Recursive Estimation for Sparse Gaussian Process Regression

Manuel Schürch [a,b], Dario Azzimonti [a], Alessio Benavoli [c,a], Marco Zaffalon [a]

[a] *Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA); Manno, Switzerland*

[b] *Università della Svizzera italiana (USI); Lugano, Switzerland*

[c] *University of Limerick (UL); Limerick, Ireland*

Abstract

Gaussian Processes (GPs) are powerful kernelized methods for non-parameteric regression used in many applications. However, their use is limited to a few thousand of training samples due to their cubic time complexity. In order to scale GPs to larger datasets, several sparse approximations based on so-called inducing points have been proposed in the literature. In this work we investigate the connection between a general class of sparse inducing point GP regression methods and Bayesian recursive estimation which enables Kalman Filter like updating for online learning. The majority of previous work has focused on the batch setting, in particular for learning the model parameters and the position of the inducing points, here instead we focus on training with mini-batches. By exploiting the Kalman filter formulation, we propose a novel approach that estimates such parameters by recursively propagating the analytical gradients of the posterior over mini-batches of the data. Compared to state of the art methods, our method keeps analytic updates for the mean and covariance of the posterior, thus reducing drastically the size of the optimization problem. We show that our method achieves faster convergence and superior performance compared to state of the art sequential Gaussian Process regression on synthetic GP as well as real-world data with up to a million of data samples.

Key words: Gaussian processes; Recursive estimation; Kalman filter; Non-parametric regression; Parameter estimation.

1 Introduction

Gaussian process (GPs) regression is used in many applications, ranging from machine learning, social sciences, natural sciences and engineering, due to its modeling flexibility, robustness to overfitting and availability of well-calibrated predictive uncertainty estimates. In control engineering, for example, GPs have been used in system identification for impulse response estimation [25,9,26,24], nonlinear ARX models [19,3], learning of ODEs [1,21], latent force modeling [42] and to learn the state space of a nonlinear dynamical system [12,23,35]. However, GPs do not scale to large data sets due to their $\mathcal{O}(N^2)$ memory and $\mathcal{O}(N^3)$ computational costs, where N is the number of training samples. For this reason several sparse GP approximations have been proposed in the literature. Often such approximations are based on *inducing points* methods, where the unknown function is represented by its values at a set of $M \ll N$ pseudo-inputs, called inducing points. Among such methods, *Subset of Regressors* (SoR/DIC) approximations [32,39,33] produces overconfident predictions when leaving the training data. On the other hand, *Deterministic Training Conditional* (DTC) [11,31], *Fully Independent Training Conditional* (FITC) [34], *Fully Independent Conditional* (FIC) [27] and *Partially Independent Training Conditional* (PITC) [27] all produce sensible uncertainty estimates. These models differ from each other in the definition of their joint prior over the latent function and test values. Titsias [36], instead, proposed to retain the exact prior but to perform approximate (variational) inference for the posterior, leading to the *Variational Free Energy* (VFE) method which converges to full GP as M increases. Bui et al. [7] introduced *Power Expectation Propagation* (PEP), based on the minimization of an α-divergence, which unifies most of the previously mentioned models. Typically, inference is achieved in $\mathcal{O}(M^2N)$ time and $\mathcal{O}(MN)$ space. In order to find good parameters (inducing input points and kernel hyper-parameters), either the log marginal likelihood of the sparse models or a lower bound are numerically optimized. The previously mentioned approximations focus on the batch setting, i.e., all data is available at once and can be processed together. For big data, where the number of samples can be millions, keeping all data in memory is not possible, moreover the data might even arrive sequentially. Bui et al. [6] developed an algorithm to update hyper-parameters in

Email addresses: manuel@idsia.ch (Manuel Schürch), dario.azzimonti@idsia.ch (Dario Azzimonti), alessio.benavoli@ul.ie (Alessio Benavoli), zaffalon@idsia.ch (Marco Zaffalon).

an online fashion promising in a streaming setting, but with limited accuracy as each sample is considered only once.

We focus here on the setting where hyper-parameters are learned by reconsidering mini-batches several times. In order to speed up the optimization, we would like to update the parameters more frequently for a subset of data and update the posterior in a sequential way. In this setting, Hensman et al. [15] applied *Stochastic Variational Inference* (SVI, [17]) to an *uncollapsed* lower bound of the marginal likelihood. The resulting *Stochastic Variational Gaussian Process* (SVGP) method allows to optimize the parameters with mini-batches. Although showing high scalability and good accuracy, SVGP has two main drawbacks: i) the (variational) posterior is not given analytically, which leads to $\mathscr{O}(M^2)$ additional many parameters; ii) the uncollapsed bounds are in practice often less tight than the corresponding collapsed VFE batch bounds because the (variational) posterior is not optimally eliminated. The large number of parameters ($\approx MD+M^2$, where D is the input space dimension) leads to a hard-to-tune optimization problem which requires appropriately decaying learning rates. Even for fixed parameters, each sample still needs to be reconsidered many times. An orthogonal direction was pursued by the authors in [14] and [30], where a connection between GPs and State Space models for particular kernels was established for spatio-temporal regression problems, which allows to apply sequential algorithm such as the Kalman Filter, see also [8,2]. Inspired by this line of research, the authors in [37] focused on efficient implementation and extended the methodology to varying sampling locations over time. These approaches can deal with sequential data and solve the problem of temporal time complexity, however the space complexity is still cubic in N. In addition, the hyper-parameters are usually fixed in advance.

In this work we propose a *recursive collapsed* lower bound to the log marginal likelihood which can be optimized stochastically with mini-batches.

In this respect, the first contribution of this paper is the derivation of a *novel Kalman-filter-like* (KF) formulation for a generic sparse inducing point method. In particular we show that sparse inducing point models can be seen as a Bayesian kernelized linear regression model with input dependent observation noise, a particular choice of basis functions and noise covariance. Given the model hyper-parameters, KF allows to train sparse GP methods analytically and exactly in an online setting (considering each sample only once, as opposed to the work in [15] and [16]). In this formulation the posterior distribution obtained online is equivalent to full batch methods. This constitutes an interesting technique on its own for applications where hyper-parameters are given, however the analysis above provides a key insight for parameter estimation.

Our second main contribution is a *recursive approach to hyper-parameter estimation* based on the KF formulation. It is based on recursively exploiting the chain rule for derivatives by recursively propagating the analytical gradients of the posterior which enables us to compute the derivatives of the lower bound sequentially. We show that, when computing the gradients of the recursive collapsed bound in a non-stochastic way, they exactly match the corresponding batch ones. This new *Stochastic Recursive Gradient Propagation* (SRGP) [1] approach constitutes an efficient method to train a very general class of sparse GP regression models with much fewer parameters to be estimated numerically ($\approx MD$) than state of the art sequential GP regression methods ($\approx MD+M^2$). Since the number M of inducing points determines the quality of the approximation to full GP, this reduction in number of parameters from M^2 to M is crucial and results in more accurate and faster convergence than state of the art approaches such as SVGP. For example, in the application to learn the input output behavior of a non-linear plan presented in Sect. 5 the number of parameters estimated by SVGP is ≈ 10500 while our approach only estimates ≈ 500 parameters due to the analytical updates.

2 Background on GP Regression

Consider a training set $\mathscr{D} = \{y_i, \boldsymbol{x}_i\}_{i=1}^N$ of N pairs of inputs $\boldsymbol{x}_i \in \mathbb{R}^D$ and noisy scalar outputs y_i generated by adding independent Gaussian noise to a latent function $f(\boldsymbol{x})$, that is $y_i = f(\boldsymbol{x}_i) + \varepsilon_i$, where $\varepsilon_i \sim \mathscr{N}\left(0, \sigma_n^2\right)$. We denote by $\boldsymbol{y} = [y_1, \ldots, y_N]^T$ the vector of observations and by $\boldsymbol{X} = [\boldsymbol{x}_1^T, \ldots, \boldsymbol{x}_N^T]^T \in \mathbb{R}^{N\times D}$ the input points.
We model f with a *Gaussian Process* (GP), a stochastic process defined by its mean function $m(\boldsymbol{x})$ and covariance kernel $k(\boldsymbol{x}, \boldsymbol{x}')$. The kernel k is a positive definite function [28], such as, for instance, the *squared exponential* (SE) kernel with individual lengthscales l_i for each dimension, that is $k(\boldsymbol{x}, \boldsymbol{x}') = \sigma_0^2 \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}')^T \operatorname{Diag}\left[l_1^2, \ldots, l_D^2\right](\boldsymbol{x}-\boldsymbol{x}')\right)$. We assume $m(x) \equiv 0$ for the sake of simplicity and we use the SE kernel throughout this paper however all methods work with any positive definite kernel. Given the training values $\boldsymbol{f} = f(\boldsymbol{X}) = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N)]^T$ and a test latent function value $f_* = f(\boldsymbol{x}_*)$ at a test point $\boldsymbol{x}_* \in \mathbb{R}^D$, then the joint distribution $p(\boldsymbol{f}, f_*)$ is Gaussian. Our likelihood is Gaussian, $p(\boldsymbol{y}|\boldsymbol{f}) = \mathscr{N}\left(\boldsymbol{y}|\boldsymbol{f}, \sigma_n^2 \mathbb{I}\right)$, and with Bayes theorem (see e.g. [28]) we obtain analytically the posterior predictive distribution $p(f_*|\boldsymbol{y}) = \mathscr{N}(f_*|\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*)$ with

$$\begin{aligned} \boldsymbol{\mu}_* &= \boldsymbol{K}_{*\boldsymbol{X}}\left(\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} + \sigma_n^2\mathbb{I}\right)^{-1}\boldsymbol{y}, \\ \boldsymbol{\Sigma}_* &= \boldsymbol{K}_{**} - \boldsymbol{K}_{*\boldsymbol{X}}\left(\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} + \sigma_n^2\mathbb{I}\right)^{-1}\boldsymbol{K}_{\boldsymbol{X}*}, \end{aligned} \tag{1}$$

where $[\boldsymbol{K}_{\boldsymbol{A}\boldsymbol{B}}]_{ij} = k(\boldsymbol{a}_i, \boldsymbol{b}_j)$ for any $\boldsymbol{A} \in \mathbb{R}^{M_1\times D}$ and $\boldsymbol{B} \in \mathbb{R}^{M_2\times D}$ with the corresponding rows $\boldsymbol{a}_i, \boldsymbol{b}_j$. For brevity, we use $*$ to indicate $\boldsymbol{x}_*$. The GP depends via the kernel matrices on the hyper-parameters $\boldsymbol{\phi} = \{\sigma_0, l_1, \ldots, l_D, \sigma_n\}$ typically estimated by maximizing the log marginal likelihood

$$\log p(\boldsymbol{y}|\boldsymbol{\phi}) = \log \mathscr{N}\left(\boldsymbol{y}|\boldsymbol{0}, \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} + \sigma_n^2\mathbb{I}\right). \tag{2}$$

Note that the computations for inference require the inversion of the matrix in Eq. (1) which scales as $\mathscr{O}(N^3)$ in time and $\mathscr{O}(N^2)$ for memory (given $\boldsymbol{\phi}$).

[1] Code is available at https://github.com/manuelIDSIA/SRGP.

2.1 Batch Sparse GP Regression

Sparse GP regression methods based on *inducing points* approaches reduce the computational complexity by introducing $M \ll N$ inducing points $\boldsymbol{u} \in \mathbb{R}^M$ that optimally summarize the dependency of the whole training data. The inducing *inputs* $\boldsymbol{R} \in \mathbb{R}^{M \times D}$ are in the D-dimensional input data space and the inducing *outputs* $\boldsymbol{u} := f(\boldsymbol{R})$ are the corresponding GP-function values, see also Fig. 1. The GP prior over $\boldsymbol{f}$ and f_* is augmented with the inducing outputs $\boldsymbol{u}$, leading to a joint $p(\boldsymbol{f}, f_*, \boldsymbol{u})$ and marginal $p(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{K_{RR}})$ prior. By marginalizing out the inducing points, the original prior $p(\boldsymbol{f}, f_*) = \int p(\boldsymbol{f}, f_* | \boldsymbol{u})\, p(\boldsymbol{u})\, \mathrm{d}\boldsymbol{u}$ is recovered. The fundamental approximation in all sparse GP models is that given the inducing outputs $\boldsymbol{u}$, $\boldsymbol{f}$ and f are conditionally independent. Consequently, inference in these models can be done in $\mathcal{O}(M^2 N)$ time and $\mathcal{O}(MN)$ space [34].

We briefly recall here the sparse predictive distribution and the variational lower bound to the log marginal likelihood for the *Power Expectation Propagation* (PEP) model [7] because it unifies the main sparse inducing points approaches. The variational lower bound is used for optimizing the parameters $\boldsymbol{\theta} := \{\phi, \boldsymbol{R}\}$. In the following, we denote $\boldsymbol{Q_{AB}} = \boldsymbol{K_{AR}} \boldsymbol{K_{RR}^{-1}} \boldsymbol{K_{RB}}$ and $\boldsymbol{D_A} = \boldsymbol{K_{AA}} - \boldsymbol{Q_{AA}}$ for any $\boldsymbol{A}, \boldsymbol{B}$. The predictive distribution $p(f_* | \boldsymbol{y}) = \mathcal{N}(f_* | \boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*)$ of PEP is given by

$$\begin{aligned} \boldsymbol{\mu}_* &= \boldsymbol{Q}_{*\boldsymbol{X}} \left(\overline{\boldsymbol{K}}_{\boldsymbol{XX}} + \sigma_n^2 \mathbb{I} \right)^{-1} \boldsymbol{y}, \\ \boldsymbol{\Sigma}_* &= \boldsymbol{K}_{**} - \boldsymbol{Q}_{*\boldsymbol{X}} \left(\overline{\boldsymbol{K}}_{\boldsymbol{XX}} + \sigma_n^2 \mathbb{I} \right)^{-1} \boldsymbol{Q}_{\boldsymbol{X}*}, \end{aligned} \tag{3}$$

where $\overline{\boldsymbol{K}}_{\boldsymbol{XX}} = \boldsymbol{Q}_{\boldsymbol{XX}} + \alpha \mathrm{Diag}[\boldsymbol{D}_{\boldsymbol{X}}]$. A lower bound to the sparse log marginal likelihood is analytically available

$$\begin{aligned} \mathscr{L}_{PEP}(\boldsymbol{\theta}) = &\log \mathcal{N}\left(\boldsymbol{y} | \boldsymbol{0}, \overline{\boldsymbol{K}}_{\boldsymbol{XX}} + \sigma_n^2 \mathbb{I} \right) \\ &- \frac{1-\alpha}{2\alpha} \sum_{i=1}^{N} \log \left(1 + \frac{\alpha}{\sigma_n^2} [\boldsymbol{D}_{\boldsymbol{X}}]_{ii} \right), \end{aligned} \tag{4}$$

where we omit the explicit dependency on $\boldsymbol{\theta}$ via $\overline{\boldsymbol{K}}_{\boldsymbol{XX}}$ and $\boldsymbol{D}_{\boldsymbol{X}}$ for the sake of brevity. This bound can be used to learn the parameters $\boldsymbol{\theta}$, similarly to Eq. (2) for full GP.

The special case $\alpha \to 0$ was originally introduced in [36] where the author proposed to maximize a variational lower bound to the true GP marginal likelihood, obtaining the *Variational Free Energy* (VFE) or the *collapsed lower bound*

$$\mathscr{L}_{VFE}(\boldsymbol{\theta}) = \log \mathcal{N}\left(\boldsymbol{y} | \boldsymbol{0}, \boldsymbol{Q}_{\boldsymbol{XX}} + \sigma_n^2 \mathbb{I} \right) - \frac{\mathrm{Tr}[\boldsymbol{D}_{\boldsymbol{X}}]}{2\sigma_n^2}. \tag{5}$$

In (5) the variational distribution over the inducing points is optimally eliminated and analytically available. The rightmost term in (5) acts as a regularizer that prevents overfitting and has the effect that the sparse GP predictive distribution (3) converges [36] to the exact GP predictive distribution (1) as the number of inducing points increases, when optimizing $\boldsymbol{\theta}$ with (5). See also [7,20,27,28] for recent reviews on the subject.

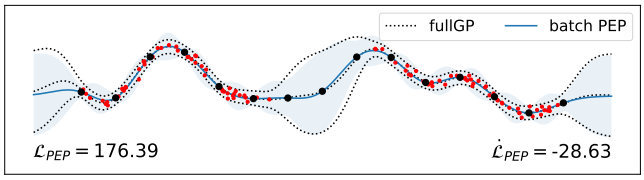

Fig. 1. Full GP and batch sparse GP regression with PEP model ($\alpha = 0.5$). $N = 100$ data samples are summarized with 15 equidistant inducing points (black dots). A slightly smaller than optimal lengthscale was selected and no parameters $\boldsymbol{\theta}$ were optimized. The numbers in the left and right corner indicate the lower bound to the log marginal likelihood in (4) and its derivative with respect to the lengthscale, respectively.

2.2 Sequential Sparse GP Regression

The optimization for $\boldsymbol{\theta}$ of the collapsed lower bound (5) requires to process the whole dataset, which is very inefficient and not feasible for large N. We would like to update the parameters more frequently, therefore, we split the data $\mathscr{D} = \{\boldsymbol{y}_k, \boldsymbol{X}_k\}_{k=1}^K$ into K mini-batches of size B and denote $\boldsymbol{f}_k$ the corresponding sparse GP value. *Stochastic Variational Gaussian Process* (SVGP) [15] achieves this result by applying stochastic optimization to an *uncollapsed lower bound* to the log marginal likelihood

$$\begin{aligned} \mathscr{L}_{SVGP}(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\theta}) = &-\mathrm{KL}\left[\boldsymbol{q}(\boldsymbol{u}) || p(\boldsymbol{u} | \boldsymbol{\theta}) \right] \\ &+ K \sum_{k=1}^{K} \int \boldsymbol{q}(\boldsymbol{u}) p(\boldsymbol{f}_k | \boldsymbol{u}, \boldsymbol{\theta})\, p(\boldsymbol{y}_k | \boldsymbol{f}_k, \boldsymbol{\theta})\, \mathrm{d}\boldsymbol{u}, \end{aligned} \tag{6}$$

where the variational distribution $q(\boldsymbol{u})$ is part of the bound and explicitly parametrized as $\boldsymbol{q}(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u} | \boldsymbol{\mu}, \boldsymbol{\Sigma})$. This uncollapsed bound satisfies $\mathscr{L}_{SVGP}(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\theta}) \leq \mathscr{L}_{VFE}(\boldsymbol{\theta})$ with equality when inserting the optimal mean and covariance of the variational distribution of VFE. The key property of this bound is that it can be written as a sum of K terms, which allows *Stochastic Variational Inference* (SVI, [17]). Note that collapsing the bound, i.e. inserting the optimal distribution, reintroduces dependencies between the observations, and eliminates the global parameter $\boldsymbol{u}$ which is needed for SVI. For this reason, all variational parameters are numerically estimated by following the noisy gradients of a stochastic estimate of the lower bound $\mathscr{L}_{SVGP}$. By passing through the training data a sufficient number of times, the variational distribution converges to the batch solution of VFE method. This approach, however, requires a large number of parameters: in addition to the parameters $\boldsymbol{\theta}$, all entries in the mean vector $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$ have to be estimated numerically, which is in order $\mathcal{O}(M^2)$.

3 Recursive Sparse GP Regression

In this section we establish the connection between Bayesian recursive estimation and sparse inducing point GP models. We recall the *weight-space view* for a large class of sparse inducing point GP models, which we present here as a particular kernelized version of a Bayesian linear regression model. See [28, Ch. 2.1], for an analogous discussion on the full GP model. Here, however, we show how to ex-

ploit the KF to train many sparse methods analytically either in an online setting for fixed hyper-parameters. This allows us to introduce a recursive log marginal likelihood with a model specific regularization term for parameter estimation.

3.1 Weight-Space View of Generic Sparse GP

For a mini-batch $\boldsymbol{X} \in \mathbb{R}^{B\times D}$ of size B, consider the generic sparse GP model

$$f(\boldsymbol{X}) = H(\boldsymbol{X})\,\boldsymbol{u} + \gamma(\boldsymbol{X}) \tag{7}$$

where the sparse GP value $f(\boldsymbol{X})$ is modeled by a linear combination of basis-functions $H(\boldsymbol{X}) \in \mathbb{R}^{B\times M}$, (stochastic) weights $\boldsymbol{u} \in \mathbb{R}^M$ with a prior $p(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_0)$ and an input dependent error term $\gamma(\boldsymbol{X}) \sim \mathcal{N}(0, V(\boldsymbol{X}))$ that takes into account the sparse approximation. For $k = 1,\dots,K$, the noisy observations $\boldsymbol{y}_k$ are obtained by adding independent noise $\varepsilon_k \sim \mathcal{N}\left(0, \sigma_n^2\mathbb{I}\right)$ to $f(\boldsymbol{X}_k)$, yielding the model

$$\boldsymbol{y}_k = \boldsymbol{f}_k + \varepsilon_k; \tag{8}$$

$$\boldsymbol{f}_k = \boldsymbol{H}_k\boldsymbol{u} + \gamma_k \quad \text{and} \quad \boldsymbol{f}_* = \boldsymbol{H}_*\boldsymbol{u} + \gamma_*, \tag{9}$$

where we distinguish the training $\boldsymbol{f}_k = f(\boldsymbol{X}_k)$ and test $\boldsymbol{f}_* = f(\boldsymbol{X}_*)$ cases depending on the input $\boldsymbol{X}_k$ and $\boldsymbol{X}_*$. Assuming γ_k, γ_* and ε_k are independent, by linearity and Gaussianity we can compactly write

$$p(\boldsymbol{y}_k|\boldsymbol{f}_k) = \mathcal{N}\left(\boldsymbol{y}_k|\boldsymbol{f}_k, \sigma_n^2\mathbb{I}\right); \tag{10}$$

$$p(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_0); \tag{11}$$

$$p(\boldsymbol{f}_k|\boldsymbol{u}) = \mathcal{N}\left(\boldsymbol{f}_k|\boldsymbol{H}_k\boldsymbol{u}, \overline{\boldsymbol{V}}_k\right); \tag{12}$$

$$p(\boldsymbol{f}_*|\boldsymbol{u}) = \mathcal{N}(\boldsymbol{f}_*|\boldsymbol{H}_*\boldsymbol{u}, \boldsymbol{V}_*), \tag{13}$$

Combining (10) and (12) and by integrating out $\boldsymbol{f}_k$ we obtain the likelihood $p(\boldsymbol{y}_k|\boldsymbol{u}) = \mathcal{N}(\boldsymbol{y}_k|\boldsymbol{H}_k\boldsymbol{u}, \boldsymbol{V}_k)$ where $\boldsymbol{V}_k = \overline{\boldsymbol{V}}_k + \sigma_n^2\mathbb{I}$. This shows that a generic sparse GP regression model can be seen as a Bayesian non-linear regression model with additional input dependent observation noise, a particular choice of basis functions $\boldsymbol{H}_k$ and covariance structures $\overline{\boldsymbol{V}}_k$, $\boldsymbol{V}_*$. For inducing inputs $\boldsymbol{R} \in \mathbb{R}^{M\times D}$, we have $\boldsymbol{\Sigma}_0 = \boldsymbol{K_{RR}}$, $\boldsymbol{H}_k = \boldsymbol{K_{X_kR}}\boldsymbol{K_{RR}^{-1}}$ and $\boldsymbol{H}_* = \boldsymbol{K_{*R}}\boldsymbol{K_{RR}^{-1}}$. Different choices of the quantities $\overline{\boldsymbol{V}}_k$ and $\boldsymbol{V}_*$ lead to a range of sparse GP models summarized in the bottom table in Fig. 2.

3.2 Training

Given the prior $p(\boldsymbol{u})$ and the likelihood $p(\boldsymbol{y}_k|\boldsymbol{u})$, the posterior over the weights $\boldsymbol{u}$ conditioned on the data $\boldsymbol{y}_{1:k}$ can be computed either in a batch or in a recursive manner.

3.2.1 Batch Estimation

The batch likelihood is $p(\boldsymbol{y}|\boldsymbol{u}) = \prod_{k=1}^K p(\boldsymbol{y}_k|\boldsymbol{u}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{H}\boldsymbol{u}, \boldsymbol{V})$ with $\boldsymbol{H} = \left[\boldsymbol{H}_1^T, \dots, \boldsymbol{H}_K^T\right]^T \in \mathbb{R}^{N\times M}$ and $\boldsymbol{V}$ a block-diagonal matrix with blocks $\boldsymbol{V}_k$. The posterior over $\boldsymbol{u}$ given the data $\boldsymbol{y}$ can be obtained by Bayes' rule, i.e.

$$p(\boldsymbol{u}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{u})\,p(\boldsymbol{u}) \propto \mathcal{N}(\boldsymbol{u}|\boldsymbol{\mu}_K, \boldsymbol{\Sigma}_K), \tag{14}$$

with $\boldsymbol{\Sigma}_K = \left(\boldsymbol{\Sigma}_0^{-1} + \boldsymbol{H}^T\boldsymbol{V}^{-1}\boldsymbol{H}\right)^{-1}$ and $\boldsymbol{\mu}_K = \boldsymbol{\Sigma}_K\boldsymbol{H}^T\boldsymbol{V}^{-1}\boldsymbol{y}$.

a) parameters

Σ_0	H_k	H_*
K_{RR}	$K_{X_kR}K_{RR}^{-1}$	$K_{*R}K_{RR}^{-1}$

b) transformed parameters

$\tilde{\Sigma}_0$	$\tilde{H}_k$	$\tilde{H}_*$
K_{RR}^{-1}	K_{X_kR}	K_{*R}

	I) training	II) prediction	III) optimization
	$\overline{V}_k$	V_*	a_k
DIC [31]	0	0	0
DTC [9]	0	D_*	0
FITC [32]	$Diag[D_{X_k}]$	D_*	0
FIC [26]	$Diag[D_{X_k}]$	$Diag[D_*]$	0
PITC [26]	D_{X_k}	D_*	0
VFE [34]	0	D_*	$\frac{1}{2\sigma_n^2}Tr[D_{X_k}]$
PEP [6]	$\alpha Diag[D_{X_k}]$	D_*	$\frac{1-\alpha}{2\alpha}\sum_i \log(1+\frac{\alpha}{\sigma_n^2}[D_{X_k}]_{ii})$
PEP_B	αD_{X_k}	D_*	$\frac{1-\alpha}{2\alpha}\log\lvert I+\frac{\alpha}{\sigma_n^2}D_{X_k}\rvert$

Fig. 2. Summary of parameters for sparse GP models for recursive estimation. For all models, we have $\boldsymbol{\mu}_0 = \boldsymbol{0}$, $\boldsymbol{V}_k = \overline{\boldsymbol{V}}_k + \sigma_n^2\mathbb{I}$ and $\boldsymbol{\Sigma}_0$, $\boldsymbol{H}_k$, $\boldsymbol{H}_*$ from the table a) or a transformed version b). Using the model specific quantities for the observation noise $\overline{\boldsymbol{V}}_k$, the prediction covariance $\boldsymbol{V}_*$ and the regularization term a_k from the bottom table allows the training with the recursive approaches.

3.2.2 Recursive Estimation

An equivalent solution can be obtained by propagating recursively $p(\boldsymbol{u}|\boldsymbol{y}_{1:k-1})$. By interpreting this previous posterior as the prior, the updated posterior can be recursively computed by

$$p(\boldsymbol{u}|\boldsymbol{y}_{1:k}) \propto p(\boldsymbol{y}_k|\boldsymbol{u})\,p(\boldsymbol{u}|\boldsymbol{y}_{1:k-1}) \propto \mathcal{N}(\boldsymbol{u}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{15}$$

where $\boldsymbol{\mu}_k = \boldsymbol{\Sigma}_k\left(\boldsymbol{H}_k^T\boldsymbol{V}_k^{-1}\boldsymbol{y}_k + \boldsymbol{\Sigma}_{k-1}^{-1}\boldsymbol{\mu}_{k-1}\right)$ and $\boldsymbol{\Sigma}_k = \left(\boldsymbol{\Sigma}_{k-1}^{-1} + \boldsymbol{H}_k^T\boldsymbol{V}_k^{-1}\boldsymbol{H}_k\right)^{-1}$.

Kalman Filter like updating: the KF constitutes an efficient way to update the mean and covariance of $p(\boldsymbol{u}|\boldsymbol{y}_{1:k})$. Applying the Woodbury identity to $\boldsymbol{\Sigma}_k$ in Eq. (15) and introducing temporary variables yields

$$\begin{aligned} \boldsymbol{r}_k &= \boldsymbol{y}_k - \boldsymbol{H}_k\boldsymbol{\mu}_{k-1}; \\ \boldsymbol{S}_k &= \boldsymbol{H}_k\boldsymbol{\Sigma}_{k-1}\boldsymbol{H}_k^T + \boldsymbol{V}_k; \\ \boldsymbol{G}_k &= \boldsymbol{\Sigma}_{k-1}\boldsymbol{H}_k^T\boldsymbol{S}_k^{-1}; \end{aligned} \qquad \begin{aligned} \boldsymbol{\mu}_k &= \boldsymbol{\mu}_{k-1} + \boldsymbol{G}_k\boldsymbol{r}_k; \\ \boldsymbol{\Sigma}_k &= \boldsymbol{\Sigma}_{k-1} - \boldsymbol{G}_k\boldsymbol{S}_k\boldsymbol{G}_k^T. \end{aligned} \tag{16}$$

Starting the recursion with $\boldsymbol{\mu}_0 = \boldsymbol{0}$ and $\boldsymbol{\Sigma}_0$, the posterior distribution at step K is equivalent to (14) independent of the order of the data. We want to emphasize that the only difference in the estimation part between the sparse GP models is the form of the additional noise $\boldsymbol{V}_k = \overline{\boldsymbol{V}}_k + \sigma_n^2\mathbb{I}$.

Transformation: instead of running a KF with $\boldsymbol{\Sigma}_0 = \boldsymbol{K_{RR}}$, $\boldsymbol{H}_k = \boldsymbol{K_{X_kR}}\boldsymbol{K_{RR}^{-1}}$ and $\boldsymbol{H}_* = \boldsymbol{K_{*R}}\boldsymbol{K_{RR}^{-1}}$, an equivalent predictive distribution is also obtained when using $\tilde{\boldsymbol{\Sigma}}_0 = \boldsymbol{K_{RR}^{-1}}$ and $\tilde{\boldsymbol{H}}_k = \boldsymbol{K_{X_kR}}$ together with $\tilde{\boldsymbol{H}}_* = \boldsymbol{K_{*R}}$. For any k we then propagate a transformed posterior distribution $\tilde{\boldsymbol{\mu}}_k = \boldsymbol{K_{RR}^{-1}}\boldsymbol{\mu}_k$, $\tilde{\boldsymbol{\Sigma}}_k = \boldsymbol{K_{RR}^{-1}}\boldsymbol{\Sigma}_k\boldsymbol{K_{RR}^{-1}}$ and $\boldsymbol{\mu}_k = \boldsymbol{K_{RR}}\tilde{\boldsymbol{\mu}}_k$, $\boldsymbol{\Sigma}_k =$

$\boldsymbol{K_{RR}}\tilde{\boldsymbol{\Sigma}}_k\boldsymbol{K_{RR}}$, respectively. This parametrization constitutes a computational shortcut, since the basis functions are very easy to interpret and do not include any matrix multiplication. Note that also the log marginal likelihood discussed below is not affected by this transformation.

3.3 Prediction

Given a new $\boldsymbol{X}_* \in \mathbb{R}^{A\times D}$, the predictive distribution after seeing $\boldsymbol{y}_{1:k}$ of the sparse GP methods can be obtained by

$$\begin{aligned} p(\boldsymbol{f}_*|\boldsymbol{y}_{1:k}) &= \int p(\boldsymbol{f}_*|\boldsymbol{u})\, p(\boldsymbol{u}|\boldsymbol{y}_{1:k})\, \mathrm{d}\boldsymbol{u} \\ &= \mathcal{N}\left(\boldsymbol{f}_*|\boldsymbol{H}_*\boldsymbol{\mu}_k, \boldsymbol{H}_*\boldsymbol{\Sigma}_k\boldsymbol{H}_*^T + \boldsymbol{V}_*\right) \end{aligned} \tag{17}$$

using $p(\boldsymbol{f}_*|\boldsymbol{u}) = \mathcal{N}(\boldsymbol{f}_*|\boldsymbol{H}_*\boldsymbol{u}, \boldsymbol{V}_*)$ with $\boldsymbol{H}_* = \boldsymbol{K}_{*\boldsymbol{R}}\boldsymbol{K}_{\boldsymbol{RR}}^{-1}$ and $\boldsymbol{V}_*$ the model specific prediction covariance. The predictions for $\boldsymbol{y}^*$ are obtained by adding $\sigma_n^2\mathbb{I}$ to the covariance of $\boldsymbol{f}_*|\boldsymbol{y}_{1:k}$. At step K, by applying the Woodbury identity to the batch covariance $\boldsymbol{\Sigma}_K$ in (14), we get for the predictive distribution in (17)

$$\begin{aligned} \boldsymbol{\mu}_K^* &= \boldsymbol{H}_*\boldsymbol{\Sigma}_0\boldsymbol{H}^T\boldsymbol{\Sigma}\boldsymbol{y}, \\ \boldsymbol{\Sigma}_K^* &= \boldsymbol{H}_*\boldsymbol{\Sigma}_0\boldsymbol{H}_*^T - \boldsymbol{H}_*\boldsymbol{\Sigma}_0\boldsymbol{H}^T\boldsymbol{\Sigma}\boldsymbol{H}\boldsymbol{\Sigma}_0\boldsymbol{H}_*^T + \boldsymbol{V}_*, \end{aligned} \tag{18}$$

where $\boldsymbol{\Sigma} = \left(\boldsymbol{H}\boldsymbol{\Sigma}_0\boldsymbol{H}^T + \boldsymbol{V}\right)^{-1}$. Inserting the particular choices for $\boldsymbol{\Sigma}_0$, $\boldsymbol{H}$ and $\boldsymbol{H}_*$ yields the usual formulation for the sparse predictive distribution

$$\begin{aligned} \boldsymbol{\mu}_K^* &= \boldsymbol{Q}_{*\boldsymbol{X}}\left(\boldsymbol{Q}_{\boldsymbol{XX}} + \boldsymbol{V}\right)^{-1}\boldsymbol{y}; \\ \boldsymbol{\Sigma}_K^* &= \boldsymbol{Q}_{**} - \boldsymbol{Q}_{*\boldsymbol{X}}\left(\boldsymbol{Q}_{\boldsymbol{XX}} + \boldsymbol{V}\right)^{-1}\boldsymbol{Q}_{\boldsymbol{X}*} + \boldsymbol{V}_*. \end{aligned} \tag{19}$$

Depending on the choice of the covariances $\overline{\boldsymbol{V}}$ and $\boldsymbol{V}_*$, we obtain for instance (3) for PEP, or the analogous predictions for VFE and FITC, respectively.

3.4 Online learning

This connection between sparse GP models and recursive estimation allows us to train the sparse GP models analytically online for streaming data for fixed $\boldsymbol{\theta}$.

As an illustrative example consider $N = 100$ data samples in $D = 1$, we are interested in training a PEP model with $\alpha = 0.5$ with $M = 15$ inducing points with fixed $\boldsymbol{\theta}$. Let's assume that the data samples $\{\boldsymbol{x}_k, y_k\}$ arrive sequentially in a stream. Thus we have $B = 1$ and $K = 100$. Here we use the transformation and we apply, for each data sample k, the recursion in (16). We note that here $\tilde{\boldsymbol{r}}_k$, $\tilde{\boldsymbol{V}}_k$ and $\tilde{\boldsymbol{S}}_k$ are numbers as $B = 1$ and $\tilde{\boldsymbol{H}}_k, \tilde{\boldsymbol{G}}_k \in \mathbb{R}^{15}$.

We obtain predictions for new data $\boldsymbol{X}_*$, by applying (17) with $\tilde{\boldsymbol{H}}_* = \boldsymbol{K}_{*\boldsymbol{R}}$ and $\boldsymbol{V}_* = \boldsymbol{K}_{*\boldsymbol{R}}\boldsymbol{K}_{\boldsymbol{RR}}^{-1}\boldsymbol{K}_{\boldsymbol{R}*}$. Note that there is no need to transform back the posterior over the inducing points, since it is already taken into account in the prediction step. After processing all N samples, the predictive distribution and the cumulative bound of log marginal likelihood correspond to the batch version, as shown in Fig. 3.

3.5 Marginal Likelihood

In the batch setting, the log marginal likelihood $\log p(\boldsymbol{y})$ can be computed by marginalizing out $\boldsymbol{u}$, that is

$$\begin{aligned} \log p(\boldsymbol{y}) &= \log \int p(\boldsymbol{y}|\boldsymbol{u})\, p(\boldsymbol{u})\, \mathrm{d}\boldsymbol{u} \\ &= \log \mathcal{N}\left(\boldsymbol{y}|\boldsymbol{0}, \boldsymbol{H}\boldsymbol{\Sigma}_0\boldsymbol{H}^T + \boldsymbol{V}\right). \end{aligned} \tag{20}$$

In the recursive setting, $p(\boldsymbol{y})$ can be factorized into $\prod_{k=1}^K p(\boldsymbol{y}_k|\boldsymbol{y}_{1:k-1})$, where

$$\begin{aligned} p(\boldsymbol{y}_k|\boldsymbol{y}_{1:k-1}) &= \mathcal{N}\left(\boldsymbol{y}_k|\boldsymbol{H}_k\boldsymbol{\mu}_{k-1}, \boldsymbol{H}_k\boldsymbol{\Sigma}_{k-1}\boldsymbol{H}_k^T + \boldsymbol{V}_k\right) \\ &= \mathcal{N}(\boldsymbol{r}_k|\boldsymbol{0}, \boldsymbol{S}_k). \end{aligned} \tag{21}$$

The log of the joint marginal likelihood involving all terms of (21) can be explicitly written as

$$\begin{aligned} \log \prod_{k=1}^K p(\boldsymbol{y}_k|\boldsymbol{y}_{1:k-1}) &= \sum_{k=1}^K \log \mathcal{N}(\boldsymbol{r}_k|\boldsymbol{0}, \boldsymbol{S}_k) \\ = -\frac{N}{2}\log 2\pi - \frac{1}{2}\sum_{k=1}^K \log|\boldsymbol{S}_k| &+ \boldsymbol{r}_k^T\boldsymbol{S}_k^{-1}\boldsymbol{r}_k. \end{aligned} \tag{22}$$

The iterative maximization of a lower bound of the recursive factorized marginal likelihood in (21) leads to the recursive KF updates in (16) for the posterior and to the lower bound

$$\psi(\boldsymbol{\theta}) = \sum_{k=1}^K \log \mathcal{N}\left(\boldsymbol{r}_k^{\boldsymbol{\theta}}|\boldsymbol{0}, \boldsymbol{S}_k^{\boldsymbol{\theta}}\right) - a_k(\boldsymbol{\theta}) \tag{23}$$

which includes a model specific regularization term a_k (see the right-most column in Fig. 2). We refer to this as the *recursive collapsed bound* and a detailed derivation for the VFE model is given in App. B. Using the model specific quantities $\overline{\boldsymbol{V}}_k$ and a_k, this recursive computation of the lower bound of the marginal likelihood are equivalent to the batch counterparts for all sparse models, for instance (5) and (4) for VFE and PEP, respectively.

4 Hyper-parameters Estimation

The previous section presented an online procedure for training sparse GP models at fixed hyper-parameters $\boldsymbol{\theta}$. Here we show that, by exploiting the connections highlighted before, we can optimize $\boldsymbol{\theta}$ sequentially. The *recursive collapsed bound* in (23) decomposes into a recursive sum over the mini-batches which allows to optimize the hyper-parameters $\boldsymbol{\theta}$ sequentially as opposed to the collapsed bound (5). Our bound (23) enables the application of stochastic optimization without needing to estimate all entries in the posterior mean vector and covariance matrix as in SVGP. Compared to the uncollapsed bound in (6), the variational distribution is recursively and analytically eliminated, thus reducing the number of parameters to be numerically estimated drastically from $\mathcal{O}(MD + M^2)$ to $\mathcal{O}(MD)$.

Finding a maximizer $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ of an objective function $\Psi(\boldsymbol{\theta}) = \sum_{k=1}^K \psi_k(\boldsymbol{\theta})$ can be achieved by applying *Stochastic*

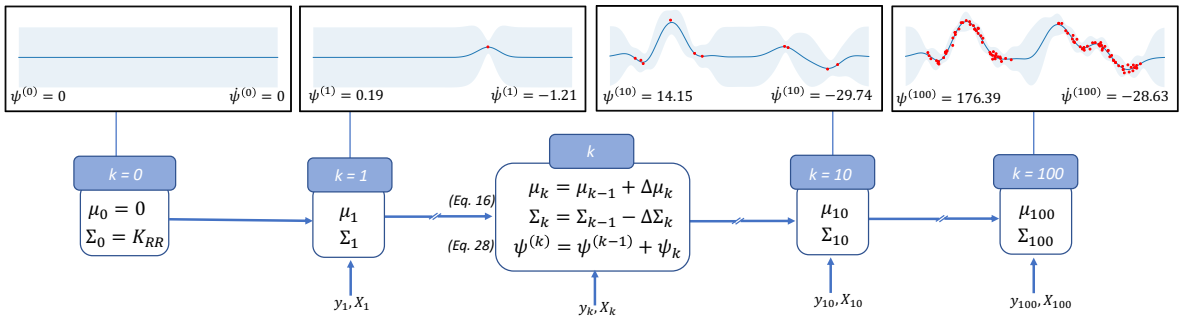

Fig. 3. Online learning for the toy example in Sect. 3.4 for fixed $\boldsymbol{\theta}$ with batch size $B = 1$. In each step k, the sample $\boldsymbol{y}_k, \boldsymbol{X}_k$ is updated to the current posterior represented by $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ according to Eq. (16) together with the cumulative recursive bound $\psi^{(k)}$ (Eq. (25) and the numbers in the left corners of the plots). For recursive parameter estimation as discussed in Sect. 4.1, in addition to the posterior and the cumulative bound, the recursive derivatives are propagated for each k. The derivatives of this bound w.r.t. to the lengthscale are indicated in the right corners of the plots. The cumulative bound and its derivative at step $k = 100$ are equal to the corresponding batch version in Fig. 1.

Gradient Descent(SGD), with the update

$$\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)} - \gamma^{(t-1)} \frac{\partial \psi_k}{\partial \boldsymbol{\theta}}_{|\boldsymbol{\theta}=\boldsymbol{\theta}^{(t-1)}}, \tag{24}$$

where $\gamma^{(t-1)}$ might be a sophisticated function of $\boldsymbol{\theta}^{(0)}, \ldots, \boldsymbol{\theta}^{(t-1)}$ (for instance using ADAM [18], where also a bias correction term is included). We call one pass over the K mini-batches an epoch. We denote $\boldsymbol{\theta}^{(e,k)} \in \boldsymbol{\Theta}^{(e,k)}$ the estimate of $\boldsymbol{\theta}$ in epoch $e \in E$ for mini-batch k.

4.1 Recursive Gradient Propagation (RGP)

We rewrite the *recursive collapsed bound* in (23) as

$$\psi^{(K)}(\boldsymbol{\theta}) = \sum_{k=1}^{K} d_k(\boldsymbol{\theta}) - a_k(\boldsymbol{\theta}) = \sum_{k=1}^{K} \psi_k(\boldsymbol{\theta}) \tag{25}$$

where $d_k(\boldsymbol{\theta}) = \log \mathcal{N}\left(\boldsymbol{r}_k^{\boldsymbol{\theta}} | \boldsymbol{0}, \boldsymbol{S}_k^{\boldsymbol{\theta}}\right)$ and $a_k(\boldsymbol{\theta})$ the model specific regularization term. Since $\psi^{(K)}(\boldsymbol{\theta})$ decomposes into a (recursive) sum over the mini-batches, we directly compute the derivative of $\psi_k(\boldsymbol{\theta})$ w.r.t. $\boldsymbol{\theta} \in \boldsymbol{\Theta}$. The derivative of a_k is straightforward, for d_k we have

$$\frac{\partial d_k(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -\frac{1}{2} \frac{\partial \log \left|\boldsymbol{S}_k^{\boldsymbol{\theta}}\right|}{\partial \boldsymbol{\theta}} - \frac{1}{2} \frac{\partial (\boldsymbol{r}_k^{\boldsymbol{\theta}})^T (\boldsymbol{S}_k^{\boldsymbol{\theta}})^{-1} \boldsymbol{r}_k^{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}} \tag{26}$$

with $\boldsymbol{r}_k^{\boldsymbol{\theta}} = \boldsymbol{y}_k - \boldsymbol{H}_k^{\boldsymbol{\theta}} \boldsymbol{\mu}_{k-1}^{\boldsymbol{\theta}}$ and $\boldsymbol{S}_k^{\boldsymbol{\theta}} = \boldsymbol{H}_k^{\boldsymbol{\theta}} \boldsymbol{\Sigma}_{k-1}^{\boldsymbol{\theta}} (\boldsymbol{H}_k^{\boldsymbol{\theta}})^T + \boldsymbol{V}_k^{\boldsymbol{\theta}}$. It is important to note that ignoring naively the dependency of $\boldsymbol{\theta}$ through $\boldsymbol{\mu}_{k-1}^{\boldsymbol{\theta}}$ and $\boldsymbol{\Sigma}_{k-1}^{\boldsymbol{\theta}}$ completely forgets the past and thus results in overfitting the current mini-batch. In order to compute the derivatives of $\boldsymbol{\mu}_k^{\boldsymbol{\theta}}$ and $\boldsymbol{\Sigma}_k^{\boldsymbol{\theta}}$, we exploit the chain rule for derivatives and recursively propagate the gradients of the mean and the covariance over time, that is

$$\begin{aligned} \frac{\partial \boldsymbol{u}_k}{\partial \theta} &= \frac{\partial \boldsymbol{u}_{k-1}}{\partial \theta} + \frac{\partial \boldsymbol{G}_k}{\partial \theta} \boldsymbol{r}_k + \boldsymbol{G}_k \frac{\partial \boldsymbol{r}_k}{\partial \theta} \\ \frac{\partial \boldsymbol{\Sigma}_k}{\partial \theta} &= \frac{\partial \boldsymbol{\Sigma}_{k-1}}{\partial \theta} - \frac{\partial \boldsymbol{G}_k}{\partial \theta} \boldsymbol{S}_k \boldsymbol{G}_k^T - \boldsymbol{G}_k \frac{\partial \boldsymbol{S}_k}{\partial \theta} \boldsymbol{G}_k - \boldsymbol{G}_k \boldsymbol{S}_k \frac{\partial \boldsymbol{G}_k^T}{\partial \theta}, \end{aligned} \tag{27}$$

where $\frac{\partial \boldsymbol{G}_k}{\partial \theta}$, $\frac{\partial \boldsymbol{r}_k}{\partial \theta}$ and $\frac{\partial \boldsymbol{S}_k}{\partial \theta}$ are computed recursively according to (16). Computing the derivatives of d_k as explained in (26) and (27), the stochastic gradient

$$\frac{\partial \psi_k(\boldsymbol{\theta})}{\partial \theta} = \frac{\partial d_k(\boldsymbol{\theta})}{\partial \theta} - \frac{\partial a_k(\boldsymbol{\theta})}{\partial \theta} \tag{28}$$

can be computed for each mini-batch k.

Proposition 1 *Consider a fixed parameter vector $\bar{\boldsymbol{\theta}} \in \boldsymbol{\Theta}$, the gradients of the full batch lower bound $\mathscr{L}_{PEP}(\bar{\boldsymbol{\theta}})$ in (4) with respect to $\boldsymbol{\theta}$ are equal to the cumulative partial derivatives $\frac{\partial \psi^{(K)}}{\partial \boldsymbol{\theta}}$ of the recursive collapsed bound with the above learning procedure. That is, it holds $\frac{\partial \mathscr{L}_{PEP}(\bar{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}} = \frac{\partial \psi^{(K)}(\bar{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}}$.*

This shows that, when the gradients are cumulated over all data samples, each gradient step of our recursive procedure is equivalent to a gradient step for $\mathscr{L}_{PEP}$ and, therefore, follows the gradients of an optimal collapsed lower bound.

The toy example in Fig. 3 also shows this equivalence. The numbers in the bottom left and right corners show the cumulative recursive collapsed bound $\psi^{(k)}$ and its cumulative derivative $\frac{\partial \psi^{(k)}}{\partial l}$ (abbreviated as $\dot{\psi}^{(k)}$) with respect to the lengthscale. The lower bound of the marginal likelihood as well as its derivatives are exactly the same value as the corresponding batch counterpart in Fig. 1.

For *Stochastic Recursive Gradient Propagation* (SRGP), in each epoch e and mini-batch k, we interleave the update step of the inducing points in Eq. (15) with the SGD update (24) of the parameters $\boldsymbol{\theta}^{(e,k)}$, i.e.

$$p\left(\boldsymbol{u} | \boldsymbol{y}_{1:k}, \boldsymbol{\theta}^{(e,k)}\right) \propto p\left(\boldsymbol{y}_k | \boldsymbol{u}, \boldsymbol{\theta}^{(e,k)}\right) p\left(\boldsymbol{u} | \boldsymbol{y}_{1:k-1}, \boldsymbol{\theta}^{(e,k-1)}\right).$$

More concretely, we update after each mini-batch k the parameters $\boldsymbol{\theta}^{(e,k)}$ with (28),(24) and propagate recursively the posterior with (16) and its derivative (27). In order to compute all the derivatives with respect to $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, we exploit several matrix derivative rules which simplify the computation significantly, see App. A. Finally note that the form

of the gradients in eq. (26) and (28) implies that the noise in the stochastic gradient at step k depends on the noise at step k-1, thus excluding standard convergence proofs such as [5]. Recent results on non-convex optimization problems [10,40,41] show convergence proofs for function classes that include our objective, nonetheless the Markov noise of our problem still excludes it from this general theory. For this reason we leave a convergence analysis to future work.

4.2 Computational complexity

In the following, we assume that the batch size B is larger than the number of inducing points M. For one mini-batch, the time complexity to update the posterior is dominated by matrix multiplications of size B and M, thus $\mathcal{O}(B^2M)$. In order to propagate the gradients of the posterior and to compute the derivative of the bound needs $\mathcal{O}(BM^2)$ for a mini-batch and a parameter $\boldsymbol{\theta} \in \boldsymbol{\Theta}$. Thus, updating a mini-batch including all $\mathcal{O}(MD)$ parameters costs $\mathcal{O}(BM^3D + B^2M)$ for the SRGP method. Since SRGP stores the gradients of the posterior, it requires $\mathcal{O}(M^3D + BM)$ storage.
On the other hand, SVGP needs $\mathcal{O}(M^2 + BM)$ storage and $\mathcal{O}(BM^3 + B^2M)$ time per mini-batch, where the latter can be broken down into once $\mathcal{O}(B^2M)$ and $\mathcal{O}(BM)$ for each of the $\mathcal{O}(M^2)$ parameters. This means, for moderate dimensions, our algorithm has the same time complexity as state of the art method SVGP. However, due to the analytic updates of the posterior we achieve an higher accuracy and less epochs are needed as shown in Fig. 4 and in Sect. 5 empirically.

Fig. 4 shows the convergence of SRGP on a 1-D toy example with $N = 1000$ data samples and $M = 15$ inducing points. The parameters are sequentially optimized with our recursive approach (blue) and as comparison with SVGP (green) with a mini-batch size of $B = 100$ over several epochs. The root-mean-squared-error (RMSE) computed on test points, the bound of the log marginal likelihood (LML) as well as the hyper-parameters converge in a few iterations to the corresponding batch values of VFE (red). Due to the analytic updates of the posterior, the accuracy is higher and SRGP needs much less epochs until convergence.

4.3 Mini-batch size

The size of the mini-batches has an impact on the speed of convergence of the algorithm. Proposition 1 tells us that if we use a full batch, our algorithm requires the same number of gradient updates as a full batch method to converge. On the other hand smaller batches should require more updates and should lead to a higher variance in the results. Fig. 5 shows a comparison of different mini-batch sizes on a 1-D toy example with $N = 10'000$ data samples generated with the same parameters as in Sect. 5.1. The convergence to the full batch value is slower as the batch size decreases. Moreover the variance of the error, over the repetitions, is much larger for smaller batch sizes: in the last 10 normalized gradient updates, the standard deviation of the error is on average 3.8×10^{-3} for $B = 100$ and 8.1×10^{-4} for $B = 5'000$, denoting a more stable procedure for higher batch sizes. As the mini-batch size increases the computational cost for each gradient update also increases. In this example one gradient update requires on average 4.9×10^{-3} sec and 5.8×10^{-2} sec with $B = 100$ and $B = 5'000$ respectively. [2] These considerations suggest that a reasonable choice is a large mini-batch size within the computational and time budgets.

5 Experiments

We first benchmark our method with $N = 100'000$ synthetic data samples generated by a GP in several dimensions. Next, we apply our approach to the Airline data used in [15] with a million of data samples. Finally a more realistic setup is presented where we use up to a million data samples to train a nonlinear plant. We compare our SRGP method to full GP and sparse batch method VFE for a subset of data (using the implementation in GPy [13]) and to the state of the art stochastic parameter estimation method SVGP implemented in GPflow [22]. Our algorithm works also for many other sparse models, however, only large-scale implementations of standard SVGP are available (corresponding to the VFE model), thus we restrict the investigation to this model.

5.1 GP Simulation

In this section we test our proposed learning procedure on simulated GP data. We generate $N = 100'000$ data samples from a zero-mean (sparse) GP with SE covariance kernel with hyper-parameters $\sigma_0 = 1, \sigma_n = 0.1$ and $l = \{0.1, 0.2, 0.5\}$ in $D = \{1, 2, 5\}$ dimensions. The initial $M = \{20, 50, 100\}$ inducing points are randomly selected points from the data and the hyper-parameters of a SE kernel with individual lengthscales for each dimension are initialized to the same values for both algorithms ($\sigma_0 = 1, \sigma_n = 1, l_1, \ldots, l_D = 1$). All parameters are sequentially optimized with our recursive approach and with SVGP with a mini-batch size of $B = 5000$. The stochastic gradient descent method ADAM [18] is employed for both methods with learning rates $\{0.001, 0.005, 0.005\}$ for SVGP and $\{0.0001, 0.001, 0.005\}$ for SRGP (based on some preliminary experiments). Each experiment is replicated 10 times.

Fig. 6 shows the bound to the log marginal likelihood, the RMSE and the coverage of $10'000$ test points for the data dimensions $D = \{1, 2, 5\}$ of both methods over 50 epochs. The shaded lines indicates the 10 repetitions and the thick line correspond to the mean. The recursive propagation of the gradients achieves faster convergence and more accurate performance regarding mean RMSE and smaller values for the log marginal likelihood. The higher accuracy and faster convergence can possibly be explained by the analytic updates of the posterior mean and covariance which leads to less parameters to be optimized numerically.

5.2 Airline Data

For the second example we apply our recursive method to the Airline Data used in [15]. It consists of flight arrival and departure times for more than 2 millions flights in the USA from January 2008 and April 2008. We preprocessed

[2] The times are measured on a laptop with a Intel i5-7300U CPU @ 2.6 GHz.

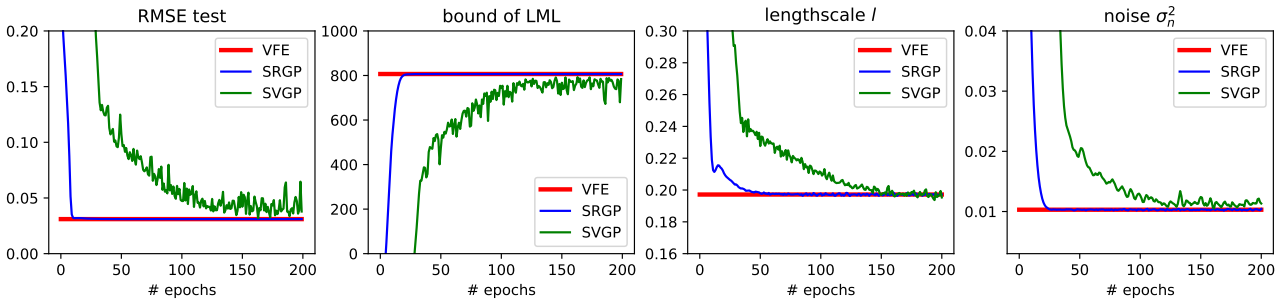

Fig. 4. Convergence of SRGP (blue) on a 1-D toy to batch version VFE (red). Compared to SVGP (green), the convergence of the root-mean-squared-error (RMSE) of test points, the bound of the log marginal likelihood (LML) as well as the hyper-parameters is faster and more accurate.

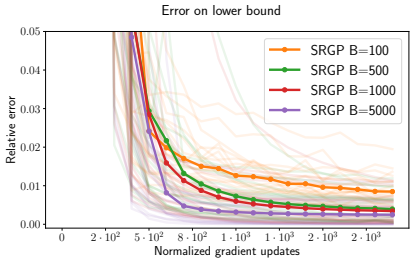

Fig. 5. Relative error in the bound of the log marginal likelihood between full batch and SRGP. Average over 20 repetitions.

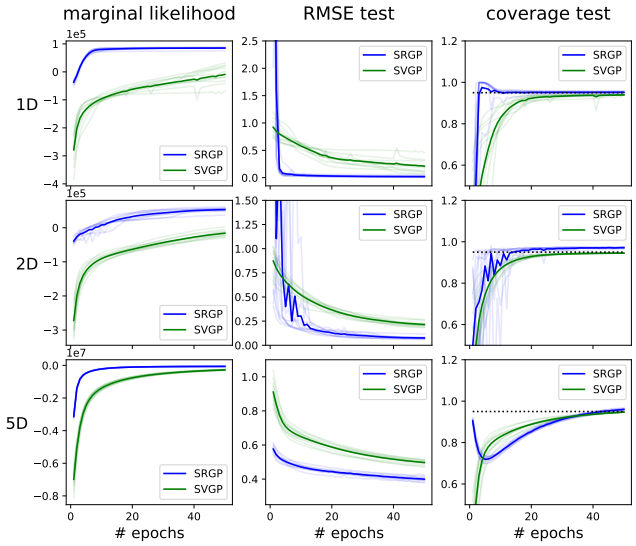

Fig. 6. Convergence over 50 epochs for $N = 100'000$ synthetic GP data samples in several dimensions obtained by SVGP and our proposed method SRGP.

the data as similar as possible as described in [15] resulting in 8 variables: age of the aircraft, distance that needs to be covered, airtime, departure time, arrival time, day of the week, day of the month and month. We trained our recursive method as well as SVGP with an SE kernel on $N = 1'000'000$ data samples with $M = 500$ inducing points randomly selected from the data and a mini-batch size of $B = 10'000$. The ADAM learning rates are set to 0.005 for both methods and the size of the test set is $50'000$. For 5 different repetitions, the RMSE as a function of epochs is depicted in Fig. 7. The mean coverage on test data (at 95%) is comparable for both methods with values of 0.92 and 0.97 for SVGP and SRGP respectively. The overall performance of SRGP is superior to SVGP.

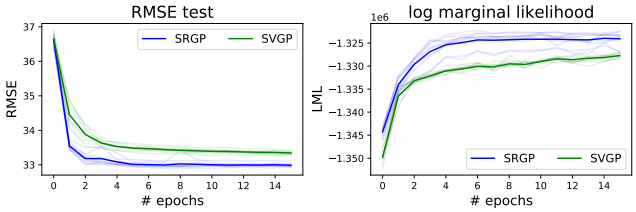

Fig. 7. Convergence over several epochs of RMSE and bound to log marginal likelihood for $N = 1'000'000$ samples from the Airline data for SRGP and SVGP.

5.3 *Non-Linear Plant*

GPs are a powerful way to model complex functions in a non-parametric way, thus they are suitable to learn the complex input output behavior of a non-linear plant. However, with full or even sparse batch GP methods the use is restricted to a few thousands of samples. With our sequential learning method, we are able to exploit the huge amount of available data by training with up to a million of samples.

We consider a Continuous Stirred Tank Reactor (CSTR). The dynamic model of the plant is

$$\frac{d}{dt}h(t) = w_1(t) + w_2(t) - 0.2\sqrt{h(t)}$$
$$\frac{d}{dt}C_b(t) = (C_{b1} - C_b(t))\frac{w_1(t)}{h(t)} + (C_{b2} - C_b(t))\frac{w_2(t)}{h(t)} - \frac{k_1 C_b(t)}{(1+k_2 C_b(t))^2},$$

where $C_b(t)$ is the product concentration at the output of the process, $h(t)$ is the liquid level, $w_1(t)$ is the flow rate of concentrated feed C_{b1}, and $w_2(t)$ is the flow rate of the diluted feed C_{b2}. The input concentrations are $C_{b1} = 24.9$ and $C_{b2} = 0.1$. The constants associated with the rate of consumption are $k_1 = k_2 = 1$. The objective of the controller is to maintain the product concentration by changing the flow $w_1(t)$. To simplify the example, we assume that $w_2(t) = 0.1$ and that the level of the tank $h(t)$ is not controlled. We denote the controlled outputs $C_b(t), C_b(t-1), \ldots, C_b(t-p)$ as $f_t, f_{t-1}, \ldots, f_{t-p}$

and the control variables as $w_t, w_{t-1}, \ldots, w_{t-p}$. Therefore, the plant identification problem can be shaped into the problem of estimating the non-linear function $f_t = g(f_{t-1}, \ldots, f_{t-p}, w_t, w_{t-1}, \ldots, w_{t-p})$ which depends on the p previous values as well as on the current and the p past control values w. However, we can only observe a

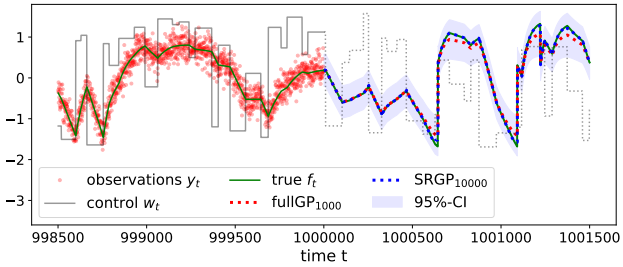

Fig. 8. Training and prediction phases for non-linear plant.

noisy version of the controlled response, that is $y_t = f_t + \varepsilon_t$ with $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$. Using a sampling rate of $0.2s$, we have generated $1'200'000$ observations (about 3 days of observations). The plant input is a series of steps, with random height (in the interval $[0,4]$), occurring at random intervals (in the interval $[5,20]s$). For different numbers N_{train}, we use the samples $y_{10^6-N_{train}}, \ldots, y_{10^6}$ for training and the last $200'000$ are used as a test set. The goal is to learn a model for the controlled response y_t given $\boldsymbol{x}_t = [y_{t-1}, y_{t-2}, w_t, w_{t-1}, w_{t-2}]^T \in \mathbb{R}^5$ for the particular choice of $p = 2$. We model the non-linear function g with a GP with a SE kernel. For comparison, we train full GP and sparse batch GP (with 100 inducing points) on a time horizon N_{train} of up to $10'000$ and $50'000$ past values, respectively. With the sequential version SVGP and our recursive gradient propagation method SRGP (both with 100 inducing points and mini-batch size of $1'000$), we use a time horizon of up to a million. This situation is depicted in Fig. 8, where for 1500 training samples y_t (red dots), the true (unknown) function f_t (green) and the control input w_t (grey) is shown together with the predicted values with full GP (red dotted) and recursive GP (blue dotted) trained on a time horizon of $1'000$ and $10'000$, respectively. In Fig. 9, the RMSE and the median computed on the test set (with 10 repetitions) is depicted for full GP, sparse GP (VFE), SVGP and SRGP trained with varying time horizons. For small and medium training sizes, when the batch methods are applicable, our recursive method achieves the same performance as the batch counterpart (VFE) and is comparable to full GP. Due to the analytic updates of the posterior, SRGP outperforms SVGP regarding both RMSE and median for all training sizes. By exploiting more than several thousand past values, a significant increase in performance of SRGP can be still observed, thus it constitutes an approach to accurately scale GPs up to a million of past values.

6 Conclusion

In this paper we introduced a recursive inference and parameter estimation method SRGP for a general class of sparse GP approximations. Since the posterior updates are

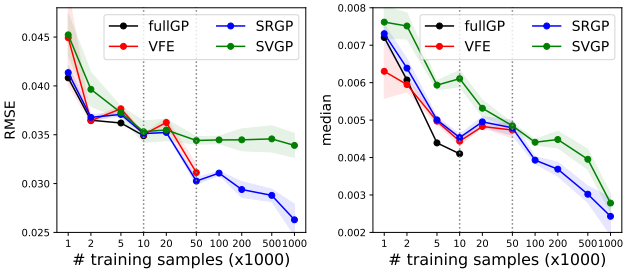

Fig. 9. RMSE and median for full GP, batch sparse GP (VFE), sequential SVGP, and our recursive method (SRGP) trained on varying time horizons (logarithmic scale). The grey dotted vertical lines at $10'000$ and $50'000$ indicate the maximal samples used for training full GP and sparse batch GP (VFE), respectively.

given analytically, one pass through the data is sufficient to compute the posterior for given parameters. For parameter estimation, we proposed a recursive collapsed bound to the log marginal likelihood that matches exactly the batch version but can be used for stochastic estimation. Due to the analytic updates of the posterior our method has much less parameters to be estimated numerically. As a consequence, the experimental section showed that our recursive method needs less epochs and has superior accuracy compared to state of the art, thus constitutes an efficient methodology for scaling GPs to big data problems.

Our approach could be enhanced in several directions. While the proposed method only exploits the update equations of the KF, an interesting direction would be to include a dynamic in a state space model that takes into account the varying hyper-parameters which makes it also applicable for the streaming setting as [6]. Moreover, we further plan to investigate distributed parameter estimation based on an information filter formulation of the problem. Finally, we aim to provide a convergence analysis of the proposed method to recursively learn the hyper-parameters using a similar approach recently employed for ADAM or AMSGrad [29,38].

Acknowledgements

This work is supported by the Swiss National Research Programme 75 "Big Data" grant n. 407540_167199 / 1.

References

[1] David Barber and Yali Wang. Gaussian processes for bayesian estimation in ordinary differential equations. In *ICML*, 2014.

[2] Alessio Benavoli and Marco Zaffalon. State space representation of non-stationary gaussian processes. 2016.

[3] Hildo Bijl, Thomas B. Schön, Jan-Willem van Wingerden, and Michel Verhaegen. Online sparse gaussian process training with input noise. *CoRR*, abs/1601.08068, 2016.

[4] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[5] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

[6] Thang D Bui, Cuong Nguyen, and Richard E Turner. Streaming sparse gaussian process approximations. In *Advances in Neural Information Processing Systems*, pages 3301–3309, 2017.

[7] Thang D Bui, Josiah Yan, and Richard E Turner. A unifying framework for sparse gaussian process approximation using power expectation propagation. *Journal of Machine Learning Research*, 18:1–72, 2017.

[8] Andrea Carron, Marco Todescato, Ruggero Carli, Luca Schenato, and Gianluigi Pillonetto. Machine learning meets kalman filtering. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 4594–4599. IEEE, 2016.

[9] Tianshi Chen, Henrik Ohlsson, and Lennart Ljung. On the estimation of transfer functions, regularizations and gaussian processes—revisited. *Automatica*, 48(8):1525–1535, 2012.

[10] Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong. On the convergence of a class of Adam-type algorithms for non-convex optimization. *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[11] Lehel Csató and Manfred Opper. Sparse online gaussian processes. *Neural computation*, 14(3):641–668, 2002.

[12] Roger Frigola, Fredrik Lindsten, Thomas B Schön, and Carl Edward Rasmussen. Bayesian inference and learning in gaussian process state-space models with particle mcmc. In *Advances in Neural Information Processing Systems*, pages 3156–3164, 2013.

[13] GPy. GPy: A gaussian process framework in python. http://github.com/SheffieldML/GPy, since 2012.

[14] Jouni Hartikainen and Simo Särkkä. Kalman filtering and smoothing solutions to temporal gaussian process regression models. In *2010 IEEE International Workshop on Machine Learning for Signal Processing*, pages 379–384. IEEE, 2010.

[15] James Hensman, Nicolo Fusi, and Neil D Lawrence. Gaussian processes for big data. In *Conference for Uncertainty in Artificial Intelligence*, 2013.

[16] Trong Nghia Hoang, Quang Minh Hoang, and Bryan Kian Hsiang Low. A unifying framework of anytime sparse gaussian process regression models with stochastic variational inference for big data. In *ICML*, pages 569–578, 2015.

[17] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] Juš Kocijan, Agathe Girard, Blaž Banko, and Roderick Murray-Smith. Dynamic systems identification with gaussian processes. *Mathematical and Computer Modelling of Dynamical Systems*, 11(4):411–424, 2005.

[20] Haitao Liu, Yew-Soon Ong, Xiaobo Shen, and Jianfei Cai. When gaussian process meets big data: A review of scalable gps. *arXiv preprint arXiv:1807.01065*, 2018.

[21] Benn Macdonald, Catherine Higham, and Dirk Husmeier. Controversy in mechanistic modelling with gaussian processes. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1539–1547. JMLR.org, 2015.

[22] Alexander G. de G. Matthews, Mark van der Wilk, Tom Nickson, Keisuke. Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18(40):1–6, apr 2017.

[23] César Lincoln C. Mattos, Zhenwen Dai, Andreas Damianou, Jeremy Forth, Guilherme A. Barreto, and Neil D. Lawrence. Recurrent Gaussian processes. In Hugo Larochelle, Brian Kingsbury, and Samy Bengio, editors, *Proceedings of the International Conference on Learning Representations*, volume 3, Caribe Hotel, San Juan, PR, 00 2016.

[24] Gianluigi Pillonetto. A new kernel-based approach to hybrid system identification. *Automatica*, 70:21–31, 2016.

[25] Gianluigi Pillonetto and Giuseppe De Nicolao. A new kernel-based approach for linear system identification. *Automatica*, 46(1):81–93, 2010.

[26] Gianluigi Pillonetto, Francesco Dinuzzo, Tianshi Chen, Giuseppe De Nicolao, and Lennart Ljung. Kernel methods in system identification, machine learning and function estimation: A survey. *Automatica*, 50(3):657–682, 2014.

[27] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6(Dec):1939–1959, 2005.

[28] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, 2006.

[29] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.

[30] Simo Sarkka, Arno Solin, and Jouni Hartikainen. Spatiotemporal learning via infinite-dimensional bayesian filtering and smoothing: A look at gaussian process regression through kalman filtering. *IEEE Signal Processing Magazine*, 30(4):51–61, 2013.

[31] Matthias Seeger, Christopher Williams, and Neil Lawrence. Fast forward selection to speed up sparse gaussian process regression. In *Artificial Intelligence and Statistics 9*, number EPFL-CONF-161318, 2003.

[32] Bernhard W Silverman. Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–52, 1985.

[33] Alex J Smola and Peter L Bartlett. Sparse greedy gaussian process regression. In *Advances in neural information processing systems*, pages 619–625, 2001.

[34] Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, pages 1257–1264, 2006.

[35] Andreas Svensson and Thomas B Schön. A flexible state–space model for learning nonlinear dynamical systems. *Automatica*, 80:189–199, 2017.

[36] Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Artificial Intelligence and Statistics*, pages 567–574, 2009.

[37] Marco Todescato, Andrea Carron, Ruggero Carli, Gianluigi Pillonetto, and Luca Schenato. Efficient spatio-temporal gaussian regression via kalman filtering. *arXiv preprint arXiv:1705.01485*, 2017.

[38] P. T. Tran and L. T. Phong. On the convergence proof of amsgrad and a new version. *IEEE Access*, 7:61706–61716, 2019.

[39] Grace Wahba, Xiwu Lin, Fangyu Gao, Dong Xiang, Ronald Klein, and Barbara Klein. The bias-variance tradeoff and the randomized gacv. In *Advances in Neural Information Processing Systems*, pages 620–626, 1999.

[40] Dongruo Zhou, Yiqi Tang, Ziyan Yang, Yuan Cao, and Quanquan Gu. On the Convergence of Adaptive Gradient Methods for Nonconvex Optimization. 2018.

[41] Fangyu Zou, Li Shen, Zequn Jie, Weizhong Zhang, and Wei Liu. A sufficient condition for convergences of Adam and RMSProp. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June(1):11119–11127, 2019.

[42] M. A. Álvarez, D. Luengo, and N. D. Lawrence. Linear latent force models using gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2693–2705, Nov 2013.

A Details for Recursive Gradient Propagation

We show here the computation for the recursive gradient propagation from Sect. 4.1 for the PEP model. For other models, a_k and $\overline{\boldsymbol{V}}$ from the Table 2 could be used correspondingly. We will use the following notation: $\operatorname{diag}[\boldsymbol{A}] = \boldsymbol{d}, d_i = a_{ii}$, $\operatorname{Diag}[\boldsymbol{d}] = \boldsymbol{A}, a_{ii} = d_i, a_{ij} = 0$, $\boldsymbol{A} \odot \boldsymbol{B} = \boldsymbol{C}, c_{ij} = a_{ij} b_{ij}$, $\boldsymbol{A} \div \boldsymbol{B} = \boldsymbol{C}, c_{ij} = \frac{a_{ij}}{b_{ij}}$, $\boldsymbol{A}^{\odot 2} = \boldsymbol{C}, c_{ij} = a_{ij}^2$, $\dot{\boldsymbol{A}} = \frac{\partial \boldsymbol{A}(\theta)}{\partial \theta}, \forall \theta \in \boldsymbol{\Theta}$, $\operatorname{sum}[\boldsymbol{A}] = \sum_{i,j} a_{ij}$, $1_{[z]} = 1$ if $z = \text{true}, 0$ other.

Initialization

$$
\begin{aligned}
\boldsymbol{\eta}_0 &= \boldsymbol{0}; & \dot{\boldsymbol{\eta}}_0 &= \boldsymbol{0};\\
\boldsymbol{\Lambda}_0 &= \boldsymbol{K}_{\boldsymbol{RR}}^{-1}; & \dot{\boldsymbol{\Lambda}}_0 &= -\boldsymbol{K}_{\boldsymbol{RR}}^{-1} \dot{\boldsymbol{K}}_{\boldsymbol{RR}} \boldsymbol{K}_{\boldsymbol{RR}}^{-1};\\
\psi_0 &= -\frac{N}{2} \log 2\pi; & \dot{\psi}_0 &= 0;\\
\boldsymbol{\Sigma}_0 &= \boldsymbol{K}_{\boldsymbol{RR}}; & \operatorname{logDet}_0 &= \log |\boldsymbol{\Lambda}_0|;
\end{aligned}
$$

Natural Mean and Precision Updates

$$
\begin{aligned}
\boldsymbol{H}_k &= \boldsymbol{K}_{\boldsymbol{X}_k \boldsymbol{R}} \boldsymbol{K}_{\boldsymbol{RR}}^{-1};\\
\boldsymbol{d}_k &= \operatorname{diag}\left[\boldsymbol{K}_{\boldsymbol{X}_k \boldsymbol{X}_k} - \boldsymbol{K}_{\boldsymbol{X}_k \boldsymbol{R}} \boldsymbol{K}_{\boldsymbol{RR}}^{-1} \boldsymbol{K}_{\boldsymbol{R}\boldsymbol{X}_k}\right];\\
\boldsymbol{v}_k &= \alpha \boldsymbol{d}_k + \sigma_n^2 \mathbb{1}; \quad \boldsymbol{V}_k^{-1} = \operatorname{Diag}[\mathbb{1} \div \boldsymbol{v}_k];\\
a_k &= \frac{1-\alpha}{\alpha} \left(\sum_{i=1}^{B} \log([\boldsymbol{v}_k]_i) - B \log \sigma_n^2 \right);
\end{aligned}
$$

$$
\begin{aligned}
\boldsymbol{r}_k &= \boldsymbol{y}_k - \boldsymbol{H}_k \boldsymbol{\Sigma}_{k-1} \boldsymbol{\eta}_{k-1};\\
\boldsymbol{\eta}_k &= \boldsymbol{\eta}_{k-1} + \boldsymbol{H}_k^T \boldsymbol{V}_k^{-1} \boldsymbol{y}_k;\\
\boldsymbol{\Lambda}_k &= \boldsymbol{\Lambda}_{k-1} + \boldsymbol{H}_k^T \boldsymbol{V}_k^{-1} \boldsymbol{H}_k;\\
\boldsymbol{\Sigma}_k, \log |\boldsymbol{\Lambda}_k| &= \boldsymbol{\Lambda}_k^{-1}, \log |\boldsymbol{\Lambda}_k|;\\
\boldsymbol{S}_k^{-1} &= \boldsymbol{V}_k^{-1} - \boldsymbol{V}_k^{-1} \boldsymbol{H}_k \boldsymbol{\Sigma}_k \boldsymbol{H}_k^T \boldsymbol{V}_k^{-1};\\
\psi_k &= \psi_{k-1} - \frac{1}{2} \left(\log |\boldsymbol{\Lambda}_k| - \log |\boldsymbol{\Lambda}_{k-1}| \right.\\
&\quad \left. - \log \left|\boldsymbol{V}_k^{-1}\right| + \boldsymbol{r}_k^T \boldsymbol{S}_k^{-1} \boldsymbol{r}_k + a_k \right)
\end{aligned}
$$

Intermediate Derivatives

$$
\begin{aligned}
\dot{L}_{d\boldsymbol{H}_k} &= 2 \left(\boldsymbol{V}_k^{-1} \boldsymbol{H}_k \boldsymbol{\Sigma}_k - \boldsymbol{S}_k^{-1} \boldsymbol{r}_k (\boldsymbol{\Sigma}_{k-1} \boldsymbol{\eta}_{k-1} \right.\\
&\quad \left. + \boldsymbol{\Sigma}_k \boldsymbol{H}_k^T \boldsymbol{V}_k^{-1} \boldsymbol{r}_k)^T \right)\\
\dot{L}_{d\boldsymbol{v}_k} &= -\left(\operatorname{diag}\left[\boldsymbol{H}_k \boldsymbol{\Sigma}_k \boldsymbol{H}_k^T\right] - \frac{1}{\alpha} \boldsymbol{v}_k \right.\\
&\quad \left. + (\boldsymbol{r}_k - \boldsymbol{H}_k \boldsymbol{\Sigma}_k \boldsymbol{H}_k^T \boldsymbol{V}_k^{-1} \boldsymbol{r}_k)^{\odot 2} \right) \div \boldsymbol{v}_k^{\odot 2}\\
\dot{L}_{d\boldsymbol{K}_{\boldsymbol{X}_k \boldsymbol{R}}} &= \dot{L}_{d\boldsymbol{H}_k} \boldsymbol{K}_{\boldsymbol{RR}}^{-1} - 2\alpha \operatorname{Diag}\left[\dot{L}_{d\boldsymbol{v}_k}\right] \boldsymbol{H}_k\\
\dot{L}_{d\boldsymbol{K}_{\boldsymbol{RR}}} &= -\boldsymbol{H}_k^T \left(\dot{L}_{d\boldsymbol{H}_k} \boldsymbol{K}_{\boldsymbol{RR}}^{-1} - \alpha \operatorname{Diag}\left[\dot{L}_{d\boldsymbol{v}_k}\right] \boldsymbol{H}_k \right)\\
\dot{L}_{d\boldsymbol{k}_{\boldsymbol{X}_k \boldsymbol{X}_k}} &= \alpha \dot{L}_{d\boldsymbol{v}_k}\\
\dot{L}_{d\boldsymbol{\Lambda}_k} &= \boldsymbol{\Sigma}_k - \boldsymbol{\Sigma}_{k-1} + 2\boldsymbol{\Sigma}_{k-1} \boldsymbol{H}_k^T \boldsymbol{S}_k^{-1} \boldsymbol{r}_k \boldsymbol{\eta}_{k-1}^T \boldsymbol{\Sigma}_{k-1}\\
&\quad + \boldsymbol{\Sigma}_k \boldsymbol{H}_k^T \boldsymbol{V}_k^{-1} \boldsymbol{r}_k \boldsymbol{r}_k^T \boldsymbol{V}_k^{-1} \boldsymbol{H}_k \boldsymbol{\Sigma}_k\\
\dot{L}_{d\boldsymbol{\eta}_k} &= -2\boldsymbol{\Sigma}_{k-1} \boldsymbol{H}_k^T \boldsymbol{S}_k^{-1} \boldsymbol{r}_k\\
\dot{L}_{d\_d_n} &= 2\sigma_n^2 \operatorname{sum}\left[\dot{L}_{d\boldsymbol{v}_k}\right] - 2B \frac{1-\alpha}{\alpha}
\end{aligned}
$$

Derivative Updates

Loop over $\theta_i \in \boldsymbol{\theta}$**:**

$$
\begin{aligned}
\dot{\psi}_k &= \dot{\psi}_{k-1} - \frac{1}{2} \left(\operatorname{sum}\left[\dot{L}_{d\boldsymbol{\eta}_k} \odot \dot{\boldsymbol{\eta}}_{k-1}\right] \right.\\
&\quad + \operatorname{sum}\left[\dot{L}_{d\boldsymbol{\Lambda}_k} \odot \dot{\boldsymbol{\Lambda}}_{k-1}\right] + \operatorname{sum}\left[\dot{L}_{d\boldsymbol{K}_{\boldsymbol{RR}}} \odot \dot{\boldsymbol{K}}_{\boldsymbol{RR}}\right]\\
&\quad + \operatorname{sum}\left[\dot{L}_{d\boldsymbol{K}_{\boldsymbol{X}_k \boldsymbol{R}}} \odot \dot{\boldsymbol{K}}_{\boldsymbol{X}_k \boldsymbol{R}}\right]\\
&\quad \left. + \operatorname{sum}\left[\dot{L}_{d\boldsymbol{k}_{\boldsymbol{X}_k \boldsymbol{X}_k}} \odot \dot{\boldsymbol{k}}_{\boldsymbol{X}_k \boldsymbol{X}_k}\right] + 1_{[\theta_k = \sigma_n]} \dot{L}_{d\_d_n} \right)\\
\dot{\boldsymbol{H}}_k &= \dot{\boldsymbol{K}}_{\boldsymbol{X}_k \boldsymbol{R}} \boldsymbol{K}_{\boldsymbol{RR}}^{-1} - \boldsymbol{K}_{\boldsymbol{X}_k \boldsymbol{R}} \boldsymbol{K}_{\boldsymbol{RR}}^{-1} \dot{\boldsymbol{K}}_{\boldsymbol{RR}} \boldsymbol{K}_{\boldsymbol{RR}}^{-1};\\
\dot{\boldsymbol{d}}_k &= \operatorname{diag}\left[\dot{\boldsymbol{K}}_{\boldsymbol{X}_k \boldsymbol{X}_k} - \dot{\boldsymbol{K}}_{\boldsymbol{X}_k \boldsymbol{R}} \boldsymbol{K}_{\boldsymbol{RR}}^{-1} \boldsymbol{K}_{\boldsymbol{R}\boldsymbol{X}_k} \right.\\
&\quad + \boldsymbol{K}_{\boldsymbol{X}_k \boldsymbol{R}} \boldsymbol{K}_{\boldsymbol{RR}}^{-1} \dot{\boldsymbol{K}}_{\boldsymbol{RR}} \boldsymbol{K}_{\boldsymbol{RR}}^{-1} \boldsymbol{K}_{\boldsymbol{R}\boldsymbol{X}_k}\\
&\quad \left. - \boldsymbol{K}_{\boldsymbol{X}_k \boldsymbol{R}} \boldsymbol{K}_{\boldsymbol{RR}}^{-1} \dot{\boldsymbol{K}}_{\boldsymbol{R}\boldsymbol{X}_k} \right];\\
\dot{\boldsymbol{V}}_k^{-1} &= -1_{[\theta_i \neq \sigma_n]} \alpha \boldsymbol{V}_k^{-1} \operatorname{Diag}\left[\dot{\boldsymbol{d}}_k\right] \boldsymbol{V}_k^{-1}\\
&\quad - 1_{[\theta_i = \sigma_n]} 2\sigma_n^2 \dot{\boldsymbol{V}}_k^{-1} \dot{\boldsymbol{V}}_k^{-1};\\
\dot{\boldsymbol{\eta}}_k &= \dot{\boldsymbol{\eta}}_{k-1} + \dot{\boldsymbol{H}}_k^T \boldsymbol{V}_k^{-1} \boldsymbol{y}_k + \boldsymbol{H}_k^T \dot{\boldsymbol{V}}_k^{-1} \boldsymbol{y}_k;\\
\dot{\boldsymbol{\Lambda}}_k &= \dot{\boldsymbol{\Lambda}}_{k-1} + \dot{\boldsymbol{H}}_k^T \boldsymbol{V}_k^{-1} \boldsymbol{H}_k\\
&\quad + \boldsymbol{H}_k^T \dot{\boldsymbol{V}}_k^{-1} \boldsymbol{H}_k + \boldsymbol{H}_k^T \boldsymbol{V}_k^{-1} \dot{\boldsymbol{H}}_k
\end{aligned}
$$

For the noise σ_n, all kernel derivatives are zero, therefore the calculations simplify significantly.

B Derivation of Recursive Collapsed Bound

We provide more details and a detailed derivation for the recursive collapsed lower bound (25) for the VFE model from Sect. 4.1. Instead of lower bounding directly the batch log marginal likelihood as done by Titsias [36] in the batch case, our approach relies on the recursive factorization of the joint log marginal likelihood $\log p(\boldsymbol{y}|\boldsymbol{\theta}) = \log \prod_{k=1}^{K} p(\boldsymbol{y}_k|\boldsymbol{y}_{1:k-1}, \boldsymbol{\theta}) = \sum_{k=1}^{K} \log p(\boldsymbol{y}_k|\boldsymbol{y}_{1:k-1}, \boldsymbol{\theta})$. The properties induced by the sparse augmented inducing point model yield $\log p(\boldsymbol{y}|\boldsymbol{\theta}) = \sum_{k=1}^{K} \log \int p(\boldsymbol{y}_k|\boldsymbol{f}_k, \boldsymbol{\theta})\, p(\boldsymbol{f}_k|\boldsymbol{u}, \boldsymbol{\theta})\, p(\boldsymbol{u}|\boldsymbol{y}_{1:k-1}, \boldsymbol{\theta})\, \mathrm{d}\boldsymbol{f}_k\, \mathrm{d}\boldsymbol{u}$. We first introduce the variational distributions $q_k(\boldsymbol{f}_k, \boldsymbol{u}) = p(\boldsymbol{f}_k|\boldsymbol{u}, \boldsymbol{\theta})\, q_k(\boldsymbol{u}) \approx p(\boldsymbol{f}_k, \boldsymbol{u}|\boldsymbol{y}_{1:k}, \boldsymbol{\theta})$, then by applying Jensen's inequality to each individual term in the true log marginal likelihood, we obtain the lower bound

$$
\begin{aligned}
\log p(\boldsymbol{y}|\boldsymbol{\theta}) &\geq \sum_{k=1}^{K} \int p(\boldsymbol{f}_k|\boldsymbol{u}, \boldsymbol{\theta})\, q_k(\boldsymbol{u}) \ldots\\
\ldots \log &\frac{p(\boldsymbol{y}_k|\boldsymbol{f}_k, \boldsymbol{\theta})\, p(\boldsymbol{f}_k|\boldsymbol{u}, \boldsymbol{\theta})\, p(\boldsymbol{u}|\boldsymbol{y}_{1:k-1}, \boldsymbol{\theta})}{p(\boldsymbol{f}_k|\boldsymbol{u}, \boldsymbol{\theta})\, q_k(\boldsymbol{u})}\, \mathrm{d}\boldsymbol{f}_k\, \mathrm{d}\boldsymbol{u}.
\end{aligned}
$$

The quantity $p(\boldsymbol{u}|\boldsymbol{y}_{1:k-1}, \boldsymbol{\theta})$ is unknown, however, we can replace it with $q_{k-1}(\boldsymbol{u})$ leading to $\mathscr{L}(q_1, \ldots, q_K, \boldsymbol{\theta})$

$$
\sum_{k=1}^{K} \int p(\boldsymbol{f}_k|\boldsymbol{u}, \boldsymbol{\theta})\, q_k(\boldsymbol{u}) \log \frac{p\left(\boldsymbol{y}_f|\boldsymbol{f}_k, \boldsymbol{\theta}\right) q_{b-1}(\boldsymbol{u})}{q_k(\boldsymbol{u})}\, \mathrm{d}\boldsymbol{f}_k\, \mathrm{d}\boldsymbol{u}. \tag{B.1}
$$

Maximizing this lower bound recursively with respect to the distributions $q_k(\boldsymbol{u})$ leads to a sequence of optimal variational

distributions $q_k^*(\boldsymbol{u})$ for the inducing outputs

$$\mathcal{N}\left(\boldsymbol{u}|\boldsymbol{\Sigma}_k\left\{\frac{1}{\sigma_n^2}\boldsymbol{H}_k^T\boldsymbol{y}_k+\boldsymbol{\Sigma}_{k-1}^{-1}\boldsymbol{\mu}_{k-1}\right\},\boldsymbol{\Sigma}_k\right), \tag{B.2}$$

where $\boldsymbol{\Sigma}_k=\left(\boldsymbol{\Sigma}_{k-1}^{-1}+\frac{1}{\sigma_n^2}\boldsymbol{H}_k^T\boldsymbol{H}_k\right)^{-1}$ and $\boldsymbol{H}_k=\boldsymbol{K}_{\boldsymbol{X}_k\boldsymbol{R}}\boldsymbol{K}_{\boldsymbol{R}\boldsymbol{R}}^{-1}$. Plugging $q_k^*(\boldsymbol{u})$ into (B.1) yields

$$\begin{aligned}\mathscr{L}_{REC}(\boldsymbol{\theta})=\sum_{k=1}^{K}\big[&\log\mathcal{N}\left(\boldsymbol{y}_k|\boldsymbol{H}_k\boldsymbol{\mu}_{k-1},\dots\right.\\&\left.\dots\boldsymbol{H}_k\boldsymbol{\Sigma}_{k-1}\boldsymbol{H}_k^T+\sigma_n^2\mathbb{I}\right)-\frac{\operatorname{Tr}\left[\boldsymbol{D}_{\boldsymbol{X}_k\boldsymbol{X}_k}\right]}{2\sigma_n^2}\Bigg].\end{aligned} \tag{B.3}$$

The recursive bound $\mathscr{L}_{REC}(\boldsymbol{\theta})$ is equivalent to the batch collapsed bound $\mathscr{L}_{VFE}(\boldsymbol{\theta})$ in (5) and it holds $\mathscr{L}_{SVGP}(\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\theta})\leq\mathscr{L}_{REC}(\boldsymbol{\theta})$ with equality when inserting the optimal variational posterior. The variational posterior update in (B.2) has the same form as the recursive updates in (15). Similarly, the recursive collapsed lower bound in (B.3) is equal to (25).

We provide below a detailed derivation that follows closely the proof in [36] for the recursive collapsed bound (B.3) as well as the sequence of optimal distributions (B.2). We assume mini-batches of size B, that is, we have training data $\mathscr{D}=\{\boldsymbol{y}_k,\boldsymbol{X}_k\}_{k=1}^K$ and the corresponding latent function values$\{\boldsymbol{f}_k\}_{k=1}^K$. We briefly recap the involved quantities and introduce abbreviations:

$$\begin{aligned}p\left(\boldsymbol{y}_k|\boldsymbol{f}_k,\boldsymbol{\theta}\right)&=\mathcal{N}\left(\boldsymbol{y}_k|\boldsymbol{f}_k,\sigma_n^2\mathbb{I}\right);\\p\left(\boldsymbol{f}_k|\boldsymbol{u},\boldsymbol{\theta}\right)&=\mathcal{N}\left(\boldsymbol{f}_k|\boldsymbol{H}_k\boldsymbol{u},\boldsymbol{K}_{\boldsymbol{X}_k\boldsymbol{X}_k}-\boldsymbol{Q}_{\boldsymbol{X}_k\boldsymbol{X}_k}\right);\\p\left(\boldsymbol{u}|\boldsymbol{\theta}\right)&=\mathcal{N}\left(\boldsymbol{u}|\boldsymbol{0},\boldsymbol{K}_{\boldsymbol{R}\boldsymbol{R}}\right)=q_0(\boldsymbol{u});\\q_{k-1}(\boldsymbol{u})&=\mathcal{N}\left(\boldsymbol{u}|\boldsymbol{\mu}_{k-1},\boldsymbol{\Sigma}_{k-1}\right)\approx p\left(\boldsymbol{u}|\boldsymbol{y}_{1:k-1},\boldsymbol{\theta}\right)\\\boldsymbol{H}_k&=\boldsymbol{K}_{\boldsymbol{X}_k\boldsymbol{R}}\boldsymbol{K}_{\boldsymbol{R}\boldsymbol{R}}^{-1};\\\boldsymbol{Q}_{\boldsymbol{X}_k\boldsymbol{X}_k}&=\boldsymbol{K}_{\boldsymbol{X}_k\boldsymbol{R}}\boldsymbol{K}_{\boldsymbol{R}\boldsymbol{R}}^{-1}\boldsymbol{K}_{\boldsymbol{R}\boldsymbol{X}_k}.\end{aligned}$$

Starting from (B.1), we have the bound

$$\sum_{k=1}^{K}\int p\left(\boldsymbol{f}_k|\boldsymbol{u},\boldsymbol{\theta}\right)q_k(\boldsymbol{u})\log\frac{p\left(\boldsymbol{y}_k|\boldsymbol{f}_k,\boldsymbol{\theta}\right)q_{k-1}(\boldsymbol{u})}{q_k(\boldsymbol{u})}\,\mathrm{d}\boldsymbol{f}_k\,\mathrm{d}\boldsymbol{u}$$

which can be rearranged to

$$\sum_{k=1}^{K}\int q_k(\boldsymbol{u})\left\{\log G(\boldsymbol{u},\boldsymbol{y}_k)+\log\frac{q_{k-1}(\boldsymbol{u})}{q_k(\boldsymbol{u})}\right\}\mathrm{d}\boldsymbol{u},$$

where $\log G(\boldsymbol{u},\boldsymbol{y}_k)=\int p\left(\boldsymbol{f}_k|\boldsymbol{u},\boldsymbol{\theta}\right)\log p\left(\boldsymbol{y}_k|\boldsymbol{f}_k,\boldsymbol{\theta}\right)\mathrm{d}\boldsymbol{f}_k$. The integral involving $\boldsymbol{f}_k$ is computed as $\log G(\boldsymbol{u},\boldsymbol{y}_k)=\int p\left(\boldsymbol{f}_k|\boldsymbol{u},\boldsymbol{\theta}\right)\log p\left(\boldsymbol{y}_k|\boldsymbol{f}_k,\boldsymbol{\theta}\right)\mathrm{d}\boldsymbol{f}_k$, which equals

$$\begin{aligned}&=\mathbb{E}_{\boldsymbol{f}_k|\boldsymbol{u}}\left[-\frac{B}{2}\log(2\pi\sigma_n^2)-\frac{1}{2}\left[\boldsymbol{y}_k-\boldsymbol{f}_k\right]^T\frac{1}{\sigma_n^2}\left[\boldsymbol{y}_k-\boldsymbol{f}_k\right]\right]\\&=-\frac{B}{2}\log(2\pi\sigma_n^2)-\frac{1}{2\sigma_n^2}\left(\boldsymbol{y}_k^T\boldsymbol{y}_k+\mathbb{E}_{\boldsymbol{f}_k|\boldsymbol{u}}\left[\boldsymbol{f}_k^T\boldsymbol{f}_k-2\boldsymbol{y}_k^T\boldsymbol{f}_k\right]\right).\end{aligned}$$

Using $\mathbb{E}\left[\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}\right]=\operatorname{Tr}[\boldsymbol{A}\boldsymbol{\Sigma}]+\boldsymbol{\mu}^T\boldsymbol{A}\boldsymbol{\mu}$ with $p(\boldsymbol{x})=\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma})$ yields

$$\begin{aligned}&\log G(\boldsymbol{u},\boldsymbol{y}_k)=-\frac{K}{2}\log(2\pi\sigma_n^2)-\frac{1}{2\sigma_n^2}\left(\boldsymbol{y}_k^T\boldsymbol{y}_k\right.\\&\left.+\operatorname{Tr}\left[\boldsymbol{K}_{\boldsymbol{X}_k\boldsymbol{X}_k}-\boldsymbol{Q}_{\boldsymbol{X}_k\boldsymbol{X}_k}\right]+\boldsymbol{u}^T\boldsymbol{H}_k^T\boldsymbol{H}_k\boldsymbol{u}-2\boldsymbol{y}_k^T\boldsymbol{H}_k\boldsymbol{u}\right)\\&=\log\left[\mathcal{N}\left(\boldsymbol{y}_k|\boldsymbol{H}_k\boldsymbol{u},\sigma_n^2\mathbb{I}\right)\right]-\frac{1}{2\sigma_n^2}\operatorname{Tr}\left[\boldsymbol{K}_{\boldsymbol{X}_k\boldsymbol{X}_k}-\boldsymbol{Q}_{\boldsymbol{X}_k\boldsymbol{X}_k}\right].\end{aligned}$$

Substitute this expression back, the lower bound becomes

$$\begin{aligned}\sum_{k=1}^{K}\Bigg[\int q_k(\boldsymbol{u})\log\frac{\mathcal{N}\left(\boldsymbol{y}_k|\boldsymbol{H}_k\boldsymbol{u},\sigma_n^2\mathbb{I}\right)q_{k-1}(\boldsymbol{u})}{q_k(\boldsymbol{u})}\,\mathrm{d}\boldsymbol{u}\\-\frac{1}{2\sigma_n^2}\operatorname{Tr}\left[\boldsymbol{K}_{\boldsymbol{X}_k\boldsymbol{X}_k}-\boldsymbol{Q}_{\boldsymbol{X}_k\boldsymbol{X}_k}\right]\Bigg].\end{aligned}$$

We can now maximize this bound with respect to $q_k(\boldsymbol{u})$. Here since we have not constrained q_b to belong to any fixed family of distributions, we can compute the optimal bound by reversing the Jensen's inequality leading to

$$\begin{aligned}\mathscr{L}_{REC}(\boldsymbol{\theta})&=\sum_{k=1}^{K}\Bigg[\log\int\mathcal{N}\left(\boldsymbol{y}_k|\boldsymbol{H}_k\boldsymbol{u},\sigma_n^2\mathbb{I}\right)q_{k-1}(\boldsymbol{u})\,\mathrm{d}\boldsymbol{u}\\&\quad-\frac{1}{2\sigma_n^2}\operatorname{Tr}\left[\boldsymbol{K}_{\boldsymbol{X}_k\boldsymbol{X}_k}-\boldsymbol{Q}_{\boldsymbol{X}_k\boldsymbol{X}_k}\right]\Bigg]\\&=\sum_{k=1}^{K}\Big[\log\mathcal{N}\left(\boldsymbol{y}_k|\boldsymbol{H}_k\boldsymbol{\mu}_{k-1},\boldsymbol{H}_k\boldsymbol{\Sigma}_{k-1}\boldsymbol{H}_k^T+\sigma_n^2\mathbb{I}\right)\\&\quad-\frac{1}{2\sigma_n^2}\operatorname{Tr}\left[\boldsymbol{K}_{\boldsymbol{X}_k\boldsymbol{X}_k}-\boldsymbol{Q}_{\boldsymbol{X}_k\boldsymbol{X}_k}\right]\Bigg]\end{aligned}$$

where we used a linear Gaussian identity (see, e.g., [4] Ch. 2.3) in the last step . This is equal to Eq. (25). The optimal distribution q_k^* that gives rise to this bound is proportional to $\mathcal{N}\left(\boldsymbol{y}_k|\boldsymbol{H}_k\boldsymbol{u},\sigma_n^2\mathbb{I}\right)q_{k-1}(\boldsymbol{u})=\mathcal{N}\left(\boldsymbol{y}_k|\boldsymbol{H}_k\boldsymbol{u},\sigma_n^2\mathbb{I}\right)\mathcal{N}\left(\boldsymbol{u}|\boldsymbol{\mu}_{k-1},\boldsymbol{\Sigma}_{k-1}\right)$ and can be analytically computed leading to

$$q_k^*(\boldsymbol{u})=\mathcal{N}\left(\boldsymbol{u}|\tfrac{1}{\sigma_n^2}\boldsymbol{\Sigma}_k\boldsymbol{H}_k^T\boldsymbol{y}_k+\boldsymbol{\Sigma}_{k-1}^{-1}\boldsymbol{\mu}_{k-1},\boldsymbol{\Sigma}_k\right)$$

where $\boldsymbol{\Sigma}_k=\left(\boldsymbol{\Sigma}_{k-1}^{-1}+\frac{1}{\sigma_n^2}\boldsymbol{H}_k^T\boldsymbol{H}_k\right)^{-1}$. This matches the result in Eq. (15) and (16) and completes the proof.

C Proof of proposition 1

We showed (App. B) how the batch approximate log-likelihood $\mathscr{L}_{PEP}$ in (4) can be recursively computed. In Sect. 4.1, we showed an equivalent way to compute it, that is, we have the cumulative bound $\psi^{(K)}(\boldsymbol{\theta})=\sum_k^K\psi_k(\boldsymbol{\theta})$ and it satisfies $\mathscr{L}_{PEP}(\boldsymbol{\theta})=\sum_k^K\psi_k(\boldsymbol{\theta})$. By induction and linearity of derivatives, we therefore get $\frac{\partial\psi^{(K)}(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}=\frac{\partial\psi_k(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}+\frac{\partial\psi^{(K-1)}(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}$ and $\frac{\partial\mathscr{L}_{PEP}}{\partial\boldsymbol{\theta}}=\frac{\partial\psi^{(K)}}{\partial\boldsymbol{\theta}}$.