Chang, Paul E.; Verma, Prakhar; John, S.T.; Solin, Arno; Emtiyaz Khan, Mohammad

Memory-Based Dual Gaussian Processes for Sequential Learning

Published in:
Proceedings of the 40th International Conference on Machine Learning

Published: 01/07/2023

Document Version
Publisher's PDF, also known as Version of record

Memory-Based Dual Gaussian Processes for Sequential Learning

Paul E. Chang [*1] **Prakhar Verma** [*1] **S.T. John** [1 2] **Arno Solin** [1 2] **Mohammad Emtiyaz Khan** [3]

Abstract

Sequential learning with Gaussian processes (GPs) is challenging when access to past data is limited, for example, in continual and active learning. In such cases, errors can accumulate over time due to inaccuracies in the posterior, hyperparameters, and inducing points, making accurate learning challenging. Here, we present a method to keep all such errors in check using the recently proposed dual sparse variational GP. Our method enables accurate inference for generic likelihoods and improves learning by actively building and updating a memory of past data. We demonstrate its effectiveness in several applications involving Bayesian optimization, active learning, and continual learning.

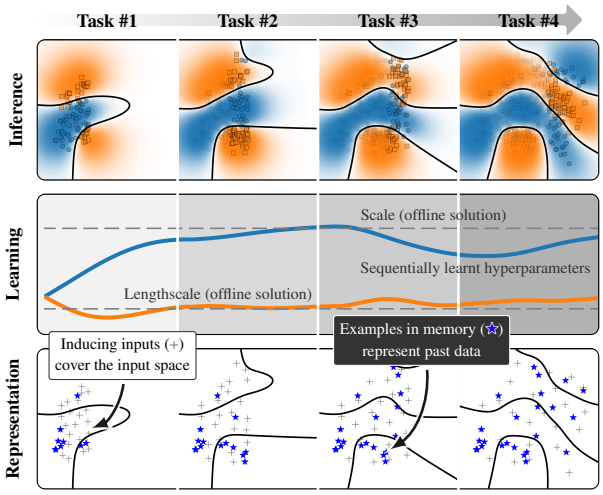


Figure 1. Our sequential learning method provides an accurate posterior (top), kernel hyperparameters (middle), and sparse representation (bottom). A key contribution is to add memory of relevant past data (shown in the bottom row with ★) in addition to inducing inputs (shown with +).

1. Introduction

Sequential decision-making requires uncertainty estimates that can be used to plan for the future. For this reason, Gaussian process (GP) models are popular for sequential problems in applications such as model-based reinforcement learning (Deisenroth & Rasmussen, 2011) and Bayesian optimization (Garnett, 2023). However, exact online inference in GPs requires access to all past data, which becomes infeasible over time as the amount of data grows (Csató & Opper, 2002). Sparse GP methods can reduce this cost, but they too assume access to all past data. For example, the popular sparse variational GP (SVGP) and variational free energy (VFE, Titsias, 2009) methods require multiple passes through the data during the stochastic training (Hensman et al., 2013). This can lead to inaccuracies for continual or active learning, where access to past data is limited and errors can accumulate over time.

Errors can arise from multiple sources: inaccurate posteriors, wrong hyperparameter values, or poor sparse representations. Various techniques can be used to control such errors, but past attempts struggled to find a coherent solution to the problem. Initially, Csató & Opper (2002) used expectation propagation (EP) for inference and a projection method for sparsity but did not estimate hyperparameters. More recently, Bui et al. (2017) did estimate hyperparameters but found EP to perform worse than variational inference. Maddox et al. (2021) did not optimize hyperparameters or use the evidence lower bound (ELBO) objective but instead resorted to a Laplace approximation for non-Gaussian likelihoods. Their method to obtain sparse representation uses a gradient method combined with pivoted Cholesky. Kapoor et al. (2021) attempt to improve performance by using a structured covariance during inference, but the cost grows with tasks which is due to the increasing size of the sparse representation. These attempts use a mix of methods to control various errors, which all lead to complications. We aim to build a single coherent method by tackling various errors simultaneously; see Fig. 1 for an overview.

We adapt the dual-SVGP method of Adam et al. (2021) to perform sequential learning with generic likelihoods. A key contribution of our work is to improve performance by

[*]Equal contribution [1]Department of Computer Science, Aalto University, Finland [2]Finnish Center for Artificial Intelligence (FCAI) [3]RIKEN Center for AI Project, Tokyo, Japan. Correspondence to: Paul Chang <paul.chang@aalto.fi>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

adding a memory set of past examples. A sufficiently large memory can achieve arbitrarily close performance to the batch case, and the performance loss can be minimized by choosing the set carefully. For such selection, we propose a new score called the *Bayesian leverage score* which uses the dual parameters of the dual-SVGP posterior to characterize relative importance of past examples. Overall, our memory-based SVGP enables us to tackle the various errors arising in sequential learning. It also avoids the complications arising in previous work on sequential learning, as discussed below:

1. Unlike Csató & Opper (2002), we aim to minimize the ELBO which aligns better with Bui et al. (2017) who also found the ELBO to give better results than EP. Our dual parameters can be seen as an estimate of those used in Csató & Opper (2002, Lemma 1).

2. Similarly to Bui et al. (2017), we obtain a pseudo-data interpretation but it is derived from the dual parameterization. Our approach directly addresses the issues they aim to solve and is more straightforward.

3. Unlike Maddox et al. (2021), we estimate hyperparameters and use a variational method for inference.

4. Unlike Kapoor et al. (2021), we use a fixed number of inducing inputs and instead increase the memory size. The cost grows linearly as opposed to a cubically increasing cost of their method. We also do not need any regularization for hyperparameters.

Our use of memory is similar to recent continual learning methods (Nguyen et al., 2018; Titsias et al., 2020; Pan et al., 2020; Khan & Swaroop, 2021), but we select the memory via an extension of leverage scores (Cook & Weisberg, 1980; Alaoui & Mahoney, 2015). We demonstrate effectiveness of our approach on several applications involving Bayesian optimization, active learning, and continual learning.

2. Sequential Learning with a GP

We consider sequential learning in models with a GP prior over functions, $f \sim \mathcal{GP}(0, \kappa_{\boldsymbol{\theta}})$. The prior is characterized by a covariance (kernel) function $\kappa_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{x}')$, where $\mathbf{x}, \mathbf{x}'$ are input vectors and $\boldsymbol{\theta}$ denotes the hyperparameters. Observations $(\mathbf{x}_i, y_i)$ are modeled by the likelihood $p(y_i \mid f_i)$ given a function value $f_i = f(\mathbf{x}_i)$. In the sequential setting, we first compute the posterior on $\mathcal{D}_{\text{old}} = (\mathbf{X}_{\text{old}}, \mathbf{y}_{\text{old}})$ where $\mathbf{X}_{\text{old}}$ is a matrix containing all the past inputs $\mathbf{x}_i^\top$ as rows and $\mathbf{y}_{\text{old}}$ is a vector containing all the past outputs y_i. Then, when new data $\mathcal{D}_{\text{new}} = (\mathbf{X}_{\text{new}}, \mathbf{y}_{\text{new}})$ is observed, our goal is to update the posterior and hyperparameters.

The posterior inference in the sequential setting is challenging because the cost grows cubically in the size of data (denoted by n). We can see this by expressing the posterior $p(f_i|\mathbf{y})$ as follows (Csató & Opper, 2002, Lemma 1),

$$\begin{aligned} \mathbb{E}_{p(f_i \mid \mathbf{y})}[f_i] &= \mathbf{k}_{\mathbf{x}i}^\top \boldsymbol{\alpha}, \\ \text{Var}_{p(f_i \mid \mathbf{y})}[f_i] &= \kappa_{ii} - \mathbf{k}_{\mathbf{x}i}^\top (\mathbf{K}_{\mathbf{xx}} + \text{diag}(\boldsymbol{\beta})^{-1})^{-1} \mathbf{k}_{\mathbf{x}i}, \end{aligned} \tag{1}$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are vectors of n dual parameters,

$$\begin{aligned} \alpha_i &= \mathbb{E}_{p(f_i \mid \mathbf{y})}[\nabla_{f_i} \log p(y_i \mid f_i)], \\ \beta_i &= \mathbb{E}_{p(f_i \mid \mathbf{y})}[-\nabla^2_{f_i} \log p(y_i \mid f_i)], \end{aligned} \tag{2}$$

respectively; a derivation is given in App. A.1. Here, we use $\mathbf{K}_{\mathbf{xx}}$ to denote the $n \times n$ matrix with $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ as the ij^{th} entry. Similarly, $\mathbf{k}_{\mathbf{x}i}$ denotes a vector where each j^{th} element is $\kappa(\mathbf{x}_i, \mathbf{x}_j)$, and $\kappa_{ii} = \kappa(\mathbf{x}_i, \mathbf{x}_i)$. The parameters (α_i, β_i) are the dual parameters that arise in the dual formulations used in, for example, support vector machines (Cortes & Vapnik, 1995), whose origins are found in work by Kimeldorf & Wahba (1971). The posterior holds for generic likelihoods, even a non-Gaussian one, and no approximations are involved.

More importantly, the expressions in Eq. (1) and Eq. (2) clearly show the challenge of sequential learning, where as data grows, so do $(\boldsymbol{\alpha}, \boldsymbol{\beta})$, and naïve computation of the posterior is now $O((n_{\text{old}} + n_{\text{new}})^3)$ due to the matrix inversion, quickly making inference infeasible. Csató & Opper (2002) suggested storing and using only a subset of past data to reduce the computation. They estimate (α_i, β_i) using a Gaussian approximation to $p(f_i \mid \mathbf{y})$, obtained by EP, but did not consider updating the hyperparameters.

In the full-batch case, scaling can be improved using the SVGP method (Titsias, 2009; Hensman et al., 2013), which optimizes an ELBO to select hyperparameters $\boldsymbol{\theta}$, inducing inputs $\mathbf{Z} = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_m)$ with $m \ll n$, and the posterior $q_{\mathbf{u}}(\mathbf{u}) = \text{N}(\mathbf{u} \mid \mathbf{m}, \mathbf{V})$ defined over functions $\mathbf{u}_i = f(\boldsymbol{z}_i)$ at $\boldsymbol{z}_i$. The ELBO is given by

$$\mathcal{L}_{\text{batch}} = \sum_{i \in \mathcal{D}_{\text{old}} \cup \mathcal{D}_{\text{new}}} \mathbb{E}_{q_{\mathbf{u}}(f_i)}[\log p(y_i \mid f_i)] - \mathbb{D}_{\text{KL}}[q_{\mathbf{u}}(\mathbf{u}) \,\|\, p_{\boldsymbol{\theta}}(\mathbf{u})], \tag{3}$$

which uses the following posterior predictive distribution,

$$q_{\mathbf{u}}(f_i) = \text{N}(f_i \mid \boldsymbol{a}_i^\top \mathbf{m}, \kappa_{ii} - \boldsymbol{a}_i^\top \mathbf{K}_{\mathbf{zz}} \boldsymbol{a}_i + \boldsymbol{a}_i^\top \mathbf{V} \boldsymbol{a}_i), \tag{4}$$

where $\boldsymbol{a}_i^\top = \mathbf{k}_{\mathbf{z}i}^\top \mathbf{K}_{\mathbf{zz}}^{-1}$, and $p_{\boldsymbol{\theta}}(\mathbf{u})$ is the GP prior over $\mathbf{u}$. The last term in the bound is the Kullback–Leibler divergence (KLD). The method assumes that all data is available throughout training, so it does not directly apply in the sequential setting.

Bui et al. (2017) extend the SVGP method in an ad-hoc way by adding two additional KLD terms to match the previous posterior and prior at the old inducing inputs. Let $\mathbf{Z}_{\text{old}}$ denote the old inducing inputs with $\mathbf{u}_{\text{old}} = f(\mathbf{Z}_{\text{old}})$ and posterior predictive $q_{\mathbf{u}_{\text{old}}}(f_i)$. Also, let $p_{\boldsymbol{\theta}_{\text{old}}}(f_i)$ denote the

prior for the old hyperparameters $\boldsymbol{\theta}_{\text{old}}$. Given the new data $\mathcal{D}_{\text{new}}$, Bui et al. modify the ELBO as follows,

$$\begin{aligned}\sum_{i\in\mathcal{D}_{\text{new}}} \mathbb{E}_{q_{\mathbf{u}}(f_i)}[\log p(y_i \mid f_i)] - \mathbb{D}_{\text{KL}}[q_{\mathbf{u}}(\mathbf{u}) \,\|\, p_{\boldsymbol{\theta}}(\mathbf{u})] \\ + \mathbb{D}_{\text{KL}}[q_{\mathbf{u}}(\mathbf{u}_{\text{old}}) \,\|\, p_{\boldsymbol{\theta}_{\text{old}}}(\mathbf{u}_{\text{old}}))] \\ - \mathbb{D}_{\text{KL}}[q_{\mathbf{u}}(\mathbf{u}_{\text{old}}) \,\|\, q_{\mathbf{u}_{\text{old}}}(\mathbf{u}_{\text{old}})],\end{aligned} \tag{5}$$

where the last two KLD terms are the newly added terms which are meant to penalize deviations from both the old prior and posterior. This is an indirect way to keep the solutions close to the full-batch case, but it does not always work well (we give empirical evidence in Sec. 4).

Maddox et al. (2021) revisit the approach of Bui et al. (2017) in the context of Bayesian optimization and active learning. They simplify the derivation for the Gaussian case via pseudo-data. They resort to a Laplace approximation to handle non-Gaussian likelihoods. Laplace approximation is a more 'local' one than those obtained by variational methods and it can give worse results (Opper & Archambeau, 2009). In addition, they do not optimize hyperparameters.

We will now show that the above difficulties can be alleviated by using the method of Adam et al. (2021): we can estimate the parameterization of Csató & Opper (2002), improve the ELBO of Bui et al. (2017), improve over Laplace approximation of Maddox et al. (2021), and, unlike Kapoor et al. (2021), keep the computational cost from exploding.

3. A Memory-based Dual-SVGP Method

We adapt the dual-SVGP method of Adam et al. (2021) to enable accurate inference in the sequential case. We first describe the dual form of the SVGP solution and then use it to derive a new memory-based objective to perform sequential learning. Then, we optimize the new objective by using the dual-SVGP algorithm to update the posterior, hyperparameters, and inducing inputs. Finally, we present the new Bayesian leverage score to select the memory.

3.1. The Dual Form of the SVGP Solution

The dual form of the sequential SVGP solution has a strikingly similar form to the full-GP case given in Eq. (1). Consider the following ELBO defined over the old data,

$$\mathcal{L}_{\text{old}} = \sum_{i\in\mathcal{D}_{\text{old}}} \mathbb{E}_{q_{\mathbf{u}}(f_i)}[\log p(y_i \mid f_i)] - \mathbb{D}_{\text{KL}}[q_{\mathbf{u}}(\mathbf{u}) \,\|\, p_{\boldsymbol{\theta}}(\mathbf{u})]. \tag{6}$$

Adam et al. (2021) show that a stationary point $q^{\text{old}}_{\mathbf{u}}(\mathbf{u}) = \mathrm{N}(\mathbf{u} \,|\, \mathbf{m}^{\text{old}}, \mathbf{V}^{\text{old}})$ has a dual form for its natural parameters:

$$(\mathbf{V}^{\text{old}})^{-1}\mathbf{m}^{\text{old}} = \textstyle\sum_{i\in\mathcal{D}_{\text{old}}} \boldsymbol{a}_i \hat{\beta}^{\text{old}}_i \hat{y}^{\text{old}}_i, \tag{7}$$

$$(\mathbf{V}^{\text{old}})^{-1} = \textstyle\sum_{i\in\mathcal{D}_{\text{old}}} \boldsymbol{a}_i \hat{\beta}^{\text{old}}_i \boldsymbol{a}_i^\top + \mathbf{K}^{-1}_{\mathbf{zz}}, \tag{8}$$

where $\hat{y}^{\text{old}}_i = \hat{\alpha}^{\text{old}}_i / \hat{\beta}^{\text{old}}_i + \boldsymbol{a}_i^\top \mathbf{m}^{\text{old}}$ is a 'pseudo' output, and $(\hat{\alpha}^{\text{old}}_i, \hat{\beta}^{\text{old}}_i)$ are defined as

$$\begin{aligned}\hat{\alpha}^{\text{old}}_i &= \mathbb{E}_{q^{\text{old}}_{\mathbf{u}}(f_i)}[\nabla_{f_i} \log p(y_i \mid f_i)], \\ \hat{\beta}^{\text{old}}_i &= \mathbb{E}_{q^{\text{old}}_{\mathbf{u}}(f_i)}[-\nabla^2_{f_i} \log p(y_i \mid f_i)].\end{aligned} \tag{9}$$

The result follows from Eq. (21) in Adam et al. (2021). A proof is given in App. A.2 where we also derive a dual-form for the predictive distribution $q^{\text{old}}_{\mathbf{u}}(f_i)$:

$$\mathbb{E}_{q^{\text{old}}_{\mathbf{u}}(f)}[f_i] = \mathbf{k}_{\mathbf{z}i}^\top \mathbf{K}^{-1}_{\mathbf{zz}} \boldsymbol{\alpha}^{\text{old}}_{\mathbf{u}}, \tag{10}$$

$$\mathrm{Var}_{q^{\text{old}}_{\mathbf{u}}(f)}[f_i] = \kappa_{ii} - \mathbf{k}_{\mathbf{z}i}^\top \left[\mathbf{K}^{-1}_{\mathbf{zz}} - \left(\mathbf{K}_{\mathbf{zz}} + \boldsymbol{B}^{\text{old}}_{\mathbf{u}}\right)^{-1}\right] \mathbf{k}_{\mathbf{z}i}.$$

The form is strikingly similar to Eq. (1) but it uses two dual parameters consisting of an m-length vector $\boldsymbol{\alpha}^{\text{old}}_{\mathbf{u}}$ and $m \times m$-size matrix $\boldsymbol{B}^{\text{old}}_{\mathbf{u}}$, defined as

$$\begin{aligned}\boldsymbol{\alpha}^{\text{old}}_{\mathbf{u}} &= \textstyle\sum_{i\in\mathcal{D}_{\text{old}}} \mathbf{k}_{\mathbf{z}i}\, \hat{\alpha}^{\text{old}}_i, \\ \boldsymbol{B}^{\text{old}}_{\mathbf{u}} &= \textstyle\sum_{i\in\mathcal{D}_{\text{old}}} \mathbf{k}_{\mathbf{z}i}\, \hat{\beta}^{\text{old}}_i\, \mathbf{k}_{\mathbf{z}i}^\top.\end{aligned} \tag{11}$$

The pair $(\boldsymbol{\alpha}^{\text{old}}_{\mathbf{u}}, \boldsymbol{B}^{\text{old}}_{\mathbf{u}})$ can be seen as 'amortized' dual parameters which do not depend on the data size n, but rather provide a distilled compact summary through m inducing inputs. Similarly, the pair $(\hat{\alpha}^{\text{old}}_i, \hat{\beta}^{\text{old}}_i)$ can be seen as an estimate of (α_i, β_i) in Eq. (2) obtained by using $q^{\text{old}}_{\mathbf{u}}(f_i)$ instead of the exact posterior $p(f_i|\mathbf{y}^{\text{old}})$.

The posterior can also be expressed in a form where likelihoods are replaced by their Gaussian approximations,

$$\begin{aligned}q^{\text{old}}_{\mathbf{u}}(\mathbf{u}) &\propto e^{\mathbf{u}^\top (\mathbf{V}^{\text{old}})^{-1}\mathbf{m}^{\text{old}} - \frac{1}{2}\mathbf{u}^\top (\mathbf{V}^{\text{old}})^{-1}\mathbf{u}} \\ &\propto p_{\boldsymbol{\theta}}(\mathbf{u})\; e^{\mathbf{u}^\top (\mathbf{V}^{\text{old}})^{-1}\mathbf{m}^{\text{old}} - \frac{1}{2}\mathbf{u}^\top \left((\mathbf{V}^{\text{old}})^{-1} - \mathbf{K}^{-1}_{\mathbf{zz}}\right)\mathbf{u}} \\ &\propto p_{\boldsymbol{\theta}}(\mathbf{u}) \textstyle\prod_{i\in\mathcal{D}_{\text{old}}} e^{-\frac{1}{2}\hat{\beta}^{\text{old}}_i (\hat{y}^{\text{old}}_i - \boldsymbol{a}_i^\top \mathbf{u})^2},\end{aligned} \tag{12}$$

where the second line is obtained by adding and subtracting $\mathbf{u}^\top \mathbf{K}^{-1}_{\mathbf{zz}} \mathbf{u}/2$ and the third line is obtained by using Eqs. (7) and (8) and then completing the squares. Previous works used EP to obtain such site parameters (Csató & Opper, 2002; Bui et al., 2017), but we obtain them with SVGP too.

Adam et al. (2021) further show that the dual form can be used to improve hyperparameter optimization. The idea is to fix the dual parameters in Eq. (12) and treat both $p_{\boldsymbol{\theta}}(\mathbf{u})$ and $\boldsymbol{a}_i = \mathbf{k}_{\mathbf{z}i}^\top \mathbf{K}^{-1}_{\mathbf{zz}}$ as functions of $\boldsymbol{\theta}$. Then, plugging Eq. (12) into Eq. (6) gives rise to an objective which, compared to the ELBO, is better aligned with the marginal likelihood; see App. B of Adam et al. (2021). They propose a stochastic expectation-maximization procedure in which the posterior is updated by maximizing Eq. (6) and hyperparameters are updated using the new objective.

In the next section, we will derive a new objective for sequential learning where memory is used to mimic the batch-SVGP of Eq. (3). Similarly to Bui et al. (2017), we also use the pseudo-data idea, but our approach is more straightforward and directly addresses the issues they aim to solve.

3.2. Memory-based Objective for Sequential Learning

Our goal is to closely mimic the batch-SVGP objective given in Eq. (3). We make two modifications to the batch objective. First, instead of $\mathcal{D}_{\text{old}}$, we use a subset of examples stored in a memory set $\mathcal{M}$:

$$\sum_{i\in\mathcal{D}_{\text{new}}\cup\mathcal{M}} \mathbb{E}_{q_{\mathbf{u}}(f_i)}[\log p(y_i \mid f_i)] - \mathbb{D}_{\text{KL}}[q_{\mathbf{u}}(\mathbf{u}) \,\|\, p_{\boldsymbol{\theta}}(\mathbf{u})]. \quad (13)$$

For $\mathcal{M} = \mathcal{D}_{\text{old}}$, we exactly recover $\mathcal{L}_{\text{batch}}$ and, for small memory size, the error can be minimized by carefully choosing $\mathcal{M}$ as a representative set of $\mathcal{D}_{\text{old}}$.

Second, we reuse the old posterior $q^{\text{old}}_{\mathbf{u}}(\mathbf{u})$ which contains a distilled summary of the old data in its dual parameters. Our key idea is to express the prior in terms of the old posterior by using Eq. (12),

$$p_{\boldsymbol{\theta}}(\mathbf{u}) \propto \frac{q^{\text{old}}_{\mathbf{u}}(\mathbf{u})}{\prod_{i\in\mathcal{D}_{\text{old}}} e^{-\frac{1}{2}\hat{\beta}^{\text{old}}_i(\hat{y}^{\text{old}}_i - \boldsymbol{a}_i^\top \mathbf{u})^2}} \propto \frac{q^{\text{old}}_{\mathbf{u}}(\mathbf{u})}{\hat{p}_{\mathcal{D}_{\text{old}}}(\mathbf{u})}, \quad (14)$$

where we rewrite the denominator as a Gaussian:

$$\hat{p}_{\mathcal{D}_{\text{old}}}(\mathbf{u}) = \mathrm{N}(\mathbf{u} \,|\, \tilde{\mathbf{y}}^{\text{old}}, \tilde{\boldsymbol{\Sigma}}^{\text{old}}), \quad (15)$$

$$\tilde{\mathbf{y}}^{\text{old}} = \mathbf{K}_{\mathbf{zz}}(\boldsymbol{B}^{\text{old}}_{\mathbf{u}})^{-1}\boldsymbol{\alpha}^{\text{old}}_{\mathbf{u}} + \mathbf{m}^{\text{old}}_{\mathbf{u}}, \quad (16)$$

$$\tilde{\boldsymbol{\Sigma}}^{\text{old}} = \mathbf{K}_{\mathbf{zz}}(\boldsymbol{B}^{\text{old}}_{\mathbf{u}})^{-1}\mathbf{K}_{\mathbf{zz}}. \quad (17)$$

A derivation is in App. A.3. The denominator can be seen as pseudo-data, similar to those derived in Bui et al. (2017). Eq. (14) is exact whenever $q^{\text{old}}_{\mathbf{u}}(\mathbf{u})$ is an exact stationary point of Eq. (6), for example, in the very first update. However, in sequential learning, $q^{\text{old}}_{\mathbf{u}}(\mathbf{u})$ is almost never exact because errors can accumulate due to lack of full access to data at subsequent updates. To handle this, we rewrite Eq. (15) where only examples in $\mathcal{M}$ are used in the denominator:

$$\frac{q^{\text{old}}_{\mathbf{u}}(\mathbf{u})}{\hat{p}_{\mathcal{D}_{\text{old}}}(\mathbf{u})} \approx \frac{\hat{q}^{\text{old}}_{\mathbf{u}}(\mathbf{u})}{\hat{p}_{\mathcal{M}}(\mathbf{u})}, \quad (18)$$

where $\hat{q}^{\text{old}}_{\mathbf{u}}$ is an estimate of $q^{\text{old}}_{\mathbf{u}}(\mathbf{u})$ (see Sec. 3.3). The denominator is defined in similar ways but using only $\mathcal{M}$. We will use this as our new prior which can be obtained from the dual parameters by removing contributions of $\mathcal{M}$,

$$\boldsymbol{\alpha}^{\text{old}\setminus\mathcal{M}}_{\mathbf{u}} = \boldsymbol{\alpha}^{\text{old}}_{\mathbf{u}} - \textstyle\sum_{i\in\mathcal{M}} \mathbf{k}_{\mathbf{z}i}\hat{\alpha}^{\text{old}}_i, \quad (19)$$

$$\boldsymbol{B}^{\text{old}\setminus\mathcal{M}}_{\mathbf{u}} = \boldsymbol{B}^{\text{old}}_{\mathbf{u}} - \textstyle\sum_{i\in\mathcal{M}} \mathbf{k}_{\mathbf{z}i}\hat{\beta}^{\text{old}}_i\mathbf{k}^\top_{\mathbf{z}i}. \quad (20)$$

Here, for notational simplicity, we use $(\boldsymbol{\alpha}^{\text{old}}_{\mathbf{u}}, \boldsymbol{B}^{\text{old}}_{\mathbf{u}})$ to denote the dual parameters of $\hat{q}^{\text{old}}_{\mathbf{u}}(\mathbf{u})$. A derivation is in App. A.3 along with expressions for the mean and covariance.

Replacing $p_{\boldsymbol{\theta}}(\mathbf{u})$ in Eq. (13) with the new prior ($\mathcal{Z}$ denotes the normalizing constant), we arrive at

$$\mathcal{L}^q_{\text{seq}} = \sum_{i\in\mathcal{D}_{\text{new}}\cup\mathcal{M}} \mathbb{E}_{q_{\mathbf{u}}(f_i)}[\log p(y_i \mid f_i)] - \mathbb{D}_{\text{KL}}\Big[q_{\mathbf{u}}(\mathbf{u}) \Big\| \frac{\hat{q}^{\text{old}}_{\mathbf{u}}(\mathbf{u})}{\mathcal{Z}\hat{p}_{\mathcal{M}}(\mathbf{u})}\Big]. \quad (21)$$

Why do we expect the new objective in Eq. (21) to be more accurate? One explanation is to see the new objective as an improved version of variational continual-learning (VCL) (Nguyen et al., 2018), by rewriting it as

$$\mathcal{L}^q_{\text{seq}} = \sum_{i\in\mathcal{D}_{\text{new}}} \mathbb{E}_{q_{\mathbf{u}}(f_i)}[\log p(y_i \mid f_i)] - \mathbb{D}_{\text{KL}}[q_{\mathbf{u}}(\mathbf{u}) \,\|\, \hat{q}^{\text{old}}_{\mathbf{u}}(\mathbf{u})] + \sum_{i\in\mathcal{M}} \mathbb{E}_{q_{\mathbf{u}}(f_i)}[\log p(y_i \mid f_i)] - \mathbb{E}_{q_{\mathbf{u}}(\mathbf{u})}[\log \hat{p}_{\mathcal{M}}(\mathbf{u})]. \quad (22)$$

The first two terms are equal to the VCL objective where the estimated old posterior is used as the new prior. VCL is exact only when the old posterior is exact, which, as discussed before, is unlikely in practice. To address this issue, in the third term we add a few representative examples of $\mathcal{D}_{\text{old}}$ using $\mathcal{M}$. The fourth term is subtracting the pseudo-data to avoid double-counting the contributions of the examples in $\mathcal{M}$.

The new objective directly addresses the issues raised by Bui et al. (2017) regarding hyperparameter learning, who add two KLD terms (see the last two lines of Eq. (5)). This essentially uses the ratio $q_{\mathbf{u}_{\text{old}}}(\mathbf{u}_{\text{old}})/p_{\boldsymbol{\theta}_{\text{old}}}(\mathbf{u}_{\text{old}})$, which is proportional to the pseudo-data. In contrast, in our objective, we use the exact likelihood (the third term in Eq. (22)) in addition to the pseudo-data (the fourth term). The two terms are added on top of the VCL objective where $\hat{q}^{\text{old}}_{\mathbf{u}}(\mathbf{u})$ is used.

In the limit of $\mathcal{M} = \mathcal{D}_{\text{old}}$, we have $\mathcal{L}^q_{\text{seq}} = \mathcal{L}_{\text{batch}}$ (assuming $\boldsymbol{Z}$ and $\boldsymbol{\theta}$ to be the same): In this case, the first terms of both Eq. (21) and Eq. (3) are equal; the second terms can also be seen to be equal using Eq. (14). By using a good representative memory of the past data, we expect to mimic the batch-SVGP objective. Using memory to improve sequential learning is unique to our approach.

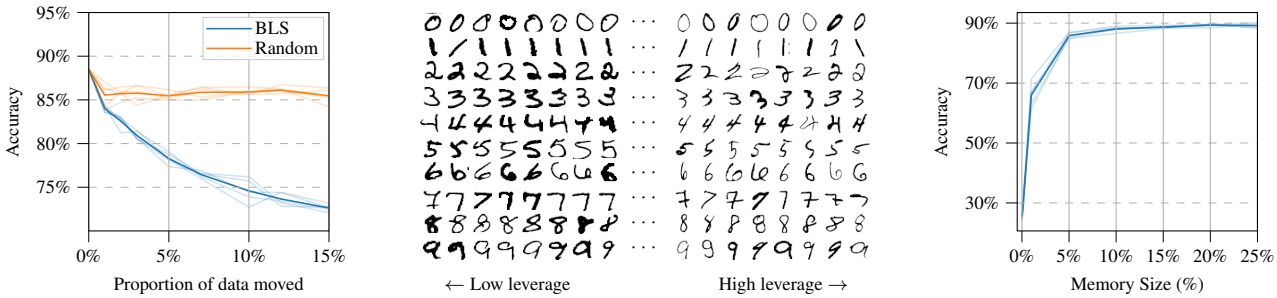
3.3. Inference using the Memory-Based Objective

We will optimize the new objective in Eq. (21) by using the Bayesian learning rule (BLR) of Khan & Rue (2021) which is a natural-gradient descent algorithm. This method is also used by Adam et al. (2021) who write the update in a natural-parameter form. We will instead use a form where we update the estimate of the dual pair $(\boldsymbol{\alpha}^{(t)}_{\mathbf{u}}, \boldsymbol{B}^{(t)}_{\mathbf{u}})$ at each iteration t. A detailed derivation is given in App. A.4 and below we give the final update,

$$\boldsymbol{\alpha}^{(t)}_{\mathbf{u}} \leftarrow (1-\rho)\boldsymbol{\alpha}^{(t-1)}_{\mathbf{u}} + \rho\Big(\boldsymbol{\alpha}^{\text{old}\setminus\mathcal{M}}_{\mathbf{u}} + \sum_{i\in\mathcal{D}_{\text{new}}} \mathbf{k}_{\mathbf{z}i}\hat{\alpha}^{(t)}_i\Big),$$

$$\boldsymbol{B}^{(t)}_{\mathbf{u}} \leftarrow (1-\rho)\boldsymbol{B}^{(t-1)}_{\mathbf{u}} + \rho\Big(\boldsymbol{B}^{\text{old}\setminus\mathcal{M}}_{\mathbf{u}} + \sum_{i\in\mathcal{D}_{\text{new}}} \mathbf{k}_{\mathbf{z}i}\hat{\beta}^{(t)}_i\mathbf{k}^\top_{\mathbf{z}i}\Big). \quad (23)$$

The update adds the contribution of the new data using the dual parameters $(\hat{\alpha}^{(t)}_i, \hat{\beta}^{(t)}_i)$ which are defined similarly to Eqs. (7) and (8) but by using the most recent posterior $q^{(t-1)}_{\mathbf{u}}(f_i)$. The mean and covariance of the posterior



(a) Accuracy vs. representation (b) Data points in each class ranked by BLS (c) Accuracy vs. memory size

Figure 2. Ablation study on *split MNIST* to explore the benefits of memory and Bayesian leverage score (BLS). (a) Evolution of test set accuracy of our method as we move data from training set to test set based on BLS ranking vs. randomly. (b) Digits from the training set with lowest↔highest BLS, high BLS digits are more unusual and more difficult to learn than digits with low BLS score. (c) Evolution of test set accuracy as the memory size is increased; memory size of just 5% achieves satisfactory performance.

needed to compute the expectations are obtained as follows,

$$\mathbf{m}^{(t)} \leftarrow \boldsymbol{\alpha}_{\mathbf{u}}^{(t)}, \quad \mathbf{V}^{(t)} \leftarrow \left(\mathbf{K}_{\mathbf{zz}}^{-1}\boldsymbol{B}_{\mathbf{u}}^{(t)}\mathbf{K}_{\mathbf{zz}}^{-1} + \mathbf{K}_{\mathbf{zz}}^{-1}\right)^{-1}. \quad (24)$$

3.4. Learning of Hyperparameter and Inducing Inputs

For hyperparameter learning, we will use the new ELBO derived by Adam et al. (2021) who show faster convergence; also see the recent work by Li et al. (2023). As before, we will derive it by assuming that $\mathcal{D}_{\text{old}}$ is available and then we will make approximations to handle the sequential case.

We follow the procedure given in Adam et al. (2021, App. B). Let $q_{\mathbf{u}}^{\text{new}}(\mathbf{u})$ denote the solution obtained by maximizing Eq. (3). The idea is to take its dual form and fix $(\hat{y}_i^{\text{new}}, \hat{\beta}_i^{\text{new}})$ while treating both $p_{\boldsymbol{\theta}}(\mathbf{u})$ and $\boldsymbol{a}_i$ as functions of $\boldsymbol{\theta}$:

$$q_{\mathbf{u}}^{\text{new}}(\mathbf{u};\boldsymbol{\theta}) \propto p_{\boldsymbol{\theta}}(\mathbf{u}) \prod_{i\in\mathcal{D}} e^{-\frac{1}{2}\hat{\beta}_i^{\text{new}}(\hat{y}_i^{\text{new}} - \mathbf{u}^\top \boldsymbol{a}_i^{\boldsymbol{\theta}})^2}, \quad (25)$$

where $\mathcal{D} = \mathcal{D}^{\text{new}} \cup \mathcal{D}^{\text{old}}$ and we have indicated explicit dependency on $\boldsymbol{\theta}$ by denoting $\boldsymbol{a}_i^{\boldsymbol{\theta}}$ and $q_{\mathbf{u}}^{\text{new}}(\mathbf{u};\boldsymbol{\theta})$. Then, plugging Eq. (25) in Eq. (6) gives the following objective (see App. B in Adam et al., 2021, for details):

$$\mathcal{L}_{\text{seq}}^{\boldsymbol{\theta}} = \log \mathcal{Z}(\boldsymbol{\theta}) + c(\boldsymbol{\theta}), \quad (26)$$

$$c(\boldsymbol{\theta}) = \sum_{i\in\mathcal{D}} \mathbb{E}_{q(f_i;\boldsymbol{\theta})}[\log p(y_i \mid f_i)] - \mathbb{E}_{q(\mathbf{u};\boldsymbol{\theta})}[\log \hat{p}_{\mathcal{D}}(\mathbf{u})],$$

$$\log \mathcal{Z}(\boldsymbol{\theta}) = -\frac{1}{2}\log\left|\mathbf{K}_{\mathbf{zz}}^{\boldsymbol{\theta}}(\boldsymbol{B}_{\mathbf{u}}^{\text{new}})^{-1}\mathbf{K}_{\mathbf{zz}}^{\boldsymbol{\theta}} + \mathbf{K}_{\mathbf{zz}}^{\boldsymbol{\theta}}\right| - \frac{1}{2}\boldsymbol{b}^\top\left(\boldsymbol{B}_{\mathbf{u}}^{\text{new}} + \mathbf{K}_{\mathbf{zz}}^{\boldsymbol{\theta}}\right)^{-1}\boldsymbol{b} + \text{const.}$$

Here, we keep $\boldsymbol{B}_{\mathbf{u}}^{\text{new}}$ fixed, and treat everything else as a function of $\boldsymbol{\theta}$. We define $\boldsymbol{b} = (\boldsymbol{B}_{\mathbf{u}}^{\text{new}})^{-1}\boldsymbol{\alpha}_{\mathbf{u}}^{\text{new}} + \mathbf{m}^{\text{new}}$ which contains the part in $\tilde{\mathbf{y}}^{\text{new}}$ that are fixed.

For the sequential setting, we modify the first term in $c(\boldsymbol{\theta})$ by using a stochastic approximation for the sum over $\mathcal{D}_{\text{old}}$. We approximate it by using $\mathcal{M}$ and scale it to get an unbiased estimate of the gradient,

$$\sum_{i\in\mathcal{D}^{\text{new}}} \mathbb{E}_{q(f_i;\boldsymbol{\theta})}[\log p(y_i \mid f_i)] + \frac{n_{\text{old}}}{n_{\mathcal{M}}} \sum_{i\in\mathcal{M}} \mathbb{E}_{q(f_i;\boldsymbol{\theta})}[\log p(y_i \mid f_i)].$$

Here, n_{old} and $n_{\mathcal{M}}$ are the size of $\mathcal{D}_{\text{old}}$ and $\mathcal{M}$ respectively. This is similar to the approach of Titsias et al. (2020).

To update the inducing inputs, we use the pivoted-Cholesky method of Burt et al. (2020). A similar method is used by Maddox et al. (2021) but in combination with gradient based optimization. We find that our framework gives much better performance using the pivoted-Cholesky alone, and we do not need any gradient based optimization of inducing points. We concatenate $[\mathbf{Z}_{\text{old}}, \mathbf{X}_{\text{new}}]$ and use pivoted-Cholesky to get the new inducing points $\mathbf{Z}_{\text{new}}$. Having moved the inducing points to $\mathbf{Z}_{\text{new}}$, we now need to adjust the variational parameters which are still defined over the old inducing points $\mathbf{Z}_{\text{old}}$. We adjust this using a projection matrix $\mathbf{P} = \mathbf{K}_{\mathbf{Z}_{\text{new}},\mathbf{Z}_{\text{old}}}\mathbf{K}_{\mathbf{Z}_{\text{old}},\mathbf{Z}_{\text{old}}}^{-1}$ to get the new dual parameters,

$$\boldsymbol{\alpha}_{\mathbf{u}} \leftarrow \mathbf{P}\boldsymbol{\alpha}_{\mathbf{u}} \quad \text{and} \quad \boldsymbol{B}_{\mathbf{u}} \leftarrow \mathbf{P}\boldsymbol{B}_{\mathbf{u}}\mathbf{P}^\top. \quad (27)$$

3.5. Memory Selection using Bayesian Leverage Score

Another key contribution of our work is to select and update a memory of the past data to improve learning. We present a new score called the Bayesian leverage score (BLS) to actively build and update the memory. The score extends the classical ridge leverage score (RLS, Alaoui & Mahoney, 2015), which is commonly used to select subsets for linear regression. We generalise it to the SVGP case, and use it to build a memory for sequential learning. Fig. 2b gives an example of ranking data instances by BLS (details in Sec. 4.3).

Algorithm 1 Dual-SVGP with memory

Initialize $\boldsymbol{\alpha}_{\mathbf{u}}, \boldsymbol{B}_{\mathbf{u}}, \mathbf{Z}_{\text{old}}, \boldsymbol{\theta}, \mathcal{M}$
for each new data **do**
  Observe new data $(\mathbf{y}_{\text{new}}, \mathbf{X}_{\text{new}})$
  Update $\mathbf{Z}$ using pivoted-Cholesky on $[\mathbf{Z}_{\text{old}}, \mathbf{X}_{\text{new}}]$
  Project old $(\boldsymbol{\alpha}_{\mathbf{u}}, \boldsymbol{B}_{\mathbf{u}})$ to new $\mathbf{Z}$ using Eq. (27)
  Update $(\boldsymbol{\alpha}_{\mathbf{u}}, \boldsymbol{B}_{\mathbf{u}})$ by using Eqs. (23) and (24)
  Update $\boldsymbol{\theta}$ by optimizing Eq. (26)
  Update memory $\mathcal{M}$ using the method from Sec. 3.5
end for

Consider the following linear model, $y_i = \boldsymbol{a}_i^\top \mathbf{u} + \epsilon_i$, where y_i are the observations, $\boldsymbol{a}_i$ are the features, $\mathbf{u}$ is the parameter vector, and $\epsilon_i \sim \mathrm{N}(0, \sigma^2)$. If we consider Bayesian linear regression in this model, *i.e.* we place a prior over the weights $\mathrm{N}(\mathbf{u}; \mathbf{0}, \mathbf{K}_{\mathbf{zz}})$, then

$$p(\mathbf{u} \,|\, \mathbf{y}) \propto \mathrm{N}(\mathbf{0}, \mathbf{K}_{\mathbf{zz}}) \textstyle\prod_{i=1}^{n} e^{-\frac{1}{2\sigma^2}(y_i - \boldsymbol{a}_i^\top \mathbf{u})^2}, \tag{28}$$

highlighting the similarity to Eq. (12). The ridge leverage score for the i^{th} example is defined as

$$h_i^{\text{RLS}} = \boldsymbol{a}_i^\top \left(\frac{1}{\sigma^2} \mathbf{A}\mathbf{A}^\top + \mathbf{K}_{\mathbf{zz}})^{-1} \right)^{-1} \boldsymbol{a}_i, \tag{29}$$

where $\mathbf{A} = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n)$. Examples with high leverage scores are often far away from other observations and have high predictive uncertainty.

Comparing the above linear model to Eq. (12), we see that the SVGP posterior is equivalent to the posterior of a linear model with a major difference: the noise is now heteroscedastic with variance $1/\hat{\beta}_i^{\text{new}}$ for the i^{th} example. This motivates the following Bayesian generalisation of RLS, which we refer to as the Bayesian leverage score (BLS),

$$h_i^{\text{BLS}} = \boldsymbol{a}_i^\top \left(\mathbf{A}\, \text{diag}(\hat{\boldsymbol{\beta}}^{\text{new}})\, \mathbf{A}^\top + \mathbf{K}_{\mathbf{zz}}^{-1} \right)^{-1} \boldsymbol{a}_i. \tag{30}$$

We do not have to invert $\mathbf{K}_{\mathbf{zz}}$ explicitly. The score can be computed without any additional cost by rewriting it in terms of predictive variance $\hat{v}_i$ of $q_{\mathbf{u}}(f_i)$, that is,

$$h_i^{\text{BLS}} = \hat{\beta}_i^{\text{new}} \hat{v}_i^{\text{new}}. \tag{31}$$

For Gaussian likelihoods, BLS in Eq. (31) reduces to RLS because $\hat{\beta}_i = 1/\sigma^2$, but unlike RLS, the BLS also applies to non-Gaussian likelihoods. When selecting the memory, we want representative examples. We achieve this by sampling memory points from the new batch weighted by the BLS score. This biases our memory towards more difficult examples whilst covering the typical set, which we find improves performance. The final algorithm is given in Alg. 1.

4. Experiments

We perform a range of experiments to show the capability of the proposed method on various sequential learning problems. For sequential decision-making tasks, we apply our method to Bayesian Optimization (BO) and Active Learning (AL) (Sec. 4.1). Building on Chang et al. (2022), we show how *'fantasy'* batch acquisition functions can be built for simple acquisition functions and demonstrate the effectiveness on the 'lunar landing' BO problem and the 'hotspot modelling' AL problem.

In streaming tasks, data comes in small batches, where the total number of data points is unknown, and all SVGP parameters should be optimized. This limits the use of the methods by Maddox et al. (2021) and Kapoor et al. (2021). Therefore, we compare the proposed method against Bui et al. (2017) on streaming UCI tasks and on a real-world robot task (Sec. 4.2). Finally, we consider the *split-MNIST* continual learning problem, where the data is non-stationary, but the total number of tasks and data points are known beforehand. Thus, in Sec. 4.3 we compare the proposed method to Kapoor et al. (2021) and Bui et al. (2017). Additionally, we include a study on the benefits of our BLS score over a random selection of memory.

4.1. Bayesian Optimization and Active Learning

The *lunar lander problem* is a challenging optimization problem that aims to successfully land a vehicle inside a specific region of a lunar surface (Moss et al., 2020). Here, every action performed by the lander results in a reward. The aim is to optimize the total reward. Various environmental components add to the model's stochasticity, making it a challenging problem. The setup with the sources of stochasticity and the multiple states of the lander is shown in Fig. 3. The search space spanning over $\mathbb{R}^{12}$ is high-

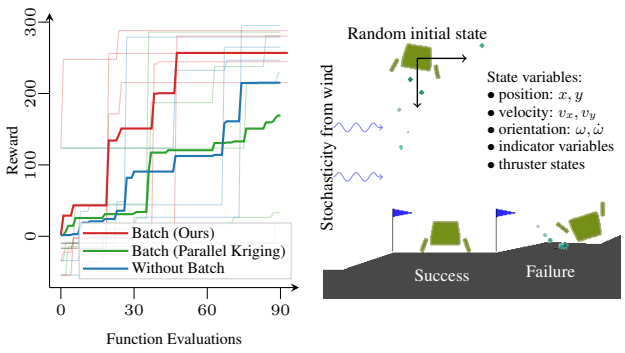


Figure 3. Bayesian optimization on a lunar landing setup. The goal is to successfully land on the surface between the flags. Compared against the non-batch solution and a batch solution using parallel Kriging (Ginsbourger et al., 2010), the proposed approach achieves higher reward in less function evaluations.

dimensional, making the task challenging. However, building a batch based on fantasy points can help overcome the difficulty in BO. These fantasy points are simulated points $(\mathbf{x}_i, \mathbf{y}_i)$ obtained from the acquisition function $\gamma(\cdot)$ by $\mathbf{x}_i = \arg\max_{\mathbf{x}} \gamma(\mathbf{x})$ and sampling the corresponding $\mathbf{y}_i$ from the current model. We then condition the fantasized point into the posterior using our dual updates, and repeat the procedure for subsequent query points, thereby constructing a batch of query points (see Alg. 2 in App. A.5).

We model the problem using a regression model that aims to maximize the reward and a classification model that models whether the landing was successful. The acquisition function is a product of the Expected Improvement of the regression model and the predictive mean of the classification model. In both the models, we make a batch of fantasy points. For details see App. B.4.

The presence of the non-Gaussian likelihood in the classification model prevents the use of many advanced batch acquisition functions which exploit properties unique to Gaussian likelihoods, such as q-KG (Wu & Frazier, 2016). Our method is agnostic to the likelihood and also allows batching of fantasy points in this challenging setup. We run two baselines: a non-batch version of the proposed method and a standard batch BO method proposed by Ginsbourger et al. (2010). Each method gets the same set of 24 initial data points and optimizes over 90 function evaluations. Both batch methods build batches of three query points. We run the experiment with five random initial observations and visualize individual and average reward curves over the BO iterations in Fig. 3. Our method improves over the non-batch setting and over the batch method of Ginsbourger et al. (2010), showing the usefulness of our batch method and robustness to the non-Gaussian likelihood.

We also consider the *schistosomiasis hotspot modelling* active learning problem from Maddox et al. (2021) (originally based on Andrade-Pacheco et al., 2020). In the same SVGP model and setup (we use Maddox et al.'s code, though we turn off their tempering, which improves all methods), our method for updating the posterior leads to clear improvement of MSE and slightly improved accuracy (Fig. 4) whilst

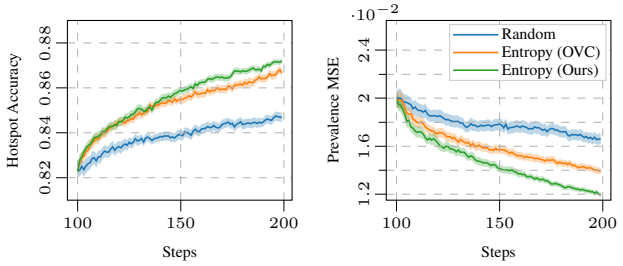


Figure 4. Schistosomiasis hotspot modelling experiment of Maddox et al. (2021)'s online variational conditioning method (OVC). We report mean ± standard error over 50 seeds, showing better performance of the proposed method.

Table 1. UCI data sets: negative log predictive density, mean (standard deviation) over 10-fold cross-validation, lower is better.

Data set	Offline	Ours	Bui et al.
ELEVATORS	.50 (.01)	.57 (.02)	.57 (.02)
BIKE	.24 (.03)	.44 (.03)	.49 (.03)
MAMMOGRAPHIC	.40 (.04)	.41 (.05)	.42 (.05)
BANK	.25 (.03)	.26 (.03)	.28 (.04)
MUSHROOM	.00 (.00)	.02 (.00)	.08 (.01)
ADULT	.32 (.00)	.35 (.01)	.36 (.00)

being faster. On the same GPU, Maddox et al.'s online variational conditioning (OVC) method took on average 85 s per step (standard deviation 11 s); ours took on average 62 s (standard deviation 7 s).

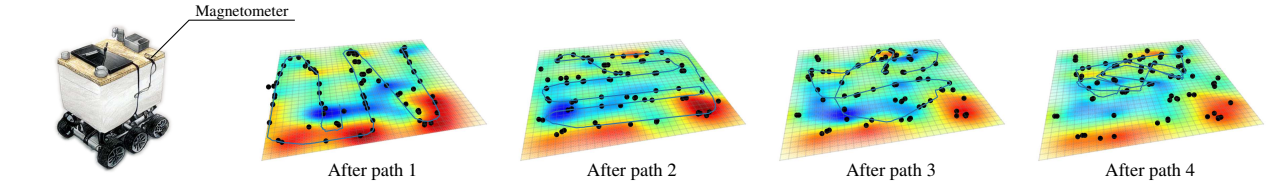
4.2. Streaming Tasks

Under streaming scenarios, where data comes in small batches and the total number of data points is unknown, we include two different experiment setups: UCI tasks and the so-called banana data set, as well as a real-world robot data set for mapping magnetic anomalies in an indoor space.

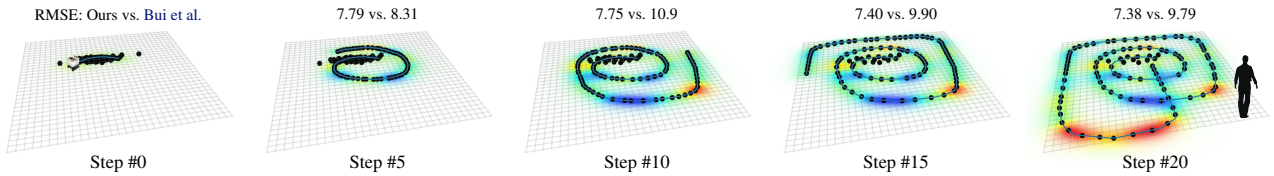
UCI and Banana We consider a setup where the *banana* data set and UCI (Dua & Graff, 2017) data sets are converted into streaming setups by splitting them into sets, $\mathcal{D}=\{\mathcal{D}_1, \mathcal{D}_2, \ldots\}$. The model at each step has access only to the current set $\mathcal{D}_k$. For UCI data sets, we sort with respect to the the first input dimension. Fig. 1 (top) shows the posterior obtained by the proposed method on the banana data set. Details about the setup and comparison with other methods can be found in App. B.1. In UCI data sets, we experiment with both regression and classification setups. The offline model has access to the whole data and can take multiple passes. Therefore, the offline model is used as a gold-standard baseline and we compare our method with Bui et al. (2017). Table 1 shows the mean test negative log predictive density (NLPD) over 10-fold cross-validation, where $\mathbf{Z}, \boldsymbol{\theta}, \boldsymbol{\alpha}_{\mathbf{u}}$, and $\boldsymbol{B}_{\mathbf{u}}$ are optimized. Other evaluation metrics, setup, and implementation details are available in App. B.5. In the streaming setting, the proposed method performs comparable to Bui et al. (2017).

Sequential Learning of Magnetic Anomalies We consider the task of online mapping of local anomalies in the ambient magnetic field. We follow the experiment setup and data provided in Solin et al. (2018), where a small wheeled robot equipped with a 3-axis magnetometer moves around in a $6\,\mathrm{m} \times 6\,\mathrm{m}$ indoor space. We assign a GP prior $\mathcal{GP}(0, \sigma_0^2 + \kappa^{\mathrm{Mat.}}_{\sigma^2,\ell}(\mathbf{x}, \mathbf{x}'))$ to the magnetic field strength over the space under the presence of Gaussian measurement noise with variance σ_{n}^2.

We include two experiments: In Fig. 5a, we simultaneously learn the hyperparameters $(\sigma_0^2, \sigma^2, \ell, \sigma_{\mathrm{n}}^2)$ and a representa-



(a) Experiment #1: Sequential estimation of magnetic field anomalies in a fixed-size domain with multiple observation paths



(b) Experiment #2: Maintaining representation in a sequential setting in an ever-expanding domain

Figure 5. Sequential estimation of anomalies by the proposed method in the local ambient magnetic field strength (10 µT 90 µT) as mapped by a small wheeled robot. Two experiments for learning the hyperparameters and representation: (a) Complete trajectories are received and the model does not have access to the previous trajectories. (b) Data is received during the exploration of the space requiring the method to spread out a fixed number of inducing points. Each step shows RMSE values comparing our method to Bui et al. (2017).

tion in terms of inducing points and memory. Our method is able to form a representation by spreading inducing points and learning the hyperparameters progressively. This has practical importance in real-world robot estimation tasks, where the robot is not constrained to a predefined area (a weakness in Solin et al., 2018). In Fig. 5b, we now receive data continuously during exploration. The visualization shows the mean estimate with marginal variance (uncertainty) controlling the opacity. We recover the same local estimate as in experiment Fig. 5a and outperform the baseline given by Bui et al. (2017). See details in App. B.3.

4.3. Continual Learning on Split MNIST

Split MNIST (Zenke et al., 2017) is a continual learning benchmark and a variant of MNIST where training data comes in five batches of two digits each, *i.e.* 0 and 1, 2 and 3, . . . , 8 and 9. Performance is measured by multi-class classification accuracy on all digits seen thus far. The model at each step has access only to the current batch of the classification task and thus should learn incrementally on different tasks without forgetting previous ones. Fig. 6 compares the accuracy during the training on each task, subdivided into batches. Our sequential model does well at remembering previous tasks (only marginal drops in accuracy in each column of Table 2), while the model of Bui et al. (2017) forgets the previous tasks. We also slightly outperform Kapoor et al. (2021), for whom compute scales cubically with the number of tasks and who need an additional hyperparameter regularization term. Therefore, we distinguish it from other baselines which do not suffer from the same complexity with the number of tasks (Table 3).

We include an experiment to show that our BLS is a useful metric to characterise and determine the difficulty of input points. We consider continual learning on *split MNIST* as before, but now we select data points from our training set and move them to the test set. Points are chosen either

Table 2. Test accuracy (and standard deviation over different random seeds) on *split MNIST* over all tasks thus far for the proposed method. High accuracy over all previous tasks shows that the method does not suffer from forgetting.

Task	0, 1	2, 3	4, 5	6, 7	8, 9
#1	.98(.005)				
#2	.68(.029)	.95(.003)			
#3	.88(.005)	.87(.004)	.98(.002)		
#4	.86(.006)	.89(.004)	.94(.007)	.97(.004)	
#5	.95(.014)	.87(.011)	.90(.011)	.93(.005)	.90(.014)

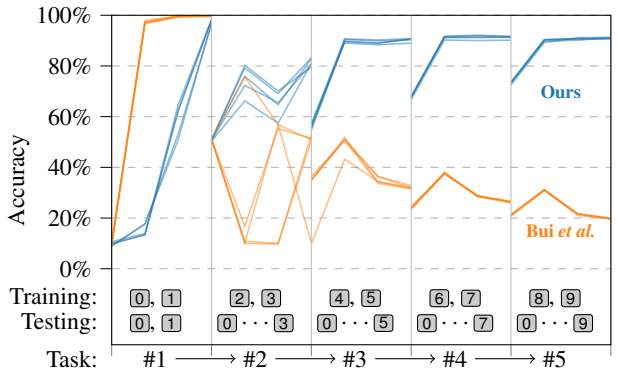


Figure 6. Accuracy on *split MNIST* over training with different random seeds. The training starts with 0 vs. 1 and each task introduces new digits while testing on all classes thus far. The overall accuracy (mean over tasks) drops when introducing a new task, but recovers and does not suffer from forgetting over tasks.

Table 3. Test final accuracy and NLPD (standard deviation over different random seeds) of various methods on the *split MNIST* task. We include Kapoor et al. (2021) as a baseline.

Method	Accuracy	NLPD
SVGP (Baseline)	.962(.001)	0.155(0.002)
Bui et al. (2017)	.200(.001)	2.150(0.019)
Ours (without memory)	.208(.007)	68.038(2.830)
Ours (with memory)	.909(.001)	0.316(0.010)
Kapoor et al. (2021)	.905(.010)	0.324(0.018)

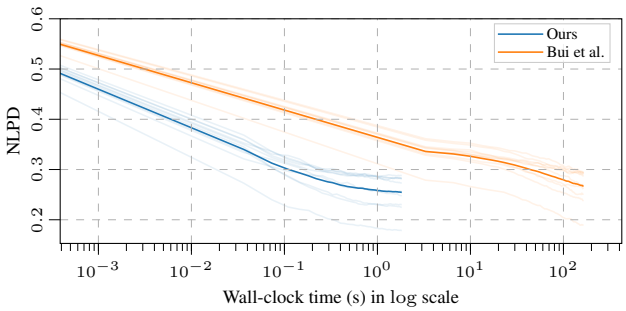


Figure 7. NLPD against wall-clock time of the proposed method and Bui et al. (2017) on 'adult' UCI data set when only variational parameters are optimized. At the same performance, the proposed method is significantly faster than Bui et al. (2017).

randomly or based on the BLS score. We then retrain the model on the reduced training set and test on the increased test set. Figure 2a shows the performance of the selection methods. Random selection of the points has a small negative effect on performance. However, using the BLS score is detrimental, showing the importance of the examples for the model that are moved to the test set. Figure 2b shows digits with the highest BLS score. We also study the effect of the size of the memory, and show how it affects the model accuracy. We train our sequential model with different memory sizes using BLS and report test accuracy (see Fig. 2c). As expected, the accuracy increases with memory size, but remarkably a memory size of just 5% achieves satisfactory performance. Experiment details can be found in App. B.2 and further evaluation of the BLS score on UCI in Table 7.

4.4. Timing Experiment

Finally, to show the benefit of natural gradient optimization for variational parameters, we perform a timing experiment against Bui et al. (2017) in Fig. 7 in a streaming setting on the 'adult' UCI data set. We see speed improvements in inference wall-clock timings with the same performance (note the x-axis is in log-scale).

5. Discussion and Conclusions

The difficulty in building a sequential sparse GP model is due to the lack of access to previous data. Approaches that do not consider the dual parameter perspective fail to see the essence of the inference problem; how to accurately infer the dual parameters in a sequential manner. The problem formulation was presented in Csató & Opper (2002), but there an EP inference scheme was used. This paper shows how to update parameters sequentially using variational approximate inference.

Furthermore, we use natural gradient updates which come easily in the dual parameter formulation allowing for a method (*i.e.*, few inference step iterations) that works with general likelihoods. Lack of access to past data also makes it challenging to learn hyperparameters $\boldsymbol{\theta}$. The problem worsens in the non-stationary continual learning setting. Here, to counter the problem of forgetting previous data, we introduce the concept of memory, a technique shown to have success in deep learning (Pan et al., 2020). We find that a small amount of memory can dramatically improve performance by replicating the offline ELBO solution.

Our memory approach is novel and different from previous work, which attempts to replace missing data with additional regularization terms in the ELBO (Bui et al., 2017; Kapoor et al., 2021). Our final method manages to control the error that come from inference, learning and representation (memory and inducing points). Furthermore, our method can be applied to a variety of sequential learning tasks. Given the importance of memory to our method, a good selection technique is paramount. We derive a new Bayesian leverage score, given its connection to Bayesian approximate inference. The score generalizes the prevalent RLS score used for kernel sampling methods (Alaoui & Mahoney, 2015). We find the importance of the dual parameters again, as they are a crucial component of the BLS. We demonstrate the applicability of our method to a complex combined batch Bayesian optimization and active learning problem, and continual learning problem.

A reference implementation of the methods presented in this paper is available at: `https://github.com/AaltoML/sequential-gp`.

Acknowledgements

This work was supported by funding from JST CREST (Grant ID JPMJ CR2112), the Academy of Finland (grant id 339730 and 324345), and the Finnish Center for Artificial Intelligence (FCAI). We acknowledge the computational resources provided by the Aalto Science-IT project and CSC – IT Center for Science, Finland. We thank Rui Li, Riccardo Mereu, Henry Moss, Thomas Möllenhof, Gianma Marconi, and Victor Picheny for helpful discussions. We also acknowldge the input from Dharmesh Tailor, Siddharth Swaroop, and Eric Nalisnick related the BLS score derivation.

References

Adam, V., Chang, P., Khan, M. E., and Solin, A. Dual parameterization of sparse variational Gaussian processes. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, pp. 11474–11486. Curran Associates, Inc., 2021.

Alaoui, A. and Mahoney, M. W. Fast randomized kernel ridge regression with statistical guarantees. In *Advances in Neural Information Processing Systems 28 (NIPS)*, pp. 775–783. Curran Associates, Inc., 2015.

Andrade-Pacheco, R., Rerolle, F., Lemoine, J., Hernandez, L., Meïté, A., Juziwelo, L., Bibaut, A. F., van der Laan, M. J., Arnold, B. F., and Sturrock, H. J. W. Finding hotspots: development of an adaptive spatial sampling approach. *Scientific Reports*, 10(1), 2020.

Bui, T. D., Nguyen, C., and Turner, R. E. Streaming sparse Gaussian process approximations. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pp. 3299–3307. Curran Associates, Inc., 2017.

Burt, D. R., Rasmussen, C. E., and van der Wilk, M. Convergence of sparse variational inference in Gaussian processes regression. *Journal of Machine Learning Research*, 21:1–63, 2020.

Chang, P. E., Verma, P., John, S. T., Picheny, V., Moss, H., and Solin, A. Fantasizing with dual GPs in Bayesian optimization and active learning. In *NeurIPS Workshop on Gaussian Processes, Spatiotemporal Modeling, and Decision-making Systems*, 2022.

Cook, R. D. and Weisberg, S. Characterizations of an empirical influence function for detecting influential cases in regression. *Technometrics*, 22(4):495–508, 1980.

Cortes, C. and Vapnik, V. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

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

Deisenroth, M. P. and Rasmussen, C. E. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pp. 465–472. Omnipress, 2011.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Garnett, R. *Bayesian Optimization*. Cambridge University Press, 2023. Cambridge, UK.

Ginsbourger, D., Le Riche, R., and Carraro, L. *Kriging Is Well-Suited to Parallelize Optimization*, pp. 131–162. Springer Berlin Heidelberg, 2010.

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

Kapoor, S., Karaletsos, T., and Bui, T. D. Variational auto-regressive Gaussian processes for continual learning. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pp. 5290–5300. PMLR, 2021.

Khan, M. E. and Rue, H. The bayesian learning rule. *arXiv preprint arXiv:2107.04562*, 2021.

Khan, M. E. and Swaroop, S. Knowledge-adaptation priors. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, pp. 19757–19770. Curran Associates, Inc., 2021.

Kimeldorf, G. and Wahba, G. Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33(1):82–95, 1971.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Li, R., John, S. T., and Solin, A. Improving hyperparameter learning under approximate inference in Gaussian process models. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*. PMLR, 2023.

Maddox, W. J., Stanton, S., and Wilson, A. G. Conditioning sparse variational Gaussian processes for online decision-making. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, pp. 6365–6379. Curran Associates, Inc., 2021.

Moss, H. B., Leslie, D. S., and Rayson, P. BOSH: Bayesian optimization by sampling hierarchically. In *The International Conference on Machine Learning: Workshop on Real World Experimental Design and Active Learning*, 2020.

Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. Variational continual learning. In *International Conference on Learning Representations (ICLR)*, 2018.

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

Pan, P., Swaroop, S., Immer, A., Eschenhagen, R., Turner, R., and Khan, M. E. Continual deep learning by functional regularisation of memorable past. In *Advances in*

Neural Information Processing Systems 33 (NeurIPS), pp. 4453–4464. Curran Associates, Inc., 2020.

Solin, A., Kok, M., Wahlström, N., Schön, T. B., and Särkkä, S. Modeling and interpolation of the ambient magnetic field by Gaussian processes. *IEEE Transactions on Robotics*, 34(4):1112–1127, 2018.

Titsias, M. K. 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*, pp. 567–574. PMLR, 2009.

Titsias, M. K., Schwarz, J., Matthews, A. G. d. G., Pascanu, R., and Teh, Y. W. Functional regularisation for continual learning with Gaussian processes. In *International Conference on Learning Representations (ICLR)*, 2020.

Wu, J. and Frazier, P. The parallel knowledge gradient method for batch Bayesian optimization. In *Advances in Neural Information Processing Systems 29 (NIPS)*, volume 29, pp. 3126–3134. Curran Associates, Inc., 2016.

Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research, pp. 3987–3995. PMLR, 2017.

Appendix

We include technical details of the methods that were omitted for brevity in the main paper. Additionally, we provide details on the experiments and evaluation setup for reproducing the results in the main paper, and include further results, figures, and tables that extend the evaluation.

A. Method Details

A.1. Equivalence to Csató and Opper

The result from Csató & Opper (2002) (*cf.* Lemma 1) assumes a Gaussian prior $p(\mathbf{f})$ with a mean function $m(\mathbf{x})$ and covariance $\kappa(\mathbf{x}, \mathbf{x}')$, with likelihood $p(\mathcal{D} \mid \mathbf{f})$ where $\mathbf{f} = [f_1, f_2, \ldots, f_n]$ is a vector of function values $f_i = f(\mathbf{x}_i)$. For this case, they use the following two parameterization,

$$\begin{aligned} q_i &= \int \nabla_{f_i} p(\mathcal{D} \mid \mathbf{f}) \frac{p(\mathbf{f})}{\int p(\mathcal{D} \mid \mathbf{f}) p(\mathbf{f}) \, \mathrm{d}\mathbf{f}} \, \mathrm{d}\mathbf{f}, \\ R_{i,j} &= \int \nabla^2_{f_i, f_j} p(\mathcal{D} \mid \mathbf{f}) \frac{p(\mathbf{f})}{\int p(\mathcal{D} \mid \mathbf{f}) p(\mathbf{f}) \, \mathrm{d}\mathbf{f}} \, \mathrm{d}\mathbf{f} - q_i q_j. \end{aligned} \tag{32}$$

For the case when $p(\mathcal{D} \mid \mathbf{f}) = \prod_{i=1}^{N} p(y_i \mid f_i)$, we can show the following,

$$q_i = \alpha_i \text{ and } R_{i,j} = \begin{cases} \beta_i, & \text{when } i = j \\ 0, & \text{when } i \neq j \end{cases},$$

where α_i and β_i are as defined in Eq. (2). We first show the proof for q_i,

$$\begin{aligned} q_i &= \int \nabla_{f_i} p(\mathcal{D} \mid \mathbf{f}) \frac{1}{p(\mathcal{D} \mid \mathbf{f})} \frac{p(\mathcal{D} \mid \mathbf{f}) p(\mathbf{f})}{\int p(\mathcal{D} \mid \mathbf{f}) p(\mathbf{f}) \, \mathrm{d}\mathbf{f}} \, \mathrm{d}\mathbf{f} \\ &= \int \nabla_{f_i} \log p(\mathcal{D} \mid \mathbf{f}) p(\mathbf{f} \mid \mathcal{D}) \, \mathrm{d}\mathbf{f} \\ &= \sum_{j=1}^{N} \int \nabla_{f_i} \log p(y_j \mid f_j) p(\mathbf{f} \mid \mathcal{D}) \, \mathrm{d}\mathbf{f} \\ &= \int \nabla_{f_i} \log p(y_i \mid f_i) p(\mathbf{f} \mid \mathcal{D}) \, \mathrm{d}\mathbf{f} \\ &= \int \nabla_{f_i} \log p(y_i \mid f_i) p(f_i \mid \mathcal{D}) \, \mathrm{d}f_i \\ &= \mathbb{E}_{p(f_i \mid \mathbf{y}_n)} [\nabla_{f_i} \log p(y_i \mid f_i)]. \end{aligned} \tag{33}$$

Here, the first line is obtained by simply multiplying and dividing by $p(\mathcal{D} \mid \mathbf{f})$, and we get the second line by using the definition of the posterior $p(\mathbf{f} \mid \mathcal{D})$ and the fact that $\nabla \log p(\mathcal{D} \mid \mathbf{f}) = [\nabla p(\mathcal{D} \mid \mathbf{f})] / p(\mathcal{D} \mid \mathbf{f})$. The third line follows from the assumption that the likelihood factorizes over data examples, and the fourth line obtained by noting that $\nabla_{f_i} \log p(y_j \mid f_j)$ is non-zero only when $i = j$. The fifth line follows by marginalizing out all f_j other than f_i, and the final line is just a different way to write the expectation.

For $R_{i,j}$, we proceed in a similar fashion. We will use the following identity to write $\nabla_{f_i, f_j} p(\mathcal{D} \mid \mathbf{f})$ in terms $\nabla_{f_i} \log p(\mathcal{D} \mid \mathbf{f})$:

$$\frac{1}{p(\mathcal{D} \mid \mathbf{f})} \nabla^2_{f_i f_j} p(\mathcal{D} \mid \mathbf{f}) = \nabla^2_{f_i f_j} \log p(\mathcal{D} \mid \mathbf{f}) + \left[\nabla_{f_i} \log p(\mathcal{D} \mid \mathbf{f})\right] \left[\nabla_{f_j} \log p(\mathcal{D} \mid \mathbf{f})\right]. \tag{34}$$

This can be proved by rearranging the derivative of $\nabla^2_{f_i f_j} \log p(\mathcal{D} \mid \mathbf{f})$ by using the fact that $\nabla \log p(\mathcal{D} \mid \mathbf{f}) =$

$[\nabla p(\mathcal{D} \mid \mathbf{f})]/p(\mathcal{D} \mid \mathbf{f})$ (which we also used in the derivation of q_i above). Using this we can simplify $R_{i,j}$,

$$\begin{aligned}
R_{i,j} &= \int \nabla^2_{f_i,f_j} p(\mathcal{D} \mid \mathbf{f}) \frac{1}{p(\mathcal{D} \mid \mathbf{f})} \frac{p(\mathcal{D} \mid \mathbf{f}) p(\mathbf{f})}{\int p(\mathcal{D} \mid \mathbf{f}) p(\mathbf{f}) \, \mathrm{d}\mathbf{f}} \, \mathrm{d}\mathbf{f} - q_i q_j \\
&= \int \left[\nabla^2_{f_i f_j} \log p(\mathcal{D} \mid \mathbf{f}) + [\nabla_{f_i} \log p(\mathcal{D} \mid \mathbf{f})] \left[\nabla_{f_j} \log p(\mathcal{D} \mid \mathbf{f}) \right] \right] p(\mathbf{f} \mid \mathcal{D}) \, \mathrm{d}\mathbf{f} - q_i q_j \\
&= \int \left[\nabla^2_{f_i f_j} \log p(\mathcal{D} \mid \mathbf{f}) \right] p(\mathbf{f} \mid \mathcal{D}) \, \mathrm{d}\mathbf{f} + \int [\nabla_{f_i} \log p(\mathcal{D} \mid \mathbf{f})] \left[\nabla_{f_j} \log p(\mathcal{D} \mid \mathbf{f}) \right] p(\mathbf{f} \mid \mathcal{D}) \, \mathrm{d}\mathbf{f} - q_i q_j \\
&= \sum_{k=1}^{n} \int \left[\nabla^2_{f_i f_j} \log p(y_k \mid f_k) \right] p(\mathbf{f} \mid \mathcal{D}) \, \mathrm{d}\mathbf{f} + \sum_{l,m} \int [\nabla_{f_i} \log p(y_l \mid f_l)] \left[\nabla_{f_j} \log p(y_m \mid f_m) \right] p(\mathbf{f} \mid \mathcal{D}) \, \mathrm{d}\mathbf{f} - q_i q_j,
\end{aligned} \tag{35}$$

where in the first line we multiply and divide by $p(\mathcal{D} \mid \mathbf{f})$ and use Eq. (34) and definition of the posterior to get to the second line. The third line is a rearrangement, and the last line is obtained by using the factorial property of the likelihood.

For the second term we get it to be non-zero when $i = l$ and $j = m$, and then integrating over all function values, except f_i and f_j, the term reduces to $q_i q_j$, which can be cancelled out. This gives us the following expression,

$$\begin{aligned}
R_{i,j} &= \sum_{k=1}^{n} \int \left[\nabla^2_{f_i f_j} \log p(y_k \mid f_k) \right] p(\mathbf{f} \mid \mathcal{D}) \, \mathrm{d}\mathbf{f} \\
&= \begin{cases} \mathbb{E}_{p(f_i \mid \mathbf{y}_n)}[\nabla^2_{f_i f_j} \log p(y_i \mid f_i)], & \text{when } i = j, \\ 0, & \text{when } i \neq j. \end{cases}
\end{aligned} \tag{36}$$

This is because the derivative inside is only nonzero when $i = j = k$, and we can express the integral only over f_i. This proves the required result.

A.2. Derivation of the Dual-form of the SVGP Stationary Point

We will derive the dual-form of the stationary point of the following SVGP objective:

$$\mathcal{L}(q) = \sum_{i \in \mathcal{D}} \mathbb{E}_{q_{\mathbf{u}}(f_i)}[\log p(y_i \mid f_i)] - \mathbb{D}_{\mathrm{KL}}[q_{\mathbf{u}}(\mathbf{u}) \,\|\, p_{\boldsymbol{\theta}}(\mathbf{u})]. \tag{37}$$

Derivation simplifies by using natural and expectation parameters of $q_{\mathbf{u}}(\mathbf{u}) = \mathrm{N}(\mathbf{u}|\mathbf{m}, \mathbf{V})$, denoted by $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ respectively,

$$\boldsymbol{\lambda} = (\boldsymbol{\lambda}^{(1)}, \boldsymbol{\lambda}^{(2)}) = (\boldsymbol{V}^{-1}\mathbf{m}, \ -\frac{1}{2}\boldsymbol{V}^{-1}), \qquad \boldsymbol{\mu} = (\boldsymbol{\mu}^{(1)}, \boldsymbol{\mu}^{(2)}) = (\mathbf{m}, \ \mathbf{m}\mathbf{m}^\top + \boldsymbol{V}),$$

Taking the gradient with respect to $\boldsymbol{\mu}$ of Eq. (37) at a stationary point $q^*_{\boldsymbol{\mu}}(\mathbf{u})$, we get

$$\begin{aligned}
\mathbf{0} &= \sum_{i \in \mathcal{D}} \nabla_{\boldsymbol{\mu}} \mathbb{E}_{q^*_{\mathbf{u}}(f_i)}[\log p(y_i \mid f_i)] - \nabla_{\boldsymbol{\mu}} \mathbb{D}_{\mathrm{KL}}[q^*_{\mathbf{u}}(\mathbf{u}) \,\|\, p_{\boldsymbol{\theta}}(\mathbf{u})], \\
&= \sum_{i \in \mathcal{D}} \nabla_{\boldsymbol{\mu}} \mathbb{E}_{q^*_{\mathbf{u}}(f_i)}[\log p(y_i \mid f_i)] - (\boldsymbol{\lambda}_* - \boldsymbol{\lambda}_{\text{prior}}), \\
&\implies \boldsymbol{\lambda}_* = \boldsymbol{\lambda}_{\text{prior}} + \sum_{i \in \mathcal{D}} \nabla_{\boldsymbol{\mu}} \mathbb{E}_{q^*_{\mathbf{u}}(f_i)}[\log p(y_i \mid f_i)],
\end{aligned} \tag{38}$$

where $\boldsymbol{\lambda}_{\text{prior}} = (0, -\mathbf{K}^{-1}_{\mathbf{zz}}/2)$ is the natural parameter of the prior $p_{\boldsymbol{\theta}}(\mathbf{u})$.

We can expand this equation and write two equations corresponding to the two natural parameters $(\mathbf{V}_*^{-1}\mathbf{m}_*, \ -\mathbf{V}_*^{-1}/2)$. We get the following for the first natural parameter,

$$\begin{aligned}
\mathbf{V}_*^{-1}\mathbf{m}_* &= \sum_{i \in \mathcal{D}} \nabla_{\boldsymbol{\mu}^{(1)}} \mathbb{E}_{q^*_{\mathbf{u}}(f_i)}[\log p(y_i \mid f_i)] \\
&= \sum_{i \in \mathcal{D}} \boldsymbol{a}_i \mathbb{E}_{q^*_{\mathbf{u}}(f_i)}[\nabla \log p(y_i \mid f_i)] - \sum_{i \in \mathcal{D}} \boldsymbol{a}_i \mathbb{E}_{q^*_{\mathbf{u}}(f_i)}[\nabla^2 \log p(y_i \mid f_i)] \boldsymbol{a}_i^\top \mathbf{m}_* \\
&= \sum_{i \in \mathcal{D}} \boldsymbol{a}_i (\hat{\alpha}^*_i + \hat{\beta}^*_i \boldsymbol{a}_i^\top \mathbf{m}_*)
\end{aligned} \tag{39}$$

In the second line we used the identity given in Khan & Rue (2021, Eq. 10) which uses chain-rule and Bonnet's result. This gives us Eq. (7). This also matches with Eq. 21 of Adam et al. (2021) who define $\lambda_{1,i}^* = \hat{\alpha}_i^* + \hat{\beta}_i^* \boldsymbol{a}_i^\top \mathbf{m}_*$.

Similarly for the second natural parameter,

$$\begin{aligned}
-\frac{1}{2}\mathbf{V}_*^{-1} &= -\frac{1}{2}\mathbf{K}_{\mathbf{zz}}^{-1} + \sum_{i\in\mathcal{D}} \nabla_{\boldsymbol{\mu}^{(2)}} \mathbb{E}_{q_{\mathbf{u}}^*(f_i)}[\log p(y_i \mid f_i)] \\
&= -\frac{1}{2}\mathbf{K}_{\mathbf{zz}}^{-1} + \frac{1}{2}\sum_{i\in\mathcal{D}} \boldsymbol{a}_i \mathbb{E}_{q_{\mathbf{u}}^*(f_i)}[\nabla^2 \log p(y_i \mid f_i)] \boldsymbol{a}_i^\top \\
&= -\frac{1}{2}\mathbf{K}_{\mathbf{zz}}^{-1} - \frac{1}{2}\sum_{i\in\mathcal{D}} \boldsymbol{a}_i \hat{\beta}_i^* \boldsymbol{a}_i^\top
\end{aligned} \tag{40}$$

Here again, in the second line we used the identity given in Khan & Rue (2021, Eq. 11) which uses chain-rule and Price's result. This gives us Eq. (8). This too matches with Eq. 21 of Adam et al. (2021).

We will now derive the dual-form similar to Eq. (1). Rearranging and substituting Eq. (40) in to Eq. (39), we get,

$$\begin{aligned}
[\mathbf{V}_*^{-1} - \boldsymbol{a}_i \hat{\beta}_i^* \boldsymbol{a}_i^\top]\mathbf{m}_* = \textstyle\sum_{i\in\mathcal{D}} \boldsymbol{a}_i \hat{\alpha}_i^* \quad &\implies \mathbf{K}_{\mathbf{zz}}^{-1}\mathbf{m}_* = \mathbf{K}_{\mathbf{zz}}^{-1} \underbrace{\textstyle\sum_{i\in\mathcal{D}} \mathbf{k}_{\mathbf{z}i} \hat{\alpha}_i^*}_{=\boldsymbol{\alpha}_{\mathbf{u}}} \\
\mathbf{V}_*^{-1} = \mathbf{K}_{\mathbf{zz}}^{-1} + \mathbf{K}_{\mathbf{zz}}^{-1}(\textstyle\sum_{i\in\mathcal{D}} \mathbf{k}_{\mathbf{z}i} \hat{\beta}_i^* \mathbf{k}_{\mathbf{z}i}^\top)\mathbf{K}_{\mathbf{zz}}^{-1} \quad &\implies \mathbf{V}_*^{-1} = \mathbf{K}_{\mathbf{zz}}^{-1} + \mathbf{K}_{\mathbf{zz}}^{-1} \underbrace{\textstyle\sum_{i\in\mathcal{D}} \mathbf{k}_{\mathbf{z}i} \hat{\beta}_i \mathbf{k}_{\mathbf{z}i}^\top}_{=\boldsymbol{B}_{\mathbf{u}}} \mathbf{K}_{\mathbf{zz}}^{-1}
\end{aligned} \tag{41}$$

Now substituting these results into Eq. (4), we get the results:

$$\begin{aligned}
\mathbb{E}_{q_{\mathbf{u}}^*(f)}[f_i] &= \mathbf{k}_{\mathbf{z}i}^\top \mathbf{K}_{\mathbf{zz}}^{-1} \mathbf{m}^* = \mathbf{k}_{\mathbf{z}i}^\top \mathbf{K}_{\mathbf{zz}}^{-1} \boldsymbol{\alpha}_{\mathbf{u}}, \\
\mathrm{Var}_{q_{\mathbf{u}}^*(f)}[f_i] &= \kappa_{ii} - \mathbf{k}_{\mathbf{z}i}^\top \left[\mathbf{K}_{\mathbf{zz}}^{-1} - \mathbf{K}_{\mathbf{zz}}^{-1} \mathbf{V}_* \mathbf{K}_{\mathbf{zz}}^{-1}\right] \mathbf{k}_{\mathbf{z}i} = \kappa_{ii} - \mathbf{k}_{\mathbf{z}i}^\top \left[\mathbf{K}_{\mathbf{zz}}^{-1} - (\mathbf{K}_{\mathbf{zz}} + \boldsymbol{B}_{\mathbf{u}}^*)^{-1}\right] \mathbf{k}_{\mathbf{z}i}
\end{aligned} \tag{42}$$

Thus proving that we can rewrite the posterior process in terms of the sparse dual parameters.

A.3. Derivation of the Pseudo-Data and New Prior

We first derive the distribution form used in Eq. (14) for the following,

$$\hat{p}_{\mathcal{D}}(\mathbf{u}) \propto \prod_{i\in\mathcal{D}} e^{-\frac{1}{2}\hat{\beta}_i^*(\hat{y}_i^* - \boldsymbol{a}_i^\top \mathbf{u})^2} = \mathrm{N}(\mathbf{u} \mid \tilde{\mathbf{y}}^*, \tilde{\boldsymbol{\Sigma}}^*), \tag{43}$$

where $(\hat{\beta}_i^*, \hat{y}_i^*)$ are obtained by training on the data $\mathcal{D}$. We start by writing the product in a matrix form,

$$\begin{aligned}
\log \hat{p}_{\mathcal{D}}(\mathbf{u}) &= -\frac{1}{2}\mathbf{u}^\top (\tilde{\boldsymbol{\Sigma}}^*)^{-1}\mathbf{u} + \mathbf{u}^\top (\tilde{\boldsymbol{\Sigma}}^*)^{-1}\tilde{\mathbf{y}} + \text{const.} \\
\log \prod_{i\in\mathcal{D}} e^{-\frac{1}{2}\hat{\beta}_i^*(\hat{y}_i^* - \boldsymbol{a}_i^\top \mathbf{u})^2} &= -\frac{1}{2}\mathbf{u}^\top \mathbf{A}^\top \mathrm{diag}(\hat{\boldsymbol{\beta}}^*)\mathbf{A}\mathbf{u} + \mathbf{u}^\top \mathbf{A}^\top \mathrm{diag}(\hat{\boldsymbol{\beta}}^*)\hat{\mathbf{y}}^* + \text{const.}
\end{aligned} \tag{44}$$

where $\mathbf{A} = \mathbf{K}_{\mathbf{xz}}\mathbf{K}_{\mathbf{zz}}^{-1}$, and $\hat{\mathbf{y}}^*$ and $\hat{\boldsymbol{\beta}}^*$ are vectors of all $\hat{y}_i^*$ and $\hat{\beta}_i^*$ respectively. Then, we simply match the terms in $\mathbf{u}$. First, by matching the quadratic term, we get,

$$(\tilde{\boldsymbol{\Sigma}}^*)^{-1} = \mathbf{A}^\top \mathrm{diag}(\hat{\boldsymbol{\beta}}^*)\mathbf{A} = \mathbf{K}_{\mathbf{zz}}^{-1}\mathbf{K}_{\mathbf{zx}}\mathrm{diag}(\hat{\boldsymbol{\beta}}^*)\mathbf{K}_{\mathbf{xz}}\mathbf{K}_{\mathbf{zz}}^{-1} = \mathbf{K}_{\mathbf{zz}}^{-1}\boldsymbol{B}_{\mathbf{u}}^*\mathbf{K}_{\mathbf{zz}}^{-1} \tag{45}$$

which gives us the third equation in Eq. (15). Next, by matching the linear term in $\mathbf{u}$, we get,

$$\begin{aligned}
\tilde{\mathbf{y}}^* &= \tilde{\boldsymbol{\Sigma}}^* \mathbf{A}^\top \mathrm{diag}(\hat{\boldsymbol{\beta}}^*)\hat{\mathbf{y}}^* \\
&= \mathbf{K}_{\mathbf{zz}}[\boldsymbol{B}_{\mathbf{u}}^*]^{-1}\mathbf{K}_{\mathbf{zz}}\mathbf{K}_{\mathbf{zz}}^{-1}\mathbf{K}_{\mathbf{zx}}\mathrm{diag}(\hat{\boldsymbol{\beta}}^*)\hat{\mathbf{y}}^* \\
&= \mathbf{K}_{\mathbf{zz}}[\boldsymbol{B}_{\mathbf{u}}^*]^{-1}\mathbf{K}_{\mathbf{zx}}\mathrm{diag}(\hat{\boldsymbol{\beta}}^*)[\mathrm{diag}(\hat{\boldsymbol{\beta}}^*)^{-1}\hat{\boldsymbol{\alpha}}^* + \mathbf{A}\mathbf{m}^*] \\
&= \mathbf{K}_{\mathbf{zz}}[\boldsymbol{B}_{\mathbf{u}}^*]^{-1}\mathbf{K}_{\mathbf{zx}}\hat{\boldsymbol{\alpha}}^* + \mathbf{K}_{\mathbf{zz}}[\boldsymbol{B}_{\mathbf{u}}^*]^{-1}\mathbf{K}_{\mathbf{zx}}\mathrm{diag}(\hat{\boldsymbol{\beta}}^*)\mathbf{K}_{\mathbf{xz}}\mathbf{K}_{\mathbf{zz}}^{-1}\mathbf{m}^* \\
&= \mathbf{K}_{\mathbf{zz}}[\boldsymbol{B}_{\mathbf{u}}^*]^{-1}\boldsymbol{\alpha}_{\mathbf{u}}^* + \mathbf{K}_{\mathbf{zz}}[\boldsymbol{B}_{\mathbf{u}}^*]^{-1}\boldsymbol{B}_{\mathbf{u}}^*\mathbf{K}_{\mathbf{zz}}^{-1}\mathbf{m}^* \\
&= \mathbf{K}_{\mathbf{zz}}[\boldsymbol{B}_{\mathbf{u}}^*]^{-1}\boldsymbol{\alpha}_{\mathbf{u}}^* + \mathbf{m}^*,
\end{aligned} \tag{46}$$

which recovers the second equation in Eq. (15). Note that we can use different forms as well. For example, an equivalent way to write Eq. (15) is

$$\hat{p}_{\mathcal{D}_{\text{old}}}(\mathbf{u}) = \mathrm{N}(\mathbf{u}|\tilde{\mathbf{\Sigma}}^{\text{old}}(\mathbf{V}^{\text{old}})^{-1}\mathbf{m}^{\text{old}},\ \tilde{\mathbf{\Sigma}}^{\text{old}}), \tag{47}$$

$$\tilde{\mathbf{y}}^{\text{old}} = \tilde{\mathbf{\Sigma}}^{\text{old}}(\mathbf{V}^{\text{old}})^{-1}\mathbf{m}^{\text{old}}, \tag{48}$$

$$\tilde{\mathbf{\Sigma}}^{\text{old}} = \left[(\mathbf{V}^{\text{old}})^{-1} - \mathbf{K}_{\mathbf{zz}}^{-1}\right]^{-1},$$

which is in terms of the mean and covariance of the posterior. This expression is similar to the one used in Bui et al. (Eq. 7 2017). All such forms are equivalent. We use Eq. (15) because it uses dual parameters which are readily obtained through the natural-gradient method of Adam et al. (2021).

We can write $\hat{p}_{\mathcal{M}}(\mathbf{u})$ in a similar way,

$$\hat{p}_{\mathcal{M}}(\mathbf{u}) \propto \prod_{i\in\mathcal{M}} e^{-\frac{1}{2}\hat{\beta}_i^{\text{old}}(\hat{y}_i^{\text{old}} - \boldsymbol{a}_i^\top \mathbf{u})^2} = \mathrm{N}(\mathbf{u}\,|\,\tilde{\mathbf{y}}^{\mathcal{M}}, \tilde{\mathbf{\Sigma}}^{\mathcal{M}}). \tag{49}$$

The expressions for $\tilde{\mathbf{y}}^{\mathcal{M}}$ and $\tilde{\mathbf{\Sigma}}^{\mathcal{M}}$ are derived by repeating the same derivation but now only involving $i \in \mathcal{M}$. For example,

$$\begin{aligned}
(\tilde{\mathbf{\Sigma}}^{\mathcal{M}})^{-1} &= \sum_{i\in\mathcal{M}} \boldsymbol{a}_i \hat{\beta}_i^{\text{old}} \boldsymbol{a}_i^\top = \mathbf{K}_{\mathbf{zz}}^{-1} \underbrace{\left(\textstyle\sum_{i\in\mathcal{M}} \mathbf{k}_{\mathbf{z}i}\hat{\beta}_i^{\text{old}}\mathbf{k}_{\mathbf{z}i}^\top\right)}_{=\boldsymbol{B}_{\mathbf{u}}^{\mathcal{M}}} \mathbf{K}_{\mathbf{zz}}^{-1} \\
\tilde{\mathbf{y}}^{\mathcal{M}} &= \tilde{\mathbf{\Sigma}}^{\mathcal{M}} \sum_{i\in\mathcal{M}} \boldsymbol{a}_i \hat{\beta}_i^{\text{old}} \hat{y}_i^{\text{old}} = \mathbf{K}_{\mathbf{zz}}[\boldsymbol{B}_{\mathbf{u}}^{\mathcal{M}}]^{-1} \sum_{i\in\mathcal{M}} \mathbf{k}_{\mathbf{z}i}(\hat{\alpha}_i^{\text{old}} + \hat{\beta}_i^{\text{old}} \boldsymbol{a}_i^\top \mathbf{m}^{\text{old}}) \\
&= \mathbf{K}_{\mathbf{zz}}[\boldsymbol{B}_{\mathbf{u}}^{\mathcal{M}}]^{-1} \underbrace{\sum_{i\in\mathcal{M}} \mathbf{k}_{\mathbf{z}i}\hat{\alpha}_i^{\text{old}}}_{=\boldsymbol{\alpha}_{\mathbf{u}}^{\mathcal{M}}} + \left[\mathbf{K}_{\mathbf{zz}}[\boldsymbol{B}_{\mathbf{u}}^{\mathcal{M}}]^{-1}\left(\sum_{i\in\mathcal{M}} \mathbf{k}_{\mathbf{z}i}\hat{\beta}_i^{\text{old}}\mathbf{k}_{\mathbf{z}i}\right)\mathbf{K}_{\mathbf{zz}}^{-1}\right]\mathbf{m}^* \\
&= \mathbf{K}_{\mathbf{zz}}[\boldsymbol{B}_{\mathbf{u}}^{\mathcal{M}}]^{-1}\boldsymbol{\alpha}_{\mathbf{u}}^{\mathcal{M}} + \mathbf{m}^*.
\end{aligned} \tag{50}$$

Next, we derive the mean and covariance of the new prior $\hat{q}_{\mathbf{u}}^{\text{old}}(\mathbf{u})/\hat{p}_{\mathcal{M}}(\mathbf{u}) \propto \mathrm{N}(\mathbf{u}|\mathbf{m}_{\text{prior}}, \mathbf{V}_{\text{prior}})$. Let us denote the current estimate of the dual pair by $(\boldsymbol{\alpha}_{\mathbf{u}}^{\text{old}}, \boldsymbol{B}_{\mathbf{u}}^{\text{old}})$ as defined in Eq. (11). There, the whole $\mathcal{D}_{\text{old}}$ is used but, in reality, the dual pairs are learned sequentially and may not represent the exact dual parameters. Still, we use the same notation for convenience. The current natural-parameters of the posterior can be then written by rewriting Eqs. (7) and (8) in terms of the dual pairs,

$$\begin{aligned}
(\hat{\mathbf{V}}^{\text{old}})^{-1}\hat{\mathbf{m}}^{\text{old}} &= \mathbf{K}_{\mathbf{zz}}^{-1}\boldsymbol{\alpha}_{\mathbf{u}}^{\text{old}} + \mathbf{K}_{\mathbf{zz}}^{-1}\boldsymbol{B}_{\mathbf{u}}^{\text{old}}\mathbf{K}_{\mathbf{zz}}^{-1}\mathbf{m}^{\text{old}}, \\
(\hat{\mathbf{V}}^{\text{old}})^{-1} &= \mathbf{K}_{\mathbf{zz}}^{-1}\boldsymbol{B}_{\mathbf{u}}^{\text{old}}\mathbf{K}_{\mathbf{zz}}^{-1} + \mathbf{K}_{\mathbf{zz}}^{-1}.
\end{aligned} \tag{51}$$

Similarly, we can write the natural parameters of $\hat{p}_{\mathcal{M}}(\mathbf{u})$ in terms (of the current estimate) of $(\boldsymbol{\alpha}_{\mathbf{u}}^{\mathcal{M}}, \boldsymbol{B}_{\mathbf{u}}^{\mathcal{M}})$,

$$\begin{aligned}
(\hat{\mathbf{V}}^{\mathcal{M}})^{-1}\hat{\mathbf{m}}^{\mathcal{M}} &= (\tilde{\mathbf{\Sigma}}^{\mathcal{M}})^{-1}\tilde{\mathbf{y}}^{\mathcal{M}} = \mathbf{K}_{\mathbf{zz}}^{-1}\boldsymbol{\alpha}_{\mathbf{u}}^{\mathcal{M}} + \mathbf{K}_{\mathbf{zz}}^{-1}\boldsymbol{B}_{\mathbf{u}}^{\mathcal{M}}\mathbf{K}_{\mathbf{zz}}^{-1}\mathbf{m}^*, \\
(\hat{\mathbf{V}}^{\mathcal{M}})^{-1} &= (\tilde{\mathbf{\Sigma}}^{\mathcal{M}})^{-1} = \mathbf{K}_{\mathbf{zz}}^{-1}\boldsymbol{B}_{\mathbf{u}}^{\mathcal{M}}\mathbf{K}_{\mathbf{zz}}^{-1}.
\end{aligned} \tag{52}$$

The natural parameters of the new prior are simply obtained by subtracting the natural parameters given above,

$$\begin{aligned}
(\mathbf{V}^{\text{prior}})^{-1}\mathbf{m}^{\text{prior}} &= (\hat{\mathbf{V}}^{\text{old}})^{-1}\hat{\mathbf{m}}^{\text{old}} - (\hat{\mathbf{V}}^{\mathcal{M}})^{-1}\hat{\mathbf{m}}^{\mathcal{M}} = \mathbf{K}_{\mathbf{zz}}^{-1}\boldsymbol{\alpha}_{\mathbf{u}}^{\text{old}\setminus\mathcal{M}} + \mathbf{K}_{\mathbf{zz}}^{-1}\boldsymbol{B}_{\mathbf{u}}^{\text{old}\setminus\mathcal{M}}\mathbf{K}_{\mathbf{zz}}^{-1}\mathbf{m}^{\text{old}}, \\
(\mathbf{V}^{\text{prior}})^{-1} &= (\hat{\mathbf{V}}^{\text{old}})^{-1} - (\hat{\mathbf{V}}^{\mathcal{M}})^{-1} = \mathbf{K}_{\mathbf{zz}}^{-1}\boldsymbol{B}_{\mathbf{u}}^{\text{old}\setminus\mathcal{M}}\mathbf{K}_{\mathbf{zz}}^{-1} + \mathbf{K}_{\mathbf{zz}}^{-1},
\end{aligned} \tag{53}$$

where $(\boldsymbol{\alpha}_{\mathbf{u}}^{\text{old}\setminus\mathcal{M}},\ \boldsymbol{B}_{\mathbf{u}}^{\text{old}\setminus\mathcal{M}})$ are defined in Eqs. (19) and (20).

A.4. Derivation of the Natural-Gradient Descent Algorithm

We will optimize the new objective in Eq. (21) by using the Bayesian learning rule (BLR) of Khan & Rue (2021) which is a natural-gradient descent algorithm. We start by denoting the natural and expectation parameters of a posterior $q_{\mathbf{u}}^{(t)}(\mathbf{u})$

obtained in the t'th iteration by $\boldsymbol{\lambda}^{(t)}$ and $\boldsymbol{\mu}^{(t)}$ respectively. We denote the natural parameters of the prior $\hat{q}_{\mathbf{u}}^{\text{old}}(\mathbf{u})/\hat{p}_{\mathcal{M}}(\mathbf{u}) \propto \mathrm{N}(\mathbf{u}|\mathbf{m}_{\text{prior}}, \mathbf{V}_{\text{prior}})$ by $\boldsymbol{\lambda}_{\text{prior}}$; an expression is given in Eq. (53). With these, the BLR update can be written as the following,

$$\boldsymbol{\lambda}^{(t)} \leftarrow (1-\rho)\boldsymbol{\lambda}^{(t-1)} + \rho\Big(\sum_{i\in\mathcal{D}_{\text{new}}} \nabla_{\boldsymbol{\mu}}\, \mathbb{E}_{q_{\mathbf{u}}(f_i)}[\log p(y_i \mid f_i)]\big|_{\boldsymbol{\mu}=\boldsymbol{\mu}^{(t)}} + \boldsymbol{\lambda}_{\text{prior}}\Big). \tag{54}$$

To simplify the implementation, we will write the updates in terms of the estimate of the dual pair $(\boldsymbol{\alpha}_{\mathbf{u}}^{(t)}, \boldsymbol{B}_{\mathbf{u}}^{(t)})$ at iteration t. We make use of the fact that each iterate $\boldsymbol{\lambda}^{(t)}$ has the same dual form as in Eq. (52). This is written below,

$$\begin{aligned}(\mathbf{V}^{(t)})^{-1}\mathbf{m}^{(t)} &= \mathbf{K}_{\mathbf{zz}}^{-1}\boldsymbol{\alpha}_{\mathbf{u}}^{(t)} + \mathbf{K}_{\mathbf{zz}}^{-1}\boldsymbol{B}_{\mathbf{u}}^{(t)}\mathbf{K}_{\mathbf{zz}}^{-1}\mathbf{m}^{(t)},\\ (\mathbf{V}^{(t)})^{-1} &= \mathbf{K}_{\mathbf{zz}}^{-1}\boldsymbol{B}_{\mathbf{u}}^{(t)}\mathbf{K}_{\mathbf{zz}}^{-1} + \mathbf{K}_{\mathbf{zz}}^{-1}.\end{aligned} \tag{55}$$

As shown in Eq. (53), the prior $\boldsymbol{\lambda}_{\text{prior}}$ too has the same form written in terms of the dual parameters $(\boldsymbol{\alpha}_{\mathbf{u}}^{\text{old}\setminus\mathcal{M}}, \boldsymbol{B}_{\mathbf{u}}^{\text{old}\setminus\mathcal{M}})$. Finally, as shown in Eqs. (39) and (40), the natural-gradients too can be written in the same form,

$$\textstyle\sum_{i\in\mathcal{D}_{\text{new}}} \nabla_{\boldsymbol{\mu}^{(1)}}\mathbb{E}_{q_{\mathbf{u}}(f_i)}[\log p(y_i \mid f_i)]\big|_{\boldsymbol{\mu}=\boldsymbol{\mu}_t} = \mathbf{K}_{\mathbf{zz}}^{-1}\left(\sum_{i\in\mathcal{D}_{\text{new}}} \mathbf{k}_{\mathbf{z}i}\hat{\alpha}_i^{(t)}\right) + \mathbf{K}_{\mathbf{zz}}^{-1}\left(\sum_{i\in\mathcal{D}_{\text{new}}} \mathbf{k}_{\mathbf{z}i}\hat{\beta}_i^{(t)}\mathbf{k}_{\mathbf{z}i}^{\top}\right)\mathbf{K}_{\mathbf{zz}}^{-1}\mathbf{m}^{(t)}, \tag{56}$$

$$\textstyle\sum_{i\in\mathcal{D}_{\text{new}}} \nabla_{\boldsymbol{\mu}^{(2)}}\mathbb{E}_{q_{\mathbf{u}}(f_i)}[\log p(y_i \mid f_i)]\big|_{\boldsymbol{\mu}=\boldsymbol{\mu}_t} = \mathbf{K}_{\mathbf{zz}}^{-1}\left(\sum_{i\in\mathcal{D}_{\text{new}}} \mathbf{k}_{\mathbf{z}i}\hat{\beta}_i^{(t)}\mathbf{k}_{\mathbf{z}i}^{\top}\right)\mathbf{K}_{\mathbf{zz}}^{-1}, \tag{57}$$

where $\hat{\alpha}_i^{(t)}$ and $\hat{\beta}_i^{(t)}$ are defined similarly to Eqs. (7) and (8), but now by using $q_{\mathbf{u}}^{(t-1)}(f_i)$,

$$\hat{\alpha}_i^{(t)} = \mathbb{E}_{q_{\mathbf{u}}^{(t-1)}(f_i)}[\nabla_{f_i}\log p(y_i \mid f_i)], \qquad \hat{\beta}_i^{(t)} = \mathbb{E}_{q_{\mathbf{u}}^{(t-1)}(f_i)}[-\nabla_{f_i}^2\log p(y_i \mid f_i)]. \tag{58}$$

We can use these to simply write the update in terms of the dual pair. Essentially, we use the following equivalent update,

$$\begin{aligned}\boldsymbol{\alpha}_{\mathbf{u}}^{(t)} &\leftarrow (1-\rho)\boldsymbol{\alpha}_{\mathbf{u}}^{(t-1)} + \rho\left(\boldsymbol{\alpha}_{\mathbf{u}}^{\text{old}\setminus\mathcal{M}} + \textstyle\sum_{i\in\mathcal{D}_{\text{new}}} \mathbf{k}_{\mathbf{z}i}\hat{\alpha}_i^{(t)}\right),\\ \boldsymbol{B}_{\mathbf{u}}^{(t)} &\leftarrow (1-\rho)\boldsymbol{B}_{\mathbf{u}}^{(t-1)} + \rho\left(\boldsymbol{B}_{\mathbf{u}}^{\text{old}\setminus\mathcal{M}} + \textstyle\sum_{i\in\mathcal{D}_{\text{new}}} \mathbf{k}_{\mathbf{z}i}\hat{\beta}_i^{(t)}\mathbf{k}_{\mathbf{z}i}^{\top}\right).\end{aligned} \tag{59}$$

This is followed by an update of the mean and the covariance given below,

$$\mathbf{m}^{(t)} \leftarrow \boldsymbol{\alpha}_{\mathbf{u}}^{(t)}, \qquad \mathbf{V}^{(t)} \leftarrow \left(\mathbf{K}_{\mathbf{zz}}^{-1}\boldsymbol{B}_{\mathbf{u}}^{(t)}\mathbf{K}_{\mathbf{zz}}^{-1} + \mathbf{K}_{\mathbf{zz}}^{-1}\right)^{-1}. \tag{60}$$

This is derived by using Eq. (55) and simplifying similarly to the first equation in Eq. (41).

A.5. Bayesian Optimization / Active Learning Algorithm

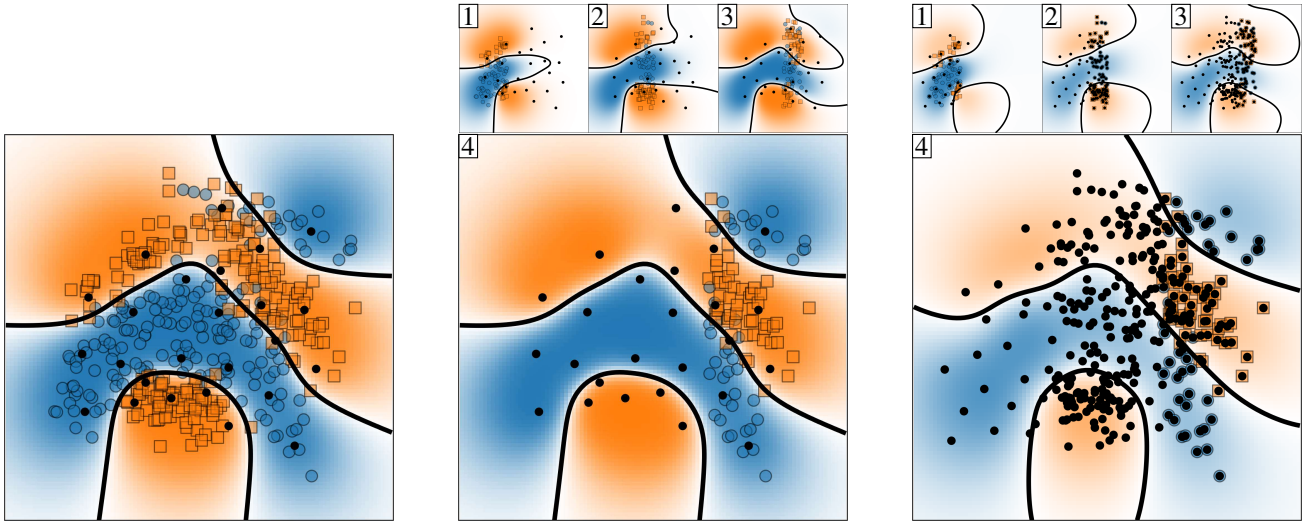
In Alg. 2, we include the algorithm that is used in the Bayesian optimization experiment in the main paper (Sec. 4.1), where we fantasize a batch with dual conditioning. The algorithm uses the method outlined in the paper combined with any simple acquisition function $\gamma(\cdot)$.

Algorithm 2 Fantasizing a batch with Dual Conditioning.

Input: current model parameters $\boldsymbol{\theta}$, $\mathbf{Z}$, $(\boldsymbol{\alpha}_{\mathbf{u}}, \boldsymbol{B}_{\mathbf{u}})$, acquisition function $\gamma(\cdot)$, batch size k
Initialize: $\mathbf{X}_b = \emptyset$

1: **for** i in $1, 2, \ldots, k$ **do** ▷ k is desired number of query points
2: $\quad\mathbf{x}_i = \arg\max_{\mathbf{x}} \gamma(\mathbf{x})$ ▷ Calculate $\gamma(\mathbf{x})$ using prediction function Eq. (4) at $\mathbf{x}$
3: $\quad y_i = \mathbb{E}[f(\mathbf{x}_i)]$ ▷ Fantasized y is mean of the GP at $\mathbf{x}_i$
4: $\quad\mathcal{D}_{\text{new}} = (\mathbf{x}_i, y_i)$ ▷ The fantasized data point is treated as new data
5: $\quad$Compute $(\boldsymbol{\alpha}_{\mathbf{u}}, \boldsymbol{B}_{\mathbf{u}})$ using $\mathcal{D}_{\text{new}}$ using method in Adam et al. (2021) ▷ Dual conditioning
6: $\quad\mathbf{X}_{\text{b}} \leftarrow \mathbf{X}_{\text{b}} \cup \mathbf{x}_i$ ▷ $\mathbf{x}_i$ is added to the current batch points
7: **end for**

Return: $\mathbf{X}_{\text{b}}$ ▷ $\mathbf{X}_{\text{b}}$ is the chosen batch of points.



(a) Offline SVGP model (baseline)

(b) Streaming dual SVGP (ours)

(c) OVC (Maddox et al., 2021)

Figure 8. Conditioning on streaming banana data set; data appears batch by batch (1–4). The plot shows the decision boundary and the predictive class probability, with colour shading and increasing the more certain the model is about the class. The inducing points are overlaid as black dots. (a) Offline SVGP model trained with full data. (b) Dual SVGP model conditioned on the data appearing in batches. (c) Online Variational Conditioning (OVC, Maddox et al., 2021) model on batched data.

B. Experiment Details

In Sec. 4, we performed a series of experiments and ablation studies to showcase the capability of our proposed method in various setups. We also compared against other methods, in particular Bui et al. (2017) and Maddox et al. (2021). Here, we provide further details regarding the setup and the experiments performed.

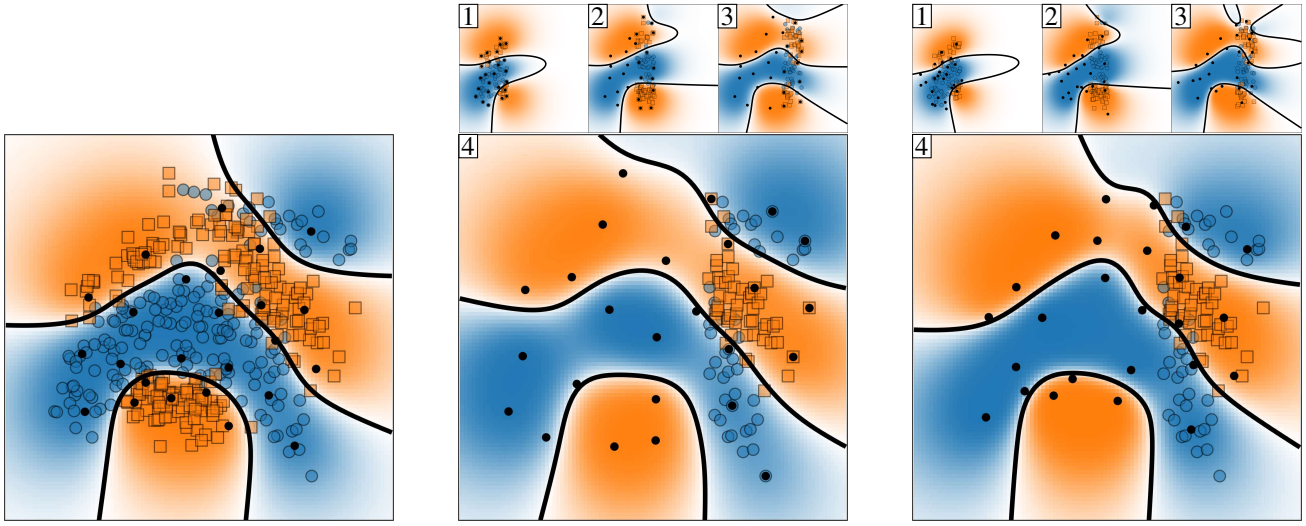
B.1. Streaming Banana Data Set

The streaming banana classification experiment was used by both Bui et al. (2017) and Maddox et al. (2021). The data set is divided into four batches of 100 points each. First, we compare within the setup of Maddox et al. (2021), who focused on fast conditioning, but kept the hyperparameters fixed. Second, we compare with Bui et al. (2017), who address hyperparameter learning without considering the speed of conditioning.

Fast Conditioning For a fair comparison against Maddox et al. (2021) in this experiment we also keep the hyperparameters fixed. The problem's challenge is that previous batches are not accessible; only the inferred variational parameters are available. Therefore, online models are needed to condition on new data. As an oracle baseline, we trained an offline SVGP model on the full data (see Fig. 8a). All three models are intialized with 25 inducing points and a Matérn-5/2 kernel. The hyperparameters for our streaming dual SVGP and Maddox et al. (2021)'s OVC model are taken from the full offline model: only conditioning is performed when the batches of data are received.

We compare decision boundary and predictive class probability of the three models in Fig. 8. The OVC method as introduced by Maddox et al. (2021) essentially initializes new models on each batch and then combines them, hence the increasing number of inducing points for this model. The evolution of the class probability of dual SVGP and OVC is shown in Fig. 8b and Fig. 8c, respectively. The class probability obtained by the dual SVGP model after seeing the final batch closely matches that of the offline SVGP model. In contrast, the OVC method does not recover the full-data decision boundary, and its uncertainty does not match the offline baseline well.

Hyperparameter Learning Next, we conduct an experiment in which we learn all the hyperparameters $(\boldsymbol{\alpha}_\mathbf{u}, \boldsymbol{B}_\mathbf{u}, \boldsymbol{\theta}, \mathbf{Z})$, in contrast to the previous experiment where the hyperparameters were fixed. We compare to Bui et al. (2017), who previously considered a similar test setup. Again, as an oracle baseline, an offline SVGP model is trained on the full data (see Fig. 9a). All three models are intialized with 25 inducing points and Matérn-5/2 kernel. The hyperparameters for the dual SVGP are optimized using Adam optimizer (Kingma & Ba, 2015) with learning rate 10^{-2} and for Bui et al. (2017) we use L-BFGS.



(a) Offline SVGP model (b) Streaming dual SVGP (c) Bui et al. (2017)

Figure 9. Streaming banana data set when $(\boldsymbol{\alpha}_{\mathbf{u}}, \boldsymbol{B}_{\mathbf{u}}, \boldsymbol{\theta}, \mathbf{Z})$ are learnt; data appears batch by batch (1–4). The plot shows the decision boundary and the predictive class probability, with colour shading and increasing the more certain the model is about the class. The inducing points are overlaid as black dots.

(for Bui et al. (2017)'s model, we tried both Adam optimizer and L-BFGS optimizer; L-BFGS gave better results).

We compare the decision boundary and predictive class probability of the three models in Fig. 9. The evolution of the class probability of our dual SVGP and Bui et al. (2017) is shown in Fig. 9b and Fig. 9c.

B.2. Split MNIST

For split MNIST (see Sec. 4.3), we use the standard MNIST data provided by TensorFlow. We concatenate the standard train and test set provided and split it $80:20$ for training and testing. Each task is sub-divided into batches of 4000.

Both the models, our proposed model and the Bui et al. model, use a Matérn-$\frac{5}{2}$ kernel initialized with unit variance and lengthscale. We use 10 latent GPs which matches the number of classes, with 300 inducing variables, and use a softmax likelihood.

For hyperparameter learning in our proposed model, we use the Adam optimizer (Kingma & Ba, 2015) with learning rate 10^{-2} for 100 iterations for each set of data. The number of memory points for each set of tasks is set to 400. We also found that for this task the removal of memory set from the variational parameters actually gave worse performance for split MNIST so do not perform the removal the last term of Eq. (13), we suspect this is due to the highly non-stationary nature of the problem and how double counting dual variables alleviates the difficulty.

For Bui et al., we experiment with the L-BFGS optimizer as well as Adam optimizer with 50 and 100 iterations; in this case we found that Adam works better (L-BFGS fails to learn subsequent tasks). The accuracy over tasks for different numbers of iterations can be seen in Tables 4 and 5.

B.3. Magnetic Anomaly Modelling

For the robot experiment for learning magentic field anomalies, we use the data from Solin et al. (2018) that is available at `https://github.com/AaltoML/magnetic-data`. The exact metric position of the robot is provided time-synced to magnetometer samples from a low-cost/high-noise Invensense MPU-9150 magnetometer sampled at 50 Hz. As discussed in Solin et al. (2018), GPs provide a principled approach for modelling smooth anomalies in the magnetic vector field.

We consider the task of online mapping of local anomalies in the ambient Earth magnetic field. These anomalies are induced by the bedrock and magnetic material in the building structure indoors. We follow the experiment setup and data provided in Solin et al. (2018), where a small wheeled robot equipped with a 3-axis magnetometer moves around in an indoor space

measuring roughly $6\,\mathrm{m} \times 6\,\mathrm{m}$. We consider a simplified model, where we assign a GP prior to the magnetic field strength $\|\mathbf{H}\| \sim \mathcal{GP}(0, \sigma_0^2 + \kappa^{\mathrm{Mat.}}_{\sigma^2,\ell}(\mathbf{x}, \mathbf{x}'))$ (in $\mu\mathrm{T}$) over the space under presence of Gaussian measurement noise with variance σ_{n}^2.

We include two separate experiments with different data paths in the same space. The first experiment (Fig. 5a) uses four separate paths of the robot that cover slightly different parts of the space. Under the sequential learning framework, we simultaneously learn the four hyperparameters $(\sigma_0^2, \sigma^2, \ell, \sigma_{\mathrm{n}}^2)$ and the representation in terms of inducing points and memory. The high measurement noise renders the value of single data points small and stresses the importance of the sparse approach. The problem is sequential, as we only receive information as the robot moves through the input space and do not have access to all previous data. Our sequential method is able to form a representation by spreading inducing points and learning the hyperparameters progressively. Showing the method has practical importance in real-world robot estimation tasks, where the robot is not constrained to a predefined area. This has been a weakness in previous approaches that have considered a fixed domain to form an efficient sparse basis function decomposition of the problem in that domain. In Fig. 5b, the set up of the second experiment is similar, but now we receive data during each robot path instead after one exploration. The visualization shows the mean estimate with marginal variance (uncertainty) controlling the opacity. We recover the same local estimate as in experiment Fig. 5a.

The data is a set of 9 trajectories out of which we use 5 for the experiments. For Experiment #1, Fig. 5a, we use trajectory 1, 2, 4, and 5 (with $n = 8875, 9105, 7332, 8313$, respectively) and for Experiment #2, Fig. 5b, we use trajectory 3 ($n = 9404$). For both the experiments, we use a sum of two kernels: constant kernel and a Matérn-5/2 kernel. For the constant kernel, we set the initial variance as 500. The number of inducing variables is set to 100 in both the experiments, and we use a Gaussian likelihood initialized with a noise variance of 0.1. For optimization of the hyperparameters, we use the Adam optimizer with learning rate 10^{-2} for 20,000 iterations. For Experiment #2, we split trajectory 3 into 20 sets, thus each set has around 470 data points.

B.4. Lunar Landing

For the lunar landing experiment (see Sec. 4.1), we use a combination of two models: a regression model with the aim to increase the reward and a classification model for success or failure. Both the models are the proposed dual SVGP model with Gaussian likelihood and Bernoulli likelihood, respectively. For the regression model, we use an ARD Matérn-5/2 kernel with initial variance calculated from the initial observation data and the lengthscale initialized with 0.2. We use a Gaussian likelihood with an initial unit noise variance. For the classification model, we use an ARD squared exponential kernel with the magnitude initialized with 100 and lengthscale with 0.2. For the classification model, we fix the magnitude. Both models use 100 inducing points. The acquisition function is the product of ExpectedImprovement from the regression model and the ProbabilityOfValidity from the classification model. Initial data for both the batch version and the non-batch version is the same set of 24 points. For the batch models, we use a batch-size of 3 and both the models are optimized for 90 function evaluations. We run the experiment with 5 random initial observations and plot the mean and individual rewards along with the BO iterations in Fig. 3.

B.5. UCI Data Sets

We benchmark on UCI data sets both for regression and classification tasks (see Sec. 4.2). We report negative log predictive density (NLPD), root mean square error (RMSE) for regression tasks, and classification error for classification tasks in

Table 4. Test accuracy (and standard deviation over different random seeds) on *split MNIST* over all tasks thus far for Bui et al. (2017). Low accuracy over all previous tasks shows that the method suffers from forgetting. (Adam optimizer, 50 iteration loops)

Task	0, 1	2, 3	4, 5	6, 7	8, 9
#1	.72(.01)				
#2	.11(.01)	.36(.26)			
#3	.04(.02)	.02(.00)	.95(.01)		
#4	.02(.01)	.01(.00)	.03(.01)	.98(.00)	
#5	.06(.04)	.06(.04)	.05(.04)	.10(.04)	.55(.34)

Table 5. Test accuracy (and standard deviation over different random seeds) on *split MNIST* over all tasks thus far for Bui et al. (2017). Low accuracy over all previous tasks shows that the method suffers from forgetting. (Adam optimizer, 100 iteration loops)

Task	0, 1	2, 3	4, 5	6, 7	8, 9
#1	.99(.00)				
#2	.01(.00)	.96(.00)			
#3	.02(.04)	.02(.04)	.80(.40)		
#4	.00(.00)	.00(.00)	.00(.00)	.99(.00)	
#5	.00(.00)	.00(.00)	.00(.00)	.00(.00)	.97(.00)

Table 1 and Table 6 over 10-fold cross-validation. We run three models: **offline** model and two online models (**Ours**, and **Bui et al.**). The offline model has access to the whole data and is used as baseline. **Ours** is our proposed method where all the parameters $(\boldsymbol{\alpha}_{\mathbf{u}}, \boldsymbol{B}_{\mathbf{u}}, \boldsymbol{\theta}, \mathbf{Z})$ are learnt. **Bui et al.** is the model proposed by Bui et al. (2017).

All the models use a Matérn-5/2 kernel with lengthscale and variance initialized to 1.0, a Gaussian likelihood with initial noise variance 0.1, and 100 inducing points. For converting the data sets into a streaming setting, we sort the data on the first dimension and split the data set into 50 subsets for all data sets apart from *Mammographic* (20 subsets). For variational parameters $(\boldsymbol{\alpha}_{\mathbf{u}}, \boldsymbol{B}_{\mathbf{u}})$ of our proposed model, we use natural gradient updates, with learning rate 0.8 and 2 update steps for regression and learning rate 0.5 and 4 update steps for classification. For learning hyperparameters $(\theta, \mathbf{Z})$, we use the Adam optimizer (Kingma & Ba, 2015) with learning rate 10^{-2} and 100 update steps. For optimizing the parameters of Bui et al.'s model, we tried both Adam and second-order optimization using L-BFGS. In our experiments, we found that L-BFGS with 100 iteration steps or until the default convergence condition is met works the best, and used this for the final results. The offline model is trained using Adam optimizer with learning rate 10^{-2}.

Table 6. Root mean square error (RMSE) (for regression tasks, ^{R}) and classification error (for classification tasks, ^{C}) on 10-fold cross-validation for UCI data sets, lower is better.

Data set	**Dimension** (N, D)	**Offline**	**Ours**	**Bui et al.**
Elevators^R	$(16599, 18)$	.39(.00)	.42(.01)	.42(.01)
Bike^R	$(17379, 17)$	.29(.01)	.37(.01)	.38(.01)
Mammographic^C	$(961, 6)$	.18(.01)	.81(.03)	.81(.04)
Bank^C	$(4521, 17)$	.11(.01)	.88(.01)	.89(.02)
Mushroom^C	$(8124, 22)$	.00(.00)	.97(.03)	.99(.01)
Adult^C	$(48842, 15)$	.16(.00)	.82(.01)	.83(.01)

B.6. UCI Data sets: BLS and Random

We experiment with two different techniques for updating memory: random and the proposed Bayesian leverage score (BLS). We report the negative log predictive density (NLPD) (Table 7) and showcase BLS outperforming random memory technique.

All the models use a Matérn-5/2 kernel with lengthscale and variance initialized to 1.0, a Gaussian likelihood with initial noise variance 0.1, and 100 inducing points. For converting the data sets into a streaming setting, we sort the data on the first dimension and split the data set into 10 subsets for all data sets. For variational parameters $(\boldsymbol{\alpha}_{\mathbf{u}}, \boldsymbol{B}_{\mathbf{u}})$ of our proposed model, we use natural gradient updates, with learning rate 0.8 and 2 update steps for regression and learning rate 0.2 and 10 update steps for classification. For learning hyperparameters $(\theta, \mathbf{Z})$, we use the Adam optimizer (Kingma & Ba, 2015) with learning rate 10^{-2} and 100 update steps. For optimizing the parameters of Bui et al.'s model, we tried both Adam and second-order optimization using L-BFGS. In our experiments, we found that L-BFGS with 100 iteration steps or until the default convergence condition is met works the best, and used this for the final results. The offline model is trained using Adam optimizer with learning rate 10^{-2}.

Table 7. Negative log predictive density (NLPD) on 5-fold cross-validation for UCI data sets, lower is better, for both random and Bayesian leverage score.

Data set	**Random**	**BLS**
Adult	.35(.01)	.34(.01)
Bank	.29(.03)	.27(.03)
Mushroom	.03(.00)	.03(.00)
Mammographic	.45(.03)	.43(.02)
Elevators	.63(.02)	.63(.04)
Bike	.47(.05)	.46(.03)