# Fast and Scalable Spike and Slab Variable Selection in High-Dimensional Gaussian Processes

**Hugh Dance**
University College London

**Brooks Paige**
University College London

## Abstract

Variable selection in Gaussian processes (GPs) is typically undertaken by thresholding the inverse lengthscales of automatic relevance determination kernels, but in high-dimensional datasets this approach can be unreliable. A more probabilistically principled alternative is to use spike and slab priors and infer a posterior probability of variable inclusion. However, existing implementations in GPs are very costly to run in both high-dimensional and large-$n$ datasets, or are only suitable for unsupervised settings with specific kernels. As such, we develop a fast and scalable variational inference algorithm for the spike and slab GP that is tractable with arbitrary differentiable kernels. We improve our algorithm's ability to adapt to the sparsity of relevant variables by Bayesian model averaging over hyperparameters, and achieve substantial speed ups using zero temperature posterior restrictions, dropout pruning and nearest neighbour minibatching. In experiments our method consistently outperforms vanilla and sparse variational GPs whilst retaining similar runtimes (even when $n = 10^6$) and performs competitively with a spike and slab GP using MCMC but runs up to 1000 times faster.

## 1 INTRODUCTION

Gaussian processes (GPs) are a powerful Bayesian nonparametric approach to regression used widely in machine learning and statistics. GPs enable the user to directly model the distribution over a function $f(\cdot)$ they aim to approximate. The covariance function $k(x, x')$ of the GP enables properties such as stationarity and smoothness to be encoded directly, and flexible function classes can be represented with few (hyper)parameters (Rasmussen and Williams, 2006).

To infer variable relevance in GPs 'automatic relevance determination' (ARD) kernels are often implemented, which use per-dimension inverse lengthscale parameters $\{\theta_j\}_{j=1}^d$ as a relevance measure. Learning $\{\theta_j\}_{j=1}^d$ by maximising the marginal likelihood (ML-II) often sees $\theta_j \to 0$ for irrelevant dimensions due to the marginal likelihood's intrinsic complexity penalty. This makes it trivial to select relevant variables by thresholding $\{\theta_j\}_{j=1}^d$ (Rasmussen and Williams, 2006). However, we find this shrinkage does not hold sufficiently in high-dimensional input designs, making it difficult to perform variable selection reliably. This agrees with recent evidence (Ober et al., 2021; Mohammed and Cawley, 2017) that the marginal likelihood can overfit in heavily parameterised regimes.

An alternative approach is to place spike and slab priors on the inverse lengthscales and infer a per-variable posterior inclusion probability (PIP). Existing implementations in GPs perform excellently but either use Markov Chain Monte Carlo (MCMC) schemes which are very costly to run in high-dimensional or large-$n$ datasets (Savitsky et al., 2011), or are only suitable for latent input selection in the unsupervised setting (and when using specific kernels) (Dai et al., 2015).

**Our Contribution**: We develop a fast and scalable variational inference algorithm for GPs with spike and slab variable selection priors, which we call the spike and slab variational Gaussian process (SSVGP). By using a continuous approximation to the Dirac spike and a structured variational approximation to the posterior, we are able to derive an otherwise intractable approximate co-ordinate ascent variational inference (CAVI) (Blei et al., 2017) algorithm with similar complexity to ML-II training. We subsequently improve our algorithm's ability to adapt to the sparsity in the data by Bayesian model averaging over spike and slab

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151. Copyright 2022 by the author(s).

hyperparameter values. We achieve substantial speed-ups and $\mathcal{O}(nf(n)d)$ complexity with $f(n) \in [\log(n), n]$ by using a combination of zero temperature posterior restrictions, dropout pruning, and nearest neighbour minibatching. Our method can be used to improve GP regression predictive accuracy in high-dimensional scenarios, or enable probabilistically principled variable selection through inference on the PIPs - without the usual spike and slab implementation costs.

We find across a range of experiments that our SSVGP consistently outperforms standard GP regression and sparse variational GP regression (Titsias, 2009; Hensman et al., 2013) in terms of predictive accuracy whilst keeping similar runtimes. Our method also outperformed benchmark variable selection algorithms and matched the performance of a spike and slab GP with MCMC (Savitsky et al., 2011), but can run up to 1000 times faster. Code for our method is provided at the Github repository https://github.com/HWDance/SSVGP.

## 2 BACKGROUND AND RELATED WORK

In this section we review GP regression, spike and slab priors, variational inference and implementation challenges with spike and slab GPs, as well as related work.

### 2.1 Gaussian Process Regression

A Gaussian process is a random function $f \sim \mathcal{GP}(m(\boldsymbol{x}), k_{\boldsymbol{\alpha}}(\boldsymbol{x}, \boldsymbol{x}'))$ with mean function $m(\cdot) : \mathcal{X} \to \mathbb{R}$ and positive semi-definite covariance function ('kernel') $k(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, such that any finite subset of points are multivariate Gaussian: $\boldsymbol{f} = [f(\boldsymbol{x}_1), ..., f(\boldsymbol{x}_n)] \sim \mathcal{N}(\boldsymbol{m}, K_{XX})$, $[\boldsymbol{m}]_i = m(\boldsymbol{x}_i)$, $[K_{XX}]_{ij} = k_{\boldsymbol{\alpha}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Here $\boldsymbol{\alpha}$ are kernel parameters.

Given a dataset $\mathcal{D} = (\boldsymbol{y}, X) \in \mathbb{R}^n \times \mathbb{R}^{n \times d}$, we define a Gaussian process model as follows

$$\boldsymbol{y}|\boldsymbol{f} \sim p(\boldsymbol{y}|\boldsymbol{f}) \quad (1)$$
$$f(\cdot)|\boldsymbol{\alpha} \sim \mathcal{GP}(m(\boldsymbol{x}), k_{\boldsymbol{\alpha}}(\boldsymbol{x}, \boldsymbol{x}')) \quad (2)$$

In standard GP regression (GPR), the observation model is a Gaussian centred at $\boldsymbol{f}$: $p(\boldsymbol{y}|\boldsymbol{f}) = \prod_{i=1}^n \mathcal{N}(y_i|f_i, \sigma^2)$. In this case the marginal likelihood has closed form $p_{\boldsymbol{\alpha},\sigma}(\boldsymbol{y}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{m}, K_{XX} + \sigma^2 I)$ and can be maximised to learn kernel parameters $\boldsymbol{\alpha}$ and noise variance $\sigma^2$ (aka ML-II optimisation). The predictive distribution at any $\boldsymbol{x}_* \in \mathbb{R}^d$ is also Gaussian with closed form moments (Rasmussen and Williams, 2006).

To do variable selection in GPR, automatic relevance determination (ARD) kernels (MacKay et al., 1998) are traditionally used. Most popular ARD kernels (e.g. squared exponential (SE), Matérn, Cauchy) fall into a subclass of anisotropic stationary kernels of the form

$$k_{\tau,\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}') = \tau h(||\boldsymbol{\theta} \odot (\boldsymbol{x} - \boldsymbol{x}')||_2). \quad (3)$$

Here $\boldsymbol{\theta} \in \mathbb{R}_+^d$ are per-dimension inverse lengthscales, which are used a measure of variable relevance (assuming inputs are scaled to have unit variance), $\tau \in \mathbb{R}_{>0}$ is the scale parameter of the kernel and $h(\cdot)$ is a continuous (often monotonically decreasing) function. ML-II optimisation often sees $\theta_j \to 0^+$ for irrelavent dimensions due to the marginal likelihood's intrinsic complexity penalty (Rasmussen and Williams, 2006), and so variable selection can be undertaken by hard-thresholding $\{\theta_j\}_{j=1}^d$. However, we demonstrate that in high-dimensional designs inferring the appropriate threshold can be challenging, and failing to threshold reliably can substantially reduce predictive accuracy.

We note alternative measures of variable relevance have been developed using posterior predictive sensitivity (Paananen et al., 2019), however these approaches suffer the same drawback of needing to infer an appropriate threshold for binary selection decisions.

### 2.2 Spike and Slab Priors

Spike and slab priors (Mitchell and Beauchamp, 1988) are considered a gold standard for high-dimensional variable selection problems due to their excellent properties (Narisetty and He, 2014; Castillo et al., 2015), empirical performance and interpretability. Given a model $p(\boldsymbol{y}|\boldsymbol{\theta})$ with per-dimension 'relevance' parameters $\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^d$, a spike and slab prior augments the model with binary inclusion variables $\boldsymbol{\gamma} \in \{0, 1\}^d$

$$p(\boldsymbol{\theta}, \boldsymbol{\gamma}) = \prod_{j=1}^d [\mathcal{P}_{slab}(\theta_j)\pi]^{\gamma_j} [\delta_0(\theta_j)(1-\pi)]^{1-\gamma_j} \quad (4)$$

Here $\mathcal{P}_{slab}(\cdot)$ is the slab distribution (usually set to be approximately uniform over $\text{dom}(\theta_j)$), the spike distribution is a Dirac point mass at zero, and $\pi$ is the prior probability of inclusion. Since $p(\theta_j = 0|\gamma_j = 0) = 1$, variable relevance can be inferred through the posterior inclusion probability (PIP) $p(\gamma_j = 1|\boldsymbol{y})$ for input dimension $j$

$$p(\gamma_j = 1|\boldsymbol{y}) = \frac{\sum_{\boldsymbol{\gamma}_{\neg j}} \int_{\boldsymbol{\theta}} p(\boldsymbol{y}|\boldsymbol{\theta}) p(\boldsymbol{\theta}, \boldsymbol{\gamma})}{\sum_{\boldsymbol{\gamma}} \int_{\boldsymbol{\theta}} p(\boldsymbol{y}|\boldsymbol{\theta}) p(\boldsymbol{\theta}, \boldsymbol{\gamma})}$$

The posterior mode $\boldsymbol{\gamma}^* = \text{argmax}_{\boldsymbol{\gamma} \in \{0,1\}^d} \{p(\boldsymbol{\gamma}|\boldsymbol{y})\}$ can also be used for variable selection (if available), since this by definition selects the posterior likeliest model. Unfortunately the posterior $p(\boldsymbol{\gamma}|\boldsymbol{y})$ is generally intractable but can be approximated using Markov Chain Monte Carlo (MCMC) (Brooks et al., 2011) or variational inference (Blei et al., 2017), of which the latter we briefly review below.

### 2.3 Variational Inference and Intractabilities in Spike and Slab GPs

Given a generative model $p(\mathcal{D}|\mathcal{Z})p(\mathcal{Z})$ over latent variables $\mathcal{Z}$ and data $\mathcal{D}$, variational inference can be used to approximate the posterior $p(\mathcal{Z}|\mathcal{D})$ (if intractable) with a surrogate $q(\mathcal{Z}) \in \mathcal{Q}$, where $\mathcal{Q}$ is restricted by parametric or independence assumptions. The optimal $q^* \in \mathcal{Q}$ is chosen by minimising the Kullback-Leibler divergence $KL[q(\mathcal{Z})||p(\mathcal{Z}|\mathcal{D})]$, which is equivalent to maximising the evidence lower bound (or Free Energy) $\langle \log p(\mathcal{D}|\mathcal{Z})\rangle_{q(\mathcal{Z})} - KL[q(\mathcal{Z})||p(\mathcal{Z})]$ (Blei et al., 2017).

Variational inference appears a natural choice to achieve fast and scalable inference in GPs with spike and slab priors. Factorising the posterior over dimensions $q(\boldsymbol{\theta}, \boldsymbol{\gamma}) = \prod_{j=1}^{d} q(\theta_j, \gamma_j)$ reduces the variable selection problem from a $2^d$ search problem to optimising $\mathcal{O}(d)$ variational parameters (after optimisation marginal PIPs and the posterior mode can be recovered trivially due to the factorisation). However, even in standard GPR models, optimisation remains either challenging or entirely intractable depending on the independence assumptions used. To see this, consider the evidence lower bound induced for a GPR model with Gaussian marginal likelihood $p(\boldsymbol{y}|\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{m}, K_{XX}(\boldsymbol{\theta}) + \sigma^2 I)$, spike and slab prior $p(\boldsymbol{\theta}, \boldsymbol{\gamma})$ and variational approximation $q(\boldsymbol{\theta}, \boldsymbol{\gamma})$

$$\mathcal{F} = \langle \log p(\boldsymbol{y}|\boldsymbol{\theta})\rangle_{q(\boldsymbol{\theta}|\boldsymbol{\gamma})q(\boldsymbol{\gamma})} - KL[q(\boldsymbol{\theta}, \boldsymbol{\gamma})||p(\boldsymbol{\theta}, \boldsymbol{\gamma})]$$

If keeping $(\theta_j, \gamma_j)$ coupled such that $q(\boldsymbol{\theta}, \boldsymbol{\gamma}) = \prod_j q(\theta_j, \gamma_j)$, but $q(\theta_j, \gamma_j) \neq q(\theta_j)q(\gamma_j)$ — i.e. the "paired mean field" (PMF) approximation of Carbonetto and Stephens (2012) — then the expected marginal likelihood term is intractable as it requires computing the expected kernel inverse $\langle K^{-1}\rangle_{q(\theta,\gamma)}$. Due to the dependence of this term on $q(\boldsymbol{\gamma})$, only Black Box variational Inference (BBVI) can be used to unbiasedly approximate its gradient (Ranganath et al., 2014), which suffers from high variance. On the other hand, if using a fully factorised (mean field) approximation $q(\boldsymbol{\theta}, \boldsymbol{\gamma}) = \prod_j q(\theta_j)q(\gamma_j)$, then the KL term is undefined for any $q(\theta_j) \neq \delta_0(\theta_j)$ due to the need to compute $\langle \log \delta_0(\theta_j)\rangle_{q(\theta_j)\neq\delta_0(\theta_j)}$.

### 2.4 Related Work on Spike and Slab GPs

Several previous works have used MCMC for inference with spike and slab priors in GPs (Linkletter et al., 2006; Savitsky et al., 2011; Qamar and Tokdar, 2014), motivated by its asymptotic convergence guarantees. Whilst performance is typically excellent, these algorithms are costly to run both in large-$d$ datasets as they stochastically search over a $2^d$ model space, and in large-$n$ datasets they use many model evaluations with $\mathcal{O}(n^3)$ cost. For example, Savitsky et al. (2011) report runtimes of nearly 3 hours when $(n = 100, d = 1000)$. Although scalable MCMC schemes have since been developed for GPR using Hamiltonian Monte Carlo (Hensman et al., 2015; Rossi et al., 2021), these schemes require differentiable random variables and so can't be used off-the-shelf with spike and slab priors without bespoke augmentation strategies (Pakman and Paninski, 2013). The need to store a large number of $d$-dimensional Monte Carlo samples also makes MCMC unattractive for high-dimensional scenarios.

Dai et al. (2015) are able to develop a scalable variational inference algorithm for selecting latent inputs in the GP latent variable model (GP-LVM) using the PMF approximation of Carbonetto and Stephens (2012). By using the sparse inducing points strategy of Titsias (2009) they are able to recover a closed form variational lower bound to optimise with $\mathcal{O}(nm^2)$ complexity. Unfortunately, the same strategy can't be used in the supervised setting when sparsifying the inverse lengthscales, as the expected inverse of the inducing variable gram matrix $\langle K_{ZZ}^{-1}\rangle_{q(\theta,\gamma)}$ is intractable. Titsias and Lázaro-Gredilla (2011) also develop an efficient variational EM algorithm for GPs with spike and slab priors in the multi-task learning setting. However their algorithm finds sparse linear mixtures of GPs and so can take advantage of linear model tractabilities.

By contrast to previous work, our method runs faster than stochastic variational GPR (Hensman et al., 2013) up to $n = 10^6$, has typically $\mathcal{O}(n \log(n) d)$ complexity, and works with any differentiable kernel.

## 3 THE SPIKE AND SLAB VARIATIONAL GAUSSIAN PROCESS (SSVGP)

In this section we derive and present our method in detail. We first present our generative model and variational inference strategy used to overcome the aforementioned intractabilities, as well as our approach for scaling to large-$n$ datasets. We then address our algorithm's sensitivity to hyperparameters and derive a Bayesian Model averaging (BMA) procedure to adapt them reliably to the degree of input relevance sparsity. In the process we speed up our algorithm substantially through several modifications. We demonstrate performance throughout using a toy example. For key equation derivations see the appendix.

### 3.1 Model and Inference Algorithm

We focus on the standard GPR setting with a Gaussian noise model $\mathcal{N}(y|f, \sigma^2)$ and zero-mean GP prior $\mathcal{GP}(f|0, k(\boldsymbol{x}, \boldsymbol{x}'))$, giving rise to closed form marginal likelihood. We place a Gaussian spike and slab prior on the kernel inverse lengthscales. In doing so, we choose to approximate the Dirac spike with a Gaussian with mass concentrated near zero, which is a strategy that has been previously been used in linear models to improve tractability (George and McCulloch, 1997; Ročková and George, 2014; Bai et al., 2021). We also place a Beta prior over the prior inclusion probability. For $n$ samples $(y_i, \boldsymbol{x}_i)_{i=1}^n$, this results in the model

$$
\begin{aligned}
p_{\boldsymbol{\phi}}(\boldsymbol{y}|\boldsymbol{\theta}) &= \mathcal{N}(\boldsymbol{y}|\mathbf{0}, \tilde{K}_{XX}) \\
p(\theta_j, \gamma_j|\pi) &= \left[\mathcal{N}\left(\theta_j|0, \frac{1}{cv}\right)\pi\right]^{\gamma_j}\left[\mathcal{N}\left(\theta_j|0, \frac{1}{v}\right)(1-\pi)\right]^{1-\gamma_j} \\
p(\pi) &= \mathcal{B}eta(\pi|a, b).
\end{aligned}
$$

Here $\boldsymbol{\theta} \in \mathbb{R}^d$ are the inverse lengthscales[1], $\boldsymbol{\phi} = \{\tau, \sigma^2\} \in \mathbb{R}^2_{>0}$ are the scale and noise parameters, $\boldsymbol{\gamma} \in \{0,1\}^d$ are the binary inclusion indicators, $[\tilde{K}_{XX}]_{ij} = k_{\tau,\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j) + \delta_1(i=j)\sigma^2$, $v \gg 1$ is the precision of the spike distribution, and $cv \ll 1$ is the precision of the slab distribution.

The Gaussian spike approximation enables us to derive a fast approximate co-ordinate ascent variational inference (CAVI) algorithm under the minimal factorisation: $p_{\boldsymbol{\phi}}(\boldsymbol{\theta}, \boldsymbol{\gamma}, \pi|\boldsymbol{y}) \approx q(\boldsymbol{\theta})q(\boldsymbol{\gamma})q(\pi)$, since now the KL term of the evidence lower bound

$$
\mathcal{F} = \langle \log p(\boldsymbol{y}|\boldsymbol{\theta})\rangle_{q(\boldsymbol{\theta})} - KL[q(\boldsymbol{\theta})q(\boldsymbol{\gamma})||p(\boldsymbol{\theta}, \boldsymbol{\gamma})] \tag{5}
$$

is well defined. In particular, if $q(\boldsymbol{\theta})$ is chosen such that $\langle \theta_j^2 \rangle_{q(\theta)}$ has closed form, exact co-ordinate ascent updates for $q(\boldsymbol{\gamma})$, $q(\pi)$ are available

$$
\begin{aligned}
q^*(\gamma_j) &= \mathcal{B}ern(\gamma_j|\lambda_j) \\
\lambda_j &= \left(1 + c^{-\frac{1}{2}} e^{-\frac{v}{2}\langle \theta_j^2\rangle_{q(\theta)}(1-c) + \left\langle \log\frac{1-\pi}{\pi}\right\rangle_{q(\pi)}}\right)^{-1} &(6) \\
q^*(\pi) &= \mathcal{B}eta(\pi|\xi_a, \xi_b) \\
(\xi_a, \xi_b) &= (a + \sum\nolimits_{j=1}^d \lambda_j, b + d - \sum\nolimits_{j=1}^d \lambda_j)) &(7)
\end{aligned}
$$

Note $\lambda_j = q(\gamma_j = 1)$ is the PIP for dimension $j$. Whilst exact CAVI updates are not available for $q(\boldsymbol{\theta})$ due to the non-conjugacy of the model, for a suitable parametric restriction $q_{\boldsymbol{\psi}}(\boldsymbol{\theta})$ we can approximate its co-ordinate update using stochastic gradient-based optimisation of $\mathcal{F}$ with the reparameterisation trick (Kingma and Welling, 2013). The reparameterisation trick enables us to unbiasedly approximate the intractable gradient of the expected likelihood term $\nabla_{\boldsymbol{\psi}}\langle \log p(\boldsymbol{y}|\boldsymbol{\theta})\rangle_{q_{\psi}(\boldsymbol{\theta})}$ by parameterising $q_{\boldsymbol{\psi}}(\cdot)$ such that

[1]Note that since most stationary kernel functions are a function of the square of $\theta$ (i.e. $k_\theta(x, x') = k_{-\theta}(x, x')$), we let the inverse lengthscales be unconstrained in domain, which enables a Gaussian spike and slab to be used. For any kernel which violates this we place the prior on $\sqrt{\theta}$.

**Algorithm 1:** a-CAVI for SSVGP training

```
Input  : Data: $(\boldsymbol{y}, X)$, initialisation:
         $(\boldsymbol{\psi}^{(0)}, \boldsymbol{\lambda}^{(0)}, \boldsymbol{\xi}^{(0)}, \boldsymbol{\phi}^{(0)})$, learning rates: $\boldsymbol{\eta}$,
         hyperparameters: $(v, c, a, b)$, # a-CAVI
         iters: $K$, # gradient iters: $T$, # MC
         samples: $S$
1 for $K$ iterations do
2     for $T$ steps do
3         sample $\boldsymbol{\epsilon}^{(1:S)} \sim \mathcal{N}(\mathbf{0}, I)$.
4         get $\hat{\nabla}_{\boldsymbol{\psi}}\mathcal{F}$ ,$\hat{\nabla}_{\boldsymbol{\phi}}\mathcal{F}$ using eq. (8) and (9).
5         update $(\boldsymbol{\psi}, \boldsymbol{\phi})$ using equation (10).
6     end
7     update $\boldsymbol{\lambda}$ using equation (6).
8     update $\boldsymbol{\xi}$ using equation (7).
9 end
Output: $\boldsymbol{\psi}, \boldsymbol{\lambda}, \boldsymbol{\xi}, \boldsymbol{\phi}$
```

$\boldsymbol{\theta} = \mathcal{T}_{\boldsymbol{\psi}}(\boldsymbol{\epsilon})$, where $\mathcal{T}_{\boldsymbol{\psi}}(\cdot)$ is differentiable, $\boldsymbol{\epsilon} \sim q(\boldsymbol{\epsilon})$ and $q(\boldsymbol{\epsilon})$ does not depend on $\boldsymbol{\psi}$. We also choose $q_{\boldsymbol{\psi}}(\boldsymbol{\theta})$ to ensure the KL term has closed form gradients. In this case, given $S$ samples $\boldsymbol{\epsilon}^{(1:S)} \sim q(\boldsymbol{\epsilon})$, $\nabla_{\boldsymbol{\psi}}\mathcal{F}$ can be unbiasedly approximated as

$$
\begin{aligned}
\hat{\nabla}_{\boldsymbol{\psi}}\mathcal{F} = \frac{1}{S}\sum\nolimits_{s=1}^S \left(\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\phi}}(\boldsymbol{y}|\boldsymbol{\theta}^{(s)})\nabla_{\boldsymbol{\psi}}\mathcal{T}_{\boldsymbol{\psi}}(\boldsymbol{\epsilon}^{(s)})\right) \\
- \nabla_{\boldsymbol{\psi}} KL[q(\boldsymbol{\theta})q(\boldsymbol{\gamma})||p(\boldsymbol{\theta}|\boldsymbol{\gamma})] \qquad (8)
\end{aligned}
$$

Scale and noise parameters $\boldsymbol{\phi}$ can be optimised jointly using a similar unbiased approximation

$$
\hat{\nabla}_{\boldsymbol{\phi}}\mathcal{F} = \frac{1}{S}\sum\nolimits_{s=1}^S \nabla_{\boldsymbol{\phi}} \log p_{\boldsymbol{\phi}}(\boldsymbol{y}|\boldsymbol{\theta}^{(s)}). \tag{9}
$$

Thus, our approximate CAVI algorithm (a-CAVI) iterates between 1) *exactly* maximising $\mathcal{F}$ with respect to $\{q(\boldsymbol{\gamma}), q(\pi)\}$ using Eqns. (6) and (7), and 2) *approximately* maximising $\mathcal{F}$ with respect to $\{q_{\boldsymbol{\psi}}(\boldsymbol{\theta}), \boldsymbol{\phi}\}$ using gradient based optimisation with Eqns. (8), (9), and the overall parameter update

$$
(\boldsymbol{\psi}, \boldsymbol{\phi}) \leftarrow (\boldsymbol{\psi}, \boldsymbol{\phi}) + \boldsymbol{\eta} \odot (\hat{\nabla}_{\boldsymbol{\psi}}\mathcal{F}, \hat{\nabla}_{\boldsymbol{\phi}}\mathcal{F}). \tag{10}
$$

Algorithm 1 summarises the steps. In practice we set learning rates $\boldsymbol{\eta}$ using ADAM (Kingma and Ba, 2014) and set $q(\boldsymbol{\theta})$ as a Mean Field Gaussian (MFG) to ensure $\mathcal{O}(d)$ time and storage complexity. In this case $q(\boldsymbol{\epsilon}) = \mathcal{N}(\mathbf{0}, I)$ and $\mathcal{T}_{\boldsymbol{\psi}}(\boldsymbol{\epsilon}) = \boldsymbol{\sigma} \odot \boldsymbol{\epsilon} + \boldsymbol{\mu}$. Fortunately reparameterisation gradient variance is sufficiently low such that setting $S = 1$ enables efficient learning.

**Complexity Analysis**: When $q(\boldsymbol{\theta})$ is MFG and $S = 1$, gradients (8) and (9) cost $\mathcal{O}(n^3)$ for inverting $K_{XX}$ and $\mathcal{O}((n^2+1)d)$ for matrix-vector products, which is similar complexity to ML-II gradients. The remaining CAVI updates for $q(\boldsymbol{\gamma})$, $q(\pi)$ are $\mathcal{O}(d)$ and generally extremely cheap to perform. If we had instead used the Dirac spike and PMF approximation as in Dai et al. (2015), then no exact CAVI updates would be tractable and only score gradients (i.e. BBVI (Ranganath et al., 2014)) would be available to optimise $q(\boldsymbol{\gamma})$ (without relying on continuous relaxations), which typically require $S \gg 1$ samples per iteration for efficient learning (Mohamed et al., 2020). Table 1 demonstrates our method significantly outperforms using a Dirac spike with PMF+BBVI on a toy example, whilst retaining much faster runtimes.

**Predictions**: At test time, given $S_*$ samples $\boldsymbol{\theta}^{(1:S_*)} \sim q_{\boldsymbol{\psi}}(\boldsymbol{\theta})$ and learned $\boldsymbol{\phi}$, the predictive distribution can be approximated by a discrete mixture of standard GPR posterior predictives $\{p_{\boldsymbol{\phi}}(y_*|\mathcal{D}, \boldsymbol{x}_*, \boldsymbol{\theta}^{(s)})\}_{s'=1}^{S_*}$, which has closed form moments (see the appendix)

$$p(y_*|\mathcal{D}, \boldsymbol{x}_*) \approx \frac{1}{S} \sum\nolimits_{s'=1}^{S_*} p_{\boldsymbol{\phi}}(y_*|\mathcal{D}, \boldsymbol{x}_*, \boldsymbol{\theta}^{(s')}) \tag{11}$$

**Interpreting the Posterior Factorisation**: Our factorisation over inverse lengthscales $\boldsymbol{\theta}$ and inclusion variables $\boldsymbol{\gamma}$ mean that $q(\boldsymbol{\gamma})$ is not used at test time. Whilst this may seem counter-intuitive, note that $q(\boldsymbol{\theta})$ automatically embeds information from $q(\boldsymbol{\gamma})$ during a-CAVI iterations in 'message passing' fashion. In the MFG case, the KL term acts as an L2-regulariser on $\boldsymbol{\psi} = \{\boldsymbol{\mu}, \boldsymbol{\sigma}\}$, where the penalty scale varies with $\boldsymbol{\lambda}$. This can be seen by examining the parts of the KL term that depend on variational parameters $\{\mu_j, \sigma_j\}$

$$KL_{\mu_j} = -0.5(\lambda_j v c \mu_j^2 + (1-\lambda_j) v \mu_j^2) \tag{12}$$

$$KL_{\sigma_j} = -0.5(\lambda_j v c \sigma_j^2 + (1-\lambda_j) v \sigma_j^2) + H[q] \tag{13}$$

As a result, when $\lambda_j \approx 0$, we often have $\boldsymbol{\mu}, \boldsymbol{\sigma} \to 0$ for large enough spike precision $v$. Thus, our algorithm can be interpreted as adaptively regularising the posterior over inverse lengthscales based on learned PIPs.

### 3.2 Scaling to Large-$n$ Datasets

Since our algorithm suffers $\mathcal{O}(n^3)$ complexity, in large-$n$ datasets we use the approach taken by Chen et al. (2020) of approximating $\nabla \log p_{\boldsymbol{\phi}}(\boldsymbol{y}|\boldsymbol{\theta})$ at each iteration using a rescaled $m$-sized minibatch $\boldsymbol{y}_m$ comprised of a uniformly sampled point $y_i$ and its $m-1$ nearest neighbours $\boldsymbol{y}_{m,\neg i}$ : $\frac{n}{m}\nabla \log p_{\boldsymbol{\phi}}(\boldsymbol{y}_m|\boldsymbol{\theta})$. This reduces the inversion complexity bottleneck to $\mathcal{O}(m^3)$ at the expense of an $\mathcal{O}(nd)$ nearest neighbour search. Although minibatch gradients are biased due to the non-decomposable likelihood, Chen et al. (2020) were able to outperform a range of state-of-the-art scalable GP approximations on various datasets, and their approach is easily integrated into our base algorithm. When the kernel is monotonically decreasing in a distance metric, we use this metric to find the neighbours with the posterior means $\langle \boldsymbol{\theta} \rangle_{q(\boldsymbol{\theta})} = \boldsymbol{\mu}$ in place of inverse lengthscales $\boldsymbol{\theta}$. For example using kernels of the form given in eq. (3) we use distance metric $d(\boldsymbol{x}, \boldsymbol{x}') = ||\boldsymbol{\mu} \odot (\boldsymbol{x} - \boldsymbol{x}')||_2$.

Since making predictions also suffers an $\mathcal{O}(n^3)$ inversion cost, in $n \gg 10^4$ datasets we follow Jankowiak and Pleiss (2021) and truncate the predictive distribution in eq. (11) using $m_*$ nearest neighbours of $\boldsymbol{x}_*$, per each sampled $\boldsymbol{\theta}^{(s)}$. We again use the distance metric in the kernel but replace $\boldsymbol{\mu}$ with $\boldsymbol{\theta}^{(s)}$. This reduces inversion complexity to $\mathcal{O}(S_* m_*^3)$ per test point, again at the expense of an $\mathcal{O}(S_* nd)$ nearest neighbour search.

### 3.3 Toy Example Demonstration

We now demonstrate our SSVGP on a toy example where we draw $n = 300$ samples of 100-dimensional input $\boldsymbol{x} \sim \mathcal{N}(0, I)$ and set response using 5/100 inputs as $y = \sum_{j=1}^{5} sin(a_j x_j) + \epsilon$, where $\{a_j\}$ are grid spaced over $[0.5, 1]$, and the noise to signal ratio is set to 0.05. We set the spike precision to $v = 10^4$, slab precision to $cv = 10^{-4}$, and Beta prior hyperparameters to $a = b = 10^{-3}$. Note, whilst a $\mathcal{B}eta(a, a)$ prior is strongly informative in the tails as $a \to 0^+$, setting

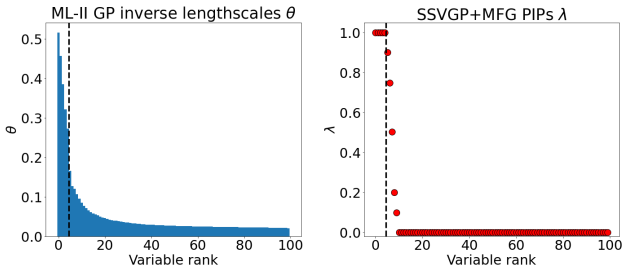

Figure 1: Toy example size ordered profile for ML-II GP inverse lengthscales (LHS) and SSVGP+MFG PIPs (RHS). First 5 variables (LHS of black dashed line) are constrained to be the 5 relevant variables. Profiles are averaged over 10 trials. Whilst the ML-II GP profile fails to shrink many irrelevant inverse lengthscales, the SSVGP learns to exclude nearly all irrelevant variables with high probability and includes all relevant variables with near certainty.

| Method | MSE | Runtime (s) |
|---|---|---|
| ML-II GP | 0.096 ± 0.014 | **5.0** ± 0.7 |
| PMF+BBVI(10) | 0.088 ± 0.019 | 71.6 ± 1.6 |
| PMF+BBVI(100) | 0.077 ± 0.013 | 595.3 ± 4.9 |
| SSVGP+MFG | **0.068** ± 0.011 | 27.8 ± 0.5 |

Table 1: Toy example mean ± sd results (10 trials) for ML-II GP, Dirac spike and slab GP with PMF+BBVI using 10 and 100 score-grad samples respectively, and SSVGP with Mean Field Gaussian $q(\boldsymbol{\theta})$.

$a \approx 0$ closely approximates the improper Haldane prior $\mathcal{B}eta(0,0)$, which is flat on $\log(\frac{\pi}{1-\pi})$ (the statistic used in the a-CAVI algorithm) (Zhu and Lu, 2004). We emphasise the Beta hyperprior has negligible effect on our algorithm results, as demonstrated in section 4. For comparison we implement an ML-II GP using GPyTorch (Gardner et al., 2018) and a spike and slab GP using the Dirac spike and PMF approximation trained using BBVI. See the appendix for implementation and experiment details. In all cases the SE kernel is used.

Table 1 displays (normalised) mean squared error (MSE) and runtime (mean $\pm$ sd) obtained on 10 trials using $n_* = 100$ test points. Our SSVGP outperforms both ML-II and PMF-BBVI on average, and is much closer in runtime to the ML-II GP. In Fig. 1 we also highlight how our SSVGP enables much more reliable variable selection than the ML-II GP by better distinguishing between irrelevant and relevant variables.

## 4 BAYESIAN MODEL AVERAGING FOR ADAPTIVE SPARSITY

One problem with our method is that the learned model sparsity (via PIPs) is heavily dependent on spike and slab parameter $v$. This can be seen by analysing the 'posterior point of intersection' (PPI), which is the value of transformed expected sufficient statistic $\hat{\theta} := \sqrt{\langle \theta^2 \rangle_{q(\theta)}}$ that induces an even chance of inclusion $\lambda = \frac{1}{2}$ in the a-CAVI update for $q(\boldsymbol{\gamma})$

$$\hat{\theta} = \sqrt{\frac{\log\left(\frac{1}{c}\right) + 2\langle \log\left(\frac{1-\pi}{\pi}\right)\rangle_{q(\pi)}}{v(1-c)}} \qquad (14)$$

The PPI adapts with $q(\pi)$ during training, however the adaptability is limited as the effect is through the square root of the log-odds. For example if a fixed value of $\pi$ is moved from 0.001 to 0.999, the PPI only moves from 0.021 to 0.057 for $(v, c) = (10^4, 10^{-8})$. Thus, since the PPI is $\mathcal{O}(v^{-\frac{1}{2}})$, $v$ can entirely determine the sparsity of the PIPs. This is demonstrated in Table 2, which shows that when we run our SSVGP on the same 10 trials increasing $v$ from $10^2$ to $10^6$, the average recovered PIP increases from 0 to 1. We therefore need a reliable way of adapting $v$ to the input sparsity in the data.

We found directly maximising $\mathcal{F}$ with respect to $v$ (or variational posterior $q(v)$) lead to highly initialisation dependent results, and so did not address the problem. Our solution is to instead train a set of models $\{\mathcal{M}_k\}_{k=1}^K$ with different scale parameters $v_k \in \mathcal{V}$, and Bayesian model average over the learned posteriors $\{q_k(\boldsymbol{\theta})_{k=1}^K\}, \{q_k(\boldsymbol{\gamma})_{k=1}^K\}$ using approximate posterior model probabilities $w_k \approx p(\mathcal{M}_k|\boldsymbol{y})$. Marginal

| $v$ | $10^2$ | $10^3$ | $10^4$ | $10^5$ | $10^6$ |
|---|---|---|---|---|---|
| average PIP | 0 | 0.05 | 0.07 | 0.34 | 1 |

Table 2: Toy example average PIP recovered for SSVGP with varying spike and slab parameter $v$.

PIPs $\{\bar{\lambda}_j\}_{j=1}^d = \{\sum_{k=1}^K \lambda^{(k)} w_k\}_{j=1}^d$ can then be used to do variable selection as in the original algorithm.

However, for this procedure to work we need (1) a fast enough algorithm to enable multiple ($\geq 10$) trials without compromising speed and scalability, and (2) a good approximation to the posterior model probability $p(\mathcal{M}_k|\boldsymbol{y})$. We address each of these aspects below.

### 4.1 Speeding Up a-CAVI

We first modify our a-CAVI algorithm to enable order of magnitude speed-ups, whilst only negligibly impacting performance when $v$ is appropriately set.

**Zero Temperature Restrictions**: We zero temperature (ZT) restrict the inverse lengthscale posterior: $q(\boldsymbol{\theta}) = \delta_{\boldsymbol{\mu}}(\boldsymbol{\theta})$. This eliminates the need for Monte-Carlo sampling at training and test time as both the expected likelihood term $\langle p_{\boldsymbol{\phi}}(\boldsymbol{y}|\boldsymbol{\theta})\rangle_{q(\boldsymbol{\theta})}$ and predictive distribution now have closed form at $\boldsymbol{\theta} = \boldsymbol{\mu}$.

**Dropout Pruning**: We prune variables with small PIPs during training iterations. This type of procedure has been used to great effect in sparsifying neural networks whilst only negligibly impacting performance (Hoefler et al., 2021). Specifically after every a-CAVI iteration, we permanently enforce $\langle \theta_j \rangle = \mu_j = 0$ if $\lambda_j \leq \epsilon$ for some chosen $\epsilon \in (0,1)$. This reduces training complexity from $\mathcal{O}(d)$ to $\mathcal{O}(p_t)$, where $p_t = \{\#\lambda_j^{(t)} > \epsilon\}$, leading to dramatic speed ups when few inputs are relevant. We also prove in the appendix that under certain conditions pruning will only negligibly affect converged solutions of a-CAVI. In practice we set $\epsilon = 0.5$ to ensure non-pruned variables are those with at least an even probability of inclusion, although our algorithm is insensitive[2] to the pruning threshold.

We also use $m < n$ nearest neighbour minibatching to further improve runtime even when $n < 10^3$, as we find for large enough $m$ (i.e. $m \geq \frac{n}{4}$) this does not harm and in some cases improves performance. We believe this is because if $m$ is sufficiently large, the gradient bias remains small enough for the intrinsic regularisation benefits of SGD (Smith et al., 2020) to be leveraged with minimal cost. Table 3 demonstrates the obtainable speed ups using these modifications on the toy example, with pruning threshold $\epsilon = 0.5$ and

[2]This is because the PIP update takes the form of generalised logistic funtion $\lambda(\theta) = (1 + Ae^{-B\theta^2})^{-1}$ with large positive constants $A, B$. Thus typically $\lambda^* \to \{0 \cup 1\}$.

| | MF | ZT | ZT+drop | ZT+drop(m=n/4) |
|---|---|---|---|---|
| MSE | 0.068 | 0.068 | 0.064 | 0.065 |
| avg PIP | 0.07 | 0.05 | 0.06 | 0.05 |
| Runtime(s) | 27.8 | 24.1 | 10.5 | 1.3 |

Table 3: Toy example average results for SSVGP implementing a-CAVI modifications sequentially (left to right). Using all modifications speeds up runtime by $\sim 20\times$, whilst performance remains similar.

$m = \frac{n}{4}$ minibatching. Whilst average performance remains similar, average runtime drops to $1.3s$.

## 4.2 Approximating Posterior Weights with the Leave-One-Out Predictive Density

Rather than use the evidence lower bound $\mathcal{F}$ to approximate the posterior model weights (under a uniform prior over models $e^{F_k} \leq p(\boldsymbol{y}|\mathcal{M}_k) \propto p(\mathcal{M}_k|\boldsymbol{y})$), we use the leave-one-out predictive density (LOOPD): $\prod_{i=1}^n p(y_i|\boldsymbol{y}_{\neg i}, \mathcal{M}_k)$ as in Yao et al. (2018). We do not use $\mathcal{F}$ as its preference for model sparsity is heavily influenced by hyperparameter $c$, which controls the KL divergence between the spike and slab of the prior. This arises due to the fixed variable inclusion cost of $\mathcal{O}(\log(\frac{1}{c}))$ from the normaliser of the slab distribution. Additionally, the LOOPD known to be more robust against overfitting and model mis-specification than the evidence $p(\boldsymbol{y}|\mathcal{M}_k)$ itself (Jankowiak and Pleiss, 2021; Rasmussen and Williams, 2006).

Under the ZT posterior, inverse lengthscales $\boldsymbol{\theta}$ can be treated as parameters fixed at $\boldsymbol{\mu}_k$ for model $\mathcal{M}_k$. In this case the LOOPD for $\mathcal{M}_k$ is that of a standard GPR evaluated at $(\boldsymbol{\theta} = \boldsymbol{\mu}_k, \boldsymbol{\phi} = \boldsymbol{\phi}_k)$. To compute the LOOPD in $\mathcal{O}(n^3)$ time in small datasets we use the Bürkner et al. (2021) algorithm, which uses Rank-1 updates to $K_{XX}$. In large datasets we nearest-neighbour truncate the LOOPD following the success of this approach in Jankowiak and Pleiss (2021). That is, for each $(y_i, \boldsymbol{x}_i)$, we condition on $\tilde{m}$ nearest neighbours of $\boldsymbol{x}_i$ found using the distance metric in the kernel. This reduces complexity to $\mathcal{O}(m^3 n)$ but introduces an $\mathcal{O}(nf(n)p_k)$ query cost, where $p_k$ is the count of non-pruned variables in $\mathcal{M}_k$ and $f(n) \in [\log(n), n]$ is the query cost using Ball-trees (Omohundro, 1989) or KD-trees (Bentley, 1975) (our preferred algorithms).

Given the recovered posteriors and model weights, the predictive distribution is a discrete mixture of standard GP predictive posteriors evaluated at $\{\boldsymbol{\mu}_k, \boldsymbol{\phi}_k\}$,

$$\begin{aligned} p(\boldsymbol{y}_*|X_*, \mathcal{D}) &\approx \sum\nolimits_{k=1}^K \int p_{\boldsymbol{\phi}_k}(\boldsymbol{y}_*|X_*, \boldsymbol{\theta}, \mathcal{D}) q_k(\boldsymbol{\theta}) d\boldsymbol{\theta} w_k \\ &= \sum\nolimits_{k=1}^K p_{\boldsymbol{\phi}_k}(\boldsymbol{y}_*|X_*, \boldsymbol{\theta} = \boldsymbol{\mu}_k, \mathcal{D}) w_k. \end{aligned} \tag{15}$$

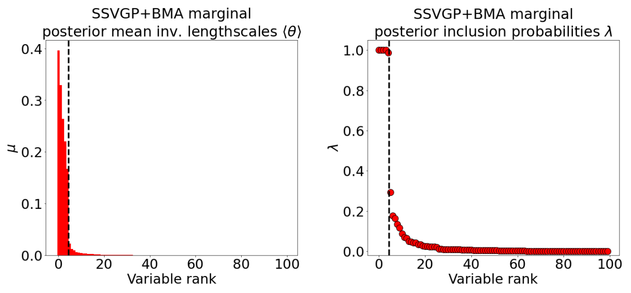

Figure 2: Toy example average size ordered profiles for (LHS) SSVGP+BMA posterior mean inverse lengthscales and (RHS) SSVGP+BMA PIPs. The PIP profile is better than the SSVGP+MFG (Fig. 1) and the posterior mean inverse lengthscales are shrunk to near-zero for all irrelevant dimensions due to the regularisation from $\{q_k(\boldsymbol{\gamma})\}_{k=1}^K$ and learned model weighting.

| Method | MSE | MCC | Runtime (s) |
|---|---|---|---|
| ML-II GP | 0.096 ± 0.014 | - | **5.0** ± 0.7 |
| PMF+BBVI(100) | 0.077 ± 0.013 | 0.62 ± 0.06 | 595.3 ± 4.9 |
| SSVGP+MFG | 0.068 ± 0.011 | 0.82 ± 0.09 | 27.8 ± 0.5 |
| SSVGP+BMA | **0.064** ± 0.01 | **0.98** ± 0.04 | 14.6 ± 0.4 |

Table 4: Toy example mean ± sd results. SSVGP+BMA outperforms all other methods in predictive accuracy (MSE) and variable selection accuracy (MCC), and runs in similar time to ML-II GP.

To speed up prediction time we stochastically threshold the recovered posterior model weights $\boldsymbol{w}$ by updating $\boldsymbol{w} \to \tilde{\boldsymbol{w}} = \frac{\boldsymbol{z}}{S}$, where $\boldsymbol{z}$ is drawn from a Multinomial distribution using probability vector $\boldsymbol{w}$ and $S$ trials. Thus we have that $\mathbb{E}[\tilde{\boldsymbol{w}}] = \boldsymbol{w}$ and $p(\tilde{w}_i = 0|w_i = \frac{1}{S}) = \left(\frac{S-1}{S}\right)^S \approx \frac{1}{3}$ for $S \geq 100$. Additionally, when predictive accuracy is the core focus we simply select the best model. We use nearest-neighbour truncation at test time for large-$n$ datasets (e.g. $n > 10^4$).

## 4.3 BMA on the Toy Example

We demonstrate our SSVGP with BMA on the same 10 trials of the toy example using 11 values of $v$ grid spaced over $10^4 \times 2^{\{-log_2(1000),\ldots,log_2(1000)\}}$ such that $10^1 \leq v \leq 10^7$, and use the ZT posterior with $m = \frac{n}{4}$ minibatching and pruning threshold $\epsilon = 0.5$.

We display in Table 4 performance results against previous models and in Fig. 2 the average profile of PIPs and posterior mean inverse lengthscales. Note we also report the Matthews Correlation Coefficient (MCC $\in [-1, 1]$) (Matthews, 1975) to measure variable selection accuracy when using $\bar{\lambda} = 0.5$ as an inclusion threshold. Our SSVGP with BMA outperformed all

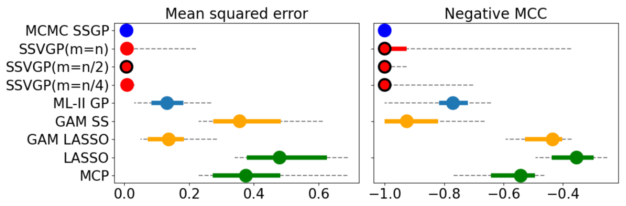

Figure 3: Experiment 1 median (circle), 25-75 percentiles (line), and 10-90 percentiles (dashed line) results. Black border denotes best median performance. Left is better. SSVGPs outperform all implemented comparators and matched single trial performance of Savitsky et al. (2011) MCMC spike and slab GP.

| MCMC | SSVGP(n) | SSVGP(n/2) | SSVGP(n/4) | ML-II |
|---|---|---|---|---|
| 10224s | 20.3s | 11.0s | 8.4s | 3.9s |

Table 5: Experiment 1 average runtimes for different methods and single trial time of Savitsky et al. (2011) spike and slab GP when running $5 \times 10^5$ MCMC iters.

other methods in terms of predictive and variable selection accuracy, learned a better quality PIP profile than the original SSVGP+MFG without BMA and also ran faster on average ( $\sim 15s$). Thus, our BMA procedure and a-CAVI modifications induced better and faster performance than using our original algorithm, even with a well calibrated choice of $v$.

## 5 EXPERIMENTS

We now test our SSVGP in several experiments (we use BMA with ZT and dropout by default from now on). We consider a synthetic small-$n$ experiment, a synthetic large-$n$ experiment and benchmark real datasets. See the appendix for experiment and implementation details. The SE kernel is used throughout, but our method works with any differentiable kernel.

### 5.1 Experiment 1: A Small-Scale, High Dimensional Variable Selection Problem

We first test our SSVGP on 50 replications of Savitsky et al. (2011) main experiment to assess whether our algorithm can compete with their spike and slab GP using a Dirac spike and MCMC. The experiment draws $n = 100$ training and $n_* = 20$ test samples of $d = 1000$ dimensional inputs $\boldsymbol{x} \sim Unif[0,1]^d$ and sets the response as an additive function of 6/1000 inputs: $y = x_1 + x_2 + x_3 + x_4 + sin(3x_5) + sin(5x_6) + \epsilon$ for $\epsilon \sim \mathcal{N}(0, 0.05)$. We also implement the ML-II GP and several benchmark variable selection algorithms: LASSO (Tibshirani, 1996), MCP (Zhang, 2010), GAM

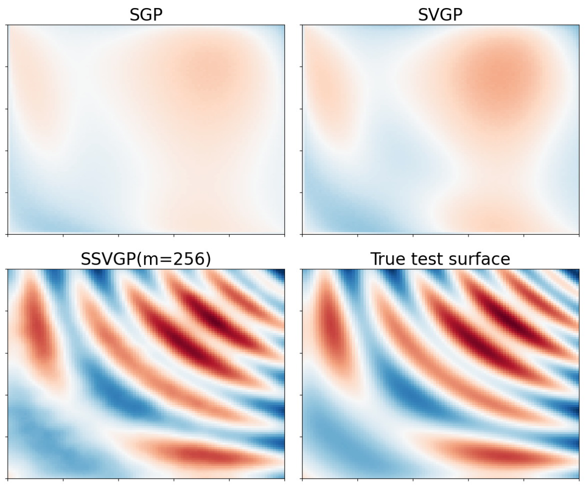

Figure 4: Experiment 2 average prediction surfaces as a function of the two relevant inputs when $n = d = 10^4$, and true test surface for latent (noiseless) $\boldsymbol{f}_*$. The SGP and SVGP cannot accurately recover the surface due to the number of noise dimensions, whilst our SSVGP recovers a near perfect surface.

with group LASSO penalty (Huang et al., 2009) and GAM with group spike and slab LASSO penalty (Bai et al., 2020). We implement our SSVGP with mini-batch sizes $m \in \{\frac{n}{4}, \frac{n}{2}, n\}$ and the same settings as previously. For the ML-II GP we report MCC using the *best* performing threshold from $0.1^{\{0,0.5,1,1.5,2\}}$ as indicative 'best-case' performance with thresholding.

In Fig. 3 and Table 5 we present MSE, negative MCC and runtime percentiles for all models against Savitsky et al. (2011) single completed trial. All SSVGPs outperformed the implemented comparators, and the median trial performance is near identical to the single trial of Savitsky et al's MCMC spike and slab GP but with average runtimes of $< 20$ seconds rather than $> 10^4$ seconds. When $m = \frac{n}{2}$ performance is best but $m = \frac{n}{4}$ outperformed $m = n$ on average.

### 5.2 Experiment 2: A Large Scale Sparse Prediction Challenge

We next test the scalability and predictive performance of our method in a large-scale synthetic experiment against the Sparse GP (SGP) of Titsias (2009) and the Stochastic Variational GP (SVGP) of Hensman et al. (2013), as two popular best-practice benchmarks in the scalable GP literature. We generate synthetic data from a 2d input interactive function $y(\boldsymbol{x}) = \tan(x_1) + \tan(x_2) + \sin(2\pi x_1) + \sin(2\pi x_2) + \cos(4\pi^2 x_1 x_2) + \tan(x_1 x_2) + \epsilon$ where the noise to signal ratio is set to $\frac{1}{3}$ and $x_1, x_2 \sim \mathcal{U}[0,1]$. We then augment with $d-2$ noise dimensions correlated 0.5 with

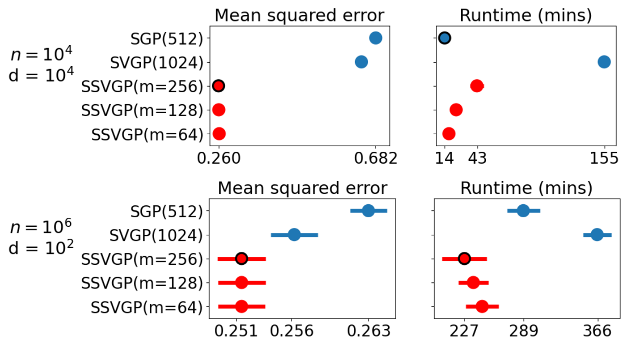

Figure 5: Experiment 2 mean (circle) ± standard deviation (line) results. Black border denotes best on average. Our SSVGPs perform better on average and produce competitive runtimes. Note $m = 256$ is fastest when $n = 10^6$ as runtime is dominated by LOOPD computation, and since $m = 256$ selected fewer variables per model in BMA this reduced compute time.

$x_1, x_2$. We fix $nd = 10^8$ and run 3 trials of (i) an ultra-high dimensional design ($n = 10^4, d = 10^4$), and (ii) a one million training point design ($n = 10^6, d = 10^2$). We test on $n_* = 10^4$ datapoints but with $x_1, x_2$ grid spaced on $[0, 1]$. We implement the SGP and SVGP using GPyTorch (Gardner et al., 2018) with 512 and 1024 inducing points respectively consistent with standard practice (Chen et al., 2020; Wang et al., 2019; Jankowiak and Pleiss, 2021). We implement the SSVGP with $m = \{64, 128, 256\}$ minibatch sizes, and use 64 and 256 neighbours in LOOPD and predictive distribution truncation respectively. In Figure 4 we present average prediction surfaces as a function of the two relevant inputs, against the latent test function $\boldsymbol{f}_*$. Figure 5 displays MSE and runtime results.

Our SSVGP outperformed the SGP and SVGP on average in both designs and consistently ran faster than the SVGP. Whilst the MSE improvement is minimal in the $d = 10^2$ case, when $d = 10^4$ the SGP and SVGP are non-competitive. This demonstrates the gains to using our method as input dimensionality and sparsity grows. SSVGP performance is similar across minibatch sizes, but $m = 256$ is best as expected.

**Computational Cost:** As $n$ grows, SSVGP runtime is dominated by LOOPD computation due to the $\mathcal{O}(nf(n)p_k)$ nearest neighbour search cost where $f(n) \in [\log(n), n]$ depending on $p_k$ (# non-pruned variables in model $k$). As $p_k$ grows, complexity converges to $\mathcal{O}(n^2 p_k)$ and so in large-$p_k$ scenarios our method may by slower than the SVGP when $n \gg 10^6$. However in scenarios where $p_k \ll 100$ for most of the $K$ models in BMA we expect our method to remain faster than SVGP even when $n \gg 10^6$.

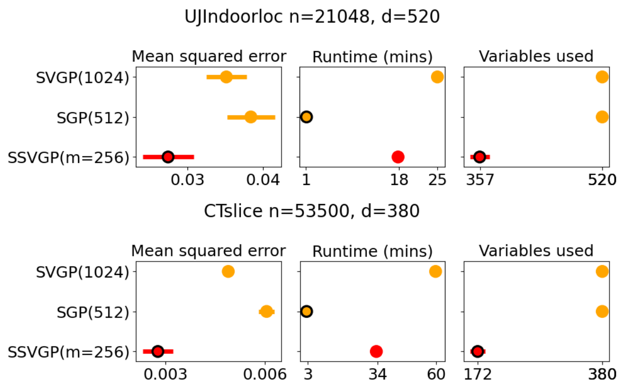

Figure 6: Real dataset mean (circle) ± standard deviation (line) results on CTslice and UJindoorloc. Black border denotes best average performance. The SSVGP outperformed both the SGP and SVGP whilst predicting with a sparse subset of identified relevant variables.

### 5.3 UCI Repository Datasets

We test our method on two high-dimensional datasets from the UCI repository: UJINDOORLOC $(n, d) = (21048, 520)$ and CTSLICE $(n, d) = (53500, 380)$. We implement the SSVGP with $m = 256$ minibatching, and again use 64 and 256 nearest neighbours respectively in LOOPD and predictive distribution truncation. In Figure 6 we produce MSE and runtime results from 10 trials of random 4:1 training/test splits, and compare against the the SGP(512) and SVGP(1024). Our SSVGP performed best on average in both datasets and again ran faster than the SVGP.

## 6 CONCLUSION

We introduced the spike and slab variational Gaussian process (SSVGP): a fast and scalable training and inference method for Gaussian processes with spike and slab variable selection priors. Unlike previous spike and slab GP implementations for variable selection, our method runs in similar time to sparse variational GPs even on $n = 10^6$ sized datasets, has typically $\mathcal{O}(n \log(n) d)$ complexity, and works with any differentiable kernel. Thus, our SSVGP captures most of the benefits of spike and slab priors over ARD in high-dimensional designs, without the typical inference cost. Future research could explore Bayesian optimisation (Shahriari et al., 2015) for efficient model search, or extensions to classification and deep kernel learning (Wilson et al., 2016). For classification the Pólya Gamma auxilliary variable strategy in Jankowiak and Pleiss (2021) could represent a sensible starting point, but we leave this to future work.

## References

Bai, R., Moran, G. E., Antonelli, J. L., Chen, Y., and Boland, M. R. (2020). Spike-and-slab group lassos for grouped regression and sparse generalized additive models. *Journal of the American Statistical Association*, pages 1–14.

Bai, R., Ročková, V., and George, E. I. (2021). Spike-and-slab meets lasso: A review of the spike-and-slab lasso. *Handbook of Bayesian Variable Selection*, pages 81–108.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877.

Brooks, S., Gelman, A., Jones, G., and Meng, X.-L. (2011). *Handbook of markov chain monte carlo*. CRC press.

Bürkner, P.-C., Gabry, J., and Vehtari, A. (2021). Efficient leave-one-out cross-validation for bayesian non-factorized normal and student-t models. *Computational Statistics*, 36(2):1243–1261.

Carbonetto, P. and Stephens, M. (2012). Scalable variational inference for bayesian variable selection in regression, and its accuracy in genetic association studies. *Bayesian analysis*, 7(1):73–108.

Castillo, I., Schmidt-Hieber, J., and Van der Vaart, A. (2015). Bayesian linear regression with sparse priors. *The Annals of Statistics*, 43(5):1986–2018.

Chen, H., Zheng, L., Al Kontar, R., and Raskutti, G. (2020). Stochastic gradient descent in correlated settings: A study on gaussian processes. *Advances in Neural Information Processing Systems*, 33.

Chicco, D. and Jurman, G. (2020). The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC genomics*, 21(1):1–13.

Dai, Z., Hensman, J., and Lawrence, N. (2015). Spike and slab gaussian process latent variable models. *arXiv preprint arXiv:1505.02434*.

Gardner, J., Pleiss, G., Weinberger, K. Q., Bindel, D., and Wilson, A. G. (2018). Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. *Advances in neural information processing systems*, 31.

George, E. I. and McCulloch, R. E. (1997). Approaches for bayesian variable selection. *Statistica sinica*, pages 339–373.

Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, pages 282–290.

Hensman, J., Matthews, A. G., Filippone, M., and Ghahramani, Z. (2015). Mcmc for variationally sparse gaussian processes. *Advances in Neural Information Processing Systems*, 28.

Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., and Peste, A. (2021). Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124.

Huang, J., Ma, S., Xie, H., and Zhang, C.-H. (2009). A group bridge approach for variable selection. *Biometrika*, 96(2):339–355.

Jankowiak, M. and Pleiss, G. (2021). Scalable cross validation losses for gaussian process models. *arXiv preprint arXiv:2105.11535*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Linkletter, C., Bingham, D., Hengartner, N., Higdon, D., and Ye, K. Q. (2006). Variable selection for gaussian process models in computer experiments. *Technometrics*, 48(4):478–490.

Liu, R., Regier, J., Tripuraneni, N., Jordan, M., and Mcauliffe, J. (2019). Rao-blackwellized stochastic gradients for discrete distributions. In *International Conference on Machine Learning*, pages 4023–4031. PMLR.

MacKay, D. J. et al. (1998). Introduction to gaussian processes. *NATO ASI series F computer and systems sciences*, 168:133–166.

Matthews, B. W. (1975). Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451.

Mitchell, T. J. and Beauchamp, J. J. (1988). Bayesian variable selection in linear regression. *Journal of the american statistical association*, 83(404):1023–1032.

Mohamed, S., Rosca, M., Figurnov, M., and Mnih, A. (2020). Monte carlo gradient estimation in machine learning. *J. Mach. Learn. Res.*, 21(132):1–62.

Mohammed, R. O. and Cawley, G. C. (2017). Over-fitting in model selection with gaussian process regression. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 192–205. Springer.

Narisetty, N. N. and He, X. (2014). Bayesian variable selection with shrinking and diffusing priors. *The Annals of Statistics*, 42(2):789–817.

Ober, S. W., Rasmussen, C. E., and van der Wilk, M. (2021). The promises and pitfalls of deep kernel learning. In *Uncertainty in Artificial Intelligence*, pages 1206–1216. PMLR.

Omohundro, S. M. (1989). *Five balltree construction algorithms*. International Computer Science Institute Berkeley.

Paananen, T., Piironen, J., Andersen, M. R., and Vehtari, A. (2019). Variable selection for gaussian processes via sensitivity analysis of the posterior predictive distribution. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1743–1752. PMLR.

Pakman, A. and Paninski, L. (2013). Auxiliary-variable exact hamiltonian monte carlo samplers for binary distributions. In *Proceedings of the 26th International Conference on Neural Information Processing Systems-Volume 2*, pages 2490–2498.

Qamar, S. and Tokdar, S. T. (2014). Additive gaussian process regression. *arXiv preprint arXiv:1411.7009*.

Ranganath, R., Gerrish, S., and Blei, D. (2014). Black box variational inference. In *Artificial intelligence and statistics*, pages 814–822. PMLR.

Rasmussen, C. E. and Williams, C. K. (2006). Rasmussen and christopher ki williams. gaussian processes for machine learning. *MIT Press*, 211:212.

Ročková, V. and George, E. I. (2014). Emvs: The em approach to bayesian variable selection. *Journal of the American Statistical Association*, 109(506):828–846.

Rossi, S., Heinonen, M., Bonilla, E., Shen, Z., and Filippone, M. (2021). Sparse gaussian processes revisited: Bayesian approaches to inducing-variable approximations. In *International Conference on Artificial Intelligence and Statistics*, pages 1837–1845. PMLR.

Salimans, T. and Knowles, D. A. (2014). On using control variates with stochastic approximation for variational bayes and its connection to stochastic linear regression. *arXiv preprint arXiv:1401.1022*.

Savitsky, T., Vannucci, M., and Sha, N. (2011). Variable selection for nonparametric gaussian process priors: Models and computational strategies. *Statistical science: a review journal of the Institute of Mathematical Statistics*, 26(1):130.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and De Freitas, N. (2015). Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175.

Smith, S. L., Dherin, B., Barrett, D., and De, S. (2020). On the origin of implicit regularization in stochastic gradient descent. In *International Conference on Learning Representations*.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.

Titsias, M. (2009). Variational learning of inducing variables in sparse gaussian processes. In *Artificial intelligence and statistics*, pages 567–574. PMLR.

Titsias, M. and Lázaro-Gredilla, M. (2011). Spike and slab variational inference for multi-task and multiple kernel learning. *Advances in neural information processing systems*, 24.

Wang, K., Pleiss, G., Gardner, J., Tyree, S., Weinberger, K. Q., and Wilson, A. G. (2019). Exact gaussian processes on a million data points. *Advances in Neural Information Processing Systems*, 32:14648–14659.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. (2016). Deep kernel learning. In *Artificial intelligence and statistics*, pages 370–378. PMLR.

Yao, Y., Vehtari, A., Simpson, D., and Gelman, A. (2018). Using stacking to average bayesian predictive distributions (with discussion). *Bayesian Analysis*, 13(3):917–1007.

Zhang, C.-H. (2010). Nearly unbiased variable selection under minimax concave penalty. *The Annals of statistics*, 38(2):894–942.

Zhu, M. and Lu, A. Y. (2004). The counter-intuitive non-informative prior for the bernoulli family. *Journal of Statistics Education*, 12(2).

# Appendix for ‘Fast and Scalable Spike and Slab Variable Selection in High-Dimensional Gaussian Processes’

## Contents

## A MATHEMATICAL APPENDIX

### A.1 Kernel parameterisation

In this work we treat the inverse lengthscales of the kernel as being unconstrained in their parameterisation: $\boldsymbol{\theta} \in \mathbb{R}^d$. We note that although it is typical for inverse lengthscales to be strictly positive $\boldsymbol{\theta} \in \mathbb{R}^d_{>0}$, most kernel functions are invariant to the sign of $\theta_j$ for standard parameterisations. For example any stationary kernel function of the form given in eq. 3 in the main text (e.g. squared exponential, Matérn, Cauchy) is a function $g_{\boldsymbol{x},\boldsymbol{x}'}(\theta_1^2, ...\theta_d^2)$. The linear kernel, polynomial kernel and periodic wrappers on anisotropic stationary kernels are also invariant to $\text{sign}(\theta_j)$. For any kernel where this invariance does not hold our procedure reparameterises the kernel in terms of $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta}^2$, which restores this property. Thus initial use of a Mean-Field Gaussian posterior approximation $q_{\boldsymbol{\psi}}(\boldsymbol{\theta}) = \prod_{j=1}^d \mathcal{N}(\theta_j|\mu_j, \sigma_j^2)$ over the inverse lengthscales is appropriate. Thus, any plots of posterior means $\{\mu_j\}_{j=1}^d$ using our method (the SSVGP) in the main text and this Appendix plot $\{|\mu_j|\}_{j=1}^d$.

### A.2 SSVGP a-CAVI Algorithm

Here we derive the mathematical form of the reparameterisation gradient approximations of the evidence lower bound (ELBO) $\nabla_{\boldsymbol{\psi}}\hat{\mathcal{F}}$, $\nabla_{\boldsymbol{\alpha}}\hat{\mathcal{F}}$ and fixed point updates for variational parameters $\boldsymbol{\lambda}, \boldsymbol{\xi}$ and in a-CAVI algorithm 1. We derive these for the case that $q_{\boldsymbol{\psi}}(\boldsymbol{\theta})$ Mean-Field Gaussian, and discuss simplifications to this when the zero-temperature posterior restriction is used (i.e. in our main method).

#### A.2.1 ELBO Gradients

We first derive expressions 8 and 9 in the main text for $\nabla_{\boldsymbol{\psi}}\hat{\mathcal{F}}$ and $\nabla_{\boldsymbol{\phi}}\hat{\mathcal{F}}$ respectively. We start with expression 5 in the main text for the evidence lower bound:

$$\mathcal{F} = \langle \log p_{\boldsymbol{\phi}}(\boldsymbol{y}|\boldsymbol{\theta})\rangle_{q_{\boldsymbol{\psi}}(\boldsymbol{\theta})} - KL[q_{\boldsymbol{\psi}}(\boldsymbol{\theta})q(\boldsymbol{\gamma})q(\pi)||p(\boldsymbol{\theta},\boldsymbol{\gamma},\pi)] \tag{1}$$

Taking gradients wrt. $\boldsymbol{\psi}$ we have:

$$\nabla_{\boldsymbol{\psi}}\mathcal{F} = \nabla_{\boldsymbol{\psi}}\langle \log p_{\boldsymbol{\phi}}(\boldsymbol{y}|\boldsymbol{\theta})\rangle_{q_{\boldsymbol{\psi}}(\boldsymbol{\theta})} - \nabla_{\boldsymbol{\psi}}KL[q_{\boldsymbol{\psi}}(\boldsymbol{\theta})q(\boldsymbol{\gamma})q(\pi)||p(\boldsymbol{\theta},\boldsymbol{\gamma},\pi)] \tag{2}$$

The first term involves the gradient of an expectation of both (i) a log-determinant and (ii) inverse of the gram matrix $\tilde{K}_{XX} = K_{XX} + \sigma^2 I$. This is intractable to compute for most kernel functions, and thus needs to be approximated We use the reparameterisation trick (Kingma and Welling, 2013) for this. The KL gradient however is exactly tractable.

For the reparameterisation trick to apply, $q_{\boldsymbol{\psi}}(\boldsymbol{\theta})$ must be parameterised such that $\boldsymbol{\theta} = \mathcal{T}_{\boldsymbol{\psi}}(\boldsymbol{\epsilon})$ and $\boldsymbol{\epsilon} \sim q(\boldsymbol{\epsilon})$. Here $\mathcal{T}_{\boldsymbol{\psi}}(\cdot)$ is a transformation differentiable in $(\boldsymbol{\psi}, \boldsymbol{\epsilon})$, and $q(\boldsymbol{\epsilon})$ is a base distribution that does not depend on $\boldsymbol{\psi}$. In this case, using the law of the unconscious statistician and the chain rule we have the following:

$$\begin{aligned}
\nabla_{\boldsymbol{\psi}}\langle \log p_{\boldsymbol{\phi}}(\boldsymbol{y}|\boldsymbol{\theta})\rangle_{q_{\boldsymbol{\psi}}(\boldsymbol{\theta})} &= \nabla_{\boldsymbol{\psi}}\int q_{\boldsymbol{\psi}}(\boldsymbol{\theta})\log p_{\boldsymbol{\phi}}(\boldsymbol{y}|\boldsymbol{\theta})d\boldsymbol{\theta} &(3)\\
&= \nabla_{\boldsymbol{\psi}}\int q(\boldsymbol{\epsilon})\log p_{\boldsymbol{\phi}}(\boldsymbol{y}|\boldsymbol{\theta})\Big|_{\boldsymbol{\theta}=\mathcal{T}_{\boldsymbol{\psi}}(\boldsymbol{\epsilon})} d\boldsymbol{\epsilon} &(4)\\
&= \int q(\boldsymbol{\epsilon})\nabla_{\boldsymbol{\theta}}\log p_{\boldsymbol{\phi}}(\boldsymbol{y}|\boldsymbol{\theta})\Big|_{\boldsymbol{\theta}=\mathcal{T}_{\boldsymbol{\psi}}(\boldsymbol{\epsilon})}\nabla_{\boldsymbol{\psi}}\mathcal{T}_{\boldsymbol{\psi}}(\boldsymbol{\epsilon})d\boldsymbol{\epsilon} &(5)\\
&= \langle \log p_{\boldsymbol{\phi}}(\boldsymbol{y}|\boldsymbol{\theta})\nabla_{\boldsymbol{\psi}}\mathcal{T}_{\boldsymbol{\psi}}(\boldsymbol{\epsilon})\rangle_{q(\boldsymbol{\epsilon})} &(6)
\end{aligned}$$

Thus, given $S$ Monte Carlo samples $\boldsymbol{\epsilon}^{(1:S)} \sim q(\boldsymbol{\epsilon})$ our unbiased approximation to $\nabla_{\boldsymbol{\psi}}\mathcal{F}$ is:

$$\hat{\nabla}_{\boldsymbol{\psi}}\mathcal{F} = \frac{1}{S}\sum_{s=1}^{S}\left(\nabla_{\boldsymbol{\theta}}\log p_{\boldsymbol{\phi}}(\boldsymbol{y}|\boldsymbol{\theta})\Big|_{\boldsymbol{\theta}=\mathcal{T}_{\boldsymbol{\psi}}(\boldsymbol{\epsilon}^{(s)})}\nabla_{\boldsymbol{\psi}}\mathcal{T}_{\boldsymbol{\psi}}(\boldsymbol{\epsilon}^{(s)})\right) - \nabla_{\boldsymbol{\psi}}KL[q_{\boldsymbol{\psi}}(\boldsymbol{\theta})q(\boldsymbol{\gamma})||p(\boldsymbol{\theta}|\boldsymbol{\gamma})] \tag{7}$$

Note here that we have used that $\nabla_{\boldsymbol{\psi}}KL[q_{\boldsymbol{\psi}}(\boldsymbol{\theta})q(\boldsymbol{\gamma})q(\pi)||p(\boldsymbol{\theta},\boldsymbol{\gamma},\pi)] = \nabla_{\boldsymbol{\psi}}KL[q_{\boldsymbol{\psi}}(\boldsymbol{\theta})q(\boldsymbol{\gamma})||p(\boldsymbol{\theta}|\boldsymbol{\gamma})]$. Note for Mean-Field Gaussian $q_{\boldsymbol{\psi}}(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, Diag(\sigma_1^2, ..., \sigma_d^2))$ we set $\mathcal{T}_{\boldsymbol{\psi}}(\boldsymbol{\epsilon}) = \boldsymbol{\sigma}\odot\boldsymbol{\epsilon} + \boldsymbol{\mu}$ and $q(\boldsymbol{\epsilon}) = \mathcal{N}(\boldsymbol{0}, I)$.

The equivalent gradient approximation for $\hat{\nabla}_{\boldsymbol{\phi}}\mathcal{F}$ given in the main text is trivial to derive:

$$\begin{aligned}
\hat{\nabla}_{\boldsymbol{\phi}}\mathcal{F} &= \nabla_{\boldsymbol{\phi}}\langle \log p_{\boldsymbol{\phi}}(\boldsymbol{y}|\boldsymbol{\theta})\rangle_{q_{\boldsymbol{\psi}}(\boldsymbol{\theta})} &(8)\\
&= \int \nabla_{\boldsymbol{\phi}}\log p_{\boldsymbol{\phi}}(\boldsymbol{y}|\boldsymbol{\theta})q_{\boldsymbol{\psi}}(\boldsymbol{\theta})d\boldsymbol{\theta} &(9)\\
\hat{\nabla}_{\boldsymbol{\phi}}\mathcal{F} &\approx \frac{1}{S}\sum_{s=1}^{S}\nabla_{\boldsymbol{\phi}}\log p_{\boldsymbol{\phi}}(\boldsymbol{y}|\boldsymbol{\theta}^{(s)}) : \boldsymbol{\theta}^{(s)} \sim q_{\boldsymbol{\psi}}(\boldsymbol{\theta}^{(s)}) &(10)
\end{aligned}$$

We now derive the closed forms of the main components in the gradient approximations above. Note that when $\log p_{\boldsymbol{\phi}}(\boldsymbol{y}|\boldsymbol{\theta}) = \log\mathcal{N}(\boldsymbol{y}|\boldsymbol{0}, \tilde{K}_{XX})$ with $n \times n$ gram matrix $\tilde{K}_{XX}$ comprised of covariance kernel function evaluations and the additive noise term $[\tilde{K}_{XX}]_{ij} = k_{\boldsymbol{\theta},\tau}(\boldsymbol{x}_i, \boldsymbol{x}_j) + \delta_0(i=j)\sigma^2$, then using standard results derived in Rasmussen and Williams (2006) we have:

$$\nabla_{\theta_j}\log_{\boldsymbol{\phi}} p(\boldsymbol{y}|\boldsymbol{\theta}) = Tr\left[(\boldsymbol{\alpha}\boldsymbol{\alpha}^T - \tilde{K}_{XX}^{-1})\frac{d\tilde{K}_{XX}}{d\theta_j}\right] \quad : \quad \boldsymbol{\alpha} = \tilde{K}_{XX}^{-1}\boldsymbol{y} \tag{11}$$

$$\nabla_{\phi_j} \log_{\boldsymbol{\phi}} p(\boldsymbol{y}|\boldsymbol{\theta}) = Tr\left[ (\boldsymbol{\alpha}\boldsymbol{\alpha}^T - \tilde{K}_{XX}^{-1}) \frac{d\tilde{K}_{XX}}{d\phi_j} \right] \quad : \quad \boldsymbol{\alpha} = \tilde{K}_{XX}^{-1}\boldsymbol{y} \tag{12}$$

Menawhile, the closed form for the gradient of the KL term wrt. $\boldsymbol{\psi}$ is:

$$\nabla_{\boldsymbol{\psi}} KL[q_{\boldsymbol{\psi}}(\boldsymbol{\theta})q(\boldsymbol{\gamma})||p(\boldsymbol{\theta}|\boldsymbol{\gamma})] = \nabla_{\boldsymbol{\psi}} \int \sum_{\boldsymbol{\gamma}} q_{\boldsymbol{\psi}}(\boldsymbol{\theta})q(\boldsymbol{\gamma}) \log p(\boldsymbol{\theta}|\boldsymbol{\gamma}) d\boldsymbol{\theta} + \nabla_{\boldsymbol{\psi}} H[q_{\boldsymbol{\psi}}(\boldsymbol{\theta})] \tag{13}$$

$$= \nabla_{\boldsymbol{\psi}} \sum_{j=1}^{d} \Big[ \langle \gamma_j \rangle_{q(\gamma_j)} \langle \log \mathcal{N}(\theta_j|0,(cv)^{-1}) \rangle_{q_{\boldsymbol{\psi}_j}(\theta_j)} + (1 - \langle \gamma_j \rangle_{q(\gamma_j)}) \langle \log \mathcal{N}(\theta_j|0, v^{-1}) \rangle_{q_{\boldsymbol{\psi}_j}(\theta_j)} \Big] + \nabla_{\boldsymbol{\psi}} H[q_{\boldsymbol{\psi}}(\boldsymbol{\theta})] \tag{14}$$

$$= -\frac{1}{2} \nabla_{\boldsymbol{\psi}} \sum_{j=1}^{d} \Big[ \lambda_j cv \langle \theta_j^2 \rangle_{q_{\boldsymbol{\psi}}(\boldsymbol{\theta})} + (1-\lambda_j) v \langle \theta_j^2 \rangle_{q_{\boldsymbol{\psi}}(\boldsymbol{\theta})} \Big] + \nabla_{\boldsymbol{\psi}} H[q_{\boldsymbol{\psi}}(\boldsymbol{\theta})] \tag{15}$$

$$= -\frac{v}{2} \nabla_{\boldsymbol{\psi}} \sum_{j=1}^{d} (\lambda_j c + 1 - \lambda_j) \langle \theta_j^2 \rangle_{q_{\boldsymbol{\psi}}(\boldsymbol{\theta})} + \nabla_{\boldsymbol{\psi}} H[q_{\boldsymbol{\psi}}(\boldsymbol{\theta})] \tag{16}$$

Note that $H[q_{\boldsymbol{\psi}}(\boldsymbol{\theta})]$ is the Shannon entropy of $q_{\boldsymbol{\psi}}(\boldsymbol{\theta})$. Substituting in $q_{\boldsymbol{\psi}}(\boldsymbol{\theta}) = \prod_{j=1}^{d} \mathcal{N}(\theta_j|\mu_j, \sigma_j^2)$, and using standard Gaussian algebra we get the equations in the main text.

Under the zero-temperature restriction ($q_{\boldsymbol{\psi}}(\boldsymbol{\theta}) = \prod_{j=1}^{d} \delta_{\mu_j}(\theta_j)$) the gradient approximations are unchanged, except now the approximation is exact for a single sample from the point mass ($\boldsymbol{\theta}^{(s)} = \boldsymbol{\mu}$), and $\boldsymbol{\psi} = \boldsymbol{\mu}$.

### A.2.2 Fixed Points

Starting with the standard CAVI update expression in Blei et al. (2017), the update for $q(\boldsymbol{\gamma})$ is given by:

$$q(\boldsymbol{\gamma}) \propto \exp\left\{ \langle \log p(\boldsymbol{\theta}, \boldsymbol{\gamma}|\pi) \rangle_{q_{\boldsymbol{\psi}}(\boldsymbol{\theta})q(\pi)} \right\}$$

$$\propto \prod_{j=1}^{d} \exp\Big\{ \gamma_j \left\langle \log \mathcal{N}(\theta_j|0,(cv)^{-1}) \right\rangle_{q_{\boldsymbol{\psi}_j}(\theta_j)} + (1-\gamma_j) \left\langle \log \mathcal{N}(\theta_j|0, v^{-1}) \right\rangle_{q_{\boldsymbol{\psi}_j}(\theta_j)} + \gamma_j \langle \log \pi \rangle_{q(\pi)} + (1-\gamma_j) \langle \log(1-\pi) \rangle_{q(\pi)} \Big\} \tag{17}$$

Passing over the Gaussian expecations, we have due to the factorisation of the RHS that:

$$q(\gamma_j = 1) = \lambda_j \propto c^{\frac{1}{2}} e^{-\frac{1}{2} cv \langle \theta_j^2 \rangle_{q_{\boldsymbol{\psi}_j}(\theta_j)} + \langle \log \pi \rangle_{q(\pi)}} \tag{18}$$

$$q(\gamma_j = 0) \propto e^{-\frac{1}{2} v \langle \theta_j^2 \rangle_{q_{\boldsymbol{\psi}_j}(\theta_j)} + \langle \log(1-\pi) \rangle_{q(\pi)}} \tag{19}$$

After renormalisation we arrive at the update used in a-CAVI algorithm 1:

$$\lambda_j = \left( 1 + \left(\frac{1}{c}\right)^{\frac{1}{2}} e^{-\frac{1}{2} \langle \theta_j^2 \rangle_{q_{\boldsymbol{\psi}_j}(\theta_j)} v(1-c) + \langle \log\left(\frac{1-\pi}{\pi}\right) \rangle_{q(\pi)}} \right)^{-1} \tag{20}$$

Where note that $\langle \boldsymbol{\theta} \odot \boldsymbol{\theta} \rangle_{q_{\boldsymbol{\psi}}(\boldsymbol{\theta})} = \boldsymbol{\mu} \odot \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\sigma}$ using standard properties of multivariate Gaussians. Similarly, since $q(\pi) = \mathcal{B}eta(\pi|\xi_a, \xi_b)$ (see below), we have using the properties of the Beta distribution that for digamma function $\tilde{\psi}_0(\cdot)$:

$$\left\langle \log\left(\frac{1-\pi}{\pi}\right) \right\rangle_{q(\pi)} = \tilde{\psi}_0(\xi_b) - \tilde{\psi}_0(\xi_a) \tag{21}$$

For the CAVI update for $q(\pi)$ we have:

$$\begin{aligned} q(\pi) &\propto exp\left\{\langle \log p(\boldsymbol{\gamma}, \pi)\rangle_{q(\boldsymbol{\gamma})}\right\} \\ &\propto exp\left\{(a-1+\sum_{j=1}^{d}\langle\gamma_j\rangle_{q(\gamma_j)})\log(\pi) + (b-1+d-\sum_{j=1}^{d}\langle\gamma_j\rangle_{q(\gamma_j)})\log(1-\pi)\right\} \\ &\propto \mathcal{B}eta\left(\pi|a+\sum_{j=1}^{d}\langle\gamma_j\rangle_{q(\gamma_j)}, b+d-\sum_{j=1}^{d}\langle\gamma_j\rangle_{q(\gamma_j)}\right) \\ &\propto \mathcal{B}eta\left(\pi|a+d\bar{\lambda}, b+d(1-\bar{\lambda})\right) \quad : \quad \bar{\lambda} = \frac{1}{d}\sum_{j=1}^{d}\lambda \end{aligned} \tag{22}$$

Therefore $\boldsymbol{\xi} = (\xi_a, \xi_b) = \left(a+\sum_{j=1}^{d}\langle\gamma_j\rangle_{q(\gamma_j)}, b+d-\sum_{j=1}^{d}\langle\gamma_j\rangle_{q(\gamma_j)}\right)$ and the expected sufficient statistic $\langle\log\frac{1-\pi}{\pi}\rangle_{q(\pi)}$ is given by expression 21 above.

### A.3 SSVGP Leave-one-out Predictive Density (LOOPD)

Here we briefly derive the mathematical form of the leave-one-out predictive density under a given model $\mathcal{M}_k$ in the BMA ensemble under the zero-temperature posterior. The easiest way to do this is to note that model $\mathcal{M}_k$ is defined by parameters $\boldsymbol{\nu}_k = (\boldsymbol{\mu}_k, \boldsymbol{\phi}_k, a, b, c, v_k)$. Note here we treat inverse lengthscales $\boldsymbol{\theta}$ as fixed at posterior means $\boldsymbol{\mu}_k$, since this is equivalent to using a zero-temperature posterior $q_{\boldsymbol{\psi}}(\boldsymbol{\theta}) = \prod_{j=1}^{d}\delta_{\mu_j}(\theta_j)$. Thus, since kernel parameters are fixed the LOOPD for model $k$ is simply that of a standard GP evaluated at $(\boldsymbol{\theta} = \boldsymbol{\mu}_k, \boldsymbol{\phi} = \boldsymbol{\phi}_k)$ (Rasmussen and Williams, 2006) (note we assume a zero mean GP prior as typical):

$$\prod_{i=1}^{n} p(y_i|\mathcal{D}_{\neg i}, \boldsymbol{x}_i, \mathcal{M}_k) = \prod_{i=1}^{n} p_{\boldsymbol{\phi}_k,\theta_k}(y_i|\mathcal{D}_{\neg i}, \boldsymbol{x}_i) \tag{23}$$

$$= \prod_{i=1}^{n}\mathcal{N}(y_i|\mu_i, \sigma_i^2) \tag{24}$$

$$\mu_i = \boldsymbol{k}_{X_{\neg i}}(\boldsymbol{x}_i)^T\left(K_{X_{\neg i}X_{\neg i}} + \sigma^2 I\right)^{-1}\boldsymbol{y}_{\neg i} \tag{25}$$

$$\sigma_i^2 = k_{\boldsymbol{\mu}_k,\tau_k}(\boldsymbol{x}_i, \boldsymbol{x}_i) - \boldsymbol{k}_{X_{\neg i}}(\boldsymbol{x}_i)^T\left(K_{X_{\neg i}X_{\neg i}} + \sigma^2 I\right)^{-1}\boldsymbol{k}_{X_{\neg i}}(\boldsymbol{x}_i) + \sigma^2 \tag{26}$$

Here $K_{X_{\neg i}X_{\neg i}}$ is the $(n-1)\times(n-1)$ gram matrix of training points when removing $(y_i, \boldsymbol{x}_i)$, and $\boldsymbol{k}_{X_{\neg i}}(\boldsymbol{x}_i)$ is an $n-1$ dimensional vector of kernel function evaluations between $\boldsymbol{x}_i$ and all other training points. As discussed in the main text we use the algorithm in Bürkner et al. (2021) in small-$n$ scenarios ($\mathcal{O}(n^3)$ complexity) and the nearest neighbour truncation algorithm in Jankowiak and Pleiss (2021) in large-$n$ scenarios ($\mathcal{O}(nf(n)d)$ complexity with $f(n) \in [log(n), n]$). In some circumstances we add a small regularisation term of $0 < \kappa \ll 1$ to the posterior variances small-$n$ scenarios to prevent outliers from blowing up the negative log LOOPD. We only find it necessary to do this in the toy example (setting $\kappa = 0.1$), but in all main experiments performance is robust to $\kappa \in [0, 1]$.

### A.4 SSVGP Predictive Distribution

Here we derive the closed form predictive moments given by the predictive distribution approximations in eq. 11 and 15 in the main text. Since both distributions are discrete mixtures of standard GP posterior predictives with the $k^{th}$ component parameterised by inverse lengthscales $\boldsymbol{\theta}_k$ kernel scale and noise parameters $\boldsymbol{\phi}_k$, we derive the moments for this distributional family. However note that for eq. 11 in the main text (i.e. the SSVGP posterior predictive with Mean-Field Gaussian $q_{\boldsymbol{\psi}}(\boldsymbol{\theta})$ and fixed $v$), the distribution is an approximation to the true predictive posterior, and thus the predictive covariance derived below is a biased but consistent estimator (it contains a product of approximate expectations).

We define the distributional family we work with as follows:

$$p(\boldsymbol{y}_*|X_*,\mathcal{D}) = \sum_{k=1}^{K} p_{\boldsymbol{\theta}_k,\boldsymbol{\phi}_k}(\boldsymbol{y}_*|X_*,\mathcal{D})w_k \tag{27}$$

Here $\{w_k\}_{k=1}^{k}$ are the mixture weights and $p_{\boldsymbol{\theta}_k,\boldsymbol{\phi}_k}(\boldsymbol{y}_*|X_*,\mathcal{D})$ is a standard GP posterior predictive given $(\boldsymbol{\theta}_k,\boldsymbol{\phi}_k)$. In this case, the posterior predictive posterior mean is computed as:

$$\begin{aligned}\boldsymbol{m}(X_*) = \mathbb{E}[\boldsymbol{y}_*|X_*,\mathcal{D}] &= \int \boldsymbol{y}_* \sum_{k=1}^{K} p_{\boldsymbol{\theta}_k,\boldsymbol{\phi}_k}(\boldsymbol{y}_*|X_*,\mathcal{D})w_k d\boldsymbol{y}_* \\ &= \sum_{k=1}^{K} w_k \int \boldsymbol{y}_* p_{\boldsymbol{\theta}_k,\boldsymbol{\phi}_k}(\boldsymbol{y}_*|X_*,\mathcal{D})d\boldsymbol{y}_* \\ &= \sum_{k=1}^{K} w_k \boldsymbol{\xi}_{\boldsymbol{\theta}_k,\boldsymbol{\phi}_k}(X_*)\end{aligned} \tag{28}$$

Where $\boldsymbol{\xi}_{\boldsymbol{\theta}_k,\boldsymbol{\phi}_k}(X_*) = \mathbb{E}_{p_{\boldsymbol{\theta}_k,\boldsymbol{\phi}_k}(\boldsymbol{y}_*|X_*,\mathcal{D})}[\boldsymbol{y}_*]$ is the mean of standard GP posterior predictive component $p_{\boldsymbol{\theta}_k,\boldsymbol{\phi}_k}(\boldsymbol{y}_*|X_*,\boldsymbol{\theta}_k,\mathcal{D})$ (see Rasmussen and Williams (2006) for details).

Similarly, using the definition of covariance: $\mathbb{V}[\boldsymbol{x}] = \mathbb{E}[\boldsymbol{x}\boldsymbol{x}^T] - \mathbb{E}[\boldsymbol{x}]\mathbb{E}[\boldsymbol{x}]^T$, the posterior covariance is given as:

$$\begin{aligned}\boldsymbol{V}(X_*) = \mathbb{V}[\boldsymbol{y}_*|X_*,\mathcal{D}] &= \sum_{k=1}^{K} w_k \int \boldsymbol{y}_*\boldsymbol{y}_*^T p_{\boldsymbol{\theta}_k,\boldsymbol{\phi}_k}(\boldsymbol{y}_*|X_*,\mathcal{D})d\boldsymbol{y}_* - \boldsymbol{m}(X_*)\boldsymbol{m}(X_*)^T \\ &= \sum_{k=1}^{K} w_k \left(\boldsymbol{\Sigma}_{\boldsymbol{\theta}_k,\boldsymbol{\phi}_k}(X_*) + \xi_{\boldsymbol{\theta}_k,\boldsymbol{\phi}_k}(X_*)\xi_{\boldsymbol{\theta}_k,\boldsymbol{\phi}_k}(X_*)^T\right) - \boldsymbol{m}(X_*)\boldsymbol{m}(X_*)^T\end{aligned} \tag{29}$$

Where $\boldsymbol{\Sigma}_{\boldsymbol{\theta}_k,\boldsymbol{\phi}_k}(X_*) = \mathbb{V}ar_{p_{\boldsymbol{\theta}_k,\boldsymbol{\phi}_k}(\boldsymbol{y}_*|X_*,\mathcal{D})}[\boldsymbol{y}_*]$ is the covariance of standard GP posterior predictive component $p_{\boldsymbol{\theta}_k,\boldsymbol{\phi}_k}(\boldsymbol{y}_*|X_*,\boldsymbol{\theta}_k,\mathcal{D})$ (see Rasmussen and Williams (2006) for details)

### A.5 SSVGP Posterior Point of Intersection (PPI)

Here we derive equation 14 in the main text for the posterior point of intersection (PPI). The PPI is given by $\hat{\theta} = \sqrt{\langle\theta^2\rangle_{q_{\boldsymbol{\psi}_j}(\theta)}}$ such that $q(\gamma=1) = q(\gamma=0) = \frac{1}{2}$. To derive the expression we substitute $\lambda = \frac{1}{2}$ into the optimal CAVI update for $\lambda$ given by equation 20:

$$\frac{1}{2} = \left(1 + \left(\frac{1}{c}\right)^{\frac{1}{2}} e^{-\frac{1}{2}\langle\theta_j^2\rangle_{q_{\boldsymbol{\psi}_j}(\theta_j)}v(1-c)+\langle\log\left(\frac{1-\pi}{\pi}\right)\rangle_{q(\pi)}}\right)^{-1} \tag{30}$$

Re-arranging terms we get:

$$1 = \left(\frac{1}{c}\right)^{\frac{1}{2}} e^{-\frac{1}{2}\langle\theta_j^2\rangle_{q_{\boldsymbol{\psi}_j}(\theta_j)}v(1-c)+\langle\log\left(\frac{1-\pi}{\pi}\right)\rangle_{q(\pi)}} \tag{31}$$

$$0 = \frac{1}{2}\log\left(\frac{1}{c}\right) - \frac{1}{2}\langle\theta_j^2\rangle_{q_{\boldsymbol{\psi}_j}(\theta_j)}v(1-c) + \left\langle \log\left(\frac{1-\pi}{\pi}\right)\right\rangle_{q(\pi)} \tag{32}$$

Taking the expected sufficient statistic of $\theta_j$ to the LHS and simplifying we get:

$$\langle\theta_j^2\rangle_{q_{\boldsymbol{\psi}_j}(\theta_j)}v(1-c) = \log\left(\frac{1}{c}\right) + 2\left\langle \log\left(\frac{1-\pi}{\pi}\right)\right\rangle_{q(\pi)} \tag{33}$$

Thus finally we arrive at the equation in the main text:

$$\sqrt{\langle\theta_j^2\rangle_{q_{\boldsymbol{\psi}_j}(\theta_j)}} = \hat{\theta} = \sqrt{\frac{\log\left(\frac{1}{c}\right) + 2\langle\log\frac{1-\pi}{\pi}\rangle_{q(\pi)}}{v(1-c)}} \tag{34}$$

## A.6 Free Energy Fixed Variable Inclusion Cost

In Section 4 we discuss that the evidence lower bound has a fixed variable 'inclusion cost' of order $\log(\frac{1}{c})$, making it unideal for model selection as $c$ can heavily impact the degree of preference for sparse models. Here we derive the source of this inclusion cost. Starting from the KL regularisation term of the evidence lower bound, note that under and arbitrary inverse lengthscale posterior $q_{\boldsymbol{\psi}}(\boldsymbol{\theta})$, if we retain only the terms that include $\{\langle\theta_j^2\rangle_{q_{\boldsymbol{\psi}_j}(\theta_j)}\}_{j=1}^d$ and $\{\lambda_j\}_{j=1}^d$, we are left with:

$$\begin{aligned}-KL[q_{\boldsymbol{\psi}}(\boldsymbol{\theta})q(\boldsymbol{\gamma})q(\pi)||p(\boldsymbol{\theta},\boldsymbol{\gamma},\pi)] = &\frac{1}{2}\sum_{j=1}^d\lambda_j\log(c) - \frac{v}{2}\sum_{j=1}^d\langle\theta_j^2\rangle_{q_{\boldsymbol{\psi}_j}(\theta_j)}(\lambda_j c+1-\lambda_j)\\ &+\sum_{j=1}^d\lambda_j\left\langle\log\left(\frac{\pi}{\lambda_j}\right)\right\rangle_{q(\pi)}+\sum_{j=1}^d(1-\lambda_j)\left\langle\log\left(\frac{1-\pi}{1-\lambda_j}\right)\right\rangle_{q(\pi)}\\ &+const.wrt.(\{\langle\theta_j^2\rangle_{q_{\boldsymbol{\psi}_j}(\theta_j)},\lambda_j\}_{j=1}^d)\end{aligned} \tag{35}$$

The contribution of variable $j$ to these components of the KL is given by:

$$\zeta_j = \lambda_j\frac{1}{2}\log(c) - \frac{v}{2}\langle\theta_j^2\rangle_{q_{\boldsymbol{\psi}_j}(\theta_j)}(\lambda_j c+1-\lambda_j)+\lambda_j\left\langle\log\left(\frac{\pi}{\lambda_j}\right)\right\rangle_{q(\pi)}+(1-\lambda_j)\left\langle\log\left(\frac{1-\pi}{1-\lambda_j}\right)\right\rangle_{q(\pi)} \tag{36}$$

Clearly when moving from $\lambda_j : 0 \to 1$, there is a fixed inclusion cost from the first term of $\frac{1}{2}\log\left(\frac{1}{c}\right)$. As a result the value of $c$ can have a substantial effect on whether $\mathcal{F}$ is higher when the average PIP is higher, which we find experimentally. This makes it challenging to use $\mathcal{F}$ reliably to do discrete model selection or averaging.

## A.7 Dirac Spike with Paired Mean Field (PMF) Approximation

In Section 4 we also implement a spike and slab GP using a Dirac point mass at zero for the spike prior distribution, and the paired mean field variational approximation used in Dai et al. (2015) and Carbonetto and Stephens (2012):

$$q(\boldsymbol{\theta}|\boldsymbol{\gamma})q(\boldsymbol{\gamma}) = \prod_{d=1}^d[\mathcal{N}(\theta_j|\mu_j,\sigma_j^2)\lambda_j]^{\gamma_j}[\delta_0(\theta_j)(1-\lambda_j)]^{1-\gamma_j} \tag{37}$$

This factorisation has the attractive property of retaining posterior dependence between the $j^{th}$ inclusion variable $\gamma_j$ and inverse lengthscale $\theta_j$. However the cost of this dependency is that exact CAVI updates for $q(\boldsymbol{\gamma})$ are no longer tractable as it requires computing expectations of the likelihood term $\log p(\boldsymbol{y}|\boldsymbol{\theta})$, which are intractable for most kernel functions:

$$q(\gamma_j) \propto \exp\left(\langle\log p(\boldsymbol{y}|\boldsymbol{\theta})\rangle_{q(\boldsymbol{\theta}|\boldsymbol{\gamma})q(\boldsymbol{\gamma}_{\neg j})}+\langle\log p(\theta_j,\gamma_j)\rangle_{q(\theta_j|\gamma_j)}\right) \tag{38}$$

Note that even if the expectations $\langle\log p(\boldsymbol{y}|\boldsymbol{\theta})\rangle_{q(\boldsymbol{\theta}|\boldsymbol{\gamma})}$ could be computed for a setting of $\boldsymbol{\gamma}$, we would need to compute $2^{d-1}$ of these quantities. The complexity of this CAVI update would therefore be $\mathcal{O}(2^d n^3)$, which is computationally infeasible for large-$n$ or even moderate $d$.

Our implementation therefore optimises $q(\boldsymbol{\theta},\boldsymbol{\gamma})$ using stochastic gradient variational inference. We use score-gradients (aka BBVI (Ranganath et al., 2014)) to estimate $\nabla_{\boldsymbol{\lambda}}\mathcal{F}$ (since $\boldsymbol{\gamma}$ are non-differentiable) and reparameterisation gradients (aka repgrad (Kingma and Welling, 2013)) to estimate $\nabla_{\boldsymbol{\psi}}\mathcal{F}$:

$$\nabla_{\boldsymbol{\psi}}\mathcal{F} \approx \frac{1}{S_0}\sum_{s=1}^{S_0}\nabla_{\boldsymbol{\theta}}\log p(\boldsymbol{y}|\boldsymbol{\theta}^{(s)})\Big|_{\boldsymbol{\theta}^{(s)}=\mathcal{T}_{\boldsymbol{\psi}}(\boldsymbol{\epsilon}^{(s)},\boldsymbol{z}^{(s)})}\nabla_{\boldsymbol{\psi}}\mathcal{T}_{\boldsymbol{\psi}}(\boldsymbol{\epsilon}^{(s)},\boldsymbol{z}^{(s)}) - \nabla_{\boldsymbol{\psi}}KL[q(\boldsymbol{\theta}|\boldsymbol{\gamma})q(\boldsymbol{\gamma})||p(\boldsymbol{\theta},\boldsymbol{\gamma})] \tag{39}$$

$$\nabla_{\boldsymbol{\lambda}}\mathcal{F} \approx \frac{1}{S_1}\sum_{s=1}^{S_1}\left(\log p(\boldsymbol{y}|\boldsymbol{\theta}^{(s)})\nabla_{\boldsymbol{\lambda}}\log q(\boldsymbol{\theta}^{(s)})\right)\Big|_{\boldsymbol{\theta}^{(s)}=\mathcal{T}_{\boldsymbol{\psi}}(\boldsymbol{\epsilon}^{(s)},\boldsymbol{z}^{(s)})} - \nabla_{\boldsymbol{\lambda}}KL[q(\boldsymbol{\theta}|\boldsymbol{\gamma})q(\boldsymbol{\gamma})||p(\boldsymbol{\theta},\boldsymbol{\gamma})] \tag{40}$$

Note that now the differentiable transformation is $\mathcal{T}_{\boldsymbol{\psi}}(\boldsymbol{\epsilon},\boldsymbol{z}) = \delta_{\mathbf{1}}(\boldsymbol{z}\leq\boldsymbol{\lambda})\odot(\boldsymbol{\epsilon}\odot\boldsymbol{\sigma}+\boldsymbol{\mu})$, whereby $\boldsymbol{z}\sim\mathcal{U}[0,1]^d$ and $\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},I)$ are drawn from the base distributions. Also note that the marginal distribution over inverse lengthscales is $q_{\boldsymbol{\psi}}(\boldsymbol{\theta}) = \prod_{j=1}^{d} \left[\lambda_j \mathcal{N}(\theta_j|\mu_j, \sigma_j^2) + (1-\lambda_j)\delta_0(\theta_j)\right]$. In practice we reparameterise the model in terms of $\boldsymbol{\kappa} \in \mathbb{R}^d$ where $\boldsymbol{\lambda} = (\mathbf{1} - \exp(-\boldsymbol{\kappa}))^{-1}$ and optimise $\boldsymbol{\kappa}$. We set $S_0 = 1$ consistent with our a-CAVI algorithm, but set $S_1 \in \{10, 100\}$ (i.e. two implementations used) due to the much higher degree of score-gradient variance. We treat $(\pi, v)$ as fixed parameters but pre-tune these to optimise performance for comparison against our SSVGP with the a-CAVI algorithm. See Appendix B.1 for implementation details.

### A.8 Proposition and Proof of a-CAVI Shrinkage Property

In this section we present a shrinkage property of a-CAVI Algorithm 1 along with a simple proof. We subsequently discuss how this property can be used to motivate the dropout pruning procedure we introduce in Section 4 of the main text. For simplicity our proof is for a 1d-input scenario but can be extended to multiple input dimensions.

The conditions required for this property to hold only apply under a simplification of the a-CAVI algorithm. However they mathematically formalise the notion that for a particular range of the spike and slab parameter $v$, small enough PIP values $\lambda_j < \epsilon \in (0,1)$ can enforce that $\mu_j$ converges to an $\delta$-neighbourhood of zero.

For the proof we require boundedness and Lipschitz continuity on the gradient of the log likelihood $\nabla_\theta \log p(\boldsymbol{y}|\theta)$. We define these assumptions as follows:

**DEFINITION 1** *For a real valued differentiable function $f : \mathcal{X} \to \mathbb{R}$ where $\mathcal{X} \subseteq \mathbb{R}^d, d \in \mathbb{N}_{>0}$, the gradient $\nabla f(\boldsymbol{x})$ is $B$-bounded if $\exists B \in \mathbb{R}_{\geq 0}$ such that $||\nabla f(\boldsymbol{x})||_2 \leq B \quad \forall \boldsymbol{x} \in \mathcal{X}$.*

**DEFINITION 2** *For a real valued differentiable function $f : \mathcal{X} \to \mathbb{R}$ where $\mathcal{X} \subseteq \mathbb{R}^d, d \in \mathbb{N}_{>0}$, the gradient $\nabla f(x)$ is $C$-Lipschitz continuous if $\exists C \in \mathbb{R}_{\geq 0}$ such that $||\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})||_2 \leq C||\boldsymbol{x} - \boldsymbol{y}||_2 \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$.*

Going forward we use $C$-smoothness to denote $C$-Lipschitz continuity for convenience.

**PROPOSITION 3** *Consider the spike and slab Gaussian process model defined in Section 3 of the main text where $\boldsymbol{y} \in \mathbb{R}^n, X \in \mathbb{R}^{n\times 1}$ is the dataset, $\theta \in \mathbb{R}$ is the inverse lengthscale (where parameterisation is chosen to enable $k_\theta(x,x') = k_{-\theta}(x,x')$ - see Appendix A.1 for details), $\boldsymbol{\phi} = (\tau, \sigma^2) \in \mathbb{R}^2_{>0}$ are kernel scale and noise parameters, $\gamma \in \{0,1\}$ is the latent binary inclusion variable, $\pi \in [0,1]$, and spike and slab prior is parameterised such that $(v,c) \in \mathbb{R}^2_{>0}$ with $0 < c \ll 1$. Let the a-CAVI algorithm be defined by the following simplifications. We use zero-temperature posterior $q_{\boldsymbol{\psi}_j}(\theta) = \delta_\mu(\theta)$ such that $\theta = \mu$, let the prior inclusion probability $\pi$ be fixed at $\pi_0 \in (0,1)$[1], fix $\boldsymbol{\phi} = \hat{\boldsymbol{\phi}} \in \mathbb{R}^2_{>0}$ and use a constant learning rate $\eta \in \mathbb{R}_+$. Let the log likelihood gradient $\nabla_\theta \log p(\boldsymbol{y}|\theta)$ be $B$-bounded and $C$-smooth.*

*Then, when running a-CAVI for $T > 1$ iterations, if at any iteration $t \in \{1, ..., T-1\}$ we get $\lambda^{(t)} \leq \epsilon$, and $\exists \delta > 0, \kappa > 0, \epsilon \in (0,1)$ such that:*

$$\frac{B+\kappa}{\delta(1-\epsilon)} \leq v \leq \frac{2}{\delta^2(1-c)} \log\left(\frac{\epsilon}{c^{\frac{1}{2}}(1-\epsilon)}\right) \tag{41}$$

$$0 < \eta \leq \frac{\alpha}{B + v(\delta+\alpha)} \quad \forall \alpha \in [\min(\delta, \frac{\kappa}{C+v}), \infty) \tag{42}$$

*we have that as $T \to \infty$: $|\mu^{(T)}| \to |\mu^\infty| \leq \delta$.*

PROOF:

We prove this by showing that under the assumptions above $\forall j \geq 1$: $|\mu^{(t+j)}| < |\mu^{(t+j-1)}|$ if $|\mu^{(t+j-1)}| > \delta$, and $|\mu^{(t+j)}| \leq \delta$ otherwise, which implies $\lambda^{(t+j)} \leq \epsilon$. Taking $j \to \infty$ we recover the result using bounds on $\nabla_\mu \mathcal{F}$ when $|\mu| > \delta$ that ensures the sequence $\{\mu^{(t+j)}\}_{j=1}^\infty$ converges to $\mu^\infty \in [-\delta, \delta]$. Note $\mu^{(t+1)}$ is obtained after running $S$ gradient-steps at a-CAVI iteration $t+1$, and $\lambda(\cdot)$ is the optimal CAVI update function in equation 20.

[1]This can be simulated by setting Beta prior hyperparameters to $a = \tau$, $b = \tau\left(\frac{1}{\pi_0} - 1\right)$ and letting $\tau \to \infty$.

We first show that if $\lambda^{(t)} \leq \epsilon$ and $|\mu^{(t)}| > \delta$ we must have that $\text{sign}(\nabla_\mu \mathcal{F}(\mu^{(t)}, \lambda^{(t)})) = -\text{sign}(\mu^{(t)})$. Starting with the LHS inequality of condition 41, using $B$-boundedness of the log-likelihood gradient and that due to our kernel parameterisation $\nabla_\mu \mathcal{F}(\mu, \cdot) = -\nabla_\mu \mathcal{F}(-\mu, \cdot)$, we have

$$\begin{aligned} v(1-\epsilon) &\geq \sup_{\theta \in (\delta, \infty)} \left\{ \frac{\nabla_\theta \log p(\boldsymbol{y}|\theta)}{\theta} \right\} + \frac{\kappa}{\delta} && (43) \\ &= \sup_{|\theta| \in (\delta, \infty)} \left\{ \frac{\nabla_\theta \log p(\boldsymbol{y}|\theta)}{\theta} \right\} + \frac{\kappa}{\delta} && (44) \\ &> \frac{\nabla_\theta \log p(\boldsymbol{y}|\theta) + (-1)^{\mathbb{I}(\theta<0)}\kappa}{\theta} \quad \forall |\theta| \in (\delta, \infty) && (45) \end{aligned}$$

where $\mathbb{I}(\cdot)$ is the indicator function. Substituting in $v(1-\lambda^{(t)} + \lambda^{(t)}c) > v(1-\epsilon)$ on the LHS, multiplying by $\theta$ and re-arranging, we get $\nabla_\mu \mathcal{F}(\mu^{(t)}, \lambda^{(t)}) < -\kappa$ when $\mu^{(t)} > \delta$ and $\nabla_\mu \mathcal{F}(\mu^{(t)}, \lambda^{(t)}) > \kappa$ when $\mu^{(t)} < -\delta$. This means the gradient always points to the origin when $|\mu^{(t)}| > \delta$.

We now show that for a small enough learning rate $\eta$ we can guarantee a single gradient step in $\mu$ will strictly decrease $|\mu^{(t)}|$ if $|\mu^{(t)}| > \delta$. To achieve this we require that:

$$|\mu^{(t+1)}| < |\mu^{(t)} + \eta \nabla_\mu \mathcal{F}(\mu^{(t)}, \lambda^{(t)})| \tag{46}$$

Starting with the positive case $\mu^{(t)} = \delta + \alpha : \alpha > 0$, due to the strictly negative gradient we only need

$$-(\delta + \alpha) < \delta + \alpha + \eta \nabla_\mu \mathcal{F}(\delta + \alpha, \lambda^{(t)}) \tag{47}$$

Re-arranging and using the strictly negative gradient we get:

$$\eta < \frac{2(\delta + \alpha)}{|\nabla_\mu \mathcal{F}(\delta + \alpha, \lambda^{(t)})|} \tag{48}$$

Now, since the log-likelihood gradient is $B$-bounded note that $\nabla_{\mu \in [-a,a]} \mathcal{F}$ is $(B + va)$-bounded, where $a > 0$. Thus we need

$$\eta < \frac{2(\delta + \alpha)}{B + v(\delta + \alpha)} \forall \alpha \geq 0 \tag{49}$$

Which is satisfied by assumption 42.

The same result holds by symmetry when examining the negative case $\mu^{(t)} = -(\delta + \alpha)$. By implication, every gradient step $s = 1, ..., S$ taken during a-CAVI iteration $t + 1$ to update $\mu^{(t)} = \mu_1^{(t)} \to \mu_S^{(t)} = \mu^{(t+1)}$, must strictly decrease $|\mu_s^{(t)}|$ at any step $s$ where $|\mu_s^{(t)}| \geq \delta$.

Now we show that if $|\mu^{(t)}| \leq \delta$ then so is $|\mu^{(t+1)}|$. For the positive case, denote $\mu^{(t)} = \delta - \alpha : \alpha \in [0, \delta]$ For the result, we need to guarantee:

$$\delta - \alpha + \eta \nabla_\mu \mathcal{F}(\delta - \alpha, \lambda^{(t)}) \in [-\delta, \delta] \tag{50}$$

Now note that since the log-likelihood gradient is $C$-smooth, then $\nabla_\mu \mathcal{F}$ must satisfy $(C + v)$-smoothness (from the gradient of the KL term). Thus, we have that $\nabla_\mu \mathcal{F}(\delta - \alpha, \lambda^{(t)}) \in [-B - v(\delta - \alpha), \min(B + v(\delta - \alpha), -\kappa + \alpha(C + v)]\ \forall \alpha \in [0, \delta]$. The $\min(\cdot, \cdot)$ accounts for the boundedness of $\nabla \mathcal{F}$ when $\mu \in [-\delta, \delta]$.

Using this condition, for the lower bound of eq. 50 to hold we need:

$$\delta - \alpha - \eta(B + v(\delta - \alpha)) \geq -\delta \tag{51}$$

Which implies

$$\eta \leq \frac{2\delta - \alpha}{B + v(\delta - \alpha)} \forall \alpha \in [0, \delta] \tag{52}$$

For the upper bound to hold we need to guarantee

$$\delta - \alpha + \eta \min(B + v(\delta - \alpha), -\kappa + \alpha(C + v)) \leq \delta \tag{53}$$

Which implies (noting we only need to consider the case where the gradient is strictly positive):

$$\eta \leq \frac{\alpha}{\min(B + v(\delta - \alpha), -\kappa + \alpha(C + v))} \forall \alpha \in (\min(\frac{\kappa}{C}, \delta), \delta] \tag{54}$$

Both eqs 52 and 54 are clearly satisfied by assumption 42. By symmetry of the problem, when we consider the negative case $\mu \in [-\delta, 0]$ we recover the same inequalities.

The above results mean that we must have

$$|\mu^{(t+1)}| < \max(|\mu^{(t)}|, \delta + \beta) \quad \forall \beta > 0 \tag{55}$$

Now, since $\lambda^{(t)} \leq \epsilon$, note that if 55 holds we must also have

$$\lambda^{(t+1)} < \max(\lambda^{(t)}, \lambda(\delta) + \beta) \quad \forall \beta > 0 \tag{56}$$

Where $\lambda(\cdot)$ is the optimal CAVI update used given by equation 20. The last line simply uses that under the assumptions in this proposition $\lambda'(\cdot) > 0$ everywhere.

From the LHS inequality of condition 41 and using 56 we also have $\lambda^{(t+1)} \leq \epsilon$, since rearranging this inequality in 41 we get $\lambda(\delta) \leq \epsilon$. Thus, the same results hold for iteration $t + j$ $\forall j \geq 1$ and so we can write:

$$|\mu^{(t+j)}| < \max(|\mu^{(t+j-1)}|, \delta + \beta) \quad \forall \beta > 0, \forall j \geq 1 \tag{57}$$

$$\lambda^{(t+j)} \leq \epsilon \quad \forall j \geq 1 \tag{58}$$

Since when $|\mu| > \delta$ we have by eq. 45 that $|\nabla_\mu \mathcal{F}| > \kappa$ and $\nabla_\mu \mathcal{F}$ points to the origin, we can write that:

$$|\mu^{(t+j)}| \leq |\mu^{(t)}| - j\eta\kappa \quad \forall j \geq 1, \forall |\mu^{(t)}| > \delta \tag{59}$$

By taking $j, T \to \infty$ we recover $|\mu^{\infty}| \leq \delta$. In other words, under the assumptions in the proposition, the gradient outside of the $\delta$-neighbourhood of zero is always large enough to get from $\mu^{(t)} \in \mathbb{R}$ to the $\delta$-neighbourhood in infinite computation time, if $\lambda^{(t)} \leq \epsilon$. $\square$

**REMARK 4** *Proposition 3 states that for a certain range of spike precision values and small enough learning rate $\eta$, if $\lambda$ is set small enough at any point during a-CAVI iterations, the algorithm is guaranteed to converge the inverse lengthscale parameter $\mu$ to an $\delta$-neighbourhood of zero in infinite computation time. In particular, the spike precision must be large enough to dominate the gradient information in the log-likelihood for suitably small $\lambda$, but small enough to guarantee $\lambda(\mu) \leq \epsilon$ for any $|\mu| \leq \delta$. Notably, no assumptions have been made on the data generating process here outside of the boundedness and smoothness of the log-likelihood gradient.*

**REMARK 5** *Proposition 3 supports our use of dropout pruning, as it suggests that under certain conditions, permanently enforcing $\mu = 0$ only deviates our found local optima within a $\delta$-neighbourhood, thus enabling us to enjoy its computational speed ups without notable performance loss.*

**REMARK 6** *We note that the proposition becomes less likely to hold as $n$ grows when using nearest neighbour minibatching, as the expected likelihood term contains a $\frac{n}{m}$ rescaling factor. In fact, this rescaling factor means that as $n \to \infty$ the recovered parameters $(\boldsymbol{\mu}, \boldsymbol{\phi})$ under ZT posterior $q_{\boldsymbol{\psi}}(\boldsymbol{\theta})$ should be no different than those recovered using the minibatch SGD procedure of Chen et al. (2020) i.e. the learned PIPs have no effect on $q_{\boldsymbol{\psi}}(\boldsymbol{\theta})$. In this case we motivate dropout pruning from the perspective of reinstating the 'message passing' link between $q(\boldsymbol{\gamma})$ and $q_{\boldsymbol{\psi}}(\boldsymbol{\theta})$, by augmenting the evidence lower bound with an L0 regularisation penalty on $\mu_j$ with penalty scale $\lim_{\tau \to \infty} (\tau \delta_1(\lambda_j \leq \epsilon))$.*

## B EXPERIMENTAL AND IMPLEMENTATION DETAILS

Here we discuss in detail the experiments ran and settings used for all methods implemented. All results were run on a 2.70GHz Intel(R) Core(TM) i7-10850H CPU with 32MB RAM. All code for our SSVGP is written in Python and is available at the Github repository https://github.com/HWDance/SSVGP. Note that barring the toy example, when we refer to the SSVGP, we refer to using our Bayesian model averaging (BMA) procedure introduced in Section 4 of the main paper, where models are trained using a-CAVI Algorithm 1 with the zero-temperature posterior $q_{\boldsymbol{\psi}}(\boldsymbol{\theta})$ and dropout pruning procedure. The first toy example is used to demonstrate that our initial method (without BMA, zero-temperature and dropout) can perform well, but the adjustments are required for simultaneously reliable, fast and scalable performance.

### B.1 Toy Example

Here we consider the problem of identifying relevant variables in a high-dimensional synthetic dataset. We draw a univariate response from a sinusoidal function of 5 inputs corrupted with a small amount of noise:

$$y = f(\boldsymbol{x}) + \epsilon \tag{60}$$

$$f(\boldsymbol{x}) = \sum_{j=1}^{5} sin(a_j x_j) \tag{61}$$

$$\boldsymbol{x} \sim \mathcal{N}_d(0, I) \tag{62}$$

$$\epsilon \sim \mathcal{N}(0, \sigma^2) \tag{63}$$

We set $d = 100$ with $\{a_j\}_{j=1}^5$ grid spaced over [0.5,1] and draw datasets of size $n = 300$ using this generative process, controlling the noise variance to $\frac{1}{20}$ of the signal variance: $\sigma^2 = 0.05Var[f(X)]$. Thus only 5/100 signals are relevant. Before running all methods we standardise $\mathcal{D} = (\boldsymbol{y}, X)$ to have mean zero and unit variance.

Our initial implementation of the SSVGP on this example uses a Mean-Field Gaussian approximation for inverse lengthscale posterior: $q_{\boldsymbol{\psi}}(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}, Diag(\sigma_1^2, ..., \sigma_d^2))$ and fixed value of spike prior hyperparametrer $v$. We train using $K = 5$ a-CAVI iterations with $T = 200$ gradient steps for the first a-CAVI iteration, and $T = 100$ steps thereafter. We use default ADAM smoothing parameters of $(\beta = 0.9, \beta_2 = 0.999)$ a learning rate of $\eta = 0.01$ and $S = 1$ Monte Carlo samples in reparameterisation gradient approximations. Prior hyperparameters are set as $(a, b, c, v) = (10^{-3}, 10^{-3}, 10^{-8}, 10^4)$ so that the spike precision $v = 10^4$ approximates a Dirac spike at zero, and the slab precision $cv = 10^{-4}$ approximates a uniform distribution. We initialise $\boldsymbol{\lambda} = \mathbf{1}$ (this is critical to avoid shrinking inverse lengthscales for relevant variables in the initialisation phase due to the regularisation of the KL term), and set $\boldsymbol{\xi} = \mathbf{1}$. We initialise inverse lengthscale posterior means to $\boldsymbol{\mu} = \mathbf{1} \times d^{-\frac{1}{2}}$ (this prevents numerical underflow in kernel evaluations in high-dimensional designs), and posterior variances to $\boldsymbol{\sigma} = 2 \times 10^{-3}$ and set $\boldsymbol{\phi} = \mathbf{1}$. We also add a jitter of $10^{-3}$ to the diagonals of $K_{XX}$.

The ML-II GP is implemented using the GPytorch (Gardner et al., 2018) library for 1000 iterations of ADAM with a learning rate of 0.1 and the same initialisation as our SSVGP for shared parameters. We use our own implementation of the Dirac spike and slab GP with paired mean field (PMF) approximation since there is no existing implementation in GP libraries (see Appendix A for mathematical model details). We run two versions with $S = 10$ and $S = 100$ Monte Carlo samples used respectively for score gradient estimation of $\nabla_{\boldsymbol{\lambda}} \mathcal{F}$. Since our implementation does not use control variates (Liu et al., 2019) or Rao-Blackwellisation (Salimans and Knowles, 2014) for variance reduction, we use the $S = 100$ sample version as an indication of obtainable performance using $S = 10$ samples were these techniques used, since Mohamed et al. (2020) find these can techniques can reduce variance by close to an order of magnitude in some experiments. We also optimise performance by a-priori fixing the prior inclusion probability to the true value ($\pi = 0.05$) and pre-tuning the slab precision $v_1 \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$. We train using the same number of total gradient iterations (500) and initialisation (for shared parameters) as the SSVGP except we initialise $\boldsymbol{\lambda} = 0.5$, which performed better than 1.

We subsequently re-implement the SSVGP (our final method) using the BMA procedure, zero-temperature posterior restriction and dropout pruning introduced in Section 4. We use 11 values of spike precision $v$ grid-spaced over $\in \mathcal{V} = 10^4 \times 2^{\{-log_2(1000), log_2(1000)\}}$ such that $v \in [10, 10^7]$. We set the pruning threshold to $\epsilon = 0.5$ and use a slightly larger base learning rate of $\eta = 0.05$ (our default recommendation in small-$n$ settings for our final method). We find this performs better in the absence of gradient variance due to the zero-temperature assumption. We use $m = \frac{n}{4}$ minibatching to further reduce runtime, and as we find this does not harm performance in our experiments. No nearest neighbour truncation of the LOOPD and predictive distribution is used.

We complete 10 trials of each method on synthetic dataset draws, reporting results in Tables 1 and 4 in the main text (we include test performance on $n_* = 100$ test points drawn from the same process). Figures 1 and 2 in the main text display average size ordered profiles for parameters that can be used to infer variable relevance for different methods (e.g. inverse lengthscales for the ML-II GP and PIPs for the SSVGP implementations). These profiles are constructed by size ordering the recovered parameters within each trial and then averaging the resulting profiles over the 10 trials. However we constrain the first 5 variables in the ordering (LHS of black dashed line) to be selected from the 5 relevant variables. Thus, the $4^{th}$ column in the variable ranking corresponds to the average value of the $4^{th}$ largest parameter recovered amongst the relevant variables, and the $10^{th}$ column corresponds to the average value of the $5^{th}$ largest parameter recovered amongst the irrelevant variables.

These profiles enable us to clearly visualise the ability of our methods to better distinguish between irrelevant and relevant variables correctly vs. the ML-II GP. In particular, we see that the ML-II GP produces a smooth decline in the profile with no real discontinuity at the true inclusion/exclusion threshold, and fails to shrink the inverse lengthscales to arbitrarily small values (e.g.$< 0.01$) for the irrelevant dimensions (see Fig 1), making it very challenging to use thresholding reliably to do variable selection. By contrast, our SSVGP shows a huge discontinuity both in PIPs and (posterior mean) inverse lengthscales at the true inclusion cut-off (see Fig. 2), and also learns to include all relevant variables with PIP $\approx 1$ and include all irrelevant variables with PIP$\ll$0.5.

### B.1.1 Analysis of Posterior Point of Intersection (PPI)

In Section 4 we demonstrated the sensitivity of the a-CAVI algorithm recovered PIPs to the value of spike and slab hyperparameter $v$. We also argued that the root of the problem is that the posterior point of intersection (PPI) is generally much more insensitive to $q(\pi)$ (which adapts the PPI during a-CAVI iterations) than to $v$. Recall that the PPI is the value of the transformed expected sufficient statistic of the inverse lengthscale $\hat{\theta} = \langle \theta^2 \rangle_{q_{\psi_j}(\theta)}$ that induces an even probability of inclusion for variable $j$ in the CAVI update for $q(\gamma_j)$:

$$\hat{\theta} = \sqrt{\frac{\log\left(\frac{1}{c}\right) + 2\langle \log \frac{1-\pi}{\pi} \rangle_{q(\pi)}}{v(1-c)}} \tag{64}$$

To demonstrate the relative sensitivity to $q(\pi)$ and $v$, below in Figure 1 we display how the PPI varies with $\pi_0$ and $v$ (under the restriction that $q(\pi) = \delta_{\pi_0}(\pi)$ for simplicity). As we can see, unless $\pi_0$ is extremely small or large, the PPI varies minimally over $\pi_0$ but varies substantially with $v$ (the PPI is $\mathcal{O}(v^{-\frac{1}{2}})$. Thus $v$ can entirely determine the sparsity of the learned PIPs $\boldsymbol{\lambda}$.

The other issue is that the range of PPI values that give good variable selection accuracy may vary based on the function properties, and so it is not the case that a single value of $v$ will perform well across all designs. For example, a function that varies much more slowly may require a lower PPI to prevent false exclusions of small but relevant inverse lengthscales.

To demonstrate this, we display on Figure 2 below the PIP profiles recovered on the 10 trials of the toy example using our SSVGP with a Mean-Field Gaussian posterior, for two fixed values of $v$ ($10^4$ and $10^5$), as well as for our final method (SSVGP+BMA with zero-temperature and dropout pruning). We also display on Fig. 3 an equivalent set of profiles recovered on a modified version of the generative process where now 25/100 variables are relevant $\{a_j\}_{j=1}^{25}$ are grid spaced over [0.05,0.1]. Whilst $v = 10^4$ performs well on the original design, but poorly on the second design, the opposite holds for $v = 10^5$. However the BMA procedure is able to effectively adapt the PPI so that the PIPs more accurately reflect each generative design.

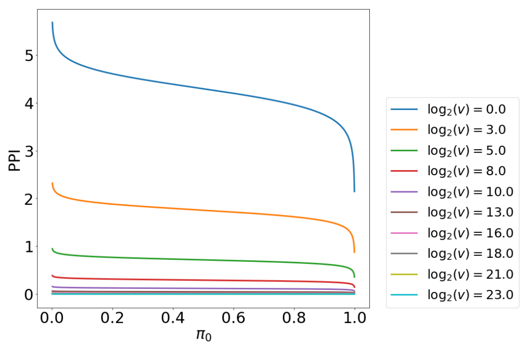

Figure 1: Posterior point of intersection (PPI) over 10 values of $v$ grid spaced over $10^{\{0,\ldots 7\}}$ and 1000 values of $\pi_0$ grid spaced over $[0.001, 0.999]$.

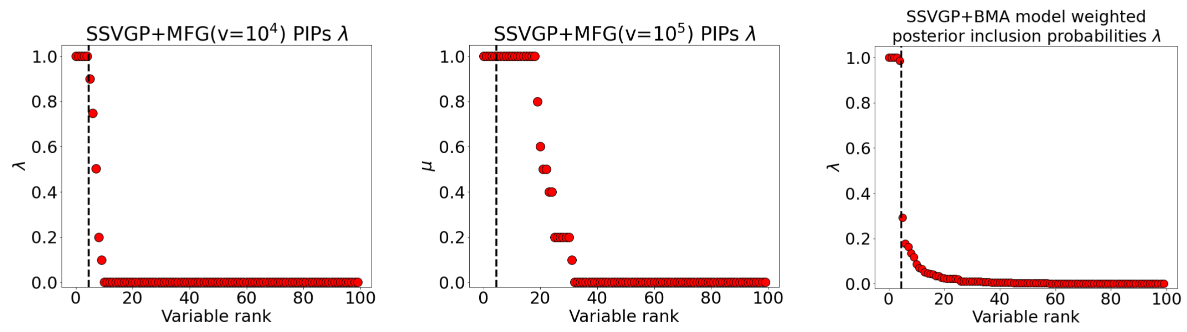

Figure 2: Toy example size ordered PIP profiles (averaged over 10 trials) using the original design (5 relevant variables) for SSVGP with (LHS) Mean-Field Gaussian and fixed $v = 10^4$, (Center) Mean-Field Gaussian and fixed $v = 10^5$, (RHS) BMA with zero-temperature posterior (our final method). Profiles are constructed by size ordering the PIPs per trial with $q = 5$ relevant variables constrained to take the first $q$ places in the ordering (LHS of black dashed line), and averaging the resulting profiles over the trials. $v = 10^4$ performs much better than $v = 10^5$, but doing BMA over 11 values of $v$ grid spaced over $10^4 \times 2^{\{-log_2(1000), log_2(1000)\}}$ performs best.

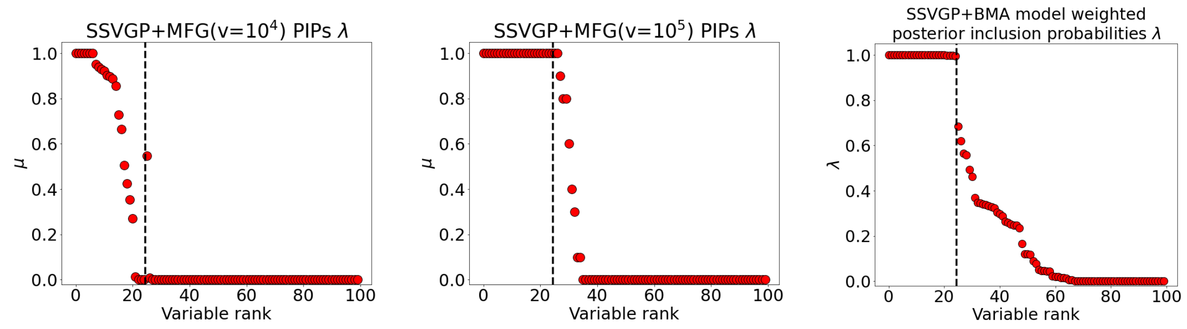

Figure 3: Toy example average size ordered PIP profile using the modified design (25 relevant variables) for SSVGP with (LHS) Mean-Field Gaussian and fixed $v = 10^4$, (Center) Mean-Field Gaussian and fixed $v = 10^5$, (RHS) BMA with zero-temperature posterior. By comparison with Fig. 2, $v = 10^5$ performs better than $v = 10^4$. However BMA performs best again, demonstrating its ability to effectively adapt to the generative process.

## B.2 Experiment 1

We next test our SSVGP on 50 trials of the additive function used in Savitsky et al's (2011) main high-dimensional experiment. They use this experiment to test their MCMC algorithm for the spike and slab GP. Since spike and slab priors with a Dirac spike and MCMC for posterior inference is considered a gold-standard in Bayesian variable selection (Narisetty and He, 2014), if our SSVGP can perform similarly well to their algorithm, we can be confident that our method can compete with best practice (but in a tiny fraction of the runtime). They were only able to complete a single trial of their method as they required $800,000$ MCMC iterations to ensure sufficient mixing of the chain (first $400,000$ discarded as burn-in). We note that they do not produce the runtime of their trial, but they do produce a runtime of $10224s$ when running the same algorithm for 500,000 iterations on an identically sized dataset ($n = 100, d = 1000$). Thus, we conclude with confidence that their runtime for the main high-dimensional experiment is likely to be $> 10000s$.

The experiment draws $n = 100$ training samples and $n_* = 20$ test samples from the following generative process:

$$y = x_1 + x_2 + x_3 + x_4 + sin(3x_5) + sin(5x_6) + \epsilon \tag{65}$$

$$\epsilon \sim \mathcal{N}(0, 0.05^2) \tag{66}$$

$$\boldsymbol{x} \sim \mathcal{U}[0,1]^{1000} \tag{67}$$

Thus, 6/1000 input variables are relevant. We implement our SSVGP (from now on with BMA, zero-temperature and pruning) and the ML-II GP using exactly the same implementation settings as in the toy example. We use three minibatch sizes of $m \in \{\frac{n}{4}, \frac{n}{2}, n\}$ to test how this affects performance. We also implement the following models with embedded variable selection: (LASSO (Tibshirani, 1996), MCP (Zhang, 2010), GAM with group LASSO penalty (Huang et al., 2009) and GAM with group spike and slab LASSO (SS LASSO) penalty (Bai et al., 2020). The LASSO and MCP were implemented using the R `ncvreg` package (**?**), the GAM LASSO and GAM spike and slab LASSO (SSwere implemented using the R package `sparseGAM` (Bai et al., 2021). In all cases default cross-validation procedures were used except for the GAM SS LASSO where 3 fold CV was used to keep runtimes manageable. All code was ran using the `rpy2` library in Python. We standardise inputs to have zero mean and unit variance following Savitsky et al. (2011).

Fig. 3 in the main text presents mean-squared error (MSE) and runtime performance percentiles for 50 trials of the experiment. We measure variable selection accuracy using the Matthews Correlation Coefficient (MCC $\in [-1, 1]$) - a best practice measure of imbalanced classification accuracy (Chicco and Jurman, 2020). We select using using a PIP threshold of $\bar{\lambda} = 0.5$ with the SSVGP. For the ML-II GP we report MCC performance using the *best* performing threshold from $0.1^{0,0.5,1,1.5,2}$, as indicative 'best-case' performance achievable with inverse lengthscale thresholding. All other methods automatically do binary variable selection. We report performance percentiles instead of mean and standard deviation as we found the distribution to be heavily skewed by a few outlier trials with poor performance. Detailed performance for different percentiles is contained in Table 1 below. For transparency, we additionally display the mean and standard deviation performance below in Table 2, but note the large standard deviations reflecting the outliers. We normalise the mean-squared error by the variance of the response as in Savitsky et al (2011).

| | 25th percentile | | | 50th percentile | | | 75th percentile | | |
|---|---|---|---|---|---|---|---|---|---|
| | MSE | MCC | Runtime | MSE | MCC | Runtime | MSE | MCC | Runtime |
| SSVGP(m=n) | 0.0046 | **1** | 19.6 | 0.0082 | **1** | 19.9 | 0.0133 | 0.93 | 20.5 |
| SSVGP(m=n/2) | **0.0043** | **1** | 10.4 | **0.0068** | **1** | 10.6 | **0.0111** | **1** | 11.1 |
| SSVGP(m=n/4) | 0.005 | **1** | 7.9 | 0.0086 | **1** | 8.2 | 0.0125 | **1** | 8.7 |
| ML-II GP | 0.0843 | 0.82 | 3.8 | 0.1321 | 0.77 | 3.9 | 0.1817 | 0.72 | 4 |
| GAM SS LASSO | 0.2749 | 1 | 386.7 | 0.3563 | 0.93 | 390.1 | 0.4831 | 0.82 | 393.6 |
| GAM LASSO | 0.0725 | 0.53 | 3.8 | 0.1373 | 0.43 | 3.9 | 0.1849 | 0.4 | 4.1 |
| LASSO | 0.3777 | 0.44 | **0.3** | 0.4796 | 0.35 | **0.3** | 0.6254 | 0.3 | **0.3** |
| MCP | 0.2731 | 0.64 | 0.3 | 0.3755 | 0.54 | 0.3 | 0.4815 | 0.49 | 0.4 |

Table 1: Experiment 1 performance percentiles for implemented models. Bold indicates best performing method per column. Note that the percentiles are reported such that the 75th percentile performance is worse than the 25th percentile performance, regardless of the metric. All SSVGP implementations outperformed the comparator set in variable selection and predictive accuracy for all percentiles reported.

| | MSE | MCC | Runtime |
|---|---|---|---|
| SSVGP(m=n) | 0.0616 ± 0.1267 | 0.85 ± 0.25 | 20.3 ± 1.2 |
| SSVGP(m=n/2) | **0.0187** ± 0.0422 | **0.94** ± 0.16 | 11.0 ± 0.9 |
| SSVGP(m=n/4) | 0.0258 ± 0.0543 | 0.93 ± 0.17 | 8.4 ± 0.6 |
| ML-II GP | 0.143 ± 0.0933 | 0.79 ± 0.13 | 3.9 ± 0.2 |
| GAM SS LASSO | 0.4174 ± 0.2154 | 0.87 ± 0.14 | 389.9 ± 4.8 |
| GAM LASSO | 0.164 ± 0.138 | 0.47 ± 0.1 | 4.0 ± 0.2 |
| LASSO | 0.5065 ± 0.1548 | 0.37 ± 0.1 | **0.3** ± 0.0 |
| MCP | 0.4103 ± 0.1885 | 0.58 ± 0.11 | 0.3 ± 0.0 |

Table 2: Experiment 1 mean ± standard deviation performance for implemented models. MSE is mean squared error and MCC is Matthews Correlation Coefficient. Bold indicates best performing method per column. All SSVGP implementations outperformed the comparator set on average variable selection and predictive accuracy, with the exception that the GAM spike and slab lasso of Bai et al. (2020) marginally outperfomed the SSVGP with $m = n$ on average with respect to variable selection accuracy (MCC).

| | MSE | MCC | Runtime (s) |
|---|---|---|---|
| Savitsky et al (2011) MCMC spike and slab GP (single trial) | 0.0067 | 1 | >10000 |

Table 3: Single trial performance of Savitsky et al's (2011) single trial of their MCMC spike and slab GP on the high-dimensional experiment. MSE is mean squared error and MCC is Matthews Correlation Coefficient.

## B.3 Experiment 2

Here we test the scalability and predictive accuracy of our SSVGP in a large-scale sparse input prediction challenge. The aim of this experiment is to demonstrate that regardless of whether variable selection is a central task, our method can substantially improve on the predictive performance of benchmark scalable GP approximations by taking advantage of input relevance sparsity when available. In this experiment we draw a univariate response from the following interactive function of two inputs:

$$y(\boldsymbol{x}) = f(\boldsymbol{x}) + \epsilon \tag{68}$$

$$f(\boldsymbol{x}) = \tan(x_1) + \tan(x_2) + \sin(2\pi x_1) + \sin(2\pi x_2) + \cos(4\pi^2 x_1 x_2) + \tan(x_1 x_2) \tag{69}$$

$$\epsilon \sim \mathcal{N}(0, \sigma^2) \tag{70}$$

$$(x_1, x_2) \sim \mathcal{U}[0,1]^2 \tag{71}$$

We then augment the dataset with $d-2$ noise dimensions $\boldsymbol{z}$, where $corr(x_i, z_j) = corr(z_j, z_k) = 0.5 \quad \forall i \in [2], \quad \forall j,k \in [d-2]$. To generate this correlation structure we pass $\boldsymbol{x}$ through an inverse standard Gaussian CDF, draw $\boldsymbol{z}|\boldsymbol{x}$ using standard Gaussian conditioning formulae, and then pass $\boldsymbol{x}, \boldsymbol{z}$ back through a standard Gaussian CDF. We fix $nd = 10^8$ and consider two variations in dimensions: (1) an ultra-high dimensional design ($n = 10^4, d = 10^4$) and (2) a one million training point design ($n = 10^6, d = 10^2$). For testing we draw an equivalent dataset of size $n_* = 10^4$ using the same generative process, but now with $x_1, x_2$ as grid spaced over $[0, 1]$. This enables a 2d prediction surface to be produced as a function of the two generating inputs and thus facilitates clear visual comparisons in results.

We implement the SSVGP using the same implementation settings as previously but use a smaller base learning rate of $\eta = 0.01$. We use the smaller learning rate in large-$n$ designs when using minibatch sizes such that $m \ll n$, as we find this leads to more stable learning due to the high degree of gradient variance. The specific minibatch sizes we choose are $m \in \{64, 128, 256\}$. We constrain the nearest neighbours in the minibatch to be chosen from a random subset of $min(n, 10^4)$ datapoints to limit the per-iteration nearest neighbour search cost. We nearest neighbour truncate the LOOPD using $\tilde{m} = 64$ neighbours, and the predictive distribution using $m_* = 256$ neighbours. We find $\tilde{m} \leq 64$ is required for competitive runtimes when $n = 10^6$ and $m_* \geq 256$ is required for (near) optimal predictive performance vs. using $m = n$. We consider $\tilde{m} = 64$ and $m_* = 256$ as default recommendations for our method. We use KD-trees (Bentley, 1975) whenever the active input dimensionality is $d < 100$, else we use Ball trees (Omohundro, 1989).

Since the experiment is focussed on predictive accuracy, we also do the following to speed up computation time: (i) We select the best model rather than averaging as this means only one set of GP predictive moments are computed, and (ii) We compress the model set such that if any $M$ models select the same variables, we keep only the model $i \in [M]$ with the best LOOPD on a random subset of $n = 1000$ training points. This leads to substantial time savings when $n = 10^6$, since as $n \to \infty$ LOOPD dominates computation time due to its superlinear complexity. These are default recommendations when predictive accuracy is the main concern.

The two competitors we use are the sparse Gaussian process (SGP) of Titsias (2009), and the stochastic variational Gaussian process (SVGP) of Hensman et al. (2013), which are popular benchmarks in the scalable GP literature. We follow the implementations in Wang et al. (2019); Chen et al. (2020) (512/1024 inducing points, 100 iterations/epochs and an ADAM learning rate of 0.1/0.01 for the SGP/SVGP respectively). However we deviate from this in the following cases : (1) In the ultra-high-dimensional design we train for 200 iterations/epochs as performance was extremely uncompetitive after 100 iterations/epochs. (2) In the one-million training point design we use 25 epochs for the SVGP as we found this did not change predictive accuracy noticeably but enabled $4\times$ savings in runtime (runtimes were not competitive for the SVGP when $n = 10^6$ using 100 epochs otherwise). We also use natural gradients for the SVGP following the GPytorch documentation recommendations. We use the same initialisation as our SSVGP for shared parameters, and standardise inputs and the response as previously.

Mean and standard deviation performance results from 3 trials of each design are in Fig. 5 in the main text. We also display in Fig. 4 of the main text the average prediction surfaces as a function of the two generating inputs for each method, against the true (noiseless) latent test surface $\boldsymbol{f}_*$. Full results tables and a breakdown of the runtime source for our SSVGP are below in Tables 4-5. Our SSVGPs all performed best on average and kept competitive runtimes in both designs (faster than both comparators when $n = 10^6$ and faster than the SVGP when $n = 10^4$). We note the relatively faster runtimes achieved by our method in the $n = 10^6$ case do not reflect better complexity with respect to $n$, but rather that more models in the BMA ensemble selected the same variables and thus the collapsed the model set was smaller for LOOPD computation.

In the ultra-high-dimensional design ($n = d = 10^4$), runtime is dominated by a-CAVI training time, and the larger minibatch size therefore substantially increases total time. However in the one-million training point design ($n = 10^6, d = 10^2$), the runtime is dominated by LOOPD computation due to its superlinear complexity. In this case, we actually find $m = 256$ runs fastest on average, as it tended to prune (select) fewer variables per BMA ensemble member (recall LOOPD compute time is linear in the number of non-pruned variables $p_k$ per model $\mathcal{M}_k$ in the BMA ensemble).

| | $(n = 10^4, d = 10^4)$ | | $(n = 10^6, d = 10^2)$ | |
|---|---|---|---|---|
| | MSE | Runtime (mins) | MSE | Runtime (mins) |
| SGP(512) | 0.682 ± 0.015 | **14.1** ± 0.3 | 0.263 ± 0.002 | 288.7 ± 17.2 |
| SVGP(1024) | 0.644 ± 0.005 | 154.8 ± 3.3 | 0.256 ± 0.002 | 365.6 ± 14.9 |
| SSVGP(m=256) | **0.26** ± 0.002 | 42.6 ± 5.9 | **0.251** ± 0.002 | **227.3** ± 23.2 |
| SSVGP(m=128) | 0.261 ± 0.003 | 24.3 ± 4.3 | 0.251 ± 0.002 | 236.6 ± 15.6 |
| SSVGP(m=64) | 0.262 ± 0.004 | 17.9 ± 2.9 | 0.251 ± 0.002 | 245.8 ± 17.0 |

Table 4: Experiment 2 mean ± standard deviation results (MSE = mean squared error) from 3 trials on the two dataset sizes tested. Bold indicates the best performing method.

| | $(n = 10^4, d = 10^4)$ | | | $(n = 10^6, d = 10^2)$ | | |
|---|---|---|---|---|---|---|
| | Train | LOOPD | Test | Train | LOOPD | Test |
| SSVGP(m=256) | 35.4 ± 2.4 | 6.9 ± 3.5 | 0.2 ± 0.0 | 11.9 ± 0.2 | 215.0 ± 23.2 | 0.4 ± 0.0 |
| SSVGP(m=128) | 17.3 ± 0.7 | 6.7 ± 3.6 | 0.2 ± 0.0 | 12.3 ± 0.2 | 223.9 ± 15.4 | 0.3 ± 0.0 |
| SSVGP(m=64) | 12.9 ± 0.7 | 4.8 ± 2.3 | 0.2 ± 0.0 | 12.1 ± 0.3 | 233.3 ± 16.7 | 0.3 ± 0.0 |

Table 5: Experiment 2 mean ± standard deviation runtime (minutes) breakdown into training time (a-CAVI), LOOPD computation, and test time from 3 trials on the two dataset sizes tested.

### B.4 Real Datasets from UCI Repository

We now test on two benchmark real datasets from the UCI repository with high-dimensional inputs ($d \gg 100$): CTSLICE (n=53500, d=380) and UJINDOORLOC (n=21048, d=520). As variable selection accuracy cannot be verified here we focus again on the predictive accuracy and runtime of our method. We implement the SSVGP against the SGP and SVGP using exactly the implementation settings as in Experiment 2 but train the SGP and SVGP using the default 100 iterations/epochs and set the SSVGP's training minibatch to $m = 256$ (the best performing in Experiment 2). We run these methods on 10 trials of random splits into training and test sets, with 4/5 used for training. As previously we standardise the data to have mean zero and unit variance. We also pass $\boldsymbol{y}$ through a Box-cox transformation due to the heavy skew in the distributions, to maximise consistency with the Gaussian modelling assumptions. Fig. 6 in the main text contains the mean and standard deviation of MSE and runtimes obtained, and we display results tables below in 6. We also highlight in Fig. 4 below the 'sparsity' of the learned solutions by our SSVGP by comparison to the SGP/SVGP by comparing the recovered inverse lengthscales (post. mean used for SSVGP).

| | CTSLICE (n=53500,d=385) | | | UJINDOORLOC (n=21048,d=520) | | |
|---|---|---|---|---|---|---|
| | MSE | Runtime | Variables used | MSE | Runtime | Variables used |
| SSVGP(m=256) | **0.003** ± 0.001 | 34 ± 2 | **142** ± 25 | **0.027** ± 0.003 | 18 ± 0 | **170** ± 32 |
| SGP(512) | 0.006 ± 0.000 | **3** ± 0 | 385 | 0.038 ± 0.003 | **1** ± 0 | 520 |
| SVGP(1024) | 0.005 ± 0.000 | 60 ± 1 | 385 | 0.035 ± 0.003 | 25 ± 0 | 520 |

Table 6: Real dataset results for our for our SSVGP with $m = 256$ minibatching, against SGP (Titsias, 2009) and SVGP (Hensman et al., 2013) on the real datasets. 'Variables used' is measured by the number of dimensions $j = (1....J)$ which affect predictions when $x_{*(j)}$ varies. Bold indicates the best performing method per column.

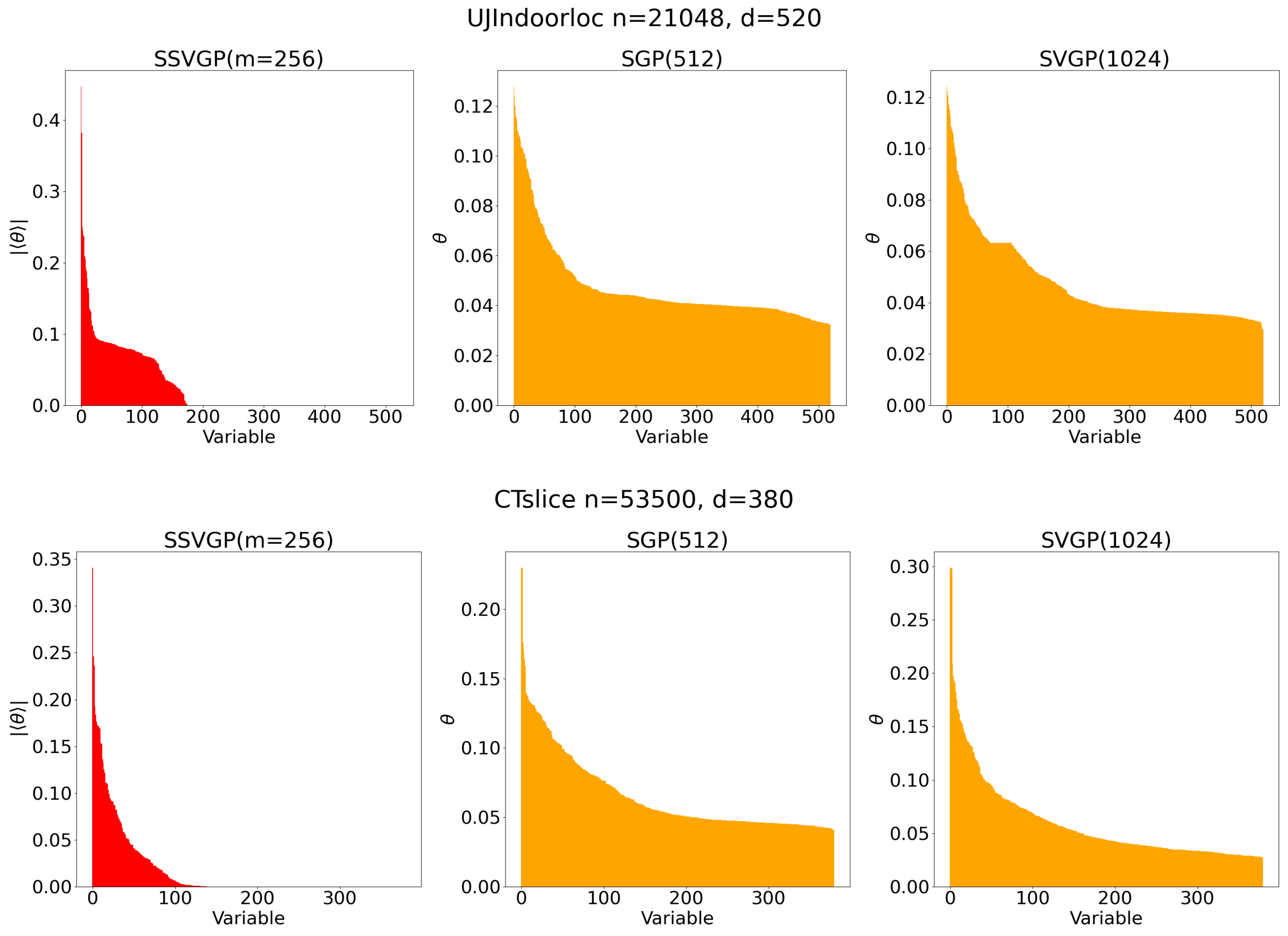

Figure 4: Size ordered inverse lengthscale profiles for our SSVGP with $m = 256$ minibatching (posterior mean of $q_\psi(\boldsymbol{\theta})$ used), against sparse GP (Titsias, 2009) and stochastic variational GP (Hensman et al., 2013) on the real datasets. Since our SSVGP is parameterised such that $\boldsymbol{\theta} \in \mathbb{R}^d$, we plot $|\langle\boldsymbol{\theta}\rangle_{q_\psi(\boldsymbol{\theta})}|$. The median trial is used to construct the profiles, meaning that the $k^{th}$ column value is the median value of the $k^{th}$ largest inverse lengthscale recovered. We use the median to indicate how the profile looks for a 'typical' trial, as the average is skewed by two trials with a larger number of selections. Our SSVGP learns an L0-sparse subset of inverse lengthscales through the BMA weighting over different models with varying aggressiveness in dropout pruning, but predicts with significantly more accuracy on average (see Table 6).