2nd Symposium on Advances in Approximate Bayesian Inference, 2019 1–15

Sparse Orthogonal Variational Inference for Gaussian Processes

Jiaxin Shi SHIJX15@MAILS.TSINGHUA.EDU.CN
Tsinghua University

Michalis Titsias MTITSIAS@GOOGLE.COM
Andriy Mnih AMNIH@GOOGLE.COM
DeepMind

Abstract

We introduce a new interpretation of sparse variational approximations for Gaussian processes using inducing points, which can lead to more scalable algorithms than previous methods. It is based on decomposing a GP as a sum of two independent processes: one in the subspace spanned by the inducing basis and the other in the orthogonal complement. We show that this formulation recovers existing methods and at the same time allows to obtain tighter lower bounds on the marginal likelihood and new stochastic variational inference algorithms. We demonstrate the efficiency of these algorithms in several GP models ranging from standard regression to multi-class classification using (deep) convolutional GPs and report state-of-the-art results on CIFAR-10 for purely GP-based models.

1. Introduction

Sparse variational GP (SVGP) methods (Titsias, 2009; Hensman et al., 2013, 2015a) based on variational learning of inducing points are widely used in scalable GP inference. Such methods require $\mathcal{O}(M^2N + M^3)$ computation with M inducing points and allow us to perform mini-batch training by sub-sampling data points. However, the computational cost of SVGP methods is still cubic with respect to M, making it difficult to improve the flexibility of posterior approximations.

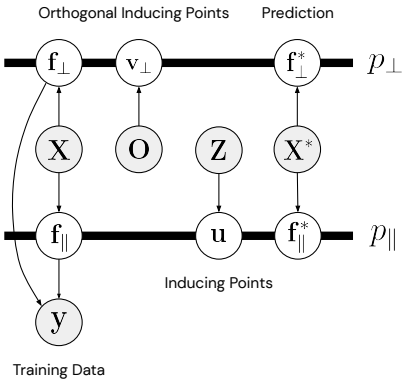


Figure 1: SOLVE-GP.

In this paper, we introduce a new framework called *Sparse OrthogonaL Variational infErence for Gaussian Processes* (SOLVE-GP), which allows increasing the number of inducing points we can use given a fixed computational budget. Our framework is based on a reinterpretation of SVGP methods as orthogonal decompositions of the GP prior. By introducing another set of inducing variables for the GP in the orthogonal complement, we can increase M at a much lower additional computational cost. For instance, doubling M leads to a 2-fold increase in computational cost with our method, compared to the 8-fold increase for the original SVGP method.

We conducted experiments on convolutional GPs and their deep variants. To the best of our knowledge, we are the first to train a purely GP-based model without any neural network components that achieves over 80% test accuracy on CIFAR-10. No data augmentation was used to obtain these results. We also evaluated our method on a range of regression datasets with tens of thousands to millions of datapoints. Our results show that SOLVE-GP is often competitive with the 4x more expensive SVGP counterpart that uses the same number of inducing points, and outperforms SVGP when given the same computational budget.

2. SOLVE-GP

2.1. Reinterpreting SVGP with Orthogonal Decomposition

We start by briefly reviewing the notations and SVGP methods in App. A, where we observe that samples from the conditional distribution $p(\mathbf{f}|\mathbf{u}) = \mathcal{N}(\mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{u}, \mathbf{K_{ff}} - \mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{K_{uf}})$ can be reparameterized as

$$\mathbf{f}_\perp \sim p_\perp(\mathbf{f}_\perp) := \mathcal{N}(\mathbf{0}, \mathbf{K_{ff}} - \mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{K_{uf}}), \quad \mathbf{f} = \mathbf{f}_\perp + \mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{u}. \tag{1}$$

Here we denote the zero-mean component that is independent of $\mathbf{u}$ as $\mathbf{f}_\perp$.[1] Now we can reparameterize $p(\mathbf{f}, \mathbf{u})$ as $\mathbf{u} \sim p(\mathbf{u})$, $\mathbf{f}_\perp \sim p_\perp(\mathbf{f}_\perp)$, $\mathbf{f} = \mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{u} + \mathbf{f}_\perp$, and the joint distribution becomes $p(\mathbf{y}, \mathbf{u}, \mathbf{f}_\perp) = p(\mathbf{y}|\mathbf{f}_\perp + \mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{u})p(\mathbf{u})p_\perp(\mathbf{f}_\perp)$. Posterior inference for $\mathbf{f}$ in the original model then turns into inference for $\mathbf{u}$ and $\mathbf{f}_\perp$. If we approximate the above model by considering a factorised approximation $q(\mathbf{u})p_\perp(\mathbf{f}_\perp)$, we arrive at the standard SVGP method. To see this, note that minimizing $\mathrm{KL}\left[q(\mathbf{u})p_\perp(\mathbf{f}_\perp)\|p(\mathbf{u}, \mathbf{f}_\perp|\mathbf{y})\right]$ is equivalent to maximizing the variational lower bound $\mathbb{E}_{q(\mathbf{u})p_\perp(\mathbf{f}_\perp)} \log p(\mathbf{y}|\mathbf{f}_\perp + \mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{u}) - \mathrm{KL}\left[q(\mathbf{u})\|p(\mathbf{u})\right]$, which is the SVGP objective (Eq. (6)) using the reparameterization in Eq. (1).

We consider improving the variational distribution for $\mathbf{f}_\perp$, noting that the complexity of inferring $\mathbf{f}_\perp$ is the same as for $\mathbf{f}$ and thus cubic. To resolve the problem, we observe that Eq. (1) corresponds to an orthogonal decomposition in the function space (see App. C for the derivation): $f = f_\parallel + f_\perp$, where

$$f_\parallel \sim p_\parallel \equiv \mathcal{GP}(0, \mathbf{k_u}(\mathbf{x})^\top \mathbf{K_{uu}^{-1}} \mathbf{k_u}(\mathbf{x}')), \tag{2}$$

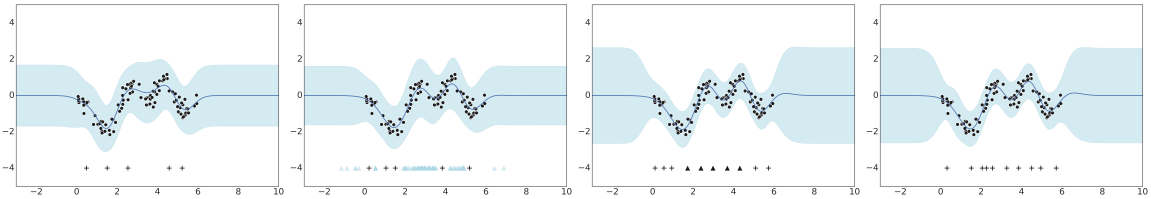
$$f_\perp \sim p_\perp \equiv \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}') - \mathbf{k_u}(\mathbf{x})^\top \mathbf{K_{uu}^{-1}} \mathbf{k_u}(\mathbf{x})). \tag{3}$$

$\mathbf{k_u}(\mathbf{x}) = [k(\mathbf{z}_1, \mathbf{x}), \ldots, k(\mathbf{z}_M, \mathbf{x})]^\top$. Marginalizing out the GPs at the training points $\mathbf{X}$, it is easy to show that $\mathbf{f}_\parallel = f_\parallel(\mathbf{X}) = \mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{u} \sim \mathcal{N}(\mathbf{0}, \mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{K_{uf}})$, and $\mathbf{f}_\perp = f_\perp(\mathbf{X}) \sim \mathcal{N}(\mathbf{0}, \mathbf{K_{ff}} - \mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{K_{uf}})$. This is exactly the decomposition we used, and the fact that $\mathbf{f}_\perp$ denotes the function values of the orthogonal process becomes clear.

2.2. SOLVE-GP Lower Bound

The orthogonal decomposition described in the previous section gives new insights for improving the variational distribution for $\mathbf{f}_\perp$. Specifically, we can introduce a second set of inducing variables $\mathbf{v}_\perp := f_\perp(\mathbf{O})$ to approximate the orthogonal process $p_\perp$, as illustrated in Fig. 1. We call this second set $\mathbf{O} = [\mathbf{o}_1, \ldots, \mathbf{o}_{M_2}]^\top \in \mathbb{R}^{M_2 \times d}$ the *orthogonal* inducing points.

1. Note that kernel matrices like $\mathbf{K_{uu}}$ depend on $\mathbf{Z}$ instead of $\mathbf{u}$.



(a) SVGP, 5 (b) ODVGP, $5 + 100$ (c) SOLVE-GP, $5 + 5$ (d) SVGP, 10

Figure 2: Posterior process on the Snelson dataset. The learned inducing locations are shown at the bottom of each figure, where + correspond to $\mathbf{Z}$; blue and dark triangles correspond to $\mathbf{O}$ in ODVGP and SOLVE-GP, respectively.

The joint distribution is then $p(\mathbf{y}|\mathbf{f}_\perp + \mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{u})p(\mathbf{u})p_\perp(\mathbf{f}_\perp|\mathbf{v}_\perp)p_\perp(\mathbf{v}_\perp)$. Notice that standard SVGP methods correspond to using the variational distribution $q(\mathbf{u})p_\perp(\mathbf{v}_\perp)p_\perp(\mathbf{f}_\perp|\mathbf{v}_\perp)$. To obtain better approximations we can replace $p_\perp(\mathbf{v}_\perp)$ with a tunable variational factor $q(\mathbf{v}_\perp)$: $q(\mathbf{u}, \mathbf{f}_\perp, \mathbf{v}_\perp) = q(\mathbf{u})q(\mathbf{v}_\perp)p_\perp(\mathbf{f}_\perp|\mathbf{v}_\perp)$. This gives the SOLVE-GP lower bound:

$$\mathbb{E}_{q(\mathbf{u})q(\mathbf{v}_\perp)p_\perp(\mathbf{f}_\perp|\mathbf{v}_\perp)}\left[\log p(\mathbf{y}|\mathbf{f}_\perp + \mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{u})\right] - \mathrm{KL}\left[q(\mathbf{u})\|p(\mathbf{u})\right] - \mathrm{KL}\left[q(\mathbf{v}_\perp)\|p_\perp(\mathbf{v}_\perp)\right]. \quad (4)$$

In the general setting, Eq. (4) can be maximized using minibatch training in $\mathcal{O}(M^3 + M_2^3)$ time per gradient update. In App. D we derive a collapsed version of this bound for GP regression and compare to the Titsias (2009) bound. For the function values at test data points $\mathbf{X}^*$, the predictive density given by our approximate posterior can be found in App. E.

Our method introduces another set of inducing points. One natural question to ask is: How does this compare to the standard SVGP algorithm with the inducing points chosen to be union of the two sets? We answer the question in App. F by interpreting our method as using a structured covariance in the variational approximation for SVGP. Interestingly, under this interpretation SOLVE-GP can be seen as a generalization of a recently proposed decoupled-inducing-points method (Salimbeni et al., 2018) (see App. H). As the decoupled method often comes with a complex dual formation in RKHS, our framework provides a simpler derivation and more intuitive understanding for it.

3. Experiments

3.1. 1D Regression

We begin by illustrating our method on a 1D regression problem (Snelson and Ghahramani, 2006) with 100 training points and batch size 20. We compare the following methods: SVGP with 5 and 10 inducing points, ODVGP ($M = 5, M_2 = 100$), and SOLVE-GP ($M = 5, M_2 = 5$). As plotted in Fig. 2, SVGP ($M = 5$) cannot fit the data well and underestimates uncertainty in regions beyond the training data. Increasing M to 10 fixes the issues, but requires 8x

Table 1: Convolutional GP for CIFAR-10 classification. Previous SOTA is 64.6%, achieved by SVGP with 1K inducing points (van der Wilk et al., 2017).

	$M(+M_2)$	Test Acc	Test LL	Time
SVGP	1K	66.07%	-1.59	0.241 s/iter
	1.6K	67.18%	-1.54	0.380 s/iter
SOLVE-GP	1K + 1K	**68.19%**	**-1.51**	0.370 s/iter
SVGP	2K*	68.06%	**-1.48**	0.474 s/iter

more computation than using 5 inducing points.[2] Decoupled methods provides a cheaper alternative and we have tried ODVGP ($M = 5, M_2 = 100$) using 100 additional inducing points for modeling the mean function. Comparing Fig. 2a and Fig. 2b, we can see that this results in a much better fit for the mean function. As ODVGP is a special case of the SOLVE-GP framework, we can improve over it in terms of covariance modeling. As seen in Fig. 2c, adding 5 orthogonal inducing points can closely approximate the results of SVGP ($M = 10$), with only 2-fold increase in time complexity relative to SVGP ($M = 5$).

3.2. (Deep) Convolutional Gaussian Process

One class of applications that benefit from SOLVE-GP is the training of large, hierarchical GP models where the posterior distribution is difficult to approximate with a small number of inducing points. Convolutional GPs (van der Wilk et al., 2017) and their deep variants (Blomqvist et al., 2018; Dutordoir et al., 2019) are such models. It is straightforward to apply SOLVE-GP to them (details in Sec. G).

Convolutional GP. We train convolutional GPs on CIFAR-10. We compare SVGP with 1K and 2K inducing points, SOLVE-GP ($M = 1\text{K}, M_2 = 1\text{K}$), and SVGP ($M = 1.6\text{K}$) which has a similar running time on GPU as SOLVE-GP. As shown in Table 1, SOLVE-GP outperforms SVGP ($M = 1\text{K}$). It also outperforms SVGP with the number of inducing points chosen to match the SOLVE-GP running time ($M = 1.6\text{K}$). SOLVE-GP also performs on par with the 4x more expensive SVGP ($M = 2\text{K}$), which indicates that the structured covariance approximation is fairly accurate for this large, non-conjugate model.

Deep Convolutional GPs. We further extend SOLVE-GP to deep convolutional GPs. We experiment with 2-layer and 3-layer models that have 1K inducing points in the output layer and 384 inducing points in each lower layer. The results are summarized in Table 2. These models are already fairly expensive on a single GPU, as can be seen from the time per iteration. SOLVE-GP allows to double the number of inducing points in each layer while only introducing a 2-fold increase in computation. This gives superior performance on both accuracy and test predictive likelihoods. The SVGP with 2x the inducing points takes a week to train and is included only for comparison purposes.

On both single-layer and deep convolutional GPs, we improve the state-of-the-art results of CIFAR-10 classification by 3-4 percentage points. This leads to more than 80% accuracy on CIFAR-10 with a purely GP-based model, without any neural network components, closing the gap between GP/kernel regression and CNN baselines presented in (Novak et al., 2019; Arora et al., 2019). All the results were obtained without data augmentation.

3.3. Regression Benchmarks

Besides classification experiments, we evaluate our method on 10 regression datasets, with size ranging from tens of thousands to millions. The results are described in detail in App. I.

2. In practice the cost is negligible in this toy problem but we are analyzing the theoretical complexity.

Table 2: Deep convolutional GPs for CIFAR-10 classification. Previous SOTA is 76.17% by a 3-layer model with 384 inducing points in all layers (Dutordoir et al., 2019).

	2-layer			3-layer		
	SVGP	SOLVE-GP	SVGP	SVGP	SOLVE-GP	SVGP
$M(+M_2)$	$384, 1\text{K}$	$384 + 384,\ 1\text{K} + 1\text{K}$	$768, 2\text{K}^*$	$384, 384, 1\text{K}$	$384 + 384, 384 + 384,$ $1\text{K} + 1\text{K}$	$768, 768, 2\text{K}^*$
Test Acc	76.35%	**77.8%**	77.46%	78.76%	**80.3%**	**80.33%**
Test LL	-1.04	**-0.98**	**-0.98**	-0.88	**-0.79**	-0.82
Time	0.392 s/iter	0.657 s/iter	1.104 s/iter	0.418 s/iter	0.752 s/iter	1.246 s/iter

References

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *arXiv preprint arXiv:1904.11955*, 2019.

Kenneth Blomqvist, Samuel Kaski, and Markus Heinonen. Deep convolutional Gaussian processes. *arXiv preprint arXiv:1810.03052*, 2018.

Ching-An Cheng and Byron Boots. Incremental variational sparse Gaussian process regression. In *Advances in Neural Information Processing Systems*, pages 4410–4418, 2016.

Ching-An Cheng and Byron Boots. Variational inference for Gaussian process models with linear complexity. In *Advances in Neural Information Processing Systems*, pages 5184–5194, 2017.

Vincent Dutordoir, Mark van der Wilk, Artem Artemev, Marcin Tomczak, and James Hensman. Translation insensitivity for deep convolutional Gaussian processes. *arXiv preprint arXiv:1902.05888*, 2019.

Trefor Evans and Prasanth Nair. Scalable Gaussian processes with grid-structured eigenfunctions (GP-GRIEF). In *International Conference on Machine Learning*, pages 1416–1425, 2018.

Jacob Gardner, Geoff Pleiss, Ruihan Wu, Kilian Weinberger, and Andrew Wilson. Product kernel interpolation for scalable Gaussian processes. In *Artificial Intelligence and Statistics*, pages 1407–1416, 2018.

Marton Havasi, José Miguel Hernández-Lobato, and Juan José Murillo-Fuentes. Deep Gaussian processes with decoupled inducing inputs. *arXiv preprint arXiv:1801.02939*, 2018.

James Hensman, Nicolo Fusi, and Neil D Lawrence. Gaussian processes for big data. *arXiv preprint arXiv:1309.6835*, 2013.

James Hensman, Alexander Matthews, and Zoubin Ghahramani. Scalable variational Gaussian process classification. In *Artificial Intelligence and Statistics*, pages 351–360, 2015a.

James Hensman, Alexander G Matthews, Maurizio Filippone, and Zoubin Ghahramani. MCMC for variationally sparse Gaussian processes. In *Advances in Neural Information Processing Systems*, pages 1648–1656, 2015b.

James Hensman, Nicolas Durrande, and Arno Solin. Variational Fourier features for Gaussian processes. *Journal of Machine Learning Research*, 18(151):1–151, 2017.

Miguel Lázaro-Gredilla and Anibal Figueiras-Vidal. Inter-domain Gaussian processes for sparse inference using inducing features. In *Advances in Neural Information Processing Systems*, pages 1087–1095, 2009.

Miguel Lázaro-Gredilla, Joaquin Quiñonero-Candela, Carl Edward Rasmussen, and Aníbal R Figueiras-Vidal. Sparse spectrum Gaussian process regression. *Journal of Machine Learning Research*, 11:1865–1881, 2010.

Iain Murray and Ryan P Adams. Slice sampling covariance hyperparameters of latent Gaussian models. In *Advances in Neural Information Processing Systems*, pages 1732–1740, 2010.

Roman Novak, Lechao Xiao, Yasaman Bahri, Jaehoon Lee, Greg Yang, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-dickstein. Bayesian deep convolutional networks with many channels are Gaussian processes. In *International Conference on Learning Representations*, 2019.

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.

Carl Edward Rasmussen and Christopher KI Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Hugh Salimbeni and Marc Deisenroth. Doubly stochastic variational inference for deep Gaussian processes. In *Advances in Neural Information Processing Systems*, pages 4588–4599, 2017.

Hugh Salimbeni, Ching-An Cheng, Byron Boots, and Marc Deisenroth. Orthogonally decoupled variational Gaussian processes. In *Advances in Neural Information Processing Systems*, pages 8711–8720, 2018.

Jiaxin Shi, Mohammad Emtiyaz Khan, and Jun Zhu. Scalable training of inference networks for Gaussian-process models. In *International Conference on Machine Learning*, pages 5758–5768, 2019.

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

Michael E Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1(Jun):211–244, 2001.

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

Mark van der Wilk, Carl Edward Rasmussen, and James Hensman. Convolutional Gaussian processes. In *Advances in Neural Information Processing Systems*, pages 2849–2858, 2017.

Ke Alexander Wang, Geoff Pleiss, Jacob R Gardner, Stephen Tyree, Kilian Q Weinberger, and Andrew Gordon Wilson. Exact Gaussian processes on a million data points. *arXiv preprint arXiv:1903.08114*, 2019.

Christopher KI Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, pages 682–688, 2001.

Andrew Wilson and Hannes Nickisch. Kernel interpolation for scalable structured Gaussian processes (KISS-GP). In *International Conference on Machine Learning*, pages 1775–1784, 2015.

Appendix A. Background

Here, we briefly review Gaussian processes and sparse variational GP methods. A GP is an uncountable collection of random variables indexed by a real-valued vector $\mathbf{x}$, of which any finite subset has a multivariate Gaussian distribution. It is defined by a mean function $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$ and a covariance function $k(\mathbf{x}, \mathbf{x}') = \mathrm{Cov}[f(\mathbf{x}), f(\mathbf{x}')]$:

$$f \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')).$$

Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]^\top \in \mathbb{R}^{N \times d}$ be (the matrix containing) the training data points and $\mathbf{f} = f(\mathbf{X}) \in \mathbb{R}^N$ denote the corresponding function values. Similarly we denote the test data points by $\mathbf{X}^*$ and their function values by $\mathbf{f}^*$. Then the joint distribution over $\mathbf{f}, \mathbf{f}^*$ is given by:

$$p(\mathbf{f}, \mathbf{f}^*) := \mathcal{N}\left(\begin{array}{c|c} \mathbf{f} & \left[\begin{array}{c} m(\mathbf{X}) \\ m(\mathbf{X}^*) \end{array} \right], \left[\begin{array}{cc} \mathbf{K_{ff}} & \mathbf{K_{f*}} \\ \mathbf{K_{*f}} & \mathbf{K_{**}} \end{array} \right] \end{array} \right),$$

where $\mathbf{K_{ff}}$ is an $N \times N$ kernel matrix with its (i,j)th entry as $k(\mathbf{x}_i, \mathbf{x}_j)$, and similarly $[\mathbf{K_{f*}}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j^*)$, $[\mathbf{K_{**}}]_{ij} = k(\mathbf{x}_i^*, \mathbf{x}_j^*)$. In practice we often observe the training function values through some noisy labels $\mathbf{y}$, generated by the likelihood function $p(\mathbf{y}|\mathbf{f})$. For regression, the likelihood usually models independent Gaussian observation noise: $y_n = f(\mathbf{x}_n) + \epsilon_n$, $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$. In this situation the exact posterior distribution $p(\mathbf{f}^*|\mathbf{y})$ can be computed in closed form:

$$\mathbf{f}^*|\mathbf{y} \sim \mathcal{N}(\mathbf{K_{*f}}(\mathbf{K_{ff}} + \sigma^2\mathbf{I})^{-1}\mathbf{y}, \mathbf{K_{**}} - \mathbf{K_{*f}}(\mathbf{K_{ff}} + \sigma^2\mathbf{I})^{-1}\mathbf{K_{f*}}). \tag{5}$$

As seen from Eq. (5), exact GP prediction involves the inverse of matrix $\mathbf{K_{ff}} + \sigma^2\mathbf{I}$, which requires $\mathcal{O}(N^3)$ computation. For large datasets, it is clear that we need to avoid the cubic complexity by resorting to approximations.

Inducing points have played a central role in previous works on scalable GP inference. The general idea is to summarize $\mathbf{f}$ with a small number of variables $\mathbf{u} = f(\mathbf{Z})$, where

$\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_M]^\top \in \mathbb{R}^{M \times d}$ is a set of parameters called inducing points in the input space. The augmented joint distribution over $\mathbf{u}, \mathbf{f}, \mathbf{f}^*$ is $p(\mathbf{f}, \mathbf{f}^*|\mathbf{u})p(\mathbf{u})$, where $p(\mathbf{u}) = \mathcal{N}(\mathbf{0}, \mathbf{K_{uu}})$ and $\mathbf{K_{uu}}$ denotes the kernel matrix of inducing points with the (i, j)th entry corresponding to $k(\mathbf{z}_i, \mathbf{z}_j)$. There is a long history in developing sparse approximations for GPs by making different independence assumptions for the conditional distribution $p(\mathbf{f}, \mathbf{f}^*|\mathbf{u})$ to reduce the computational cost (Quiñonero-Candela and Rasmussen, 2005). However, these methods made modifications to the GP prior and tended to suffer from degeneracy and overfitting problems.

Sparse variational GP methods (SVGP), first proposed in Titsias (2009) and later extended for minibatch training and nonconjugate likelihoods (Hensman et al., 2013, 2015a), provide an elegant solution to these problems. By reformulating the posterior inference problem as variational inference and restricting the variational distribution to be $q(\mathbf{f}, \mathbf{f}^*, \mathbf{u}) := q(\mathbf{u})p(\mathbf{f}, \mathbf{f}^*|\mathbf{u})$, the variational lower bound for minimizing $\mathrm{KL}\left[q(\mathbf{f}, \mathbf{f}^*, \mathbf{u})\|p(\mathbf{f}, \mathbf{f}^*, \mathbf{u}|\mathbf{y})\right]$ is simplified as:

$$\sum_{n=1}^{N} \mathbb{E}_{q(\mathbf{u})p(f_n|\mathbf{u})}\left[\log p(y_n|f_n)\right] - \mathrm{KL}\left[q(\mathbf{u})\|p(\mathbf{u})\right]. \tag{6}$$

For GP regression the bound has a collapsed form obtained by solving for the optimal $q(\mathbf{u})$ and plugging it into (6) (Titsias, 2009):

$$\log \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{Q_{ff}} + \sigma^2\mathbf{I}) - \frac{1}{2\sigma^2}\mathrm{tr}\left(\mathbf{K_{ff}} - \mathbf{Q_{ff}}\right), \tag{7}$$

where $\mathbf{Q_{ff}} = \mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{K_{uf}}$. Computing this objective requires $\mathcal{O}(M^2N + M^3)$ operations, in constrast to the $\mathcal{O}(N^3)$ complexity of exact inference. The inducing points $\mathbf{Z}$ can be learned as variational parameters by maximizing the lower bound. More generally, if we do not collapse $q(\mathbf{u})$ we obtain the uncollapsed bound suitable for minibatch training and non-Gaussian likelihoods (Hensman et al., 2013, 2015a).

Appendix B. Tighter Sparse Variational Bounds for GP Regression

Another direction to improve the variational distribution $q(\mathbf{u})p_\perp(\mathbf{f}_\perp)$ in SVGP is to add some dependence between $\mathbf{u}$ and $\mathbf{f}_\perp$. The best possible approximation of this family is obtained by the setting $q(\mathbf{u})$ to the optimal exact posterior conditional $q^*(\mathbf{u}) = p(\mathbf{u}|\mathbf{f}_\perp, \mathbf{y})$. The corresponding collapsed bound for GP regression can be derived by analytically marginalising out $\mathbf{u}$ from the joint model in $p(\mathbf{y}, \mathbf{u}, \mathbf{f}_\perp) = p(\mathbf{y}|\mathbf{f}_\perp + \mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{u})p(\mathbf{u})p_\perp(\mathbf{f}_\perp)$,

$$p(\mathbf{y}|\mathbf{f}_\perp) = \int p(\mathbf{y}|\mathbf{f}_\perp + \mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{u})p(\mathbf{u})\, d\mathbf{u} \tag{8}$$

$$= \mathcal{N}(\mathbf{y}|\mathbf{f}_\perp, \mathbf{Q_{ff}} + \sigma^2\mathbf{I}), \tag{9}$$

and then forcing the approximation $p_\perp(\mathbf{f}_\perp)$:

$$\mathbb{E}_{p_\perp(\mathbf{f}_\perp)} \log \mathcal{N}(\mathbf{y}|\mathbf{f}_\perp, \mathbf{Q_{ff}} + \sigma^2\mathbf{I}). \tag{10}$$

This bound has a closed-form as

$$\log \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{Q_{ff}} + \sigma^2\mathbf{I}) - \frac{1}{2}\mathrm{tr}\left[(\mathbf{Q_{ff}} + \sigma^2\mathbf{I})^{-1}(\mathbf{K_{ff}} - \mathbf{Q_{ff}})\right], \tag{11}$$

Applying the matrix inversion lemma to $(\mathbf{Q_{ff}} + \sigma^2\mathbf{I})^{-1}$, we have an equivalent form that can be directly compared with Eq. (7):

$$\log \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{Q_{ff}} + \sigma^2\mathbf{I}) - \frac{1}{2\sigma^2}\mathrm{tr}(\mathbf{K_{ff}} - \mathbf{Q_{ff}}) + \frac{1}{2\sigma^4}\mathrm{tr}\left[\mathbf{K_{fu}}(\mathbf{K_{uu}} + \sigma^{-2}\mathbf{K_{uf}}\mathbf{K_{fu}})^{-1}\mathbf{K_{uf}}(\mathbf{K_{ff}} - \mathbf{Q_{ff}})\right], \quad (12)$$

where the first two terms recover Eq. (7), suggesting this is a tighter bound than the Titsias (2009) bound. This bound is not amenable to large-scale datasets because of $O(N^2)$ storage requirements and $O(MN^2)$ computational time (dominated by the matrix multiplication $\mathbf{K_{uf}}\mathbf{K_{ff}}$). However, it is still of theoretical interest and can be applied to medium-sized regression datasets, just like the SGPR algorithm using the Titsias (2009) bound.

Appendix C. Orthogonal Decomposition

In this section we derive the orthogonal decomposition in the function space that corresponds to the reparameterization of $\mathbf{f}$ into $\mathbf{f}_\perp$ and $\mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{u}$. Recall that the GP prior is a distribution over functions $p : f \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$. Consider a subspace spanned by the kernel basis functions indexed by the inducing points $\mathbf{z}_1, \ldots, \mathbf{z}_M$:

$$V = \left\{ \sum_{j=1}^{M} \alpha_j k(\mathbf{z}_j, \cdot) \;\middle|\; \boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_M]^\top \in \mathbb{R}^M \right\}.$$

Samples from the GP prior can be decomposed (Cheng and Boots, 2016; Hensman et al., 2017) as

$$f = f_\parallel + f_\perp, \quad f_\parallel \in V \text{ and } f_\perp \perp V. \quad (13)$$

Since $f_\parallel \in V$, we let $f_\parallel = \sum_{j=1}^{M} \alpha_j^\parallel k(\mathbf{z}_j, \cdot)$. By $\langle f, f' \rangle_\mathcal{H} = \langle f_\parallel, f' \rangle_\mathcal{H}, \forall f' \in V$, where $\langle\rangle_\mathcal{H}$ is the inner product defined as $\langle f, k(\mathbf{x}, \cdot)\rangle_\mathcal{H} = f(\mathbf{x})$, we can solve for the coefficients: $\boldsymbol{\alpha}^\parallel = \mathbf{K_{uu}^{-1}}\mathbf{u}$. Interestingly, we can check that $f_\parallel$ is itself a sample from a GP with a zero mean function and covariance function $\mathrm{Cov}[f_\parallel(\mathbf{x}), f_\parallel(\mathbf{x}')] = \mathbf{k_u}(\mathbf{x})^\top\mathbf{K_{uu}^{-1}}\mathbf{k_u}(\mathbf{x}')$, where $\mathbf{k_u}(\mathbf{x}) = [k(\mathbf{z}_1, \mathbf{x}), \ldots, k(\mathbf{z}_M, \mathbf{x})]^\top$. Similarly we can show that $f_\perp$ is a sample from another GP and we denote these two independent GPs as $p_\parallel$ and $p_\perp$:

$$f_\parallel \sim p_\parallel \equiv \mathcal{GP}(0, \mathbf{k_u}(\mathbf{x})^\top\mathbf{K_{uu}^{-1}}\mathbf{k_u}(\mathbf{x})), \quad (14)$$

$$f_\perp \sim p_\perp \equiv \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}') - \mathbf{k_u}(\mathbf{x})^\top\mathbf{K_{uu}^{-1}}\mathbf{k_u}(\mathbf{x})). \quad (15)$$

Appendix D. The Collapsed SOLVE-GP Lower Bound

To intuitively understand the improvement over the standard SVGP methods, we derive a collapsed version of Eq. (4) for GP regression by seeking the optimal $q(\mathbf{u})$ that is independent of $\mathbf{f}_\perp, \mathbf{v}_\perp$. Then we can compare it to the Titsias (2009) bound.

First we rearrange the terms in the uncollapsed SOLVE-GP bound (Eq. (4)) as

$$\mathbb{E}_{q(\mathbf{u})}\left\{\mathbb{E}_{q(\mathbf{v}_\perp)p_\perp(\mathbf{f}_\perp|\mathbf{v}_\perp)}\left[\log \mathcal{N}\left(\mathbf{y}|\mathbf{f}_\perp + \mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{u}, \sigma^2\mathbf{I}\right)\right]\right\} - \mathrm{KL}\left[q(\mathbf{u})\|p(\mathbf{u})\right] - \mathrm{KL}\left[q(\mathbf{v}_\perp)\|p_\perp(\mathbf{v}_\perp)\right].$$

Let $\mathbf{m}_{\mathbf{f}_\perp}$ and $\mathbf{S}_{\mathbf{f}_\perp}$ denote the mean and covariance matrix of the variational predictive distribution in the orthogonal process:

$$q_\perp(\mathbf{f}_\perp) = \int q(\mathbf{v}_\perp) p_\perp(\mathbf{f}_\perp|\mathbf{v}_\perp) d\mathbf{v}_\perp. \tag{16}$$

We can compute them as

$$\mathbf{m}_{\mathbf{f}_\perp} = \mathbf{C}_{\mathbf{fv}}\mathbf{C}_{\mathbf{vv}}^{-1}\mathbf{m}_{\mathbf{v}}, \tag{17}$$

$$\mathbf{S}_{\mathbf{f}_\perp} = \mathbf{C}_{\mathbf{ff}} + \mathbf{C}_{\mathbf{fv}}\mathbf{C}_{\mathbf{vv}}^{-1}(\mathbf{S}_{\mathbf{v}} - \mathbf{C}_{\mathbf{vv}})\mathbf{C}_{\mathbf{vv}}^{-1}\mathbf{C}_{\mathbf{vf}}. \tag{18}$$

where $\mathbf{C}_{\mathbf{fv}} := \mathbf{K}_{\mathbf{fv}} - \mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K}_{\mathbf{uv}}$ is the kernel matrix of the orthogonal process on the training inputs and orthogonal inducing points and similarly for the other matrices. In the first term we can simplify the expectation over $\mathbf{v}_\perp$ and $\mathbf{f}_\perp$ as:

$$\begin{aligned}
&\mathbb{E}_{q(\mathbf{v}_\perp)p_\perp(\mathbf{f}_\perp|\mathbf{v}_\perp)} \log \mathcal{N}\left(\mathbf{y}|\mathbf{f}_\perp + \mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u}, \sigma^2\mathbf{I}\right) \\
&= \mathbb{E}_{q_\perp(\mathbf{f}_\perp)}\left[-\frac{N}{2}\log 2\pi - \frac{N}{2}\log\sigma^2 - \frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{f}_\perp - \mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u})^\top(\mathbf{y} - \mathbf{f}_\perp - \mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u})\right] \\
&= \left[-\frac{N}{2}\log 2\pi - \frac{N}{2}\log\sigma^2 - \frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{m}_{\mathbf{f}_\perp} - \mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u})^\top(\mathbf{y} - \mathbf{m}_{\mathbf{f}_\perp} - \mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u})\right] \\
&\quad - \mathbb{E}_{q_\perp(\mathbf{f}_\perp)}\left[\frac{1}{2\sigma^2}(\mathbf{f}_\perp - \mathbf{m}_{\mathbf{f}_\perp})^\top(\mathbf{f}_\perp - \mathbf{m}_{\mathbf{f}_\perp})\right] \\
&= \log\mathcal{N}(\mathbf{y}|\mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u} + \mathbf{m}_{\mathbf{f}_\perp}, \sigma^2\mathbf{I}) - \frac{1}{2\sigma^2}\mathrm{tr}(\mathbf{S}_{\mathbf{f}_\perp}).
\end{aligned} \tag{19}$$

Plugging into Eq. (D) and rearranging the terms, we have

$$\mathbb{E}_{q(\mathbf{u})}\left[\log\mathcal{N}(\mathbf{y}|\mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u} + \mathbf{m}_{\mathbf{f}_\perp}, \sigma^2\mathbf{I})\right] - \mathrm{KL}\left[q(\mathbf{u})\|p(\mathbf{u})\right] - \frac{1}{2\sigma^2}\mathrm{tr}(\mathbf{S}_{\mathbf{f}_\perp}) - \mathrm{KL}\left[q(\mathbf{v}_\perp)\|p_\perp(\mathbf{v}_\perp)\right].$$

Clearly the leading two terms form a variational lower bound of the joint distribution $\mathcal{N}(\mathbf{y}|\mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u} + \mathbf{m}_{\mathbf{f}_\perp}, \sigma^2\mathbf{I})p(\mathbf{u})$. The optimal $q(\mathbf{u})$ will turn it into the log marginal likelihood:

$$\log\int \mathcal{N}(\mathbf{y}|\mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u} + \mathbf{m}_{\mathbf{f}_\perp}, \sigma^2\mathbf{I})p(\mathbf{u})\, d\mathbf{u} = \log\mathcal{N}(\mathbf{y}|\mathbf{m}_{\mathbf{f}_\perp}, \mathbf{Q}_{\mathbf{ff}} + \sigma^2\mathbf{I}). \tag{20}$$

Plugging this back, we have the collapsed SOLVE-GP bound:

$$\log\mathcal{N}(\mathbf{y}|\mathbf{C}_{\mathbf{fv}}\mathbf{C}_{\mathbf{vv}}^{-1}\mathbf{m}_{\mathbf{v}}, \mathbf{Q}_{\mathbf{ff}} + \sigma^2\mathbf{I}) - \frac{1}{2\sigma^2}\mathrm{tr}(\mathbf{S}_{\mathbf{f}_\perp}) - \mathrm{KL}\left[\mathcal{N}(\mathbf{m}_{\mathbf{v}}, \mathbf{S}_{\mathbf{v}})\|\mathcal{N}(\mathbf{0}, \mathbf{C}_{\mathbf{vv}})\right], \tag{21}$$

This bound now can be tighter than the Titsias (2009) bound. For example, notice that when $q(\mathbf{v}_\perp)$ is equal to the prior $p_\perp(\mathbf{v}_\perp)$, i.e. $\mathbf{m}_{\mathbf{v}} = \mathbf{0}$ and $\mathbf{S}_{\mathbf{v}} = \mathbf{C}_{\mathbf{vv}}$, the bound in (21) reduces to the one in (7). Another interesting special case arises when the variational distribution has the same covariance matrix as the prior (i.e. $\mathbf{S}_{\mathbf{v}} = \mathbf{C}_{\mathbf{vv}}$), while the mean $\mathbf{m}_{\mathbf{v}}$ is learnable. Then the bound becomes

$$\log\mathcal{N}(\mathbf{y}|\mathbf{C}_{\mathbf{fv}}\mathbf{C}_{\mathbf{vv}}^{-1}\mathbf{m}_{\mathbf{v}}, \mathbf{Q}_{\mathbf{ff}} + \sigma^2\mathbf{I}) - \frac{1}{2\sigma^2}\mathrm{tr}\left(\mathbf{K}_{\mathbf{ff}} - \mathbf{Q}_{\mathbf{ff}}\right) - \frac{1}{2}\mathbf{m}_{\mathbf{v}}^\top\mathbf{C}_{\mathbf{vv}}^{-1}\mathbf{m}_{\mathbf{v}}. \tag{22}$$

Here we see that the second set of inducing variables $\mathbf{v}_\perp$ mostly determines the mean prediction over $\mathbf{y}$, which is zero in Titsias (2009) bound (Eq. (7)).

Appendix E. Predictions with SOLVE-GP

For the function values at test data points $\mathbf{X}^*$, the predictive density given by our approximate posterior is

$$p(\mathbf{f}^*|\mathbf{y}, \mathbf{X}^*) \approx \mathcal{N}(\mathbf{f}^*|\mathbf{m}_*, \mathbf{S}_*) := \int p(\mathbf{f}^*|\mathbf{K}_{*\mathbf{u}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u} + \mathbf{f}_\perp^*)q(\mathbf{u})p_\perp(\mathbf{f}_\perp^*|\mathbf{v}_\perp)q(\mathbf{v}_\perp)\, d\mathbf{u}d\mathbf{v}_\perp.$$

Then for $q(\mathbf{u}) = \mathcal{N}(\mathbf{m}_\mathbf{u}, \mathbf{S}_\mathbf{u})$, the predicted mean and covariance are

$$\mathbf{m}_* = \mathbf{K}_{*\mathbf{u}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{m}_\mathbf{u} + \mathbf{C}_{*\mathbf{v}}\mathbf{C}_{\mathbf{vv}}^{-1}\mathbf{m}_\mathbf{v}. \tag{23}$$

$$\mathbf{S}_* = \mathbf{K}_{*\mathbf{u}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{S}_\mathbf{u}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K}_{\mathbf{u}*} + \mathbf{C}_{**} - \mathbf{C}_{*\mathbf{v}}\mathbf{C}_{\mathbf{vv}}^{-1}(\mathbf{C}_{\mathbf{vv}} - \mathbf{S}_\mathbf{v})\mathbf{C}_{\mathbf{vv}}^{-1}\mathbf{C}_{\mathbf{v}*}. \tag{24}$$

Appendix F. SOLVE-GP as Structured Covariance Approximation

To obtain this structured covariance matrix, we need to express our variational approximation w.r.t. the original GP. Let $\mathbf{v} = f(\mathbf{O})$ denote the function outputs at the orthogonal inducing points. We have the following relationship between $\mathbf{u}, \mathbf{v}$ and $\mathbf{u}, \mathbf{v}_\perp$:

$$\begin{bmatrix}\mathbf{u}\\ \mathbf{v}\end{bmatrix} = \begin{bmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{K}_{\mathbf{vu}}\mathbf{K}_{\mathbf{uu}}^{-1} & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{u}\\ \mathbf{v}_\perp\end{bmatrix}. \tag{25}$$

By change-of-variable we compute the joint variational distribution over $\mathbf{u}$ and $\mathbf{v}$ that corresponds to the factorized $q(\mathbf{u})q(\mathbf{v}_\perp)$:

$$\begin{aligned} q(\mathbf{u}, \mathbf{v}) &= q(\mathbf{u})q(\mathbf{v}_\perp)\left(\det\begin{vmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{K}_{\mathbf{vu}}\mathbf{K}_{\mathbf{uu}}^{-1} & \mathbf{I}\end{vmatrix}\right)^{-1} \\ &= \mathcal{N}(\mathbf{u}|\mathbf{m}_\mathbf{u}, \mathbf{S}_\mathbf{u})\mathcal{N}(\mathbf{v} - \mathbf{K}_{\mathbf{vu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u}|\mathbf{m}_\mathbf{v}, \mathbf{S}_\mathbf{v}). \end{aligned} \tag{26}$$

Using Gaussian identities we can show that $q(\mathbf{u}, \mathbf{v})$ is a Gaussian distribution with mean $\mathbf{m}_{\mathbf{u},\mathbf{v}} = \left[\mathbf{m}_\mathbf{u}, \mathbf{m}_\mathbf{v} + \mathbf{K}_{\mathbf{vu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{m}_\mathbf{u}\right]^\top$ and covariance matrix

$$\mathbf{S}_{\mathbf{u},\mathbf{v}} = \begin{bmatrix}\mathbf{S}_\mathbf{u} & \mathbf{S}_\mathbf{u}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K}_{\mathbf{uv}}\\ \mathbf{K}_{\mathbf{vu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{S}_\mathbf{u} & \mathbf{S}_\mathbf{v} + \mathbf{K}_{\mathbf{vu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{S}_\mathbf{u}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K}_{\mathbf{uv}}\end{bmatrix}.$$

Interestingly, despite $\mathbf{S}_{\mathbf{u},\mathbf{v}}$ being a $(M + M_2) \times (M + M_2)$ matrix, it can be inverted with $\mathcal{O}(M^3 + M_2^3)$ computation, which gives a 4x speed-up over a fully parameterized multivariate Gaussian distribution for $q(\mathbf{u}, \mathbf{v})$ when $M = M_2$.

Appendix G. Extensions

A direct extension of SOLVE-GP is that we can introduce more than two sets of inducing points by repeatedly applying the orthogonal decomposition, however this adds more complexity to the implementation. Below we show that the SOLVE-GP framework can be easily extended to different GP models where the standard SVGP method applies.

Inter-domain Inducing Points and Convolutional GP. Similar to SVGP methods, SOLVE-GP can deal with inter-domain inducing points (Lázaro-Gredilla and Figueiras-Vidal, 2009) which lie in a different domain from the input space. The inducing variables

$\mathbf{u}$, which we used to represent outputs of the GP at the inducing points, is now defined as $\mathbf{u} = g(\mathbf{Z}) := [g(\mathbf{z}_1), \ldots, g(\mathbf{z}_M)]^\top$, where g is a different function from f that takes inputs in the domain of inducing points. In convolutional GPs (van der Wilk et al., 2017), the input domain is the space of images, while the inducing points are in the space of image patches. The convolutional GP function is defined as

$$f(\mathbf{x}) = \sum_p w_p g\left(\mathbf{x}^{[p]}\right), \tag{27}$$

where $\mathbf{x}^{[p]}$ is the pth patch in the image $\mathbf{x}$; $\mathbf{w} = [w_1, \ldots, w_P]^\top$ are the assigned weights to different patches. In SOLVE-GP, we can choose either $\mathbf{Z}$, $\mathbf{O}$, or both to be inter-domain as long as we can compute the covariance between $\mathbf{u}, \mathbf{v}$ and $\mathbf{f}$. When applied to convolutional GP models, we set both $\mathbf{Z}$ and $\mathbf{O}$ to be a collection of image patches. Examples of the covariance matrices we need for this model include $\mathbf{K_{vf}}$ and $\mathbf{K_{vu}}$ (used for $\mathbf{C_{vv}}$). They can be computed as

$$[\mathbf{K_{vf}}]_{ij} = \mathrm{Cov}[g(\mathbf{o}_i), f(\mathbf{x}_j)] = \sum_p w_p k(\mathbf{o}_i, \mathbf{x}_j^{[p]}), \tag{28}$$

$$[\mathbf{K_{vu}}]_{ij} = \mathrm{Cov}[g(\mathbf{o}_i), g(\mathbf{z}_j)] = k(\mathbf{o}_i, \mathbf{z}_j). \tag{29}$$

Deep GP. We show that we can integrate SOLVE-GP with popular doubly stochastic variational inference algorithms for deep GPs (Salimbeni and Deisenroth, 2017). The joint distribution of a deep GP model with inducing variables in all layers is

$$p(\mathbf{y}, \mathbf{f}^{1:L}, \mathbf{u}^{1:L}) = p(\mathbf{y}|\mathbf{f}^L) \prod_{\ell=1}^{L} \left[p(\mathbf{f}^\ell|\mathbf{u}^\ell, \mathbf{f}^{\ell-1}) p(\mathbf{u}^\ell)\right],$$

where we define $\mathbf{f}^0 = \mathbf{X}$ and $\mathbf{f}^\ell$ is the output of the ℓth-layer GP. The doubly stochastic algorithm applies SVGP methods to each layer conditioned on samples from the variational distribution in the previous layer. The variational distribution over $\mathbf{u}^{1:L}, \mathbf{f}^{1:L}$ is $q(\mathbf{f}^{1:L}, \mathbf{u}^{1:L}) = \prod_{\ell=1}^{L} \left[p(\mathbf{f}^\ell|\mathbf{u}^\ell, \mathbf{f}^{\ell-1}) q(\mathbf{u}^\ell)\right]$. This gives a similar objective as in the single layer case (Eq. (6)):

$$\mathbb{E}_{q(\mathbf{f}^L)}\left[\log p(\mathbf{y}|\mathbf{f}^L)\right] - \sum_{\ell=1}^{L} \mathrm{KL}\left[q(\mathbf{u}^\ell) \| p(\mathbf{u}^\ell)\right],$$

where $q(\mathbf{f}^L) = \int \prod_{\ell=1}^{L} \left[p(\mathbf{f}^\ell|\mathbf{u}^\ell, \mathbf{f}^{\ell-1}) q(\mathbf{u}^\ell) d\mathbf{u}^\ell\right] d\mathbf{f}^{1:L-1}$. Extending this using SOLVE-GP is straightforward by introducing orthogonal inducing variables $\mathbf{v}_\perp^{1:L}$ for all layers, which gives the lower bound:

$$\mathbb{E}_{q(\mathbf{u}^L, \mathbf{f}_\perp^L)}\left[\log p(\mathbf{y}|\mathbf{f}_\perp^L + \mathbf{K}_{\mathbf{fu}}^L (\mathbf{K}_{\mathbf{uu}}^L)^{-1} \mathbf{u}^L)\right] - \sum_{\ell=1}^{L} \left\{\mathrm{KL}\left[q(\mathbf{u}^\ell) \| p(\mathbf{u}^\ell)\right] + \mathrm{KL}\left[q(\mathbf{v}_\perp^\ell) \| p_\perp(\mathbf{v}_\perp^\ell)\right]\right\},$$

where $q(\mathbf{u}^L, \mathbf{f}_\perp^L)$ is defined as:

$$q(\mathbf{u}^L, \mathbf{f}_\perp^L) = \int \prod_{\ell=1}^{L} \left[p_\perp(\mathbf{f}_\perp^\ell|\mathbf{v}_\perp^\ell, \mathbf{f}_\perp^{\ell-1}, \mathbf{u}^{\ell-1}) q(\mathbf{v}_\perp^\ell) q(\mathbf{u}^\ell) d\mathbf{u}^\ell d\mathbf{v}_\perp^\ell\right] \prod_{\ell=1}^{L-1} d\mathbf{f}_\perp^\ell. \tag{30}$$

Appendix H. Related Work

Many approximate algorithms have been proposed to overcome the computational limitations of GPs. The simplest of these are based on sub-sampling data points, which include the naive subset-of-data training (Rasmussen and Williams, 2006) as well as the Nyström approximation to the kernel matrix (Williams and Seeger, 2001). Better sparse approximations can be constructed by learning a set of inducing points to summarize the entire dataset. These works can be divided into sparse approximations to the GP prior (SoR, DTC, FITC, etc.) (Quiñonero-Candela and Rasmussen, 2005), and sparse variational methods (Titsias, 2009; Hensman et al., 2013, 2015a).

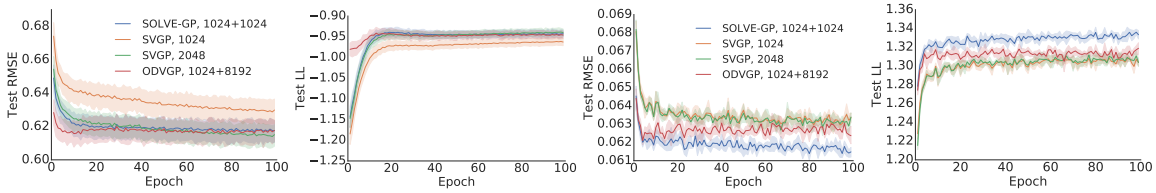
Recently there have been many attempts to reduce the $\mathcal{O}(M^3)$ complexity of computing $\mathbf{K}_{\mathbf{uu}}^{-1}$ for using a large set of inducing points. A notable line of work (Wilson and Nickisch, 2015; Evans and Nair, 2018; Gardner et al., 2018) involves imposing grid structures on the locations of $\mathbf{Z}$ to perform fast structure exploiting computations. However, to get such computational benefits $\mathbf{Z}$ need to be fixed due to the structure constraints, which often suffers from curse-of-dimensionality in the input space. Another direction for allowing the use of more inducing points is the decoupled method, first proposed in Cheng and Boots (2017), where two different set of inducing points are used for modeling the mean and the covariance function. This gives linear complexity in the number of mean inducing points which allows using many more of them. Despite the increasing interest in decoupled inducing points (Havasi et al., 2018; Salimbeni et al., 2018), the method has not been well-understood due to its complexity. We found that our SOLVE-GP framework is closely connected to a recent development of decoupled methods: the orthogonal decoupled variational GP (ODVGP) (Salimbeni et al., 2018), as explained next.

Connection with decoupled inducing points. If we set the β and γ inducing points in ODVGP (Salimbeni et al., 2018) to be $\mathbf{Z}$ and $\mathbf{O}$, their approach becomes equivalent to using the following variational distribution:

$$q'(\mathbf{u},\mathbf{v}) = \mathcal{N}\left(\begin{bmatrix}\mathbf{m_u}\\ \mathbf{m_v}+\mathbf{K_{vu}}\mathbf{K_{uu}^{-1}}\mathbf{m_u}\end{bmatrix}, \begin{bmatrix}\mathbf{S_u} & \mathbf{S_u}\mathbf{K_{uu}^{-1}}\mathbf{K_{uv}}\\ \mathbf{K_{vu}}\mathbf{K_{uu}^{-1}}\mathbf{S_u} & \mathbf{K_{vv}}+\mathbf{K_{vu}}\mathbf{K_{uu}^{-1}}(\mathbf{S_u}-\mathbf{K_{uu}})\mathbf{K_{uu}^{-1}}\mathbf{K_{uv}}\end{bmatrix}\right).$$

By comparing this covariance matrix to $\mathbf{S}_{\mathbf{u},\mathbf{v}}$, we can see that we generalize their method by introducing $\mathbf{S_v}$, which replaces the original residual $\mathbf{K_{vv}} - \mathbf{K_{vu}}\mathbf{K_{uu}^{-1}}\mathbf{K_{uv}}$, so that we allow more flexible covariance modeling while still keeping the block structure that facilitates cheap inverse. This implies that ODVGP is a special case of SOLVE-GP by restricting $q(\mathbf{v}_\perp)$ to have the same covariance $\mathbf{C_{vv}}$ as the prior.

Besides inducing points, another way to construct sparse approximations is by examining the weight space representation of GPs, i.e., Bayesian linear regression in the kernel feature space (Rasmussen and Williams, 2006). Relevance vector machines (Tipping, 2001) use finite basis functions as features, while sparse spectrum GP (Lázaro-Gredilla et al., 2010) uses random Fourier features. These two methods approximate the prior distribution. It is also possible to use the weight-space representation to approximate the posterior distribution of GPs by variational inference (Shi et al., 2019).



(*a*) Protein (*b*) HouseElectric

Figure 3: Changes of test RMSE and predictive log likelihoods during training, on (a) Protein; (b) HouseElectric.

Table 3: Test log likelihood values of regression datasets.

		Kin40k	**Protein**	**KeggDirected**	**KEGGU**	**3dRoad**	**Song**	**Buzz**	**HouseElectric**
	N	25,600	29,267	31,248	40,708	278,319	329,820	373,280	1,311,539
	d	8	9	20	27	3	90	77	9
SVGP	1024	0.0938(0.0056)	-0.9628(0.0124)	0.9673(0.0111)	0.6784(0.0085)	-0.6981(0.0051)	-1.1934(0.0019)	-0.0793(0.0040)	1.3036(0.0044)
	1536	0.1287(0.0067)	-0.9490(0.0116)	0.9442(0.0133)	0.6734(0.0100)	-0.6744(0.0056)	-1.1927(0.0018)	-0.0786(0.0041)	1.3040(0.0069)
ODVGP	$1024+1024$	0.1372(0.0061)	-0.9558(0.0116)	-0.1988(0.1499)	0.1054(0.0739)	-0.6644(0.0062)	-1.1932(0.0016)	-0.0783(0.0026)	1.3170(0.0052)
	$1024+8096$	0.1444(0.0040)	-0.9460(0.0108)	-0.1364(0.1416)	0.1091(0.0747)	**-0.6568**(0.0067)	-1.1929(0.0016)	-0.0789(0.0029)	1.3188(0.0086)
SOLVE-GP	$1024+1024$	**0.1868**(0.0050)	**-0.9429**(0.0110)	**0.9730**(0.0073)	**0.6804**(0.0074)	-0.6587(0.0034)	**-1.1918**(0.0019)	**-0.0711**(0.0033)	**1.3332**(0.0058)
SVGP	2048*	0.1374(0.0057)	**-0.9402**(0.0112)	0.9071(0.0071)	0.6648(0.0099)	-0.6689(0.0049)	-1.1924(0.0018)	-0.0788(0.0039)	1.3036(0.0056)

Appendix I. Additional Results

I.1. Regression Benchmarks

We follow the settings in Wang et al. (2019), where the results of exact GP regression have been reported on these datasets with distributed training (Wang et al., 2019). We implemented SVGP with $M = 1024, 2048$ inducing points, ODVGP and SOLVE-GP ($M = 1024, M_2 = 1024$), as well as SVGP with $M = 1536$ inducing points, which has roughly the same training time per iteration on GPU as the SOLVE-GP objective. An attractive property of ODVGP is that by restricting the covariance of $q(\mathbf{v}_\perp)$ to be the same as the prior covariance $\mathbf{C}_{\mathbf{vv}}$, it can use far larger M_2 because the complexity is linear with M_2 and sub-sampling of orthogonal inducing points can be used for each gradient update. Thus for a fair comparison, we also include ODVGP ($M_2 = 8096$), where in each iteration a subset of size 1024 is sampled from the orthogonal inducing points to estimate the gradient.

We report the predictive log likelihoods on test data in Table 3. Due to space limit we leave the results on two small datasets (Elevators, Bike) in App. I. We can see that the performance of SOLVE-GP is competitive with the 4x more expensive SVGP ($M = 2048$). Perhaps surprisingly, while SOLVE-GP uses a less flexible covariance in the variational distribution, it often outperforms SVGP ($M = 2048$). We believe this is due to optimization difficulties introduced by the 2048×2048 covariance matrix. We shall analyze this on the HouseElectric dataset later. On most datasets, using a large number of additional inducing points for modeling the mean function did improve the performance, as shown by comparison between the ODVGP ($M_2 = 1024$) and ODVGP ($M_2 = 8096$). Though more flexible covariance modeling seems to be more essential, as SOLVE-GP outperforms ODVGP ($M_2 = 8096$) on all datasets except 3dRoad.

Table 4: Test RMSE values of regression datasets.

		Kin40k	**Protein**	**KeggDirected**	**KEGGU**	**3dRoad**	**Song**	**Buzz**	**HouseElectric**
	N	25,600	29,267	31,248	40,708	278,319	329,820	373,280	1,311,539
	d	8	9	20	27	3	90	77	9
SVGP	1024	0.1933(0.0021)	0.6298(0.0092)	0.0975(0.0059)	0.1233(0.0021)	0.4825(0.0027)	0.7973(0.0018)	0.2628(0.0013)	0.0634(0.0003)
	1536	0.1824(0.0028)	0.6208(0.0086)	0.0981(0.0054)	0.1232(0.0018)	0.4703(0.0032)	0.7967(0.0017)	0.2627(0.0012)	0.0633(0.0003)
ODVGP	$1024 + 1024$	0.1827(0.0025)	0.6247(0.0087)	0.1764(0.0268)	0.1561(0.0089)	0.4665(0.0031)	0.7975(0.0016)	0.2627(0.0014)	0.0624(0.0004)
	$1024 + 8096$	0.1798(0.0014)	0.6176(0.0082)	0.1573(0.0194)	0.1567(0.0091)	**0.4620**(0.0037)	0.7973(0.0016)	0.2629(0.0017)	0.0624(0.0006)
SOLVE-GP	$1024 + 1024$	**0.1721**(0.0019)	**0.6175**(0.0083)	**0.0951**(0.0046)	**0.1229**(0.0016)	0.4639(0.0012)	**0.7964**(0.0018)	**0.2608**(0.0017)	**0.0615**(0.0003)
SVGP	2048*	0.1771(0.0026)	**0.6151**(0.0084)	0.0995(0.0060)	0.1236(0.0016)	0.4668(0.0030)	**0.7964**(0.0018)	0.2626(0.0011)	0.0634(0.0003)

In Fig. 3 we plot the change of test RMSE and test log likelihoods during training on Protein and HouseElectric. Interestingly, on both datasets ODVGP ($M_2 = 8096$) gets very good performance at the beginning, then slowly converges to less competitive results. The beginning stage is likely where the additional inducing points give good predictions but are not in the best configuration for maximizing the training lower bounds. This phenomenon is also observed on Elevators and Kin40k. We believe such mismatch between the training lower bound and predictive performance is caused by fixing the covariance matrix of $q(\mathbf{v}_\perp)$ to the prior covariance. On HouseElectric, SVGP ($M = 2048$) does not improve over SVGP ($M = 1024$) and is outperformed by SOLVE-GP. As previously mentioned, this might be due to difficulties when optimising large covariance matrices. To verify this, we tried the "whitening" trick (Murray and Adams, 2010; Hensman et al., 2015b) that is often used to improve the optimization of SVGP methods by reducing the correlation in the posterior distribution of $\mathbf{u}$. As expected, the result of SVGP ($M = 2048$) then becomes similar to SOLVE-GP, outperforming all other methods.

Due to space limit in the main text, we report the test predictive log likelihood values on Elevators and Bike in Table 5. We also include the Root Mean Squared Error (RMSE) on test data in Table 4 and Table 6.

Table 5: Test log likelihoods values of Elevators and Bike.

		Elevators	**Bike**
	N	10,623	11,122
	d	18	17
SVGP	1024	-0.5164(0.0144)	-0.2176(0.0123)
	1536	-0.5108(0.0149)	-0.2030(0.0140)
ODVGP	$1024 + 1024$	-0.5184(0.0145)	-0.1906(0.0140)
	$1024 + 8096$	-0.5231(0.0137)	**-0.1860**(0.0128)
SOLVE-GP	$1024 + 1024$	**-0.5090**(0.0150)	-0.1891(0.0130)
SVGP	2048*	**-0.5075**(0.0152)	-0.1933(0.0138)

I.2. (Deep) Convolutional Gaussian Processes

We include here the full tables for CIFAR-10 classification, where we also report the accuracies and predictive log likelihoods on the training data. Table 7 contains the results by convolutional GPs, while Table 8 and Table 9 include results of 2/3-layer deep convolutional GPs.

Table 6: Test RMSE values of Elevators and Bike.

		Elevators	**Bike**
	N	10,623	11,122
	d	18	17
SVGP	1024	0.3975(0.0092)	0.2831(0.0058)
	1536	0.3959(0.0092)	0.2789(0.0065)
ODVGP	$1024 + 1024$	0.3974(0.0093)	0.2724(0.0064)
	$1024 + 8096$	0.3992(0.0091)	**0.2703**(0.0063)
SOLVE-GP	$1024 + 1024$	**0.3950**(0.0094)	0.2724(0.0060)
SVGP	2048*	**0.3949**(0.0093)	0.2756(0.0063)

Table 7: 1-layer CIFAR-10 classification.

		Train Acc	Train LL	Test Acc	Test LL	Time
SVGP	1000	77.81%	-1.36	66.07%	-1.59	0.241 s/iter
	1600	78.44%	-1.26	67.18%	-1.54	0.380 s/iter
SOLVE-GP	$1000 + 1000$	79.32%	-1.20	**68.19%**	**-1.51**	0.370 s/iter
SVGP	2000*	79.46%	-1.22	68.06%	**-1.48**	0.474 s/iter

Table 8: 2-layer CIFAR-10 classification.

	Inducing Points	Train Acc	Train LL	Test Acc	Test LL	Time
SVGP	$384, 1K$	84.86%	-0.82	76.35%	-1.04	0.392 s/iter
SOLVE-GP	$384 + 384, 1K + 1K$	87.59%	-0.72	**77.8%**	**-0.98**	0.657 s/iter
SVGP	$768, 2K$*	87.25%	-0.74	77.46%	**-0.98**	1.104 s/iter

Table 9: 3-layer CIFAR-10 classification.

	Inducing Points	Train Acc	Train LL	Test Acc	Test LL	Time
SVGP	$384, 384, 1K$	87.7%	-0.67	78.76%	-0.88	0.418 s/iter
SOLVE-GP	$384 + 384, 384 + 384, 1K + 1K$	89.88%	-0.57	**80.3%**	**-0.79**	0.752 s/iter
SVGP	$768, 768, 2000$*	90.01%	-0.58	**80.33%**	-0.82	1.246 s/iter