pi_i \mathcal{N}(\mu_i, \Sigma_i) = \sum_o q(o) q(\mathbf{x}|o),$$

where $\theta = [\pi_1, \mu_1, \Sigma_1, \ldots, \pi_n, \mu_n, \Sigma_n]$. From now on we will omit θ for simplicity.

Recall from the previous section that minimizing the KL divergence between $q(\mathbf{x})$ and $p(\mathbf{x})$ is equivalent to minimizing the ELBO, which is here given by

$$\begin{aligned} L(\theta) &= \sum_o q(o) \int q(\mathbf{x}|o)(R(\mathbf{x}) - \log q(\mathbf{x}))\, d\mathbf{x} \\ &= \sum_o q(o) \left(\int q(\mathbf{x}|o)(R(\mathbf{x}) + \log\, q(o|\mathbf{x}))\, d\mathbf{x} + H(q(\mathbf{x}|o)) \right) + H(q(o)), \end{aligned} \tag{2.2}$$

where we used Bayes' theorem in the second equality,

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

Next, we introduce a helper distribution $\tilde{q}(o|\mathbf{x})$ to equation 2.2

$$\begin{aligned} L(\theta) &= \sum_o q(o) \left[\int q(\mathbf{x}|o)(R(\mathbf{x}) + \log \tilde{q}(o|\mathbf{x}) + \log q(o|\mathbf{x}) - \log \tilde{q}(o|\mathbf{x}))\, d\mathbf{x} + H(q(\mathbf{x}|o)) \right] \\ &\quad + H(q(o)) \\ &= \underbrace{\sum_o q(o) \left[\int q(\mathbf{x}|o)(R(\mathbf{x}) + \log \tilde{q}(o|\mathbf{x}))\, d\mathbf{x} + H(q(\mathbf{x}|o)) \right] + H(q(o))}_{\tilde{L}(\theta)} \\ &\quad + \int q(\mathbf{x})\ \mathrm{KL}(q(o|\mathbf{x})||\tilde{q}(o|\mathbf{x}))\, d\mathbf{x}. \end{aligned} \tag{2.3}$$

$\tilde{L}(\theta)$ is a lower bound to $L(\theta)$ because the KL divergence is by definition nonnegative, which means that the final term in equation 2.3 is also nonnegative. This trick allows us to maximize the ELBO by updating each component in the GMM independently of each other, which was not possible before due to the occurrence of $q(o|\mathbf{x})$.

We can now perform our optimization by iteratively alternating between tightening the lower bound $\tilde{L}(\theta)$ and maximizing $\tilde{L}(\theta)$, similar to the expectation-maximization algorithm [8]. We tighten the lower bound by setting $\tilde{q}(o|\mathbf{x}) = q(o|\mathbf{x})$. This causes $\mathrm{KL}(q(o|\mathbf{x})||\tilde{q}(o|\mathbf{x})) = 0$ in equation 2.3 and consequently $\tilde{L}(\theta) = L(\theta)$. We then keep $\tilde{q}(o|\mathbf{x})$ fixed and maximize $\tilde{L}(\theta)$ in two steps, first with respect to the component means μ_o and covariance matrices Σ_o of $q(\mathbf{x}|o)$, and after that with respect to the weights π_o. This procedure will be explained in the following subsections in detail.

2.2.1. Component update

Maximizing $\tilde{L}(\theta)$ with respect to the means and covariance matrices could be done by maximizing the term in square brackets in equation 2.3, as $q(o)$ and $H(q(o))$ that appear outside the square brackets do not depend on them. This allows us to optimize each component independently. Our optimization problem for a given component o is given by

$$\begin{aligned} &\mathrm{argmax}_{q(\mathbf{x}|o)} \int q(\mathbf{x}|o)(R(\mathbf{x}) + \log \tilde{q}(o|\mathbf{x}))\, d\mathbf{x} + H(q(\mathbf{x}|o)) \\ &\qquad \text{s.t. } \mathrm{KL}(q(\mathbf{x}|o)\,||\,q^{(i)}(\mathbf{x}|o)) < \epsilon(o). \end{aligned} \tag{2.4}$$

We add a constraint on the new distribution that ensures that it stays close to the previous distribution $q^{(i)}(\mathbf{x}|o)$. This type of constraint originates from the field of mathematical

optimization and is called a trust-region constraint. The upper bound $\epsilon(o)$ of the KL divergence controls the amount of exploration or exploitation desired and is adapted in every iteration.

To solve this optimization problem, we use a stochastic search algorithm, Model-Based Relative Entropy Stochastic Search (MORE) introduced by Abdolmaleki et al. [9], which gives us the solution in closed form. We first fit a quadratic surrogate

$$\tilde{R}(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^T\mathbf{R}\mathbf{x} + \mathbf{x}^T\mathbf{r} \approx R(\mathbf{x}) + \log \tilde{q}(o|\mathbf{x})$$

using weighted least squares, which makes use of importance sampling to enable reusing samples from previous iterations instead of having to draw new samples in each iteration. We then apply MORE to get the solution, which is a Gaussian distribution with natural parameters

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

where $\mathbf{Q}^{(i)}, \mathbf{q}^{(i)}$ are the natural parameters of the Gaussian approximation from the previous iteration, $\mathbf{R}^{(i)}, \mathbf{r}^{(i)}$ are the parameters of the quadratic surrogate $\tilde{R}(\mathbf{x})$, and η is the Lagrangian multiplier for the constraint on the KL divergence that could be found by minimizing the dual problem. For the complete derivation, we refer to [1].

2.2.2. Weight update

After the component updates, we now optimize $\tilde{L}(\theta)$ with respect to the component weights. We substitute

$$R(o) = \int q(\mathbf{x}|o)(R(\mathbf{x}) + \log \tilde{q}(o|\mathbf{x}))\, d\mathbf{x} + H(q(\mathbf{x}|o))$$

from equation 2.4 that we previously optimized into $\tilde{L}(\theta)$ and get

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

Our optimization problem is now given by

$$\text{argmax}_{q(o)} \sum_o q(o)R(o) + H(q(o)). \tag{2.5}$$

$R(o)$ contains integrals that could not be computed in closed form, therefore we approximate $R(o)$ using importance sampling similarly to section 2.2.1, which allows us to reuse samples from previous iterations. The optimal solution to this problem could be given in closed form, which is

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

where $\tilde{R}(o)$ is the estimate of $R(o)$ calculated using importance sampling.

2.2.3. Adapting the number of components

Throughout the whole optimization process, we will have to adapt the number of components to maintain an accurate approximation. We delete components that do not contribute much to the approximation due to having low weight or not showing enough improvement with respect to the objective function, and add components by drawing samples from the sample database and initializing a new component at the location of the sample with highest initial reward, which is computed using the formula

$$\tilde{R}_{\mathbf{x}_s}(o_n) = R(\mathbf{x}_s) - \log\left((1 - q(o_n))q(\mathbf{x}_s) + q(o_n)\exp\left\{\frac{1}{2}D - H_{\text{init}}\right\}\right)$$

where $\mathbf{x}_s$ denotes a drawn sample, o_n denotes the component to be intialized, D is the number of dimensions, and H_{init} is the initial entropy that is computed as a weighted average of the entropy of all previous components. We add and delete components at every predetermined n_{add} and n_{del} iterations.

2.3. Parallel Tempering

To encourage exploration of the search space, we leverage an approach used in global optimization known as parallel tempering.

Parallel tempering was first introduced in the context of MCMC by Geyer [10]. Here, parallel tempering MCMC runs multiple MCMCs at T different temperatures, and each MCMC samples from a modified posterior distribution. Periodically, the systems at different temperatures exchange configurations. The systems with lower temperatures are able to explore the search space locally while the systems with high temperatures are generally

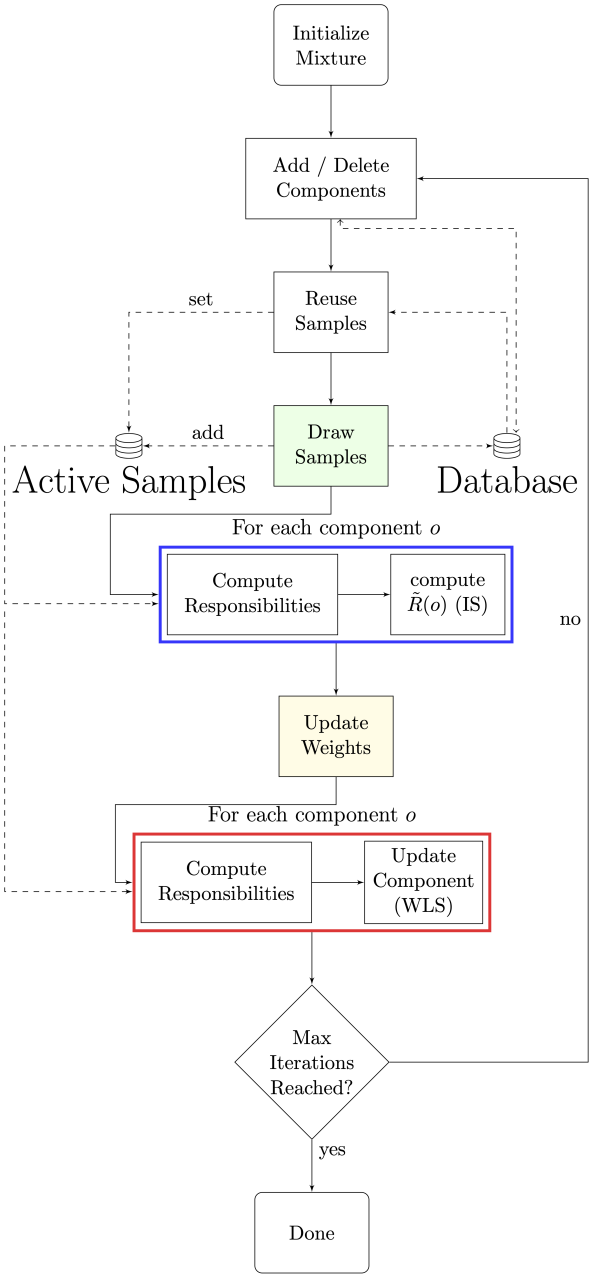

Figure 2.1.: Flowchart of the complete VIPS algorithm [1]

able to sample larger volumes of the parameter space and allow the systems with lower temperatures to escape from local minima. The simulation of T replicas requires on the order of T times more computational effort, but it has been shown that a parallel tempering simulation is more than $1/T$ times more efficient than a standard, single-temperature Monte Carlo simulation. This increased efficiency derives from allowing the lower temperature systems to sample regions of phase space that they would not have been able to access had regular sampling been conducted for a single-temperature simulation that was T times as long [4]. It is also possible to decrease computation time by running multiple simulations in parallel using CPU clusters.

3. Method

3.1. Parallel Tempering

3.1.1. Basic algorithm

To apply parallel tempering, we need to first determine a sequence of temperatures $\beta = (\beta_1, \ldots, \beta_n)$ with $\beta_1 > \cdots > \beta_n = 1$. This will be used to scale our target distribution $R(\mathbf{x})$, resulting in n target distributions $\frac{1}{\beta_1}R(\mathbf{x}), \ldots, \frac{1}{\beta_n}R(\mathbf{x})$ that will be used throughout VIPS. We then initialize n GMMs with the same initial parameters, and run VIPS in the following way:

Input: Number of iterations N_i
Input: Initial GMM parameters $\theta_1, \ldots, \theta_n$
Input: Temperatures $\beta = (\beta_1, \ldots, \beta_n)$
Initialize GMMs $q(\mathbf{x}|\theta_1), \ldots, q(\mathbf{x}|\theta_n)$
for $i = 1, \ldots, N_i$ **do**
    for $j = 1, \ldots, n$ **do**
        Perform one VIPS iteration using modified reward function $\frac{1}{\beta_j}R(\mathbf{x})$
    end
end

Algorithm 1: Parallel tempering VIPS

For each iteration of the outer loop, we run one VIPS iteration for every temperature in the order of decreasing temperature. Figure 2.1 illustrates how an iteration in VIPS is executed. Each iteration runs independently of each other and therefore optimizes each GMM independently, while sharing a single sample database.

By using new target distributions $\frac{1}{\beta_j}R(\mathbf{x})$, we create replicas of the original target $R(\mathbf{x})$ with higher entropy. As a consequence, as new samples keep being drawn from the

different GMMs and stored in the database, each GMM will have access to a larger set of samples. The GMMs with lower temperatures benefit from this, as they now have access to samples from GMMs with higher temperatures which cover a larger volume of the search space. We run the inner loop in the order of decreasing temperature so that the GMMs with lower temperatures could reuse samples from the GMMs with higher temperatures directly after they have just been updated. These samples could help prevent the GMMs with lower temperatures from getting stuck in local optima. Furthermore, an increase in temperature also causes the different modes to to overlap, making them easier to find.

3.1.2. Adding components using samples from parent GMM

Previously, to add new samples, we draw a predetermined number of random samples from the sample database, and initialize a new component at the location of the sample that has the highest initial reward as described in section 2.2.3. We could now take advantage of having multiple GMMs in the process of adding new components by only using samples from the GMM with the next higher temperature as candidates for the mean of the new component. This causes the new component to be initialized within the region of a component in this GMM and makes the search more focused. This could be done with every GMM except for the one with the highest temperature.

3.2. Tree Structure

3.2.1. KL divergence to parent component as penalty

In addition to using multiple temperatures, we also want to implement a tree structure on the component of the different GMMs, where one component in a GMM could have one or more "children" components on the next lower level. To achieve this, we introduce a penalty term to the component update objective function in equation 2.4, so that in addition to the trust region constraint, we also penalize a component for having a large KL divergence to its "parent". Our optimization problem becomes

$$\text{argmax}_{\theta_o} \int q(\mathbf{x}|o,\theta_o)(R(\mathbf{x}) + \log \tilde{q}(o|\mathbf{x}))\, d\mathbf{x} + H(q(\mathbf{x}|o,\theta_o)) - \alpha\, \text{KL}(q(\mathbf{x}|o,\theta_o) \,||\, q^{\uparrow}(\mathbf{x}|o))$$
$$\text{s.t. KL}(q(\mathbf{x}|o,\theta_o) \,||\, q(\mathbf{x}|o,\theta^{(i)})) \leq \epsilon(o)$$

which is equivalent to

$$\text{argmax}_{\theta_o} \int q(\mathbf{x}|o,\theta_o)(R(\mathbf{x}) + \log \tilde{q}(o|\mathbf{x}) + \alpha \log q^{\uparrow}(\mathbf{x}|o))\, d\mathbf{x} + (1+\alpha)H(q(\mathbf{x}|o,\theta_o))$$
$$\text{s.t. KL}(q(\mathbf{x}|o,\theta_o) \,||\, q(\mathbf{x}|o,\theta^{(i)})) \leq \epsilon(o), \tag{3.1}$$

where $q^{\uparrow}(\mathbf{x}|o)$ denotes the parent component of the current component, and α controls how strictly the penalty should be enforced. We can solve this optimization problem using MORE by first fitting a quadratic surrogate $\tilde{R}(\mathbf{x}) \approx R(\mathbf{x}) + \log \tilde{q}(o|\mathbf{x}) + \alpha \log q^{\uparrow}(\mathbf{x}|o)$.

We also make a small modification to MORE by multiplying the entropy term by $1+\alpha$. The full derivation of the solution to this new version of MORE can be found in Appendix A, which closely follows the derivation of the solution to 2.4 presented in [1].

To keep track of the parent of each component, we assign unique identifiers for each component in a GMM and keep a mapping from the component ids in one GMM to the id of its parent component in the GMM with the next higher temperature.

Here, every time a component is deleted, we also delete all its children. Therefore, we need an additional condition that a component could be deleted if all its children have low weight. This prevents the case that one temperature level has an empty GMM caused by its last component being deleted because its parent was deleted. In addition, we occasionally readjust the parent-children mapping by matching a child component with a component in the GMM on the next higher level that has the smallest KL divergence to it at the current iteration.

We also alter algorithm 1 so that instead of running one VIPS iteration for each temperature level, we now run 1000 iterations for each temperature level.

This tree structure allows for more efficient exploration and optimization, as the penalty term encourages the components on lower levels to stay close to their "parent" components and prevents different components from overlapping. This way, the GMMs with higher temperatures perform exploration, while the components in the GMMs with lower temperatures perform exploitation within the region of their parent components.

Below is a visualization of this tree structure in two dimensions. The setup consists of three GMMs with temperatures $150, 20$ and 1 respectively, where the components are shown as ellipses, and the parent-child relationships are shown using the black lines connecting two components. The target distribution is a GMM that is shown in black, but is barely visible here.

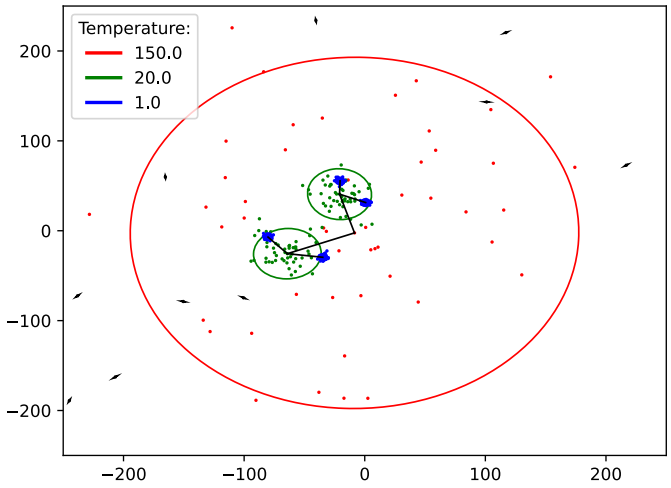

Figure 3.1.: A visualization of the tree structure in two dimensions

3.2.2. Softplus reward

We define a function

$$f(\mathbf{x}) = \sqrt{(\mathbf{x} - \mu^{\uparrow})^T (\mathbf{\Sigma}^{\uparrow})^{-1} (\mathbf{x} - \mu^{\uparrow})} - c$$

where the first term represents the Mahalanobis distance function, and apply the softplus function $g(\mathbf{x}) = \log(e^{\mathbf{x}} + 1)$ to $f(\mathbf{x})$. The softplus function is a smooth approximation of ReLU, which equals to zero if x is negative, and equals to x otherwise.

We now use $g(f(\mathbf{x}))$ to replace $\log q^{\uparrow}(\mathbf{x}|o)$ in equation 3.1. Intuitively, we implement here a penalty on our optimization objective that is linear when $\mathbf{x}$ is far from the parent mean. But if $\mathbf{x}$ is close to the parent mean, or more precisely, the Mahalanobis distance is smaller than c, $f(\mathbf{x})$ will be negative, and $g(f(\mathbf{x}))$ will be zero. This causes this penalty term to vanish. Therefore, when $\mathbf{x}$ is sufficiently close to the parent mean, we solely optimize our new component to approximate our target distribution.

4. Experiments and Results

4.1. Experiments

Our method is tested on hard exploration problems using two multimodal target distributions. We compare the exploration capacity of our work with the initial version of VIPS by calculating the number of modes successfully found by our model. The experiments were conducted on the on the high-performance computer Lichtenberg at the NHR Centers NHR4CES at TU Darmstadt.

4.1.1. Gaussian mixture model

For this experiment, our aim is to approximate an unknown 20-dimensional GMM with 10 components. We initialize the mean by randomly sampling each component within the interval $[-500, 500]$ and initialize the covariance matrix by computing $\mathbf{\Sigma} = \mathbf{X}^T\mathbf{X} + \mathbf{I}$, where each entry in $\mathbf{X}$ is sampled from $\mathcal{N}(0, 0.04)$. Additionally, we make the covariance matrix more singular by multiplying its smallest eigenvalue by 0.0001. We say that a mode μ_t of the target is found if there exists a component in our model, so that $||\mu_t - \mu|| < \phi$ for the component mean μ and a threshold value ϕ. We set the distance threshold $\phi = \|6 \cdot \mathbb{1}_{20}\|_2 = 26.83$, where $\mathbb{1}_{20}$ is a 20-dimensional vector containing ones. This experiment is suitable for testing exploration capacity due to the distant modes of the GMM and high correlation in the covariance matrices of the components.

4.1.2. Planar robot

The planar robot experiment is an experiment involving robot arms based at $[0, 0]$ with 10 links, each with length 1, and 9 joints. We have 4 goal positions in coordinates $[7, 0], [0, 7], [-7, 0]$ and $[0, -7]$ which are visualized in Figure 4.1 as red X's that we want

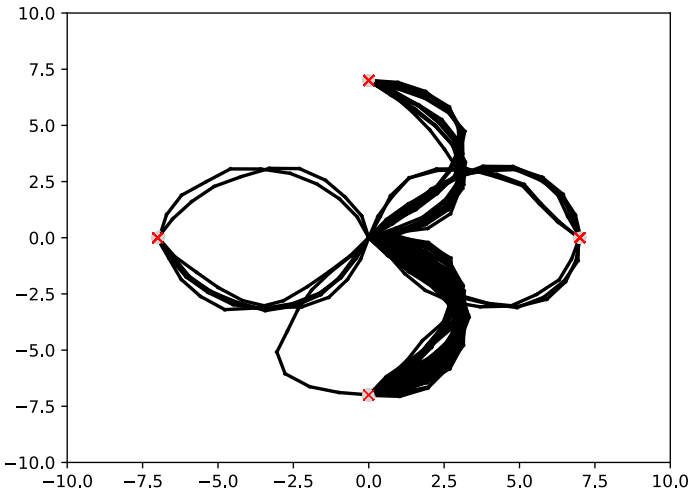

Figure 4.1.: Visualization of the planar robot setup

to reach using our robot endeffector. For each goal position, we only consider two ways to reach the goal position, namely from the left and right, or from the top and bottom. In this experiment, we construct a probability distribution for the joint angles where we use a Gaussian prior for smoothness with zero mean and variance 1 for the first joint and 0.04 for the remaining ones. We also penalize deviations from each goal position using a Gaussian on the endeffector space with the goal position as the mean and covariance matrix $0.0001 \cdot \mathbf{I}$. We consider a mode in this target distribution to be found if there is a component in our model where the first component of its mean (corresponding to the first joint angle) is in a plausible range of angles ($\pm 2\pi$) to reach the goal position from one of the two sides, and the endeffector position, also calculated using the same mean, is sufficiently close to the goal position. Therefore, there are in total 24 modes that could be found.

4.2. Results

Each experiment was run 40 times. For each experiment except the baseline experiments, we use the sequence of temperatures $\beta = (150, 20, 1)$. We plot the number of modes detected by our models and $-L(\theta)$ or $-$ELBO with respect to the number of iterations. In each plot, the mean value is shown along with a standard error interval.

4.2.1. GMM experiment

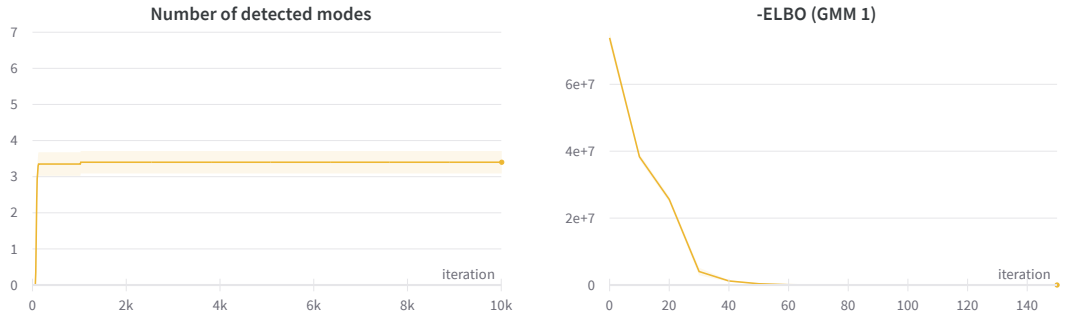

Figure 4.2.: Baseline

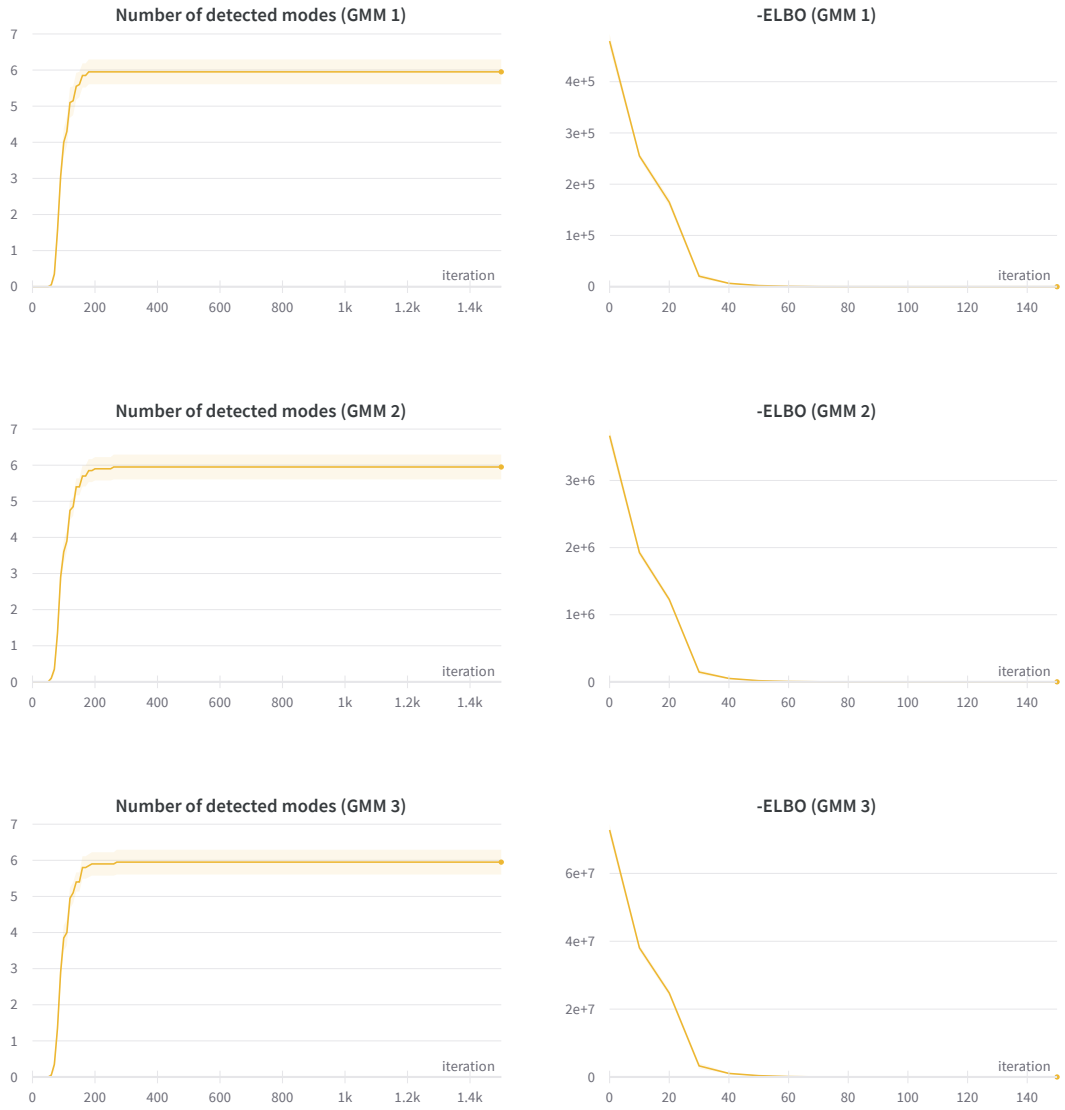

Figure 4.3.: Parallel tempering: Basic algorithm

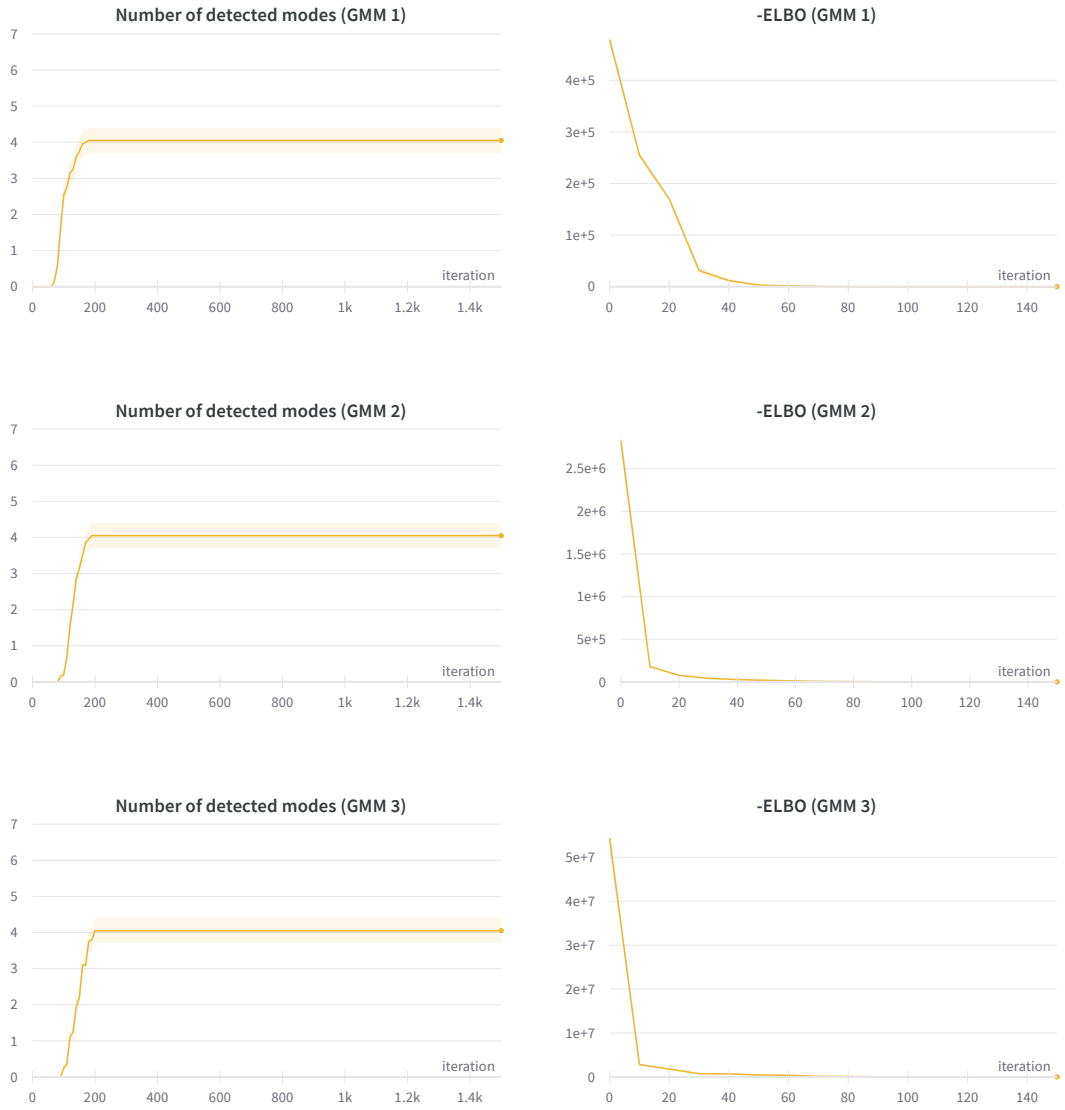

Figure 4.4.: Parallel tempering: Sampling from parent GMM

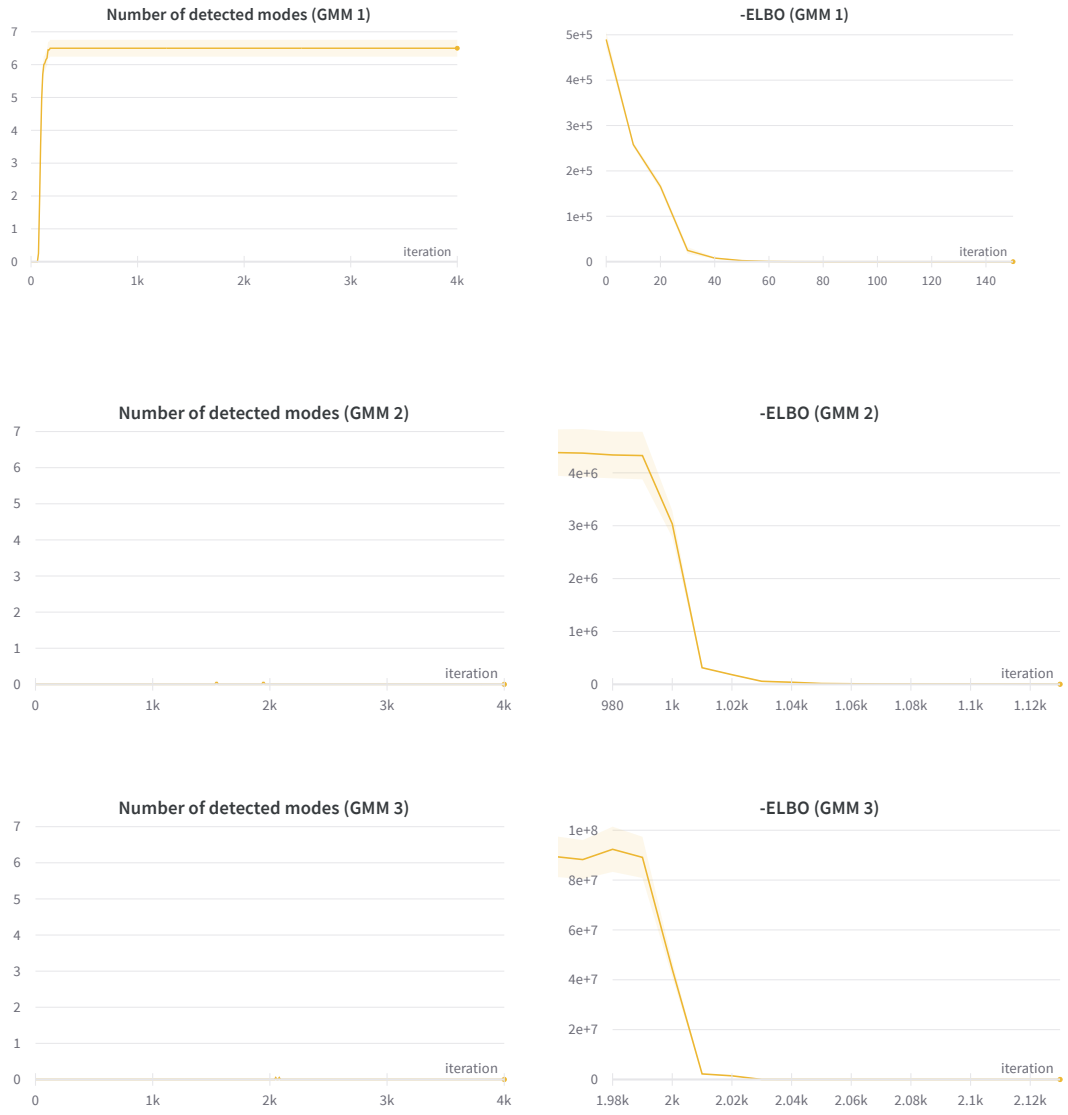

Figure 4.5.: Tree structure: KL divergence to parent component as penalty

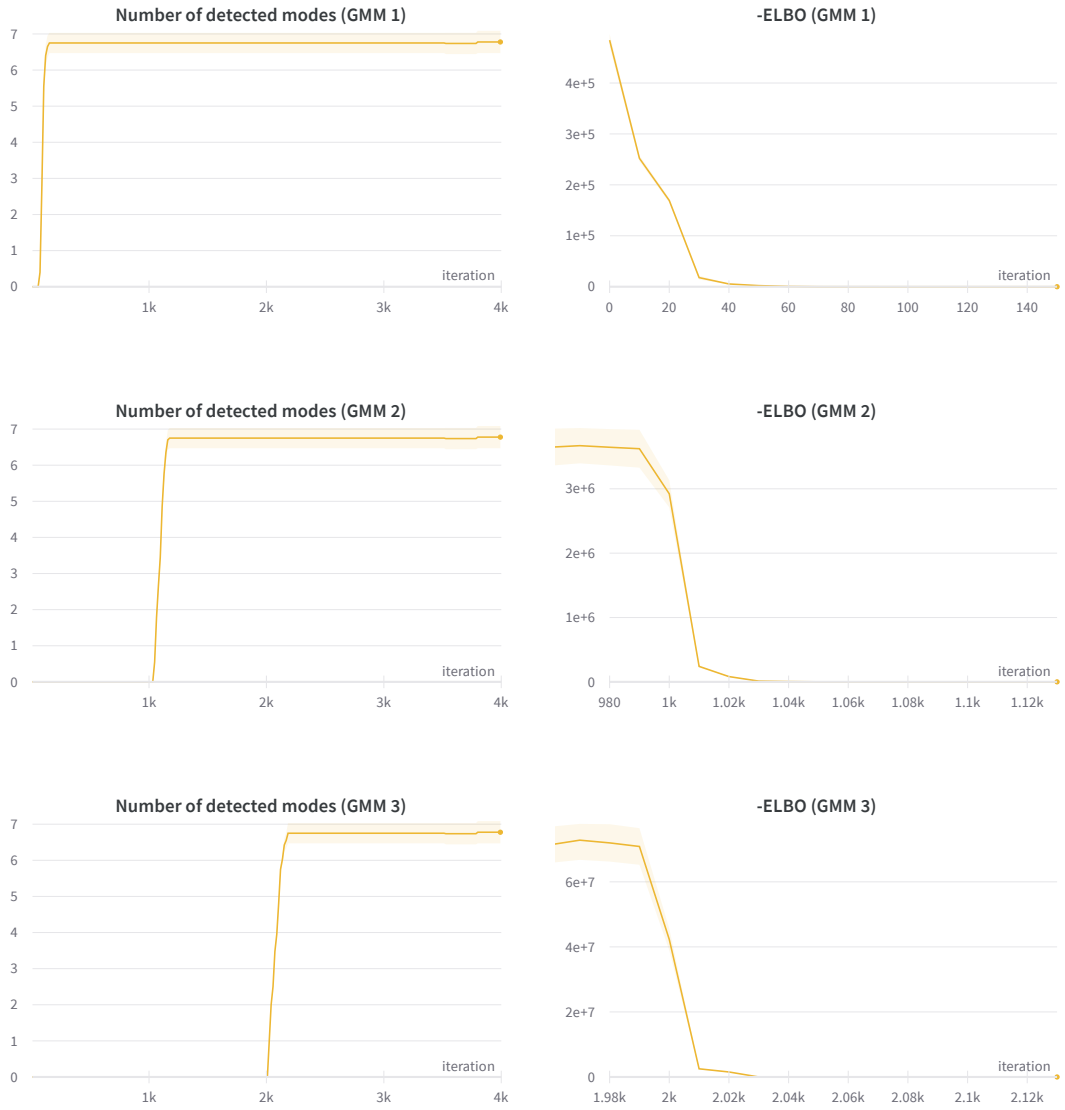

Figure 4.6.: Tree structure: Softplus reward

4.2.2. Planar robot experiment

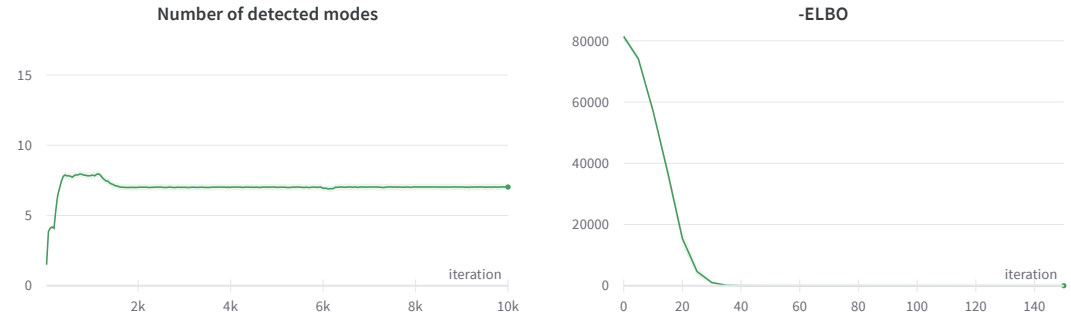

Figure 4.7.: Baseline

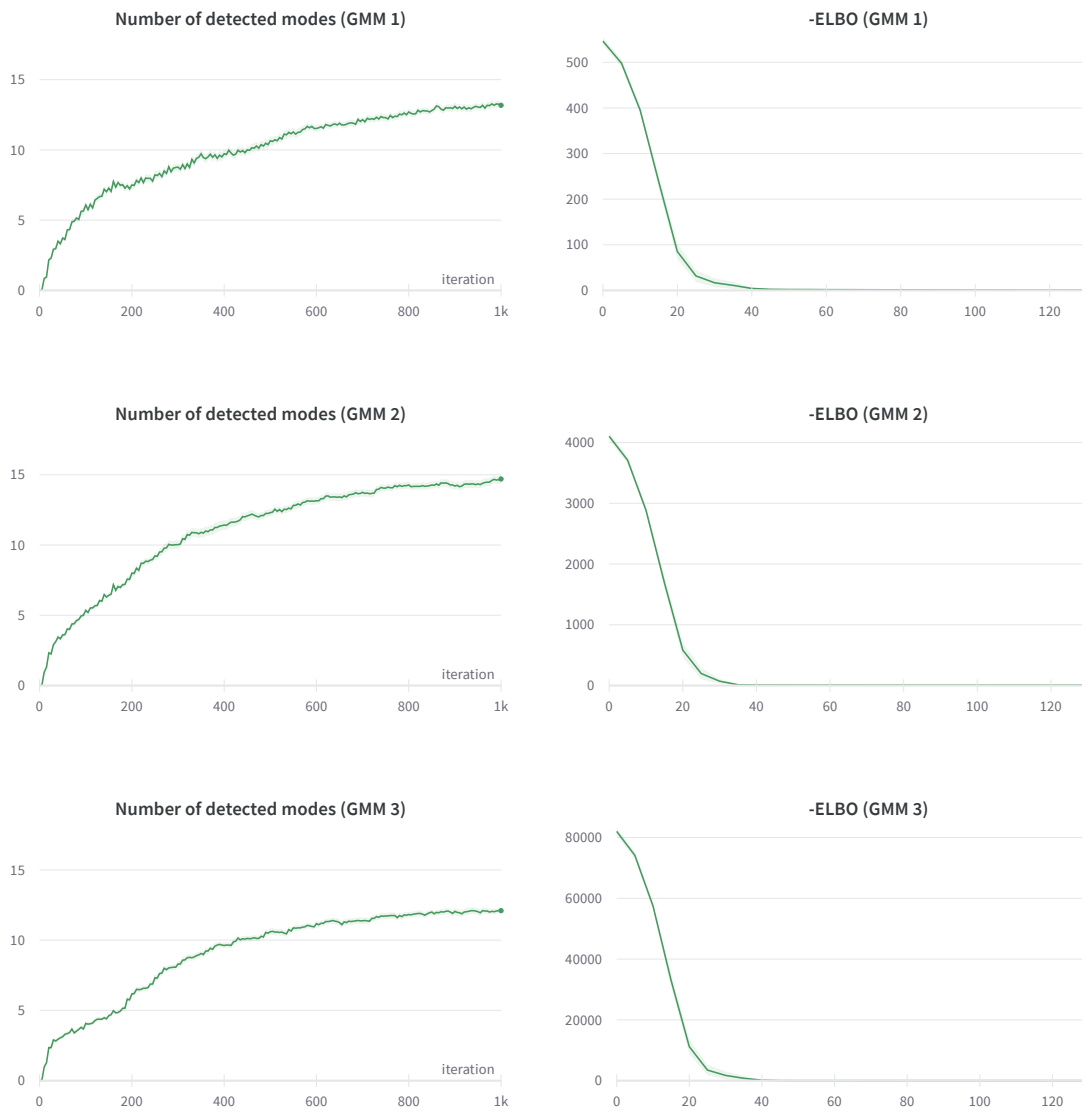

Figure 4.8.: Parallel tempering: Basic algorithm

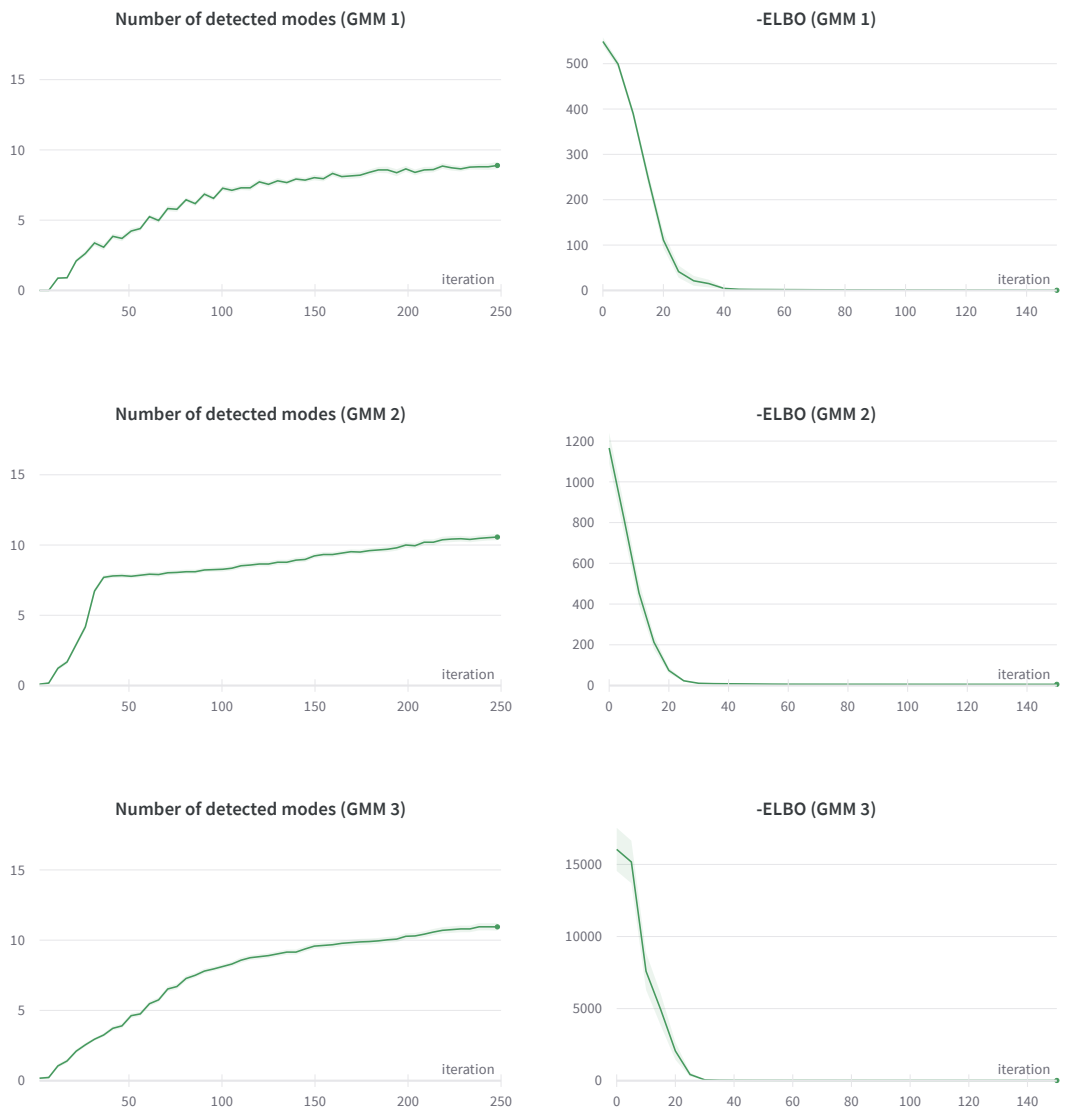

Figure 4.9.: Parallel tempering: Sampling from parent GMM

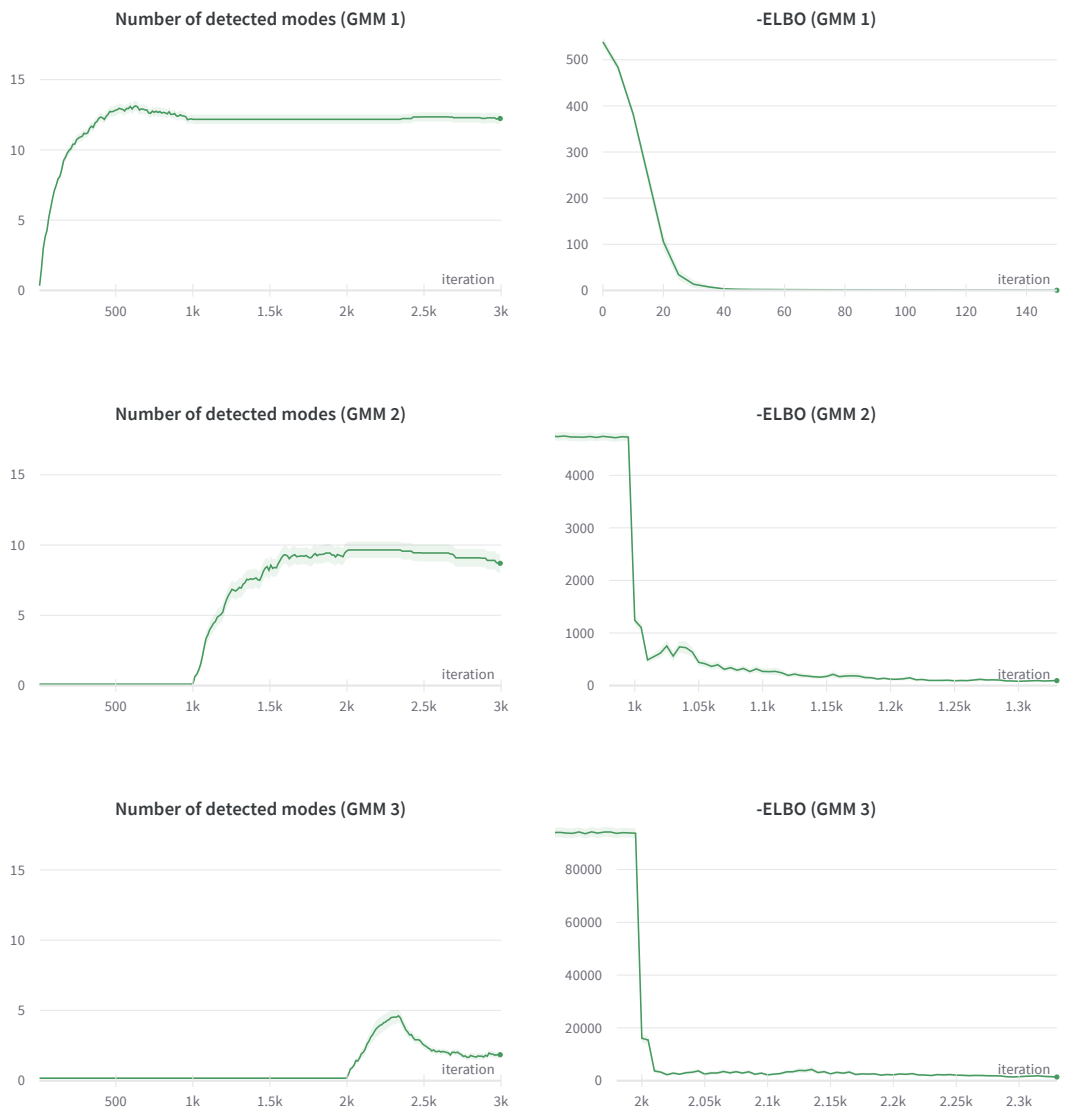

Figure 4.10.: Tree structure: KL divergence to parent component as penalty

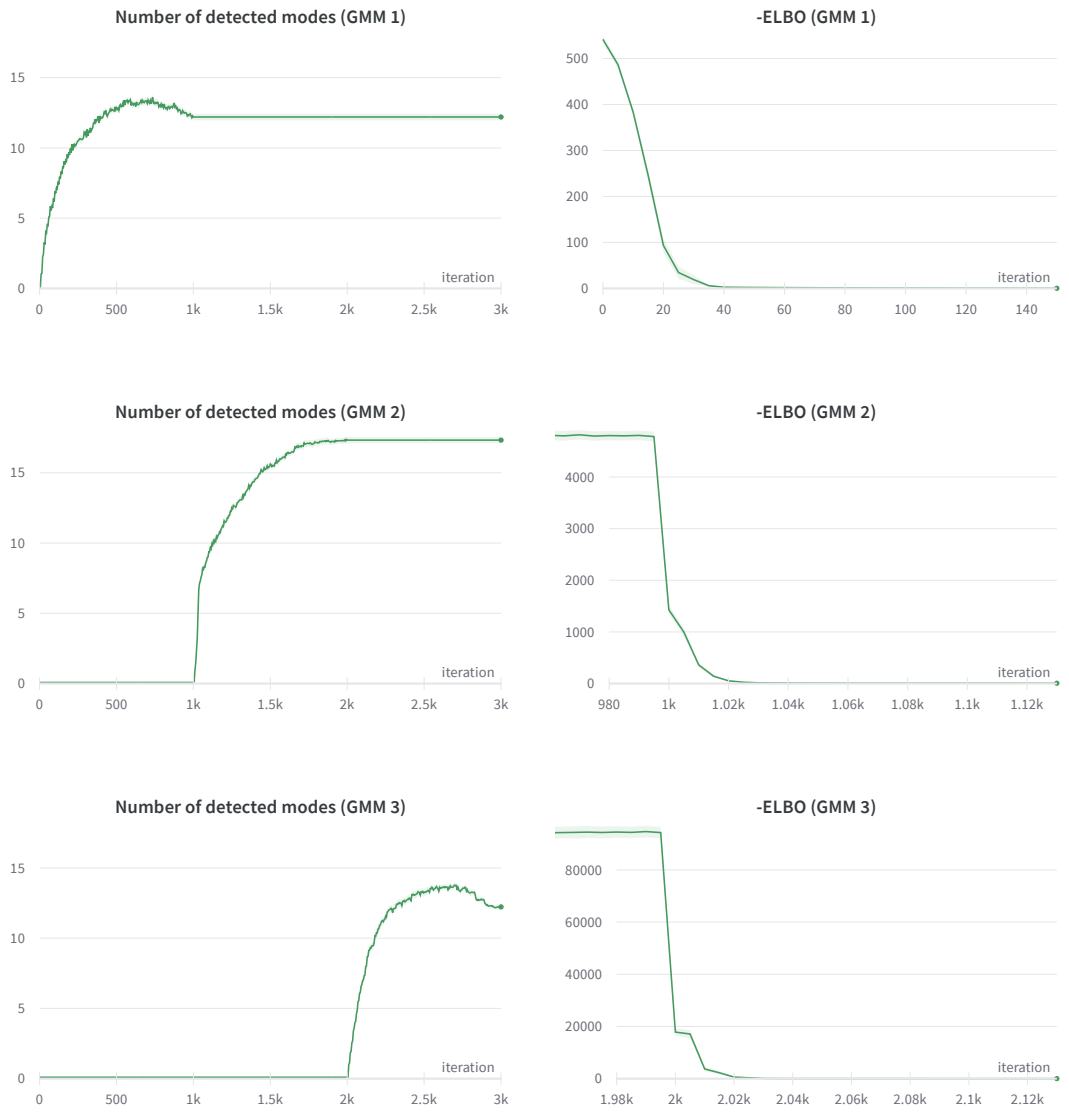

Figure 4.11.: Tree structure: Softplus reward

We first tested VIPS in the original form on the GMM experiment and planar robot experiment that we described in section 4.1, and the results are given by figures 4.2 and 4.7. We see here that VIPS is only able to find a small fraction of the total number of modes.

Next, we tested using multiple GMMs with scaled reward functions and a shared sample database, as described in section 3.1.1. We could see in the results in figures 4.3 and 4.8, that VIPS is able to find much more modes than the baseline, which indicates that using parallel tempering improves the exploration capacity of VIPS.

Subsequently, we test our changes to the heuristic for adding new components, where one GMM only uses samples from the GMM on the next higher temperature level as candidates for the mean of the new component. In figure 4.4 and 4.9, we observe that our models could find slightly less modes than basic parallel tempering, but still more than in the baseline experiments.

We then evaluated our tree structure using the KL divergence to the parent component as penalty. Our models did not perform well in this experiment. In the GMM experiment (figure 4.5), the GMMs with temperature 20 and 1 could not find any modes at all, while in the planar robot experiment (figure 4.10), GMM 3 finds significantly less modes than the baseline.

Finally, we evaluated our softplus reward function which is used to replace the log density of the parent component in the previous scenario. It is clear from figures 4.6 and 4.11 that this version performs best and could find the most modes out of all our methods.

Below, we present our data at the end of training in tabular form, which includes the ELBO values of GMM 3, which is the GMM with temperature 1, and the number of modes detected by all the models. The data is shown along with the standard error. In the leftmost column, 1 stands for the baseline version, 2 stands for parallel tempering (basic algorithm), 3 stands for parallel tempering (adding components using samples from parent GMM), 4 stands for tree structure (KL divergence to parent component as penalty), and 5 stands for tree structure (softplus reward).

	GMM	Planar Robot
1	1.72 ± 0.1	12.48 ± 0.028
2	0.98 ± 0.062	12.67 ± 0.15
3	1.85 ± 0.21	12.81 ± 0.019
4	848.75 ± 126.67	1199.46 ± 318.01
5	1.12 ± 0.07	12.33 ± 0.02

Table 4.1.: ELBO values for GMM 3 (GMM with temperature 1)

	GMM			Planar Robot		
1	3.4 ± 0.31			7.03 ± 0.21		
2	5.95 ± 0.34	5.95 ± 0.34	5.95 ± 0.34	13.18 ± 0.25	14.7 ± 0.3	12.1 ± 0.19
3	4.05 ± 0.34	4.05 ± 0.34	4.05 ± 0.34	8.89 ± 0.25	10.57 ± 0.22	10.95 ± 0.26
4	6.5 ± 0.26	0	0	12.23 ± 0.33	8.7 ± 0.58	1.84 ± 0.19
5	6.78 ± 0.29	6.78 ± 0.29	6.78 ± 0.29	12.2 ± 0.24	17.33 ± 0.23	12.23 ± 0.16

Table 4.2.: Number of detected modes for every model

5. Discussion

From our results, it could be seen that parallel tempering generally enables our model to find more modes than the original VIPS algorithm. In every experiment except for using the KL divergence to parent component as penalty, the number of detected modes are higher than in the baseline experiment. This indicates that parallel tempering could be utilized to improve exploration in VIPS.

One plausible reason for the bad results in the experiment using KL divergence penalty term is that the optimization problem is biased due to the penalty term. The new component has to compromise between accurately approximating the target distribution and being similar to its parent component, which is detrimental to the performance in finding modes of the target distribution. This is also reflected in the ELBO values in table 4.1. This problem is however fixed by using softplus reward, where within a certain region of the parent component, the penalty term does not contribute to the optimization anymore, as discussed in 3.2.2.

Looking at table 4.2, we could see that using softplus reward brings the most benefit, followed by basic parallel tempering using a shared sample database. It is then followed by parallel tempering with adding components using samples from parent GMM. This method likely did not perform as well as basic parallel tempering, as only relying on the parent GMM for samples to add new components does not allow the GMMs with lower temperature levels to find new modes that the parent GMM has not found yet.

The ELBO plots show that the optimization process is not hindered by parallel tempering. We could see in table 4.1 that the ELBO of our models with temperature 1 converge to roughly the same values as the baseline ELBO. This shows that using parallel tempering still results in the KL divergence between our model and the target distribution, as given in equation 2.1, being minimized.

6. Conclusion

In this work, we aimed to investigate how the VIPS algorithm could be extended to improve exploration of the search space. We implemented four approaches, each building on top of each other. First, we used multiple models, each with their own reward functions scaled by the corresponding temperatures and a single shared sample database. Next, we altered the heuristic for adding new components so that one GMM only uses samples from the GMM on the next higher temperature level as candidates for the mean of the new component. We then implemented a tree structure by connecting two components on consecutive temperature levels and applying a penalty term on the component update objective function to bound the KL divergence between a component and its parent component. Finally, we used a softplus function as a reward term to replace the log density of the parent component that previously appeared because of the KL divergence penalty term.

We found that parallel tempering is generally beneficial in improving exploration for the VIPS algorithm, where using the softplus reward allows VIPS to find the most modes. However, we could not show a benefit in using the KL divergence from one component to its parent as a penalty term. This is likely caused by the optimization having to compromise between two different objectives.

6.1. Future Work

One interesting point to research further is how bandit algorithms could be applied to VIPS to further balance exploration and exploitation. This could be used to get a more principled way to add new components, for example by balancing between adding components in regions with high target density and adding components at regions that are not yet explored by our model. It should also be possible to exploit our tree structure to optimize the GMMs for non-overlapping modes independently.

Bibliography

[1] O. Arenz, M. Zhong, and G. Neumann, "Trust-Region Variational Inference with Gaussian Mixture Models," in *Journal of Machine Learning Research (JMLR)*, 2020.

[2] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational Inference: A Review for Statisticians," *Journal of the American Statistical Association*, 2017.

[3] M. Sambridge, "A Parallel Tempering Algorithm for Probabilistic Sampling and Multimodal Optimization," *Geophysical Journal International*, 2013.

[4] D. J. Earl and M. W. Deem, "Parallel tempering: Theory, applications, and new perspectives," *Phys. Chem. Chem. Phys.*, 2005.

[5] T. Rainforth, Y. Zhou, X. Lu, Y. W. Teh, F. Wood, H. Yang, and J.-W. van de Meent, "Inference Trees: Adaptive Inference with Exploration," 2018.

[6] C.-W. Huang, S. Tan, A. Lacoste, and A. Courville, "Improving Explorability in Variational Inference with Annealed Variational Objectives," 2018.

[7] R. Williams and J. Peng, "Function Optimization Using Connectionist Reinforcement Learning Algorithms," *Connection Science*, 1991.

[8] C. M. Bishop, "Pattern Recognition and Machine Learning (Information Science and Statistics)," 2006.

[9] A. Abdolmaleki, R. Lioutikov, J. Peters, N. Lau, L. Reis, and G. Neumann, "Model-Based Relative Entropy Stochastic Search," in *Advances in Neural Information Processing Systems (NIPS / NeurIPS)*, MIT Press, 2015.

[10] C. J. Geyer, "Markov Chain Monte Carlo Maximum Likelihood," *Computing Science and Statistics: Proceedings of the 23rd Symposium on the Interface*, 1991.

[11] S. Boyd and L. Vandenberghe, "Convex Optimization," 2004.

A. MORE Derivation

We want to solve the constrained optimization problem

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

where $\tilde{R}(\mathbf{x})$ is a quadratic function, and $q(\mathbf{x})$ a Gaussian. The Lagrangian is then given by

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

We get the optimal distribution $q^\star(\mathbf{x},\eta,\lambda)$ by taking the derivative of the Lagrangian and setting it to zero

$$\begin{aligned} \frac{\partial \mathcal{L}(q,\eta,\lambda)}{\partial q(\mathbf{x})} &= \tilde{R}(\mathbf{x}) - (1+\alpha+\eta)\log q^\star(\mathbf{x}) + \eta \log q^{(i)}(\mathbf{x}) - \lambda - (1+\alpha+\eta) \\ &\implies q^\star(\mathbf{x},\eta,\lambda) = \exp\left(\frac{\tilde{R}(\mathbf{x}) + \eta \log q^{(i)}(\mathbf{x}) - \lambda - (1+\alpha+\eta)}{1+\alpha+\eta}\right). \end{aligned} \tag{A.1}$$

The Lagrange dual function is then given by

$$\mathcal{G}(\eta,\lambda) = \mathcal{L}(q^\star,\eta,\lambda) = (1+\alpha+\eta)\int q^\star(\mathbf{x},\eta,\lambda)\ d\mathbf{x} + \eta\epsilon + \lambda. \tag{A.2}$$

Due to Slater's condition [11], strong duality holds, which means that we could find $q^\star(\mathbf{x},\eta,\lambda)$ by minimizing A.2 with respect to η and λ, and then using the optimal $\eta^\star$

and $\lambda^\star$ to compute the optimal solution $q^\star(\mathbf{x}, \eta, \lambda)$ according to A.1. To do this, we first compute the partial derivatives, which are given by

$$\begin{aligned}\frac{\partial \mathcal{G}(\eta, \lambda)}{\partial \eta} &= \epsilon - \int q^\star(\mathbf{x}, \eta, \lambda)(\log q^\star(\mathbf{x}, \eta, \lambda) - \log q^{(i)}(\mathbf{x}))\, d\mathbf{x} \\ &= \epsilon - \mathrm{KL}(q^\star(\mathbf{x}, \eta, \lambda) \mid\mid q^{(i)}(\mathbf{x}))\end{aligned}$$

and

$$\frac{\partial g(\eta, \lambda)}{\partial \lambda} = -\int q^\star(\mathbf{x}, \eta, \lambda)\, d\mathbf{x} + 1,$$

where the optimal Lagrange multiplier $\lambda^\star(\eta)$ for a given η normalizes $q^\star(\mathbf{x}, \eta, \lambda^\star)$, because

$$\frac{\partial g(\eta, \lambda)}{\partial \lambda} = 0 \Leftrightarrow \int q^\star(\mathbf{x}, \eta, \lambda)\, d\mathbf{x} = 1. \tag{A.3}$$

We can now perform coordinate descent by alternatively updating η along its partial derivative and computing the optimal λ. This corresponds to optimizing the dual function

$$\mathcal{G}(\eta) = (1 + \alpha + \eta) \int q^\star(\mathbf{x}, \eta, \lambda^\star)\, d\mathbf{x} + \eta\epsilon + \lambda^\star(\eta) = 1 + \alpha + \eta + \eta\epsilon + \lambda^\star(\eta) \tag{A.4}$$

based on

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

To calculate our optimal distribution, we first recall that $\tilde{R}(\mathbf{x})$ has the form

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

and express the previous approximation $q^{(i)}(\mathbf{x})$ in terms of its natural parameters $\mathbf{Q}^{(i)}$ and $\mathbf{q}^{(i)}$. We use this in A.1 to get

$$\begin{aligned}q^\star(\mathbf{x}, \eta) = \exp&\left(\frac{\eta \log Z(\mathbf{Q}^{(i)}, \mathbf{q}^{(i)}) - \lambda^\star(\eta) - 1 - \alpha - \eta}{1 + \alpha + \eta}\right) \cdot \\ &\exp\left(-\frac{1}{2}\mathbf{x}^T\frac{\mathbf{R}^{(i)} + \eta\mathbf{Q}^{(i)}}{1 + \alpha + \eta}\mathbf{x} + \mathbf{x}^T\frac{\mathbf{r}^{(i)} + \eta\mathbf{q}^{(i)}}{1 + \alpha + \eta}\right)\end{aligned} \tag{A.5}$$

which is Gaussian with natural parameters

$$\mathbf{Q}(\eta) = \frac{\eta}{1 + \alpha + \eta}\mathbf{Q}^{(i)} + \frac{1}{1 + \alpha + \eta}\mathbf{R}^{(i)} \quad \text{and} \quad \mathbf{q}(\eta) = \frac{\eta}{1 + \alpha + \eta}\mathbf{q}^{(i)} + \frac{1}{1 + \alpha + \eta}\mathbf{r}^{(i)}$$

Next, we use A.5 and A.3 to get

$$\begin{aligned}\lambda^{\star}(\eta) &= -(1+\alpha+\eta)\log\int \exp\left(-\frac{1}{2}\mathbf{x}^T\frac{\mathbf{R}^{(i)}+\eta\mathbf{Q}^{(i)}}{1+\alpha+\eta}\mathbf{x}+\mathbf{x}^T\frac{\mathbf{r}^{(i)}+\eta\mathbf{q}^{(i)}}{1+\alpha+\eta}\right)\,d\mathbf{x} \\ &\quad -(1+\alpha+\eta)+\eta\log Z(\mathbf{Q}^{(i)},\mathbf{q}^{(i)}) \\ &= \eta\log Z(\mathbf{Q}^{(i)},\mathbf{q}^{(i)})-(1+\alpha+\eta)\log Z(\mathbf{Q}(\eta),\mathbf{q}(\eta))-(1+\alpha+\eta). \qquad \text{(A.6)}\end{aligned}$$

Finally, we can now express the dual function using A.6 and A.4:

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