Supplementary material: Probabilistic selection of inducing points in sparse Gaussian processes

A DERIVATION OF DEEP GAUSSIAN PROCESSES

For completeness we provide a derivation of the Deep Gaussian process models used in section 5. We follow Damianou and Lawrence [2013] and Salimbeni and Deisenroth [2017] and consider the layer-wise composition of functions where each function is endowed with a Gaussian process prior. By putting a prior and associated variational distribution on the inputs, the GP-LVM (Lawrence [2004], Titsias and Lawrence [2010]) that we consider in Section 5.3 emerges as a special case.

Considering a process with L layers (that is, function compositions) with the joint likelihood:

$$p(\mathbf{Y}, \{\mathbf{F}^\ell\}_{\ell=1}^L \mid \mathbf{X}) = p(\mathbf{Y} \mid \mathbf{F}^L)p(\mathbf{F}^L \mid \mathbf{F}^{\ell-1}) \dots p(\mathbf{F}^2 \mid \mathbf{F}^1)p(\mathbf{F}^1 \mid \mathbf{X}). \tag{1}$$

We capitalise all variable names to let it reflect that the function and observed outputs may have multiple dimensions. We assume w.l.o.g. that all functions have output dimensionality D. For notational convenience we define $\mathbf{F}_0 \triangleq \mathbf{X}$.

In our implementation, we let the function evaluations factorise across output dimensions, and so the conditional probabilities are given by:

$$p(\mathbf{F}^\ell \mid \mathbf{F}^{\ell-1}) = \prod_{1=d}^{D} \mathcal{N}\left(\mathbf{f}^{\ell,d} \mid \mu^{\ell,d}, \Sigma^{\ell,d}\right), \qquad 0 \leq \ell \leq L,$$
$$\mu_i^{\ell,d} = m^\ell(\mathbf{F}_i^{\ell-1}) \qquad \Sigma_{ij}^{\ell,d} = \kappa^\ell\left(\mathbf{F}_i^{\ell-1}, \mathbf{F}_j^{\ell-1}\right),$$

where $\mathbf{f}^{\ell,d}$ are all function evaluations of output dimension d, and $\mathbf{F}_i^\ell$ is the i'th evaluation across dimensions. Note that we could increase flexibility by using different kernel and mean functions for each dimension in a given layer, but for our experiments we have not found that necessary.

The posterior over latent variables in (1) is $p(\{\mathbf{F}\}_{\ell=1}^L \mid \mathbf{X}, \mathbf{Y})$ which we approximate with a variational posterior. As for the standard sparse GP, we first augment all vectors of function evaluations with inducing points, $\{\mathbf{U}^\ell, \mathbf{Z}^\ell\}_{\ell=1}^L$, which are to be marginalised out along with $\{\mathbf{F}^\ell\}_{\ell=1}^L$. The variational posterior, $\mathcal{Q}$, then takes the form:

$$\begin{aligned}
\mathcal{Q} &= q(\{\mathbf{F}^\ell, \mathbf{U}^\ell\}_{\ell=1}^L) \\
&= \prod_{\ell=1}^{L} p(\mathbf{F}^\ell \mid \mathbf{U}^\ell, \mathbf{F}^{\ell-1}, \mathbf{Z}^\ell) q(\mathbf{U}^\ell) \\
&= \prod_{\ell=1}^{L} \prod_{d=1}^{D} p(\mathbf{f}^{\ell,d} \mid \mathbf{u}^{\ell,d}, \mathbf{F}^{\ell-1}, \mathbf{Z}^\ell) q(\mathbf{u}^{\ell,d}).
\end{aligned}$$

Note that all dimensions for a given layer are informed by the same set of inducing inputs but different sets of inducing outputs. Inserting this into the standard ELBO derivation, we obtain:

$$\begin{aligned}
\log p(\mathbf{Y} \mid \mathbf{X}) &\geq \int \mathcal{Q} \log \frac{p(\mathbf{Y}, \{\mathbf{F}^\ell, \mathbf{U}^\ell\}_{\ell=1}^L \mid \mathbf{X}, \mathbf{Z})}{\mathcal{Q}} \, \mathrm{d}\{\mathbf{F}^\ell, \mathbf{U}^\ell\}_{\ell=1}^L \\
&= \int \mathcal{Q} \log \frac{p(\mathbf{Y} \mid \mathbf{F}^L)\{\cancel{p(\mathbf{F}^\ell \mid \mathbf{U}^\ell, \mathbf{F}^{\ell-1}, \mathbf{Z}^\ell)} p(\mathbf{U}^\ell \mid \mathbf{Z}^\ell)\}_{\ell=1}^L}{\{\cancel{p(\mathbf{F}^\ell \mid \mathbf{U}^\ell, \mathbf{F}^{\ell-1}, \mathbf{Z}^\ell)} q(\mathbf{U}^\ell)\}_{\ell=1}^L} \, \mathrm{d}\{\mathbf{F}^\ell, \mathbf{U}^\ell\}_{\ell=1}^L \\
&= \mathbb{E}_{\mathcal{Q}}\left[\log p(\mathbf{Y} \mid \mathbf{F}^L)\right] - \sum_{\ell=1}^{L} \sum_{d=1}^{D} \mathrm{KL}[q(\mathbf{u}^{\ell,d}) \parallel p(\mathbf{u}^{\ell,d} \mid \mathbf{Z}^\ell)].
\end{aligned} \tag{2}$$

Defining $q(\mathbf{u}^{\ell,d}) = \mathcal{N}\left(\mathbf{u}^{\ell,d} \mid \mathbf{m}^{\ell,d}, \mathbf{S}^{\ell,d}\right)$ with $(\mathbf{m}^{\ell,d}, \mathbf{S}^{\ell,d})$ being variational parameters, the KL divergence terms of (2) are all computable in closed form. Within each layer and for a specific dimension, the inducing outputs can be marginalised

out analytically yielding a distribution that factorises over datapoints:

$$
\begin{aligned}
q(\mathbf{f}^{\ell,d} \mid \mathbf{F}^{\ell-1}, \mathbf{Z}^{\ell}) &= \int p(\mathbf{f}^{\ell,d} \mid \mathbf{u}^{\ell,d}, \mathbf{F}^{\ell-1}, \mathbf{Z}^{\ell}) q(\mathbf{u}^{\ell,d} \mid \mathbf{Z}^{\ell}) \, \mathrm{d}\mathbf{u}^{\ell,d} \\
&= \prod_{i=1}^{N} \mathcal{N}\left(f_i^{\ell,d} \mid \mu_i^{\ell,d}, \left(\sigma_i^{\ell,d}\right)^2\right), \\
\mu_i^{\ell,d} &= m^{\ell}(\mathbf{F}_i^{\ell-1}) - \left(\alpha_i^{\ell}\right)^T \left(\mathbf{m}^{\ell,d} - m^{\ell}(\mathbf{Z}^{\ell})\right) \\
\sigma_i^{\ell,d} &= \kappa^{\ell}(\mathbf{F}_i^{\ell-1}, \mathbf{F}_i^{\ell-1}) - \left(\alpha_i^{\ell}\right)^T \left(\kappa^{\ell}(\mathbf{Z}^{\ell}, \mathbf{Z}^{\ell}) - \mathbf{S}^{\ell}\right) \alpha_i^{\ell} \\
\alpha_i^{\ell} &= \kappa^{\ell}(\mathbf{Z}^{\ell}, \mathbf{Z}^{\ell})^{-1} \kappa(\mathbf{Z}^{\ell}, \mathbf{F}_i^{\ell-1})
\end{aligned}
$$

Assuming that the likelihood is conditionally independent across both data points and dimensions, the entire expectation in (2) decomposes into one-dimensional expectations, each of which relies only on one input observation as well as the variational parameters:

$$
\begin{aligned}
\mathbb{E}_{\mathcal{Q}}\left[\log p(\mathbf{Y} \mid \mathbf{F}^{L})\right] &= \sum_{i=1}^{N} \sum_{d=1}^{D} \mathbb{E}_{q_i^d}\left[\log p(y_i^d \mid f_i^{L,d})\right], \\
q_i^d &= \prod_{\ell=1}^{L} \mathcal{N}\left(f_i^{\ell,d} \mid \mu_i^{\ell,d}, \left(\sigma_i^{\ell,d}\right)^2\right). \qquad (3)
\end{aligned}
$$

Since each distribution in (3) can be re-parameterised according to Kingma et al. [2015], the expectation in (2) is easily optimised through Monte Carlo sampling as suggested in Salimbeni and Deisenroth [2017].

B INFERENCE USING THE RE-PARAMETERISATION TRICK

In section 3 we present a general inference scheme based on score function estimation that make very few assumptions about the variational point process. However, when using a Possion point process it is possible to derive an inference scheme based on the re-parametrisaiton trick which we include here for completeness.

We build on Maddison et al. [2017] who provide a method for stochastically estimating (ϕ, θ) of the expression $\mathbb{E}_{D\sim p_\phi(d)}[g_\theta(D)]$ where $D \in \{0,1\}^L$ is a one-hot encoding of a value drawn from a multinomial distribution with probabilities $\phi = (\phi_1, \phi_2, \ldots, \phi_L)$:

$$p[D_\ell = 1] = \phi_i, \qquad \sum_{\ell=1}^{L} D_\ell = 1.$$

The approach is conceptually the same as in [Kingma et al, 2015] in that the instantiations of $D \sim p_\theta(d)$ is re-written as $D = h_\phi(T), T \sim \pi(t)$ where h_ϕ is a discrete (and so non-differentiable), deterministic function and $\pi(t)$ is an (unparameterised) distribution over the domain of h_ϕ. This allows us to move ϕ inside the expectation:

$$\mathbb{E}_{D\sim p_\phi(d)}[g_\theta(D)] = \mathbb{E}_{T\sim\pi(t)}[g_\theta(h_\phi(T))].$$

By further making a continuous relaxation of h_ϕ, the expectation can be approximated and maximised through Monte Carlo sampling. This relaxation is governed by a temperature parameter, that controls how closely the continuous approximations resembles a true one-hot vector. Lower temperature means more accurate approximations but also reduces the amount of gradient information that can be communicated between states of d.

To apply this method for our point process estimation, we first introduce a binary vector, $\mathbf{b} \in [0,1]^K$, that indicates which inducing points are present in a given sample $\mathbf{Z} \subseteq \mathbf{Z}^\star$, i.e. $b_k = 1 \Leftrightarrow \mathbf{z}_k \in \mathbf{Z}$. Next, we construct a new expression for the bound, $\hat{\mathcal{L}}(\mathbf{Z}^\star; \mathbf{b})$, that takes the entire candidate set as input and masks out those inducing points for which $b_k = 0$, such that $\hat{\mathcal{L}}(\mathbf{Z}^\star; \mathbf{b}) = \mathcal{L}(\mathbf{Z})$. This construction is presented in B.1. If we now have a distribution over binary vectors, $q_\lambda(\mathbf{b})$, that is equivalent to $q_\lambda(\mathbf{Z})$ in the sense that the same subsets are given same probabilities, then:

$$\mathbb{E}_{q_\lambda(\mathbf{Z})}\left[\mathcal{L}(\mathbf{Z})\right] = \mathbb{E}_{q_\lambda(\mathbf{b})}\left[\hat{\mathcal{L}}(\mathbf{Z}^\star; \mathbf{b})\right]. \tag{4}$$

When $q_\lambda(\mathbf{Z})$ is a discrete Poisson point process, the equivalent distribution over $\mathbf{b}$ is

$$q_\lambda(\mathbf{b}) = \prod_{k=1}^{K} \lambda_k^{b_k}(1-\lambda_k)^{1-b_k}.$$

Substituting this into (4), we have

$$\mathbb{E}_{q_\lambda(\mathbf{b})}[\hat{\mathcal{L}}(\mathbf{Z}^\star; \mathbf{b})] = \mathbb{E}_{q(b_1)}[\mathbb{E}_{q(b_2)}[\cdots[\mathbb{E}_{q(b_K)}[\hat{\mathcal{L}}(\mathbf{Z}^\star; \mathbf{b})]]\cdots]],$$

where $q(b_k) = \text{Bernoulli}(\lambda_k)$. Now, each of $q(b_k)$ can be re-parameterised, yielding an expression that is compatible with the formulation from Maddison et al. [2017].

B.1 MASKING THE BOUND

The masked bound, $\hat{\mathcal{L}}(\mathbf{Z}^\star; \mathbf{b})$, is obtained by evaluating the usual bound from (1) over the entire candidate set, $\mathbf{Z}^\star$, but under a modified GP with the following mean and kernel function:

$$\begin{aligned}
\hat{m}_{\mathbf{b}}(\mathbf{x}) &= m(\mathbf{x}) \cdot \Delta(\mathbf{x}, \mathbf{b}) \\
\hat{\kappa}_{\mathbf{b}}(\mathbf{x}, \mathbf{x}') &= \kappa(\mathbf{x}, \mathbf{x}') \cdot \Delta(\mathbf{x}, \mathbf{b}) \cdot \Delta(\mathbf{x}', \mathbf{b}) && \text{if } \mathbf{x} \neq \mathbf{x}' \\
\hat{\kappa}_{\mathbf{b}}(\mathbf{x}, \mathbf{x}) &= \Delta(\mathbf{x}, \mathbf{b})(\kappa(\mathbf{x}, \mathbf{x}) - 1) + 1 \\
\Delta(\mathbf{x}, \mathbf{b}) &= \prod_{j=1}^{M} b_j^{\delta(\mathbf{x}-\mathbf{z}_j)}.
\end{aligned}$$

Despite the somewhat convoluted expressions, the above procedures can be carried out efficiently as vector manipulations of the mean vector and covariance matrix.

The updated mean and kernel function has the effect of factorising the complement evaluations, $\mathbf{u}_C = f(\mathbf{Z}^\star) \setminus \mathbf{u}$, into a standard normal:

$$p(\mathbf{y}, \mathbf{f}, \mathbf{u}_C \mid \mathbf{X}, \mathbf{Z}^\star) = p(\mathbf{y}, \mathbf{f} \mid \mathbf{X}, \mathbf{Z}) p(\mathbf{u}_C), \qquad p(\mathbf{u}_C) = \mathcal{N}\left(\mathbf{u}_C \mid \mathbf{0}, \mathbf{I}\right).$$

Note, that this also implies that any Gram matrix produced by $\hat{\kappa}_{\mathbf{b}}$ is positive semi-definite and so $\mathcal{GP}(\hat{m}_{\mathbf{b}}, \hat{\kappa}_{\mathbf{b}})$ remains a valid Gaussian process. Assuming that a similar factorisation holds for our variational distribution s.t. $q(\mathbf{f}, \mathbf{u}_C) = q(\mathbf{f}) p(\mathbf{u}_C)$ (which is the case under e.g. the collapsed bound), we have

$$\begin{aligned}
\mathcal{L}(\mathbf{Z}^\star; \mathbf{b}) &= \int q(\mathbf{f}, \mathbf{u}_C) \log \frac{p(\mathbf{y}, \mathbf{f}, \mathbf{u}_C \mid \mathbf{X}, \mathbf{Z}^\star)}{q(\mathbf{f}, \mathbf{u}_C)} \, \mathrm{d}[\mathbf{f}, \mathbf{u}_C] \\
&= \int q(\mathbf{f}) p(\mathbf{u}_C) \log \frac{p(\mathbf{y}, \mathbf{f} \mid \mathbf{X}, \mathbf{Z}) p(\mathbf{u}_C)}{q(\mathbf{f}) p(\mathbf{u}_C)} \, \mathrm{d}[\mathbf{f}, \mathbf{u}_C] \\
&= \mathcal{L}(\mathbf{Z}).
\end{aligned}$$

As such, we have an equivalent expression for $\mathcal{L}(\mathbf{Z})$ that lends itself to the reparameterisation trick when $q_\lambda(\mathbf{b})$ is a Poisson point process.

B.2 COMPARISON TO SCORE FUNCTION ESTIMATION

Empirical evidence suggests that our method, when relying on this technique for maximising (2), produce comparable results as the score function estimator when applied to non-deep models. However, in deep models there is a considerable propagation of noise through the layers which is induced by the continuous relaxation of the Bernoulli distributions in (4). This makes it difficult to get stable estimates of λ, especially in the "deeper" layers, prompting us to focus primarily on the score function estimator in this work.

In terms of efficiency, the re-parameterisation approach does not require multiple samples per optimisation step as is the case for SFE. However, SFE only needs to evaluate $\mathcal{L}(\mathbf{Z})$ for those inducing points, that have been sampled from the from the candidate set, while the re-parameterisation approach always include all points. When a high level of sparsity is induced by the prior, this makes SFE notably more efficient.

C EXPERIMENTAL SETUPS

In the following, we provide the implementation details and experimental setups necessary for recreating our results.

C.1 IMPLEMENTATION

Our implementation relies on the GPyTorch framework (Gardner et al. [2018]) which enables modular construction of exact and sparse Gaussian process models with GPU acceleration. We ran the experiments on a RTX 2080 TI GPU system. All experiments pertaining to computation speed were run on the dedicated computing server.

All of our Gaussian process priors used a zero mean function and radial basis function (RBF) kernel with Automatic Relevance Detection (ARD). We used the Adam optimiser (Kingma and Ba [2014]) with fixed learning rate of 0.01 for all the free parameters except those related to the variational point process, which had a fixed learning rate 0.2. In each iteration of (3) we drew 4 samples from $q_\lambda(\mathbf{Z})$.

All hyper-parameters (lengthscale and variance of the RBF kernels and, for regression, Gaussian noise variance) were initialised to 1. The inducing inputs were initialised to a random subset of observed inputs. We note, however, that we compared against other common initialisation methods (e.g. Latin hyper-cube sampling and K-means clustering) without noting any difference in performance.

All input and output data was standardised before fitting the model. However, any predictive likelihood reported was evaluated in the original rather than the scaled space.

C.2 TRAINING PHASES

In the informativeness and DGP experiments we found it useful to divide the inference into three phases where we would first pre-train without the Poisson point process (PPP), then include the PPP in the training, sample a subset of inducing points from the PPP, and finally post-train with only those points and without the PPP. In the following we will refer to the number of epochs used for each phase as respectively n_{pre}, n_{PPP}, and n_{post}.

C.3 INFORMATIVENESS EXPERIMENT

C.3.1 Synthetic data

In the experiment for synthetic data of Section 5.1 we considered three generative characteristic: 1. observation noise, 2. kernel smoothness, and 3. input clustering, each of which were evaluated with different "intensity" levels. For each combination of condition and intensity, we sampled 500 observations from the following, generative model:

$$\begin{aligned}
\mathbf{x} &\sim p(\mathbf{x}), \\
\mathbf{f} &\sim \mathcal{GP}(\mathbf{0}, \kappa(\mathbf{x}, \mathbf{x}'; \gamma)), \\
\mathbf{y} &\sim \mathcal{N}\left(\mathbf{f}, \mathbf{I}\sigma^2\right).
\end{aligned}$$

Here, κ is an RBF kernel with variance 1.0 and lengthscale γ. To generate data for condition 1 and 2. we set each $p(\mathbf{x})$ to a uniform distribution over $[0, 100]$. The intensities were then given by γ for condition 1 and σ for condition 2. To generate data for condition 3, we set $p(\mathbf{x})$ to a homogeneous mixture of 5 normal distributions, $\{\mathcal{N}\left(\mathbf{u}_j, \beta^{-1}\right)\}_{j=1}^5$, with equidistant means distributed over the input domain, $\mathcal{X} = [0, 100]$. The intensity was then given by the shared precision, β; i.e. higher values yields more clustering, and as $\beta \to 0$ the mixture converges weakly to a uniform distribution. The default values for the intensity parameters were $\sigma = 0.1, \gamma = 1.0$.

All baselines models of the experiment were trained for 1000 epochs. For the adaptive method we fixed the prior parameter α to 0.05 and used $n_{\text{pre}} = 200$, $n_{\text{PPP}} = 600$, $n_{\text{post}} = 200$.

C.3.2 Real-world data

For the real-world data we corrupted the original outputs, $\mathbf{y}$, with additive, standard Gaussian noise scaled by a constant factor:

$$\hat{\mathbf{y}} = \mathbf{y} + \epsilon \cdot \hat{\sigma}(\mathbf{y}) \cdot v, \qquad \epsilon \sim \mathcal{N}(0, 1),$$

where $\hat{\sigma}(\mathbf{y})$ is the empirical standard deviation of the observed outputs. The constant $v \in [0, 1)$ determines the level of corruption and is the value reported in Figure 4. The baselines models were trained for 5000 epochs. For the adaptive method we used $n_{\text{pre}} = 2500$, $n_{\text{PPP}} = 1500$, $n_{\text{post}} = 1000$. The prior parameters α needed to be configured for each dataset as more observations will tend to diminish the influence of the prior. The configurations are listed in Table 1.

Dataset	N	D	Noise levels	α
UCI Concrete	1030	8	[0.0, 0.2, 0.3, 0.4]	0.01
UCI Energy	768	8	[0.0, 0.05, 0.1, 0.15]	0.05
UCI Kin8nm	8192	8	[0.0, 0.2, 0.3, 0.4]	0.1
UCI Protein	45730	9	[0.0, 0.3, 0.5, 0.7]	0.2

Table 1: Specifications for the informativeness experiment carried out on real-world benchmark datasets.

C.4 DEEP GAUSSIAN PROCESS

For all DGP experiment in Section 5.2 we first trained the layers individually for 200 epochs each. In the gridsearch we then fitted both layers for another 3000 epochs. For the adaptive method we used $n_{\text{pre}} = 1000$, $n_{\text{PPP}} = 500$, $n_{\text{post}} = 1500$.

C.5 GPLVM

For the GPLVM we did *not* divide into different training phases as in the other experiments. Rather, we included the PPP in the entire training and learned probability of inclusion along with the SVGP function and latent representation. The prior parameter, α, was fixed to 3. We used the qPCR dataset from `https://github.com/sods/ods` which holds 437 observations, each with 48 outputs (cell stages).

REFERENCES

A. Damianou and N. Lawrence. Deep Gaussian Processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 207–215, 2013.

J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson. GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. In *Advances in Neural Information Processing Systems*, 2018.

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

D. P. Kingma, T. Salimans, and M. Welling. Variational dropout and the local reparameterization trick. *arXiv preprint arXiv:1506.02557*, 2015.

N. D. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In *Advances in Neural Information Processing Systems*, pages 329–336, 2004.

C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017.

H. Salimbeni and M. P. Deisenroth. Doubly stochastic variational inference for deep Gaussian processes. In *Advances in Neural Information Processing Systems*, pages 4589–4600, 2017.

M. Titsias and N. D. Lawrence. Bayesian Gaussian process latent variable model. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 844–851, 2010.