Dual Parameterization of Sparse Variational Gaussian Processes

Anonymous Author(s)
Affiliation
Address
email

Abstract

Sparse variational Gaussian process (SVGP) methods are a common choice for non-conjugate Gaussian process inference because of their computational benefits. In this paper, we improve their efficiency by using a dual parameterization where each data example is assigned dual parameters, similar to site parameters used in expectation propagation. Our dual parameterization speeds-up inference using natural gradient descent, and provides a tighter evidence lower bound for hyperparameter learning. Our approach matches the low memory cost of the current SVGP method, while providing faster and more accurate inference and learning.

1 Introduction

Gaussian processes (GPs, [31]) have become ubiquitous models in the probabilistic machine learning toolbox, but their application often struggles with two issues: poor $\mathcal{O}(n^3)$ scaling in the number of data points, n, and challenging approximate inference in non-conjugate (non-Gaussian) models. In recent years, variational inference has become the go-to solution to overcome these problems, where sparse variational GP methods [37] tackle both the non-conjugacy and computation cost. The computation is reduced to $\mathcal{O}(nm^2)$ by using a small number of $m \ll n$ *inducing-input* locations. For large machine learning problems, the SVGP framework is a popular choice as it enables fast stochastic training for prediction, and reduces the cost of hyperparameter learning and training inducing-inputs to $\mathcal{O}(m^3 + n_\mathrm{b} m^2)$ per step, where n_b is the batch size [12, 13].

It is a common practice in SVGP to utilize the standard mean-covariance parameterization which requires $\mathcal{O}(m^2)$ memory. Often in practice, inference is cast as an optimization problem involving an objective $\mathcal{L}(q)$ with a Gaussian q parameterized by parameters $\boldsymbol{\xi} = (\mathbf{m}, \mathbf{L})$ where $\mathbf{m}$ is the mean and $\mathbf{L}$ is the Cholesky factor of the covariance. Throughout, we will refer to this parameterization as q-SVGP. The advantage of this formulation is that, under mild assumptions, the objective is convex and gradient-based optimization works well [4]. The optimization can further be improved by using natural gradient descent (NGD) which is shown to be less sensitive to learning rates [33]. The NGD algorithm with mean-covariance parameterization is currently the state-of-the-art and implemented in the efficient software implementations such as GPflow [23] and GPyTorch [11].

An alternate parameterization to the mean-covariance parameterization is the one where every likelihood is assigned two sets of parameters, which requires $\mathcal{O}(n)$ memory. Its existence was initially shown by Csató and Opper [9] for general GP models, and later on extended to variational objectives for GPs [26, 27], and also to general class of models by using Lagrangian duality [20, 16]. Due to this later connection, we refer to this parameterization as the *dual parameterization* where the parameters are the Lagrange multipliers, ensuring that the marginal mean and variance of each likelihood term is consistent to those obtained by the full GP [16]. Expectation propagation (EP, [24]) too naturally employ such parameterization (unrelated to duality though) where it is referred to as the *site* parameters [24]. Due to this connection to EP, we will refer to this approach as t-SVGP (the

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

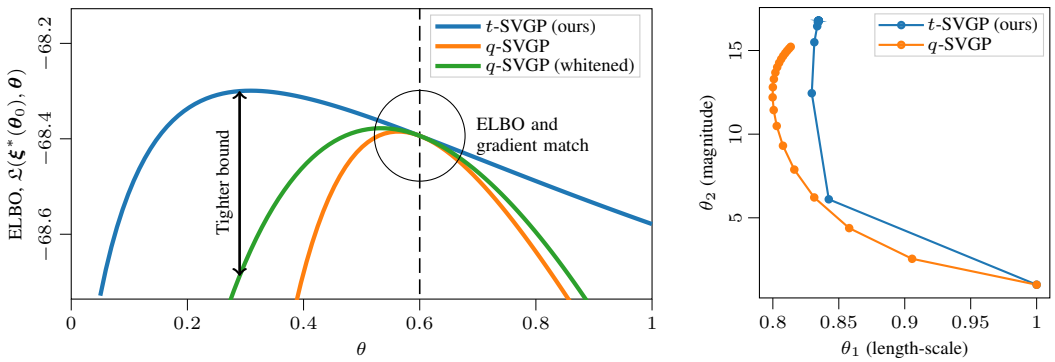


Figure 1: Our parameterization gives a tighter bound compared to standard SVGP (whitened/ unwhitened) as shown on the left for a GP classification tasks and varying kernel magnitude parameter θ. The tighter bound improves learning of hyperparameters, which is shown on the right for the *banana classification* task when performing coordinate ascent w.r.t. $\boldsymbol{\theta}$ and variational parameters $\boldsymbol{\xi}$.

letter 't' refers to the sites). To the best of our knowledge, the dual parameterization is not used in any recent works in SVGP.

The main contribution of this work is to introduce the dual parameterization for SVGP and show that it speeds up both the learning and inference. For inference, we show that the dual parameters are automatically obtained through a different formulation of NGD, written in terms of the expectation parameters [17, 15, 18]. The formulation is fast since it avoids the use of sluggish automatic differentiation to compute the natural gradients. For learning, we show that the dual parameterization results in a *tighter* lower bound to the marginal likelihood, which speed-up the hyperparameter optimization (see Fig. 1). We provide extensive evaluation on benchmark data sets, which confirms our findings. Our work attempts to revive the dual parameterization, which was popular in the early 2000s, but was somehow forgotten and not exploited in the recent SVGP algorithms.

2 Background: Variational Inference for Gaussian Processes Models

Gaussian processes (GPs, [31]) are distributions over functions, commonly used in probabilistic machine learning to endow latent functions in generative models with rich and interpretable priors. These priors provide strong inductive biases for regression tasks in the small data regime. GP models then employ a GP prior over the (latent) model functions $f(\cdot) \sim \mathcal{GP}(\mu(\cdot), \kappa(\cdot, \cdot'))$, where the prior is completely characterized by the mean function $\mu(\cdot)$ and covariance function $\kappa(\cdot, \cdot')$. We denote $f(\cdot)$ as a function evaluated at an input point $\mathbf{x} \in \mathbb{R}^d$, but occasionally suppress the notation to just f in the interest of reducing clutter. Given a data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ of input–output pairs, the function evaluation at $\mathbf{x}_i$ are passed through likelihood functions to model the outputs $y \in \mathbb{R}$, *e.g.* by specifying $p(\mathbf{y} \,|\, \mathbf{f}) := \prod_{i=1}^n p(y_i \,|\, f(\mathbf{x}_i))$. Prediction at a new test input $\mathbf{x}_*$ is obtained by computing the distribution $p(f(\mathbf{x}_*) \,|\, \mathcal{D}, \mathbf{x}_*)$. For Gaussian likelihoods $\mathrm{N}(y_i \,|\, f(\mathbf{x}_i), \sigma^2)$, is available in closed form as a Gaussian distribution $\mathrm{N}(f(\mathbf{x}_*) \,|\, m_{\text{GPR}}(\mathbf{x}_*), v_{\text{GPR}}(\mathbf{x}_*))$ with mean and variance defined as follows:

$$m_{\text{GPR}}(\mathbf{x}_*) := \mathbf{k}_{\mathbf{f}*}^\top (\mathbf{K}_{\mathbf{ff}} + \sigma^2 \mathbf{I}_n)^{-1} \mathbf{y}, \quad \text{and} \quad v_{\text{GPR}}(\mathbf{x}_*) := \kappa(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_{\mathbf{f}*}^\top (\mathbf{K}_{\mathbf{ff}} + \sigma^2 \mathbf{I}_n)^{-1} \mathbf{k}_{\mathbf{f}*}, \quad (1)$$

where $\mathbf{k}_{\mathbf{f}*}$ is a vector of $\kappa(\mathbf{x}_*, \mathbf{x}_i)$ as the ith element for all $\mathbf{x}_i \in \mathcal{D}$, and $\mathbf{K}_{\mathbf{ff}}$ is an $n \times n$ matrix with $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ as the ij^{th} entry. The hyperparameters of the kernel, denoted by $\boldsymbol{\theta}$, can be learned from the data by maximizing the log-marginal likelihood which has a closed-form expression for Gaussian likelihoods. Unfortunately, all of these quantities require inversion of matrix $\mathbf{K}_{\mathbf{ff}} + \sigma^2 \mathbf{I}_n$, which takes $\mathcal{O}(n^3)$ computations and is prohibitively expensive for large n. For non-Gaussian (non-conjugate) likelihoods, *e.g.*, in classification problems, the cost is even more prohibitive since learning requires iterative methods, such as expectation propagation (EP) or variational inference (VI), where the inversion is performed multiple times. Such high computational costs for GPs makes is a drawback in applying them to large real-world learning problems.

2.1 Variational Inference

Variational inference in probabilistic models turns inference into an optimisation problem [2]. The variational Gaussian process (VGP) approximates the posterior distribution over the process $p(f(\cdot) \mid \mathcal{D}) \propto p(f)\, p(\mathcal{D} \mid \mathbf{f})$ by a process $q(f)$,

$$q(f(\cdot)) = \int p(f(\cdot) \mid \mathbf{f})\, q_{\mathbf{f}}(\mathbf{f})\, \mathrm{d}\mathbf{f}, \tag{2}$$

where the marginal distribution $q_{\mathbf{f}}$ is restricted to belong to a class of parametric distributions $\mathcal{Q}$. For hyperparameters $\boldsymbol{\theta}$, the variational lower bound (evidence lower bound, ELBO) to the log-marginal likelihood $p_{\boldsymbol{\theta}}(\mathcal{D})$ depends on the marginal distribution $q_{\mathbf{f}}$ and is then given by:

$$\mathcal{L}(q_{\mathbf{f}}, \boldsymbol{\theta}) = \mathbb{E}_{q(f)}\left[\log \frac{p_{\boldsymbol{\theta}}(\mathcal{D}, f)}{q(f)}\right] = \mathbb{E}_{q_{\mathbf{f}}(\mathbf{f})}\left[\log p_{\boldsymbol{\theta}}(\mathcal{D} \mid \mathbf{f})\right] - \mathrm{D}_{\mathrm{KL}}[q_{\mathbf{f}}(\mathbf{f}) \,\|\, p_{\boldsymbol{\theta}}(\mathbf{f})] \;\leq \log p_{\boldsymbol{\theta}}(\mathcal{D}). \tag{3}$$

The gap in the last inequality can be expressed as $\log p_{\boldsymbol{\theta}}(\mathcal{D}) - \mathcal{L}(q_{\mathbf{f}}) = \mathrm{D}_{\mathrm{KL}}[q(f) \,\|\, p_{\boldsymbol{\theta}}(f \mid \mathcal{D})]$. This means that maximizing $\mathcal{L}$ with respect to q amounts to minimizing the KL divergence between q and the posterior $p_{\boldsymbol{\theta}}(f \mid \mathcal{D})$. In non-conjugate models, the true posterior does not belong to $\mathcal{Q}$, the optimal distribution $q^*_{\mathbf{f}} = \arg\max_{q_{\mathbf{f}} \in \mathcal{Q}} \mathcal{L}(q_{\mathbf{f}})$ provides an approximation to the posterior. The variational distribution $q_{\mathbf{f}} \in \mathcal{Q}$ has parameters called 'variational' that we denote them by $\boldsymbol{\xi}_{\mathbf{f}}$ and, depending on the choice parameterization which we review in the next section, it may additionally depend on $\boldsymbol{\theta}$. To clearly separate the contribution of the variational parameters $\boldsymbol{\xi}_{\mathbf{f}}$ and hyperparameters $\boldsymbol{\theta}$, we introduce the loss function $\mathfrak{L}(\boldsymbol{\xi}_{\mathbf{f}}, \boldsymbol{\theta}) = \mathcal{L}(q_{\boldsymbol{\xi}_{\mathbf{f}}, \boldsymbol{\theta}}, \boldsymbol{\theta})$, which is the objective we will optimize via coordinate descent as a proxy to marginal likelihood optimization.

2.2 Common parameterizations for Gaussian Variational Inference

Choosing $\mathcal{Q}$ to be the class of multivariate normal (MVN) leads to a tractable objective. The KL divergence term in Eq. (3) is available in closed form. Furthermore, the expected log-likelihood terms often referred to as 'variational expectations' are also easy to compute. For the factorized likelihood setting where each conditional $\mathbf{y}_i \mid f$ depends on a single function evaluation $f_i = f(\mathbf{x}_i)$, these are one-dimensional integral that can be efficiently approximated through Gaussian quadrature [12].

Multiple parameterizations have been proposed for $q_{\mathbf{f}}$. As part of the exponential family, MVN distributions have densities of the form $p(\mathbf{x}) = \exp(t(\mathbf{x})^\top \boldsymbol{\eta} - a(\boldsymbol{\eta}))$, where $t(\mathbf{x})$ are the sufficient statistics, $\boldsymbol{\eta}$ the natural parameters, and $a(\boldsymbol{\eta})$ the log partition function. For MVN distribution, $t(\mathbf{x}) = (\mathbf{x}, \mathbf{x}\mathbf{x}^\top)$ and so the $\mathcal{Q}$ natural parameters in terms of source parameters are $\boldsymbol{\eta} = (\mathbf{S}^{-1}\mathbf{m}, -1/2\mathbf{S}^{-1})$. Another parameterization for exponential family distributions is the *expectation parameters*, $\boldsymbol{\mu} = \mathbb{E}_p\left[t(\mathbf{x})\right] = (\mathbf{m}, \mathbf{S} + \mathbf{m}\mathbf{m}^\top)$ (see App. A for more details).

In practice, a common approach is to use the mean/covariance (source) parameterization $\boldsymbol{\xi}_{\mathbf{f}} = (\mathbf{m}_{\mathbf{f}}, \mathbf{S}_{\mathbf{f}})$ [5]. An alternative formulation is through its mean vector and the Cholesky factor of its covariance $\boldsymbol{\xi}_{\mathbf{f}} = (\mathbf{m}_{\mathbf{f}}, \mathbf{L}_{\mathbf{f}})$, where $\mathbf{S}_{\mathbf{f}} = \mathbf{L}_{\mathbf{f}}\mathbf{L}_{\mathbf{f}}^\top$, which leads to a convex objective under mild assumptions on the observation model [4] and thus lend themselves to gradient-based optimization. Another parameterization commonly used and referred to as *whitened* consists in the statistics $(\mathbf{m}_{\mathbf{v}}, \mathbf{L}_{\mathbf{v}})$ of the transformed variable $\mathbf{v}$ such that $\mathbf{f} = \mathbf{L}_{\mathbf{f}}\mathbf{v}$ [38]. The statistics of $q(\mathbf{f})$ are then given by $\mathbf{m}_{\mathbf{f}} = \mathbf{L}_{\mathbf{f}}\mathbf{m}_{\mathbf{v}}$ and $\mathbf{S}_{\mathbf{f}} = \mathbf{L}_{\mathbf{f}}\mathbf{S}_{\mathbf{v}}\mathbf{L}_{\mathbf{f}}^\top$. For Gaussian likelihoods, the maximizers $\boldsymbol{\xi}^*_{\mathbf{f}}$ of $\mathfrak{L}(\boldsymbol{\xi}_{\mathbf{f}}, \boldsymbol{\theta})$ have closed-form expressions, as shown in Eq. (1). For general likelihoods, such closed-form expressions are unavailable.

In this section we introduced multiple parameterizations for $q_{\mathbf{f}}$ in VGP. The most widely used are the $(\mathbf{m}_{\mathbf{f}}, \mathbf{L}_{\mathbf{f}})$ and the whitened $(\mathbf{m}_{\mathbf{v}}, \mathbf{L}_{\mathbf{v}})$. We will refer to the algorithms associated to these parameterizations as q-VGP.

2.3 Inference via Natural Gradients Descent

Having chosen a parameterization $\boldsymbol{\xi}$ for the variational distribution $q_{\mathbf{f}}$, we now focus on procedures to perform approximate inference by maximizing the variational objective $\mathfrak{L}(\boldsymbol{\xi})$. A common procedure to solve such an optimization problem is via gradient descent with updates: $\boldsymbol{\xi}_{t+1} \leftarrow \boldsymbol{\xi}_t - \rho_t \nabla_{\boldsymbol{\xi}} \mathfrak{L}(\boldsymbol{\xi}_t)$, where ρ_t is a sequence of scalar learning rates. This gradient descent procedure can be cast as solving the following optimization step at each iteration:

$$\boldsymbol{\xi}_{t+1} \leftarrow \arg\min_{\boldsymbol{\xi}} \left(\langle \nabla_{\boldsymbol{\xi}} \mathfrak{L}(\boldsymbol{\xi}_t), \boldsymbol{\xi} - \boldsymbol{\xi}_t \rangle + \tfrac{1}{2\rho_t}\|\boldsymbol{\xi} - \boldsymbol{\xi}_t\|_2^2\right), \tag{4}$$

where $\langle \cdot\,,\,\cdot \rangle$ denotes the Euclidean inner product. This formulation is an instance of the mirrordescent algorithm.

The ordinary gradient is, however, an unnatural direction to follow in the context of variational inference where we are optimizing the distribution $q_{\mathbf{f}}$, rather than a set of parameters $\boldsymbol{\xi}$. An alternative optimization procedure is *natural gradient* descent. Natural gradient descent can be expressed as solving a similar objective where the Euclidean penalty replaced by the KL divergence [29]:

$$\boldsymbol{\xi}_{t+1} \leftarrow \arg\min_{\boldsymbol{\xi}} \left(\langle \nabla_{\boldsymbol{\xi}} \mathfrak{L}(\boldsymbol{\xi}_t), \boldsymbol{\xi} - \boldsymbol{\xi}_t \rangle + \tfrac{1}{\rho_t} \mathrm{D}_{\mathrm{KL}}[q(\boldsymbol{\xi}) \,\|\, q(\boldsymbol{\xi}_t)] \right), \tag{5}$$

which can be solved in closed form, leading to a step, where the natural descent direction is obtained by scaling the standard gradient by the inverse of the Fisher information matrix, which for distribution $q(\boldsymbol{\xi})$ is given by $\mathbf{F}(\boldsymbol{\xi}) = \mathbb{E}_{q(\cdot)} \left[\nabla^2 \log q(\cdot) \right]$ [1]:

$$\boldsymbol{\xi}_{t+1} \leftarrow \boldsymbol{\xi}_t - \rho_t \mathbf{F}(\boldsymbol{\xi}_t)^{-1} \nabla_{\boldsymbol{\xi}} \mathfrak{L}(\boldsymbol{\xi}_t). \tag{6}$$

These updates correspond to a second-order optimization since the Fisher information needs to be computed. The current standard implementation of these updates do not directly compute $\mathbf{F}(\boldsymbol{\xi}_t)$ since in the exponential family, $\mathbf{F}(\boldsymbol{\xi}_t)^{-1} \nabla_{\boldsymbol{\xi}} \mathfrak{L}(\boldsymbol{\xi}_t) = \nabla_{\boldsymbol{\eta}} \boldsymbol{\xi}(\boldsymbol{\eta}_t) \nabla_{\boldsymbol{\mu}} \mathfrak{L}(\boldsymbol{\mu}_t)$. However it requires multiple $\mathcal{O}(n^3)$ differentiations through transformations across MVN parameterizations $\boldsymbol{\xi} \to \boldsymbol{\eta}$ and back [33].

2.4 Hyperparameter Learning via Variational Expectation–Maximization

Hyperparameter selection via maximization of the marginal likelihood $\log p_{\boldsymbol{\theta}}(\mathcal{D})$ can approximated as the optimization of the bound $\mathfrak{L}(\boldsymbol{\xi}^*(\boldsymbol{\theta}), \boldsymbol{\theta})$, where $\boldsymbol{\xi}^*(\boldsymbol{\theta})$ are the optimial variational parameters for $\boldsymbol{\theta}$. Having no closed form solution for $\xi^*(\boldsymbol{\theta})$, a common procedure to perform this optimization is to construct a sequence of two steps as in the Expectation–Maximization algorithm [10, 25], where one alternates between optimizing the variational parameters (E-step) and the hyperparameters of the model (M-step):

$$\boldsymbol{\xi}^*(\boldsymbol{\theta}_t) = \arg\max_{\boldsymbol{\xi}} \mathfrak{L}(\boldsymbol{\xi}, \boldsymbol{\theta}_t), \qquad \text{(E-step)} \tag{7}$$

$$\boldsymbol{\theta}_{t+1} = \arg\max_{\boldsymbol{\theta}} \mathfrak{L}(\boldsymbol{\xi}^*(\boldsymbol{\theta}_t), \boldsymbol{\theta}). \qquad \text{(M-step)} \tag{8}$$

From $\boldsymbol{\theta}_t$, first compute the optimal $\xi^*(\boldsymbol{\theta}_t)$, a partial E-step, then optimize $\mathfrak{L}(\xi^*(\boldsymbol{\theta}_t), \boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$. The optimization for $\boldsymbol{\theta}$ in each step can also be performed partially, *i.e.* not until convergence. This view coincides with and motivates the arguments made in Salimbeni et al. [33] in favor of such a coordinate ascent procedure for improved convergence. See App. B for more details on the EM algorithm and view in GPs.

Crucially, the M-step of the proposed learning procedure is highly dependent on the choice of the parameterization $\boldsymbol{\xi}$. To the best of our knowledge, this has not been investigated. In the following sections, we explore this dependence and demonstrate the superiority of a particular *dual* parameterization.

3 Dual Parameterization for Improved Learning and Inference

We now introduce a known, yet rarely used parameterization for Gaussian distributions in the VGP setting and explore its consequence for inference and learning. The mean-covariance parameterization can be expressed in terms of the alternate reparameterization $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ [26, 27, 16], which at the optimum q^* satisfies,

$$\mathbf{m}_* = -\mathbf{K}_{\mathbf{ff}} \alpha_*, \qquad \text{where } \boldsymbol{\alpha}_{*,i} := \mathbb{E}_{\mathrm{N}(f \,|\, m_{*,i}, S_{*,ii})} \left[-\nabla_f \log p(y_i \,|\, f_i) \right], \tag{9}$$

$$\mathbf{S}_*^{-1} = \mathbf{K}_{\mathbf{ff}}^{-1} + \mathrm{diag}(\boldsymbol{\beta}_*), \qquad \text{where } \beta_{*,i} := \mathbb{E}_{\mathrm{N}(f \,|\, m_{*,i}, S_{*,ii})} \left[-\nabla^2_{ff} \log p(y_i \,|\, f_i) \right]. \tag{10}$$

The VGP objective is however not a convex function of $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ making this parameterization ill-suited to direct gradient based optimization [14]. The parameters $\boldsymbol{\alpha}_*$ and $\boldsymbol{\beta}_*$ can also be shown to be the *Lagrange multipliers* of the following Lagrangian [16, 20], where the mean and variance of distribution $q(f_i) = \mathrm{N}(f_i; e_i, v_i)$ are matched to the marginals of the $q(\mathbf{f}) = \mathrm{N}(\mathbf{f}; \mathbf{m}, \mathbf{S})$,

$$\textstyle\sum_i \mathbb{E}_{\mathrm{N}(f_i; e_i, v_i)} \left[\log p_{\boldsymbol{\theta}}(y_i \,|\, f_i) \right] - \mathrm{D}_{\mathrm{KL}}[q(\mathbf{f}) \,\|\, p_{\boldsymbol{\theta}}(\mathbf{f})] + \boldsymbol{\alpha}^\top (\mathbf{e} - \mathbf{m}) + \boldsymbol{\beta}^\top (\mathrm{diag}(\mathbf{S}) - \mathbf{v}). \tag{11}$$

The perturbation in the constraints are measured by the optimal Lagrangian multipliers $\boldsymbol{\alpha}_*$ and $\boldsymbol{\beta}_*$. Note that this holds even when the original variational objective is not convex.

An alternate way to express this parameterization is the following (see Eq. 18 in [18]),

$$q_*(\mathbf{f} \mid \mathbf{m}_*, \mathbf{S}_*) \propto \mathrm{N}(\mathbf{f} \mid \mathbf{0}, \mathbf{K}_{\mathbf{ff}}) \prod_{i=1}^n e^{\boldsymbol{\lambda}_{1,i} f_i - \frac{1}{2}\boldsymbol{\lambda}_{2,i} f_i^2}. \tag{12}$$

where $\boldsymbol{\beta} = -2\boldsymbol{\lambda}_2$ and $\mathbf{K}_{\mathbf{ff}}\boldsymbol{\alpha} = -\boldsymbol{\lambda}_1/\boldsymbol{\lambda}_2$. This parameterization is related to the site-based parameterization of the Expectation Propagation (EP) algorithm [24], where the aim is to approximate the joint density $p(\mathbf{y}, \mathbf{f}) \approx p(\mathbf{f}) \prod_i t_i(f_i)$, where the sites t_i are unnormalized Gaussian factors, also often parameterized by their natural parameters and an additional normalizer [35].

Because of this connection, we refer to the $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2) \in \mathbb{R}^n \times \mathbb{R}^n$ variables, that parameterize the Gaussian factors in Eq. (12) as the *dual* parameterization. We also introduce $\bar{\boldsymbol{\lambda}} = (\boldsymbol{\lambda}_1, \mathrm{diag}(\boldsymbol{\lambda}_2))$, such that the natural parameters of the approximate posterior in Eq. (12) can be written as $\boldsymbol{\eta}_q = \boldsymbol{\eta}_p + \bar{\boldsymbol{\lambda}}$, where $\boldsymbol{\eta}_p$ the natural parameters of $p(\mathbf{f})$. Contrary to common wisdom, this parameterization is ideally suited for both inference and learning in VGP, which we detail in the next sections and we refer to the algorithm associated to the dual paramaterization as t-VGP.

3.1 Faster Natural Gradients for Inference

The natural gradient updates expressed as mirrordescent in Sec. 2.3 are independent of the parameterization. Here we express it in terms of the expectation parameters $\boldsymbol{\mu}$ of the variational distribution.

$$\boldsymbol{\mu}_{t+1} \leftarrow \arg\min_{\boldsymbol{\mu}} \left(\langle \nabla_{\boldsymbol{\mu}} \mathcal{L}(q_t), \boldsymbol{\mu} - \boldsymbol{\mu}_t \rangle + \tfrac{1}{\rho_t} \mathrm{D}_{\mathrm{KL}}[q(\boldsymbol{\mu}) \,\|\, q(\boldsymbol{\mu}_t)] \right). \tag{13}$$

Using the relationship between the mean and natural parameters $\nabla_\mu \mathrm{D}_{\mathrm{KL}}[\boldsymbol{\mu} \,\|\, \boldsymbol{\mu}_t] = \boldsymbol{\eta} - \boldsymbol{\eta}_t$ [19], we get a simple natural gradient update for the natural parameters in terms of their deviation from the natural parameters $\boldsymbol{\eta}_p$ of the prior distribution $\boldsymbol{\eta} = \boldsymbol{\eta}_p + \bar{\boldsymbol{\lambda}}_t$, *i.e.* the dual parameterization:

$$\bar{\boldsymbol{\lambda}}_{t+1} = r_t \nabla_{\boldsymbol{\mu}} \mathbb{E}_{q(\mathbf{f})} \left[\log p(\mathbf{y} \mid \mathbf{f}) \right] + (1 - r_t) \bar{\boldsymbol{\lambda}}_t, \tag{14}$$

where $r_t = \frac{\rho_t}{1+\rho_t}$. This formulation of natural gradient updates was introduced by Khan and Lin [17]. It is a first order update (only needs the forward gradient of the variational expectations), and is much cheaper than previous calculations of the natural gradient [33]. When the iterations in Eq. (14) converge, the optimal parameter satisfies $\boldsymbol{\lambda}_*(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\mu}^*} \mathbb{E}_{q^*(\mathbf{f})} \left[\log p(\mathbf{y} \mid \mathbf{f}) \right]$ as given in Eq. (9) and Eq. (10).

3.2 Improved Objective for Hyperparameter Learning

We now assess how the different parameterizations for the MVN variational distribution $q_{\mathbf{f}}$ affects the EM learning procedure introduced in Sec. 2.4. We show empirically that the dual parameterization $\boldsymbol{\lambda}$, provides a better objective for hyperparameter learning.

Starting from hyperparameter $\boldsymbol{\theta}_t$, a dual parameter $\boldsymbol{\lambda}_*(\boldsymbol{\theta}_t)$ and matched alternative parameterization $\boldsymbol{\xi}_t(\boldsymbol{\theta}_t)$, computing the approximate posterior (E-step) leads to matched objective and gradient:

$$\mathfrak{L}(\boldsymbol{\lambda}^*(\boldsymbol{\theta}_t), \boldsymbol{\theta}_t) = \mathfrak{L}(\boldsymbol{\xi}^*(\boldsymbol{\theta}_t), \boldsymbol{\theta}_t), \tag{15}$$

$$\nabla_\theta \mathfrak{L}(\boldsymbol{\lambda}^*(\boldsymbol{\theta}_t), \boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}_t} = \nabla_{\boldsymbol{\theta}} \mathfrak{L}(\boldsymbol{\xi}^*(\boldsymbol{\theta}_t), \boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}_t} = \partial_{\boldsymbol{\theta}} \mathcal{L}(q^*, \boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}_t}. \tag{16}$$

We conjecture that for the following M-step, the dual parameterization leads to a better update: $\max_{\boldsymbol{\theta}} \mathfrak{L}(\boldsymbol{\lambda}^*(\boldsymbol{\theta}_t), \boldsymbol{\theta}) \geq \max_{\boldsymbol{\theta}} \mathfrak{L}(\boldsymbol{\xi}^*(\boldsymbol{\theta}_t), \boldsymbol{\theta})$, and that the resulting sequence of updates converges faster to an optimum of $\mathcal{L}(q, \boldsymbol{\theta})$.

In the conjugate regression case, the posterior process is a Gaussian process and thus contained in the approximating family $\mathcal{Q}$. The Gaussian observation provides the likelihood $p(y_i \mid f_i) = \mathrm{N}(y_i; f_i, \sigma^2)$ and the optimal dual parameters exactly match the natural parameters of the likelihood function $\boldsymbol{\lambda}_i^* = (f_i/\sigma^2, -1/2\sigma^2)$. Crucially, they are independent of the kernel parameters. Thus, after one full E-step the lower bound is tight and matches the log-marginal likelihood after one E-step: $\mathfrak{L}(\boldsymbol{\lambda}^*(\boldsymbol{\theta}_t), \boldsymbol{\theta}) = \log p(\mathcal{D} \mid \boldsymbol{\theta})$. In other words, the optimal parameters for a parameter $\boldsymbol{\theta}_t$ generalize perfectly in the bound to new parameters $\boldsymbol{\theta}$. In the non-conjugate case, we observe that the objective $\mathfrak{L}(\boldsymbol{\lambda}^*(\boldsymbol{\theta}_t), \boldsymbol{\theta})$ provides a better objective for learning (see Fig. 1). We also find that the outcome of an M-step in Eq. (8) is less dependent on the starting parameter $\boldsymbol{\theta}_t$. We demonstrate this in a classification example in Fig. 2.

The key difference between the dual parameterization and the other two is that the variational distribution $q_{\mathbf{f}}$ in the former depends on the hyperparameters $\mathfrak{L}(\boldsymbol{\lambda}, \theta) = \mathcal{L}(q_{\boldsymbol{\lambda},\boldsymbol{\theta}}, \boldsymbol{\theta})$ but not

in the latter: $\mathfrak{L}(\boldsymbol{\xi}, \theta) = \mathcal{L}(q_{\boldsymbol{\xi}}, \boldsymbol{\theta})$. Although we observe it empirically, we cannot prove that $\forall \boldsymbol{\theta}, \mathfrak{L}(\boldsymbol{\lambda}^*(\boldsymbol{\theta}_t), \boldsymbol{\theta}) \geq \mathfrak{L}(\boldsymbol{\xi}^*(\boldsymbol{\theta}_t), \boldsymbol{\theta})$ in general. However we provide in App. D the conditions for it to be true around $\boldsymbol{\theta}_t$, *i.e.*, for $\nabla_{\boldsymbol{\theta}\boldsymbol{\theta}} \mathfrak{L}(\boldsymbol{\lambda}^*(\boldsymbol{\theta}_t), \boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}_t} \geq \nabla_{\boldsymbol{\theta}\boldsymbol{\theta}} \mathfrak{L}(\boldsymbol{\xi}^*(\boldsymbol{\theta}_t), \boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}_t}$.

4 Application to Sparse Gaussian Processes

Variational inference for GP models as described in the previous section suffers from the same $\mathcal{O}(n^3)$ computational bottleneck as the exact conjugate GP regression setting. The typical approach to alleviate the cubic complexity is by approximative methods which leverage a set of sparse *inducing* inputs or features [36, 34, 8, 28, 39, 30]. Within the variational inference framework, sparse GPs [37] and their stochastic extensions [12, 13, 6, 7, 32] have been proposed. In their simplest instantiation, sparse methods aim to reduce the infeasible $\mathcal{O}(n^3)$ computation cost associated with GP models by choosing $m \ll n$ representative inputs $\mathcal{Z} := \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m\}$ and by setting the class $\mathcal{Q}$ of posterior processes to be:

$$q(f(\cdot)) = \int p(f(\cdot) \mid f(\mathcal{Z}) = \mathbf{u})\, q_{\mathbf{u}}(\mathbf{u})\, \mathrm{d}\mathbf{u}. \tag{17}$$

The m-dimensional distribution $q_{\mathbf{u}}$ is over the *inducing variables* $f(\mathcal{Z})$ and is restricted to be a MVN distribution. It can be thought of a posterior distribution under a compressed alternative Gaussian observation model with likelihood $t(\mathbf{u})$, such that $q(\mathbf{u}) \propto p(\mathbf{u})\, t(\mathbf{u})$, that is learned to best approximate the posterior process as measured per the divergence $\mathrm{D}_{\mathrm{KL}}[q(f) \,\|\, p(f \mid \mathcal{D})]$. Sparse variational GP (SVGP) methods optimize a lower bound to find a Gaussian posterior approximation $q(\mathbf{f}_{\mathbf{u}})$ over function values $\mathbf{f}_{\mathbf{u}} := (f(\mathbf{z}_1), f(\mathbf{z}_2), \ldots, f(\mathbf{z}_m))$, as well as hyperparameters $\boldsymbol{\theta}$ and inducing input locations in $\mathcal{Z}$:

$$\mathcal{L}(\boldsymbol{\xi}_{\mathbf{u}}, \mathcal{Z}, \boldsymbol{\theta}) := \textstyle\sum_{i=1}^n \mathbb{E}_{q(f_i)} \left[\log p(y_i \mid f_i)\right] - \mathrm{D}_{\mathrm{KL}}[q(\mathbf{f}_{\mathbf{u}}) \,\|\, p(\mathbf{f}_{\mathbf{u}})]. \tag{18}$$

The evaluation of the marginal predictions $q(f_i)$ which can be computed as in Eq. (17). Denoting by $\mathbf{K}_{\mathbf{uu}}$ the covariance matrix such that $[\mathbf{K}_{\mathbf{uu}}]_{ij} = \kappa(\mathbf{z}_i, \mathbf{z}_j)$, $\mathbf{k}_{i\mathbf{u}} = \kappa(x_i, \mathcal{Z})$ and $\mathbf{a}_i = \mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{k}_{i\mathbf{u}}$, we have the conditional $p(f_i \mid \mathbf{u}) = \mathrm{N}\left(f_i \mid \mathbf{a}_i^\top \mathbf{u}, \kappa(\mathbf{x}_i, \mathbf{x}_i) - \mathbf{a}_i^\top \mathbf{K}_{\mathbf{uu}} \mathbf{a}_i\right)$. Given the variational distribution $q_{\mathbf{u}}(\mathbf{f}_{\mathbf{u}}) = \mathrm{N}(\mathbf{f}_{\mathbf{u}}; \mathbf{m}_{\mathbf{u}}, \mathbf{S}_{\mathbf{u}})$, the marginal is available in closed form as $q(f_i) = \mathrm{N}\left(f_i \mid \mathbf{a}_i^\top \mathbf{m}_{\mathbf{u}}, \kappa(\mathbf{x}_i, \mathbf{x}_i) - \mathbf{a}_i^\top (\mathbf{K}_{\mathbf{uu}} - \mathbf{S}_{\mathbf{u}}) \mathbf{a}_i\right)$. The variational objective for such sparse GP posteriors can be evaluated at a cost $\mathcal{O}(nm^2 + m^3)$. This complexity improvement has lead to the broad adoption of the SVGP algorithm in the community making it one of the underlying workhorse of many recent GP-based algorithms. As shown by Titsias [37], for Gaussian likelihoods and a given $\mathcal{Z}$ and $\boldsymbol{\theta}$, the maximum of the lower bound with respect to $\mathbf{m}_{\mathbf{u}}$ and $\mathbf{S}_{\mathbf{u}}$ is equivalent to the projected point approximation (PPA, [34, 8]). The maximization of the lower bound can fortunately be performed in $\mathcal{O}(m^3 + n_{\mathrm{b}} m^2)$ per iteration by using stochastic gradient methods, where n_{b} is the batch size [12]. This formulation works for general likelihood functions [13], which is another major advantage.

A commonly used parameterization for $q_{\mathbf{u}}$ is through the mean / Cholesky factor of covariance parameterization $\boldsymbol{\xi}_{\mathbf{u}} = (\mathbf{m}_u, \mathbf{L}_u)$ [12], and natural gradient descent has been used for inference [33]. In this sparse setting, the variational parameters $\boldsymbol{\xi}_{\mathbf{u}}$ and the marginals $\{(\mathbf{m}_i, \mathbf{S}_{ii})\}_{i=1}^n$ are related through the conditionals $p(f_i \mid \mathbf{u})$. As in the VGP setting, solving for $\nabla_{\boldsymbol{\xi}_{\mathbf{u}}} \mathfrak{L}(\boldsymbol{\xi}_{\mathbf{u}}) = 0$ reveals that the optimal structure of the variational posterior takes the form:

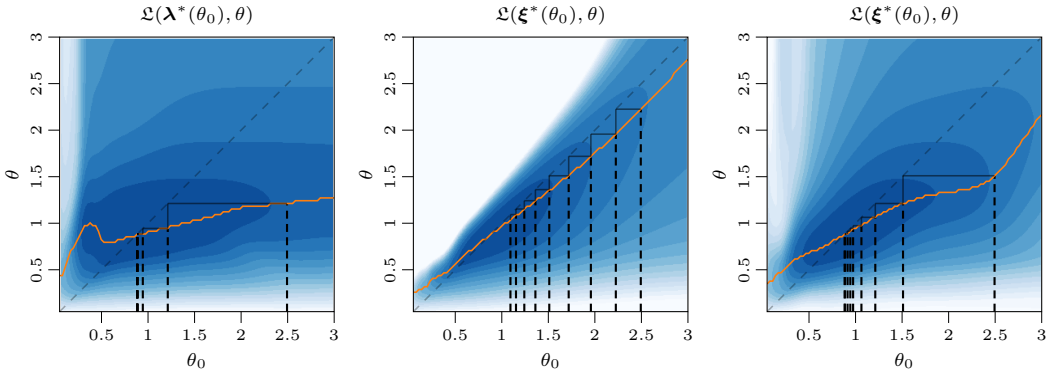
$$\mathbf{S}_{\mathbf{u}}^{-1} \mathbf{m}_{\mathbf{u}} = \textstyle\sum_{i=1}^n \mathbf{a}_i \boldsymbol{\lambda}_{1,i} \qquad \text{and} \qquad \mathbf{S}_{\mathbf{u},\mathbf{u}}^{-1} = \mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1} + \textstyle\sum_{i=1}^n \mathbf{a}_i \mathbf{a}_i^\top \boldsymbol{\lambda}_{2,i}, \tag{19}$$

where the $2n$ variables $\boldsymbol{\lambda}$ are the dual variables. It also corresponds to the optimal site-based parameterization of EP in the sparse setting [3]. As in the VGP algorithm, efficient natural gradient updates for the dual variables are obtained by performing mirror descent in the expectation parameterization of $q_{\mathbf{u}}$, as in Eq. (13), leading to the very same updates for the dual parameters as in Eq. (14). This is because the gradient of the objective with respect to the natural parameters of $q_{\mathbf{u}}$ are given by

$$\nabla_{\mathbf{m}_{\mathbf{u}}} \mathbb{E}_{q(\mathbf{f})} \left[\log p(\mathbf{y}_i \mid f_i)\right] = \mathbf{a}_i \alpha_i \qquad \text{and} \qquad \nabla_{\mathbf{S}_{\mathbf{uu}}} \mathbb{E}_{q(\mathbf{f})} \left[\log p(\mathbf{y}_i \mid f_i)\right] = \mathbf{a}_i \mathbf{a}_i^\top \beta_i, \tag{20}$$

where $\beta_i = \mathbb{E}_{q(f_i)}[-\nabla^2_{ff} \log p(\mathbf{y}_i \mid f_i)]$ and $\alpha_i = \mathbb{E}_{q(f_i)} \left[\nabla_f \log p(\mathbf{y}_i \mid f_i)\right]$. The dual variables again provide a parameterization that makes the statistics of $q_{\mathbf{u}}$ depend on the kernel hyper-parameters, leading to an improved objective for learning.

One of the advantages of the classic $\boldsymbol{\xi}_{\mathbf{u}} = (\mathbf{m}_u, \mathbf{L}_u)$ parameterization of SVGP is its $\mathcal{O}(m^2)$ storage requirement and the possibility to perform stochastic optimization via mini-batching. A



(a) t-SVGP (ours) (b) q-SVGP (c) q-SVGP (whitened)

Figure 2: For the dual (left), the standard unwhitened (middle) and whitened (right) SVGP parameterizations, the panels show (in blue) the objective $\mathfrak{L}(\boldsymbol{\xi}^*(\boldsymbol{\theta}_0), \boldsymbol{\theta})$ in the *banana classification* task with a RBF kernel. $\boldsymbol{\theta}$ is the kernel lenghthscale we are attempting to learn and $\boldsymbol{\xi}^*(\boldsymbol{\theta})$ are the optimal variational parameters for $\boldsymbol{\theta}$. The yellow curve shows the outcome of an M-step starting at $\boldsymbol{\theta}_0$ as $\boldsymbol{\theta}^*(\boldsymbol{\theta}_0) = \arg\max_{\boldsymbol{\theta}} \mathfrak{L}(\boldsymbol{\xi}^*(\boldsymbol{\theta}_0), \boldsymbol{\theta})$. The EM iterations correspond to the fixed point iterations $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}^*(\boldsymbol{\theta}_t)$. The shape of this curve reveals the advantage of using the dual parameterization $\boldsymbol{\xi} = \boldsymbol{\lambda}$ for the purpose of learning $\boldsymbol{\theta}$. The iterations converge to the point of intersection of this line with the diagonal $\boldsymbol{\theta}(\boldsymbol{\theta}_0) = \boldsymbol{\theta}_0$. Convergence is fastest when the yellow curve $\boldsymbol{\theta}^*(\boldsymbol{\theta}_0)$ is horizontal and slowest when diagonal. Starting at $\boldsymbol{\theta}_0 = 2.5$, EM converges in 3 iterations for the dual parameterization (left) and is slower for the alternative parameterizations.

parameterization that would lead to the same storage complexity consists in storing $\boldsymbol{\Lambda} = (\boldsymbol{\Lambda}_1, \boldsymbol{\Lambda}_2)$, with $\boldsymbol{\Lambda}_1 \in \mathbb{R}^m$ and $\boldsymbol{\Lambda}_2 \in \mathbb{R}^{m \times m}$ given by:

$$\boldsymbol{\Lambda}_1 = \textstyle\sum_{i=1}^n \mathbf{a}_i \boldsymbol{\lambda}_{1,i} \qquad \text{and} \qquad \boldsymbol{\Lambda}_2 = \textstyle\sum_{i=1}^n \mathbf{a}_i \mathbf{a}_i^\top \boldsymbol{\lambda}_{2,i}, \tag{21}$$

from which we construct the mean $\mathbf{m}_\mathbf{u}$ and covariance $\mathbf{S}_\mathbf{u}$ via

$$\mathbf{S}_\mathbf{u}^{-1} \mathbf{m}_\mathbf{u} = \boldsymbol{\Lambda}_1 \qquad \text{and} \qquad \mathbf{S}_\mathbf{u}^{-1} = \mathbf{K}_{\mathbf{uu}}^{-1} + \boldsymbol{\Lambda}_2. \tag{22}$$

We call this dual parameterization a *tied* parameterization because it corresponds to the *site-tying* setting in sparse EP [3, 22] that was motivated by the same goal of reducing storage complexity. We refer to the algorithm associated to the $(\mathbf{m}_\mathbf{u}, \mathbf{L}_\mathbf{u})$ parameterization as q-SVGP, and to the one associated to the dual parameterizations as t-SVGP. As in the non-sparse setting, the dual parameterizations $\boldsymbol{\lambda}$ and $\boldsymbol{\Lambda}$ provide improved objectives for hyperparameters learning which in the sparse setting consist of both $\boldsymbol{\theta}$ and $\mathcal{Z}$. The tied parameterization $\boldsymbol{\Lambda}$ reduces the coupling between the hyperparameters $\boldsymbol{\theta}$ and the statistics of $q_\mathbf{u}$, potentially reducing the gains for hyperparameter learning over q-SVGP compared to its untied counterpart. However, this is the price to pay to obtain an economical storage complexity matching that of q-SVGP. In all the experiments presented in this paper, we use the tied parameterization of t-SVGP.

5 Empirical Evaluation

We conduct experiments to highlight the advantages of using the dual parameterization. Firstly, we study the effects of the improved objective for hyperparameter learning of t-SVGP versus q-SVGP. We study the objective being optimized for a single M-step, after an E-step ran until convergence. We then show a full sequence of EM iterations on small data sets. For large-scale data, where running steps to convergence is expensive, we use partial E and M-steps and mini-batching. Our improved bound and faster natural gradient computations show benefits in both settings. It is worth noting that the E-step for both q-SVGP with natural gradients and t-SVGP are identical up to machine precision, and any differences in performance are to be attributed to the different parameterization.

The Role of the Learning Objective In Fig. 1, we learn the kernel hyperparameters $\boldsymbol{\theta}$ in a GP classification task via coordinate ascent of the lower bound $\mathfrak{L}(\boldsymbol{\xi}, \boldsymbol{\theta})$, where $\boldsymbol{\xi}$ are the variational parameters, *i.e.* via EM. Starting at hyperparameter $\boldsymbol{\theta}_0$, we denote by $\boldsymbol{\xi}^*(\boldsymbol{\theta}_0)$ the associated optimal variational parameters. Updating $\boldsymbol{\theta}$ consists in optimizing $\mathfrak{L}(\boldsymbol{\xi}^*(\boldsymbol{\theta}_0), \boldsymbol{\theta})$ which we show on the left

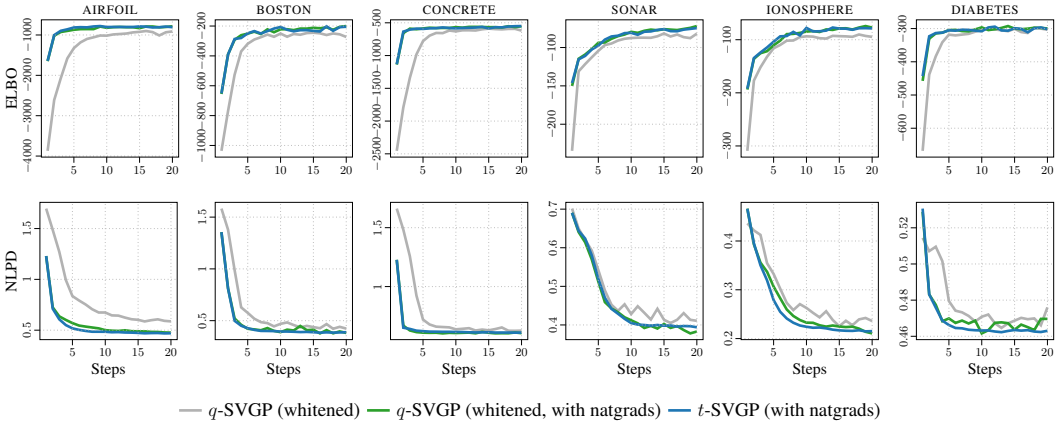


Figure 3: Comparison of convergence in terms of ELBO and negative log-predictive density (NLPD) as averages over 5-fold cross-validation runs, on UCI regression and classification tasks. All methods were trained (incl. hyperparameters and 50 inducing input locations) with matching learning rate. Natural gradient–based training is superior, and the t-SVGP parameterization improves stability.

panel for the dual parameterization (blue), the standard whitened (orange) and unwhitened (green) SVGP parameterizations. The dual parameterization leads to a tighter bound and thus to bigger steps and faster overall convergence as shown on the right for the *banana classification* task, starting at $(\theta_1, \theta_2) = (1, 1)$, in the extreme case of taking both the E and M step to convergence. For the banana data set we use $m = 50$ inducing points (see details in App. E).

In Fig. 2, we also use the banana data and parameters as in Fig. 1, but we show how the learning objective changes over iterations. The blue contours show, for all initial $\boldsymbol{\theta}_0$, the objective maximized in the M-step, *i.e.* $\mathfrak{L}(\boldsymbol{\xi}^*(\boldsymbol{\theta}_0), \boldsymbol{\theta})$. The orange lines show, for all initial $\boldsymbol{\theta}_0$, the outcome of an E-step followed by and M-step, *i.e.* $\boldsymbol{\theta}^*(\boldsymbol{\theta}_0) = \arg\max_{\boldsymbol{\theta}} \mathfrak{L}(\boldsymbol{\xi}^*(\boldsymbol{\theta}_0), \boldsymbol{\theta})$. The EM iterations converge to the fixed points of $\boldsymbol{\theta}^*$, *i.e.* its intersection with the diagonal line of the identity function. A flatter line around the optimal value is more desirable as it means the iterations converge faster to the optimum value which in this experiment is just below one, while a line close to the diagonal leads to slow convergence. Here t-SVGP has the fastest convergence, q-SVGP performs poorly, although whitening clearly helps the optimisation problem. The dark dashed lines show how optimising θ would look starting from $\theta_0 = 2.5$ and running 8 iterations for the different models.

Evaluation on UCI Classification and Regression Tasks We use common small and mid-sized UCI data sets to test the performance of our method against q-SVGP with natural gradient optimisation and normal q-SVGP trained with Adam optimizer for the variational parameters. All methods use Adam for the hyperparameters. The exact details of the data sets can be found in App. E. Here we again take the approach that the optimal way to optimize the ELBO if computational budget allows is to alternate between performing E and M-steps till convergence. We plot how the different inference schemes perform for ELBO and NLPD on a hold test set. We perform 5-fold cross validation with the results in Fig. 3 showing the mean of the folds for ELBO and NLPD. Natural gradient variants of q-SVGP clearly perform better than non natural gradient q-SVGP. Our method t-SVGP seems more stable specifically in NLPD for most data sets if not equal to q-SVGP. For q-SVGP, we have used the whitened version which as we noted helps with hyperparameter optimisation.

Improved Efficiency in Large-scale Inference To highlight *practical* benefits, we show the performance of our stochastic and sparse t-SVGP framework on the MNIST ([21], 277 available under CC BY-SA 3.0) multiclass-classification task (for details see App. E). Given the data set is $n = 70{,}000$, minibatching is needed and we adopt the parameterization of Eq. (21) meaning we match q-SVGP for parameter storage complexity. We compare against the natural gradient q-SVGP implementation but not the non-natural gradient version since it produces considerably worse performance. In large scale minibatching experiments

Table 1: NLPD on MNIST benchmarks for different learning rates and E and M steps.

NLPD		LR		STEPS	
q-SVGP	t-SVGP	E	M	#E	#M
0.304±0.015	**0.304±0.006**	0.040	0.05	1	1
0.289±0.010	**0.283±0.007**	0.035	0.10	2	1
0.293±0.020	**0.281±0.010**	0.030	0.10	3	1
0.259±0.010	**0.255±0.006**	0.025	0.03	4	2
0.282±0.007	**0.283±0.006**	0.050	0.03	4	2
0.243±0.003	**0.230±0.009**	0.030	0.03	4	1

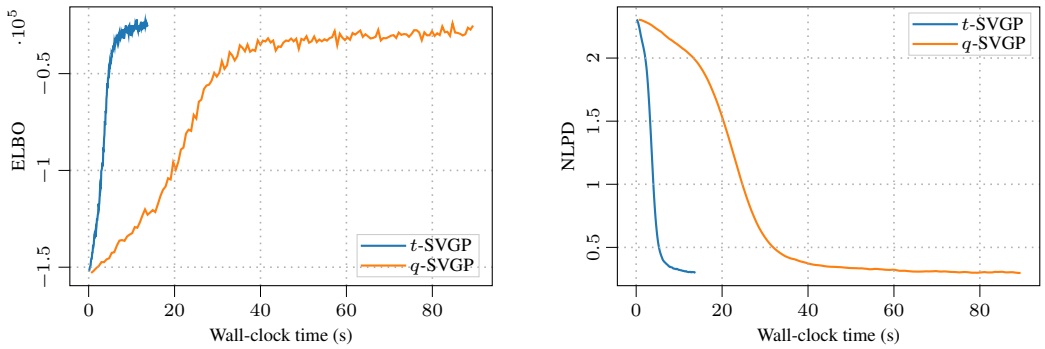


Figure 4: Comparison of practical inference and learning on MNIST. We compare training time on a laptop between t-SVGP to the q-SVGP model in GPflow in terms of wall-clock time of training for 150 steps, where both methods use natural gradient updates and share the same learning rates.

performing full E and M steps may not be efficient. Instead we perform partial steps for both. A single E and M step can be thought of as the approach outlined in [33]. All experiments are performed with a batch size of $n_\mathrm{b} = 200$ and $m = 100$ inducing points and the optimization is ran until convergence using the Adam optimizer for the hyperparameters (M-step). Table 1 shows different variations of learning rates and iterations of E- and M-steps. The results suggest some benefits in running partial EM steps. The t-SVGP formulation performs equally if not better than q-SVGP under all settings. In Fig. 4, we show the speed advantage of t-SVGP over q-SVGP due to cheaper natural gradient updates. We compare against the state-of-the-art implementation of SVGP in GPflow ([23], v2.2.1) and a closely matched implementation of our method in GPflow. We compare wall-clock time to compute 150 steps of the algorithm for both methods in terms of NLPD and ELBO taking single E and M-steps (MacBook pro, 2 GHz CPU, 16 GB RAM). Our implementation avoids the use of sluggish automatic differentiation to compute the natural gradients, and, even if our implementation is not as optimized as SVGP in GPflow, it is roughly 5 times faster on this standard benchmark.

6 Discussion and Conclusion

Sparse variational GP (SVGP) methods are the current *de facto* approach to allow GPs to scale to large problems. In this paper, we introduced an alternative parameterization to variational GPs that leads to an improved loss landscape for learning (*cf.*, Fig. 1). This improvement hinges on writing the variational problem in terms of its dual—similar to the conjugate-computation variational inference (CVI) approach by Khan and Lin [17]— parameterization to capture sites: we assume the approximate posterior decomposes into a prior contribution and a Gaussian *approximate* likelihood contribution. Variational inference under this model can conveniently be implemented by mirror descent and corresponds to natural gradient based learning, thus improving convergence in variational parameter optimization (the 'E-step'), at the same time as improving hyperparameter optimization (the 'M-step'), due to the tighter evidence lower bound.

We further show that we can derive the *sparse* equivalent of this method, which also allows for stochastic training through mini-batching, reducing the computational complexity to $\mathcal{O}(m^3 + n_\mathrm{b} m^2)$ per step. Our method matches the asymptotic computational cost of other SVGP methods, while marginally reducing compute due to simpler expressions to back-propagate through (see discussion in Sec. 5). Our empirical validation across a wide variety of regression and classification tasks confirms the benefits suggested by our theory: The proposed strategy typically allows for improved stability over gold-standard SVGP methods even when the learning rates remain the same. It allows for higher learning rates, and reduces computational cost—leading to improved learning both in terms of reduced steps as well as expected wall-clock time.

We provide a reference implementation of our method for GPflow that is to be released under the MIT license upon acceptance of the paper (see supplement for now).

References

[1] S.-i. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

[2] 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.

[3] T. D. Bui, J. Yan, and R. E. Turner. A unifying framework for Gaussian process pseudo-point approximations using power expectation propagation. *Journal of Machine Learning Research*, 18(1):3649–3720, 2017.

[4] E. Challis and D. Barber. Concave Gaussian variational approximations for inference in large-scale Bayesian linear models. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15 of *Proceedings of Machine Learning Research*, pages 199–207. PMLR, 2011.

[5] E. Challis and D. Barber. Gaussian Kullback–Leibler approximate inference. *Journal of Machine Learning Research*, 14(32):2239–2286, 2013.

[6] C.-A. Cheng and B. Boots. Incremental variational sparse Gaussian process regression. In *Advances in Neural Information Processing Systems 29 (NIPS)*, pages 4410–4418. Curran Associates, Inc., 2016.

[7] C.-A. Cheng and B. Boots. Variational inference for Gaussian process models with linear complexity. In *Advances in Neural Information Processing Systems 30 (NIPS)*, pages 5184–5194. Curran Associates, Inc., 2017.

[8] L. Csató. *Gaussian Processes: Iterative Sparse Approximations*. PhD thesis, Aston University, Birmingham, UK, 2002.

[9] L. Csató and M. Opper. Sparse on-line Gaussian processes. *Neural Computation*, 14(3):641–668, 2002.

[10] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[11] J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson. GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. In *Advances in Neural Information Processing Systems 31 (NeurIPS)*, pages 7576–7586. Curran Associates, Inc., 2018.

[12] J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian processes for big data. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 282–290. AUAI Press, 2013.

[13] J. Hensman, A. Matthews, and Z. Ghahramani. Scalable variational Gaussian process classification. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 38 of *Proceedings of Machine Learning Research*, pages 351–360. PMLR, 2015.

[14] E. Khan, S. Mohamed, and K. P. Murphy. Fast bayesian inference for non-conjugate Gaussian process regression. *Advances in Neural Information Processing Systems 25 (NIPS)*, pages 3140–3148, 2012.

[15] 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 *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 2611–2620. PMLR, 2018.

[16] M. E. Khan. Decoupled variational Gaussian inference. In *Advances in Neural Information Processing Systems 27 (NIPS)*, pages 1547–1555. Curran Associates, Inc., 2014.

[17] M. E. Khan and W. Lin. Conjugate-computation variational inference: Converting variational inference in non-conjugate models to inferences in conjugate models. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 54 of *Proceedings of Machine Learning Research*, pages 878–887. PMLR, 2017.

[18] M. E. Khan and D. Nielsen. Fast yet simple natural-gradient descent for variational inference in complex models. In *2018 International Symposium on Information Theory and Its Applications (ISITA)*, pages 31–35. IEEE, 2018.

[19] M. E. Khan and H. Rue. Learning-algorithms from Bayesian principles. *arXiv preprint arXiv*, 2021.

[20] M. E. Khan, A. Aravkin, M. Friedlander, and M. Seeger. Fast dual variational inference for non-conjugate latent Gaussian models. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, volume 28 of *Proceedings of Machine Learning Research*, pages 951–959. PMLR, 2013.

[21] Y. LeCun, C. Cortes, and C. J. Burges. The MNIST database of handwritten digits, 1998. URL http://yann.lecun.com/exdb/mnist/.

[22] Y. Li, J. M. Hernández-Lobato, and R. E. Turner. Stochastic expectation propagation. In *Advances in Neural Information Processing Systems 28 (NIPS)*, pages 2323–2331. Curran Associates, Inc., 2015.

[23] A. G. d. G. Matthews, M. van der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, and J. Hensman. GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18(40):1–6, 2017.

[24] T. P. Minka. Expectation propagation for approximate Bayesian inference. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 17, pages 362–369. AUAI Press, 2001.

[25] R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, pages 355–368. Springer, 1998.

[26] H. Nickisch and C. E. Rasmussen. Approximations for binary Gaussian process classification. *Journal of Machine Learning Research*, 9(Oct):2035–2078, 2008.

[27] M. Opper and C. Archambeau. The variational Gaussian approximation revisited. *Neural Computation*, 21 (3):786–792, 2009.

[28] J. Quiñonero-Candela and C. E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6(Dec):1939–1959, 2005.

[29] G. Raskutti and S. Mukherjee. The information geometry of mirror descent. *IEEE Transactions on Information Theory*, 61(3):1451–1457, 2015.

[30] C. E. Rasmussen and J. Quiñonero-Candela. Healing the relevance vector machine through augmentation. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pages 689–696. Association for Computing Machinery, 2005.

[31] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.

[32] H. Salimbeni, C.-A. Cheng, B. Boots, and M. Deisenroth. Orthogonally decoupled variational Gaussian processes. In *Advances in Neural Information Processing Systems 31 (NeurIPS)*, pages 8711–8720. Curran Associates, Inc., 2018.

[33] H. Salimbeni, S. Eleftheriadis, and J. Hensman. Natural gradients in practice: Non-conjugate variational inference in Gaussian process models. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 84 of *Proceedings of Machine Learning Research*, pages 689–697. PMLR, 2018.

[34] M. Seeger. *Bayesian Gaussian Process Models: PAC-Bayesian Generalisation Error Bounds and Sparse Approximations*. PhD thesis, University of Edinburgh, Edinburgh, UK, 2003.

[35] M. Seeger. Expectation propagation for exponential families. Technical report, 2005.

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

[37] M. K. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 5 of *Proceedings of Machine Learning Research*, pages 567–574. PMLR, 2009.

[38] M. van der Wilk, V. Dutordoir, S. John, A. Artemev, V. Adam, and J. Hensman. A framework for interdomain and multioutput Gaussian processes. *arXiv preprint arXiv:2003.01115*, 2020.

[39] C. K. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13 (NIPS)*, pages 682–688. MIT Press, 2001.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] The methodological aspects are presented in detail in Sec. 3.2 and results shown in Sec. 5.
   (b) Did you describe the limitations of your work? [Yes] See Sec. 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] In parts: The paper is concerned with technical basic research, but some implications are covered in the Discussion.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See the background section Sec. 2.
   (b) Did you include complete proofs of all theoretical results? [Yes] See supplement for details.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We provide code to recreate results of the paper.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See the write up in Sec. 5 and App. E.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Sec. 5 and App. E.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Sec. 5.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No] We provide code, but no new data is released.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]