Leveraging Locality and Robustness to Achieve Massively Scalable Gaussian Process Regression

Anonymous Author(s)
Affiliation
Address
email

A Theoretical GPnn Results

A.1 Preliminary results

Let $\rho(\boldsymbol{x}, \boldsymbol{x'}) = \sigma_f \sqrt{1 - c(\boldsymbol{x}/l, \boldsymbol{x'}/l)}$ be the kernel-induced distance function over $\mathbb{R}^d$ ([17]). We define $\mathbf{x}_{(j,n)}(\boldsymbol{x}^*)$ as the j^{th} nearest-neighbour random variable to a test point $\boldsymbol{x}^*$ under ρ, which we abbreviate to $\mathbf{x}_{(j)}$ when the context is clear, and $\boldsymbol{x}_{(j)}(\boldsymbol{x}^*) \in N_m(\boldsymbol{x}^*)$ as the realised j^{th} nearest-neighbour of the test point $\boldsymbol{x}^*$ from a training set X. From this we define $\epsilon_i = \rho^2(\mathbf{x}_{(i)}, \boldsymbol{x}^*)$ and $\epsilon_{ij} = \rho^2(\mathbf{x}_{(i)}, \mathbf{x}_{(j)})$.

Definition 8 (Support)**.** *Let $P_{\mathbf{x}}$ be the probability measure of $\mathbf{x}$ and $S^{\rho}_{\boldsymbol{x},\epsilon}$ the closed ball of radius $\epsilon > 0$ under the metric ρ centred at $\boldsymbol{x}$. Then we define* $\text{support}(P_{\mathbf{x}}) = \{\boldsymbol{x} : P_{\mathbf{x}}(S^{\rho}_{\boldsymbol{x},\epsilon}) > 0 \,\forall\, \epsilon > 0\}$.

Definition 9 (Weakly-faithful)**.** *We define a pair of metrics $\rho(\cdot,\cdot), \hat{\rho}(\cdot,\cdot)$ to be* weakly-faithful *w.r.t. each other if the following condition holds: The m^{th} nearest-neighbour under $\hat{\rho}$ converges to the test point as $n \to \infty$ if and only if the m^{th} nearest-neighbour under ρ converges to the test point in the limit.*

Assumptions

- **(A1)** $\mathbf{x} \overset{\text{iid}}{\sim} P_{\mathbf{x}}$ and $\mathbf{x}^* \in \text{support}(P_{\mathbf{x}})$ under the generative metric defined by $c(\cdot,\cdot)$.
- **(A2)** $c(\cdot,\cdot), \hat{c}(\cdot,\cdot)$ are stationary kernels whose induced distance functions are *weakly faithful* metrics (Definition 9).
- **(A3)** $y_i = f(\boldsymbol{x}_i) + \xi_i$ with $\xi_i \overset{\text{iid}}{\sim} P_{\xi}$, $f(\boldsymbol{x}) \sim \mathcal{WSRF}(\sigma_f^2 c(./l, ./l))$ and $y_i \,|\, f(\boldsymbol{x}_i) \sim P_{\xi}$ and $\mathrm{E}[\xi] = 0$, $\mathrm{E}[\xi^2] = \sigma_{\xi}^2$.

Note: Assumption (A2) is not overly restrictive and encompasses commonly used kernels such as all those mentioned in this paper.

Lemma 10. *$\epsilon_i \to 0$ and $\epsilon_{ij} \to 0$ as $n \to \infty$ a.e. with respect to the measure over $\mathbf{x} \in \mathbb{R}^d$, $P_{\mathbf{x}}$, for $i, j \leq m$, $\frac{m}{n} \to 0$ and under (A1-2).*

Proof. Lemma 6.1 of [9] states that $\left\|\mathbf{x}_{(m,n)}(\boldsymbol{x}) - \boldsymbol{x}\right\| \xrightarrow{n\to\infty} 0$ with probability one (with respect to $P_{\mathbf{x}}$). Their proof can be generalised immediately to state that $\rho(\mathbf{x}_{(m,n)}(\boldsymbol{x}), \boldsymbol{x}) \xrightarrow{n\to\infty} 0$ by using our definition of support, 8, that directly invokes the metric ρ. Hence $\epsilon_i \to 0$ for all $i \leq m$ (since $\boldsymbol{x}^*$ is in $\text{support}(P_{\mathbf{x}})$). Since ρ is a metric it satisfies the triangle inequality; hence $\rho(\mathbf{x}_{(i)}, \mathbf{x}_{(j)}) \leq \rho(\mathbf{x}_{(i)}, \boldsymbol{x}^*) + \rho(\mathbf{x}_{(j)}, \boldsymbol{x}^*) \xrightarrow{n\to\infty} 0$ for all $i, j \leq m$. $\square$

Lemma 11. *For an m-GPnn under the assumptions (A1-3),*

$$\lim_{n\to\infty} \boldsymbol{k}_N^{*T} K_N^{-1} \boldsymbol{k}_N^* = \sigma_f^2 - \sigma_{\xi}^2 m^{-1} + \mathcal{O}\big(m^{-2}\big)\,.$$

Proof. From Lemma 10 we have that $\lim_{n\to\infty} k(\mathbf{x}_{(j)}(\boldsymbol{x}^*), \boldsymbol{x}^*) = \lim_{n\to\infty}(\sigma_f^2 - \epsilon_i) = \sigma_f^2$ and $\lim_{n\to\infty} k(\mathbf{x}_{(i)}(\boldsymbol{x}^*), \mathbf{x}_{(j)}(\boldsymbol{x}^*)) = \lim_{n\to\infty}(\sigma_f^2 - \epsilon_{ij}) = \sigma_f^2$. As a result, $\boldsymbol{k}_N^* \to \sigma_f^2 \mathbf{1}$ and

$$K^\infty := \lim_{n\to\infty} K_N = \sigma_\xi^2 I + \sigma_f^2 \mathbf{1}\mathbf{1}^T. \tag{8}$$

Now using Sherman-Morrison and the continuity of matrix inverse and matrix-matrix products:

$$(A + \boldsymbol{b}\boldsymbol{c}^T)^{-1} = A^{-1} - \frac{A^{-1}\boldsymbol{b}\boldsymbol{c}^T A^{-1}}{1 + \boldsymbol{c}^T A^{-1}\boldsymbol{b}} \tag{9}$$

$$(K^\infty)^{-1} = (\sigma_\xi^2 I + \sigma_f^2 \mathbf{1}\mathbf{1}^T)^{-1} = \frac{1}{\sigma_\xi^2}\left(I - \sigma_f^2 \frac{\mathbf{1}\mathbf{1}^T}{\sigma_\xi^2 + \sigma_f^2 \mathbf{1}^T\mathbf{1}}\right) \tag{10}$$

$$\begin{aligned}
\mathbf{1}^T (K^\infty)^{-1}\mathbf{1} &= \frac{m}{\sigma_\xi^2}\left(1 - \frac{m\sigma_f^2}{\sigma_\xi^2 + m\sigma_f^2}\right) \\
&= \frac{m}{\sigma_\xi^2}\left(1 - m\sigma_f^2 \frac{1}{m\sigma_f^2}\left(1 - \frac{\sigma_\xi^2}{m\sigma_f^2} + \frac{\sigma_\xi^4}{m^2\sigma_f^4} - \mathcal{O}\big(m^{-3}\big)\right)\right) \\
&= \frac{1}{\sigma_f^2} - \frac{\sigma_\xi^2}{m\sigma_f^4} + \mathcal{O}\big(m^{-2}\big)\,.
\end{aligned} \tag{11}$$

Thus,

$$\lim_{n\to\infty} \boldsymbol{k}_N^{*T} K_N^{-1} \boldsymbol{k}_N^* = \sigma_f^4 \mathbf{1}^T (K^\infty)^{-1}\mathbf{1} = \sigma_f^2 - \sigma_\xi^2 m^{-1} + \mathcal{O}\big(m^{-2}\big)\,. \tag{12}$$

□

Lemma 12 ($\mathcal{WSRF}$ expectations)**.** *Under (A3),* $\mathrm{E}_{\mathbf{y},\mathrm{y}^*}\{\boldsymbol{y}y^*\} = \boldsymbol{k}^*$ *and* $\mathrm{E}_{\mathbf{y}}\{\boldsymbol{y}\boldsymbol{y}^T\} = K$.

Proof. By assumption on the covariance properties of y and the independence and zero-mean of the additive noise, $\mathrm{E}_{\mathbf{y}}\{y_i y_j\} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Extending this to the joint distribution over $\boldsymbol{y}, y^*$ is straightforward and gives the results stated. □

Lemma 12 is subsequently assumed to be in use throughout A.2.

A.2 Limit proofs

In the following statements only misspecification of type (d) and/or (e) (subsection 4.2) is considered to be at work.

Lemma 13 (MSE limit)**.** *Under the assumptions (A1-3), for fixed $m < \infty$, the predictive GPnn given in subsection 4.1 converges pointwise in the sense of MSE wrt $P_{\mathbf{x}}$-a.e. as*

$$\lim_{n\to\infty} f_n^{\mathrm{MSE}}(\boldsymbol{\theta}) = \sigma_\xi^2(1 + m^{-1}) - \mathcal{O}\big(m^{-2}\big)\,.$$

Proof. This follows from Lemma 11 by expanding the definition of MSE:

$$\begin{aligned}
\lim_{n\to\infty} f_n^{\mathrm{MSE}}(\boldsymbol{\theta}) &= \lim_{n\to\infty} \mathrm{E}_{\mathbf{y},\mathrm{y}^*}\left\{|y^* - \mu_N^*|^2\right\} \\
&= \lim_{n\to\infty}\left[\mathrm{E}_{\mathrm{y}^*}\{{y^*}^2\} + \mathrm{E}_{\mathbf{y}}\{{\mu_N^*}^2\} - 2\,\mathrm{E}_{\mathbf{y},\mathrm{y}^*}\{\boldsymbol{k}_N^{*T} K_N^{-1}\boldsymbol{y}_N y^*\}\right] \\
&= \sigma_f^2 + \sigma_\xi^2 - \lim_{n\to\infty}\mathrm{E}_{\mathbf{y}}\{{\mu_N^*}^2\} \\
&= \sigma_\xi^2(1 + m^{-1}) - \mathcal{O}\big(m^{-2}\big)\,.
\end{aligned}$$

Since $\mathrm{E}_{\mathbf{y}}\{{\mu_N^*}^2\} = \mathrm{E}_{\mathbf{y}}\{\boldsymbol{k}_N^{*T} K_N^{-1}\boldsymbol{y}_N\boldsymbol{y}_N^T K_N^{-1}\boldsymbol{k}_N^*\} = \boldsymbol{k}_N^{*T} K_N^{-1}\boldsymbol{k}_N^*$, and by assumption $\mathrm{E}_{\mathbf{y},\mathrm{y}^*}\{\boldsymbol{y}_N y^*\} = \boldsymbol{k}_N^*$, even under a $\mathcal{WSRF}$ generative model (Lemma 12). □

Corollary 14 (NLL limit)**.**

$$\lim_{n\to\infty} f_n^{\mathrm{NLL}}(\boldsymbol{\theta}) = \frac{1}{2}\log\big(\sigma_\xi^2(1 + m^{-1})\big) + \frac{1}{2} + \frac{1}{2}\log 2\pi - \mathcal{O}\big(m^{-2}\big)\,.$$

Proof. The proof follows straightforwardly from Lemma 11 and because $\sigma_N^{*2} = \sigma_f^2 + \sigma_\xi^2 - \boldsymbol{k}_N^{*T} K_N^{-1} \boldsymbol{k}_N^*$.

$$\begin{aligned}
2\,\mathrm{E}_{\mathbf{y},y^*}\left\{l_N^*\right\} &= \mathrm{E}_{\mathbf{y},y^*}\left\{\log \sigma_N^{*2} + \frac{(y^* - \mu_N^*)^2}{\sigma_N^{*2}} + \log 2\pi\right\} \\
&= \log \sigma_N^{*2} + 1 + \log 2\pi \\
\lim_{n\to\infty} 2\,\mathrm{E}_{\mathbf{y},y^*}\left\{l_N^*\right\} &= \log\big(\sigma_f^2 + \sigma_\xi^2 - (\sigma_f^2 - \sigma_\xi^2 m^{-1} + \mathcal{O}\big(m^{-2}\big))\big) + 1 + \log 2\pi \\
&= \log\left(\sigma_\xi^2(1 + m^{-1}) - \mathcal{O}\big(m^{-2}\big)\right) + 1 + \log 2\pi \\
&= \log \sigma_\xi^2 + m^{-1} + 1 + \log 2\pi - \mathcal{O}\big(m^{-2}\big)\,.
\end{aligned}$$

□

A.2.1 Full misspecification

For the remainder of A.2 we assume that the full range of possible misspecifications ((a)-(e)) outlined in subsection 4.2 are in action. We refer to this case as "fully-misspecified" and introduce the notation $\hat{\mu}_N^*, \hat{\sigma}_N^{*2}$ to be understood to mean the predictive mean and variance under these misspecifications.

Lemma 15 (Fully misspecified MSE limit)**.** *For a fully misspecified model, asymptotically*

$$\lim_{n\to\infty} f_n^{\mathrm{MSE}}(\hat{\boldsymbol{\theta}}) = \sigma_\xi^2(1 + m^{-1}) - \mathcal{O}\big(m^{-2}\big)\,.$$

provided the misspecified kernel distance metric is weakly faithful *in the sense that the m^{th} nearest-neighbour converges under both the true and misspecified metrics (Definition 9).*

Proof.

$$\begin{aligned}
\mathrm{E}_{\mathbf{y}}\left\{\mathrm{E}_{y^*}\left[(y^* - \hat{\mu}_N^*)^2 \mid \mathbf{y}\right]\right\} &= \mathrm{E}_{\mathbf{y}}\left\{\mathrm{E}_{y^*}\left[y^{*2} - 2y^*\hat{\mu}_N^* + (\hat{\mu}_N^*)^2 \mid \mathbf{y}\right]\right\} \\
&= \mathrm{E}_{\mathbf{y}}\left\{\sigma_N^{*2} + \mu_N^{*2} - 2\mu_N^*\hat{\mu}_N^* + (\hat{\mu}_N^*)^2\right\} \\
&= \underbrace{\sigma_N^{*2}}_{(a)} + \underbrace{\boldsymbol{k}_N^{*T} K_N^{-1} \boldsymbol{k}_N^*}_{(b)} - 2\,\underbrace{\boldsymbol{k}_N^{*T} \hat{K}_N^{-1} \hat{\boldsymbol{k}}_N^*}_{(c)} + \underbrace{\hat{\boldsymbol{k}}_N^{*T} \hat{K}_N^{-1} K_N \hat{K}_N^{-1} \hat{\boldsymbol{k}}_N^*}_{(d)}.
\end{aligned}$$

We can use standard results to state that $(a) + (b) = \sigma_f^2 + \sigma_\xi^2$. Then we define $\hat{\gamma} = \frac{\hat{\sigma}_f^2}{\hat{\sigma}_\xi^2 + m\hat{\sigma}_f^2}$ and expand it in terms of m^{-1}:

$$1 - m\hat{\gamma} = \frac{\hat{\sigma}_\xi^2}{m\hat{\sigma}_f^2} - \frac{\hat{\sigma}_\xi^4}{m^2\hat{\sigma}_f^4} + \mathcal{O}\big(m^{-3}\big)\,.$$

In a manner similar to Lemma 11 we use this result to compute:

$$\begin{aligned}
\lim_{n\to\infty} (c) &= \sigma_f^2 \mathbf{1}^T \hat{\sigma}_\xi^{-2}(I - \hat{\gamma}\mathbf{1}\mathbf{1}^T)\mathbf{1}\hat{\sigma}_f^2 \\
&= \frac{\sigma_f^2\hat{\sigma}_f^2}{\hat{\sigma}_\xi^2} m(1 - m\hat{\gamma}) \\
&= \frac{\sigma_f^2\hat{\sigma}_f^2}{\hat{\sigma}_\xi^2}\left(\frac{\hat{\sigma}_\xi^2}{\hat{\sigma}_f^2} - \frac{\hat{\sigma}_\xi^4}{m\hat{\sigma}_f^4}\right) + \mathcal{O}\big(m^{-2}\big) \\
&= \sigma_f^2 - \frac{\sigma_f^2\hat{\sigma}_\xi^2}{m\hat{\sigma}_f^2} + \mathcal{O}\big(m^{-2}\big)
\end{aligned}$$

and

$$\begin{aligned}
\lim_{n\to\infty}(d) &= \frac{\hat\sigma_f^4}{\hat\sigma_\xi^4}\mathbf{1}^T(I-\hat\gamma\mathbf{1}\mathbf{1}^T)(\sigma_\xi^2 I+\sigma_f^2\mathbf{1}\mathbf{1}^T)(I-\hat\gamma\mathbf{1}\mathbf{1}^T)\mathbf{1}\\
&= \frac{\hat\sigma_f^4}{\hat\sigma_\xi^4}\mathbf{1}^T\left[\sigma_\xi^2 I+\sigma_f^2\mathbf{1}\mathbf{1}^T-2\sigma_\xi^2\hat\gamma\mathbf{1}\mathbf{1}^T+\hat\gamma^2\sigma_\xi^2 m\mathbf{1}\mathbf{1}^T-2\sigma_f^2\hat\gamma m\mathbf{1}\mathbf{1}^T+\sigma_f^2\hat\gamma^2m^2\mathbf{1}\mathbf{1}^T\right]\mathbf{1}\\
&= \frac{\hat\sigma_f^4}{\hat\sigma_\xi^4}m(\sigma_\xi^2+m\sigma_f^2)\left[1-2m\hat\gamma+m^2\hat\gamma^2\right]\\
&= \frac{\hat\sigma_f^4}{\hat\sigma_\xi^4}m(\sigma_\xi^2+m\sigma_f^2)(1-m\hat\gamma)^2\\
&= \frac{\hat\sigma_f^4}{\hat\sigma_\xi^4}m(\sigma_\xi^2+m\sigma_f^2)\left(\frac{\hat\sigma_\xi^4}{m^2\hat\sigma_f^4}-2\frac{\hat\sigma_\xi^6}{m^3\hat\sigma_f^6}+\mathcal{O}\big(m^{-4}\big)\right)\\
&= \sigma_f^2+\frac{\sigma_\xi^2}{m}-2\frac{\sigma_f^2}{\hat\sigma_f^2}\frac{\hat\sigma_\xi^2}{m}-\mathcal{O}\big(m^{-2}\big)\,,
\end{aligned}$$

where we have used the expansion of $1-m\hat\gamma$ given earlier. Putting these results together gives

$$\begin{aligned}
\lim_{n\to\infty}f_n^{\mathrm{MSE}}(\hat{\boldsymbol\theta}) &= \lim_{n\to\infty}[(a)+(b)-2(c)+(d)]\\
&= \sigma_f^2+\sigma_\xi^2-2\left(\sigma_f^2-\frac{\sigma_f^2\hat\sigma_\xi^2}{m\hat\sigma_f^2}\right)+\sigma_f^2+\frac{\sigma_\xi^2}{m}-2\frac{\sigma_f^2}{\hat\sigma_f^2}\frac{\hat\sigma_\xi^2}{m}-\mathcal{O}\big(m^{-2}\big)\\
&= \sigma_\xi^2(1+m^{-1})-\mathcal{O}\big(m^{-2}\big)\,.
\end{aligned}$$

□

Lemma 16 (Calibration limit under full misspecification)**.**

$$\lim_{n\to\infty}f_n^{\mathrm{CAL}}(\hat{\boldsymbol\theta})=\frac{\sigma_\xi^2}{\hat\sigma_\xi^2}+\mathcal{O}\big(m^{-2}\big)\,.$$

Proof. We use continuity to write

$$\lim_{n\to\infty}\mathrm{E}_{\mathbf{y},\mathrm{y}^*}\left\{\frac{(y^*-\hat\mu_N^*)^2}{\hat\sigma_N^{*2}}\right\}=\left(\lim_{n\to\infty}\frac{1}{\hat\sigma_N^{*2}}\right)\left(\lim_{n\to\infty}f_n^{\mathrm{MSE}}(\hat{\boldsymbol\theta})\right).$$

By direct application of Lemma 11 $\hat\sigma_N^{*2}\xrightarrow{n\to\infty}\hat\sigma_\xi^2(1+m^{-1})-\mathcal{O}\big(m^{-2}\big)$ and thus

$$\lim_{n\to\infty}f_n^{\mathrm{CAL}}(\hat{\boldsymbol\theta})=\frac{\sigma_\xi^2}{\hat\sigma_\xi^2}+\mathcal{O}\big(m^{-2}\big)\,.$$

□

Corollary 17 (NLL limit under full misspecification)**.**

$$\lim_{n\to\infty}f_n^{\mathrm{NLL}}(\hat{\boldsymbol\theta})=\frac{1}{2}\log\big(\hat\sigma_\xi^2(1+m^{-1})\big)+\frac{1}{2}\frac{\sigma_\xi^2}{\hat\sigma_\xi^2}+\frac{1}{2}\log 2\pi-\mathcal{O}\big(m^{-2}\big)\,.$$

Proof. We start with

$$2f_n^{\mathrm{NLL}}(\hat{\boldsymbol\theta})=\mathrm{E}_{\mathbf{y},\mathrm{y}^*}\left\{\log\hat\sigma_N^{*2}+\frac{(y^*-\hat\mu_N^*)^2}{\hat\sigma_N^{*2}}+\log 2\pi\right\}.$$

For the second term we use Lemma 16 so that we have

$$\lim_{n\to\infty}2f_n^{\mathrm{NLL}}(\hat{\boldsymbol\theta})=\log\hat\sigma_\xi^2+m^{-1}+\frac{\sigma_\xi^2}{\hat\sigma_\xi^2}+\log 2\pi-\mathcal{O}\big(m^{-2}\big)\,.$$

□

Proof of Theorem 1. We construct the proof using all of the intermediate results given above. In particular item (i) follows from Lemma 15, item (ii) from Lemma 16 and item (iii) from Corollary 17.

□

B Parameter Calibration (Proof of Lemma 4)

Proof of Lemma 4. (a) Replacing parameters $\hat{\boldsymbol{\theta}} = (\hat{l}, \hat{\sigma}_\xi^2, \hat{\sigma}_f^2)$ with $\hat{\boldsymbol{\theta}}' = (\hat{l}, \alpha\hat{\sigma}_\xi^2, \alpha\hat{\sigma}_f^2)$ changes all of the ${\sigma_i^*}^2$ values to $\alpha{\sigma_i^*}^2$ and therefore changes the calibration value on C from $\alpha = \frac{1}{c}\sum_{i=1}^{c} \frac{(y_i^* - \mu_i^*)^2}{{\sigma_i^*}^2}$ to $\alpha/\alpha = 1$. (b) The NLL on C arising from parameters $(\hat{l}, \alpha\hat{\sigma}_\xi^2, \alpha\hat{\sigma}_f^2)$ is $\frac{1}{2c}\sum_{i=1}^{c}\{\log\left(\alpha\hat{\sigma}_\xi^2\right) + (y_i^* - \mu_i^*)^2/(\alpha{\sigma_i^*}^2) + \log 2\pi)\}$ which, on taking first and second derivatives w.r.t. α, is found to be uniquely minimised by $\alpha = \frac{1}{c}\sum_{i=1}^{c} \frac{(y_i^* - \mu_i^*)^2}{{\sigma_i^*}^2}$. (c) It is easily shown that replacing parameters $(\hat{\sigma}_\xi^2, \hat{\sigma}_f^2)$ by $(k\hat{\sigma}_\xi^2, k\hat{\sigma}_f^2)$ (for any $k > 0$) in the formula for μ^* (Equation 3 and Equation 6) does not alter μ^*. Hence the value of MSE $= \frac{1}{n^*}\sum_{i=1}^{n^*}(y_i^* - \mu_i^*)^2$ on any size-n^* test set is unchanged when parameters $\hat{\boldsymbol{\theta}}'$ are used in place of $\hat{\boldsymbol{\theta}}$. □

C Real world datasets

We consider a variety of datasets from the standard UCI machine learning repository[1]. These datasets are commonly used in the GP literature (see [8] for instance) and are, in principle, easily available online. In practice, we encountered some difficulties: the dataset documentation is often limited; the dataset names commonly used in other published papers do not always match the UCI database naming and important details about data pre-processing, which features to use etc, are often omitted. There are numerous attempts on GitHub and elsewhere at cataloguing these datasets along with any pre-processing, however we had limited success using them, with many appearing unmaintained. Our focus in this work is on testing our methods on a variety of real world datasets and in a way that is, as far as possible, consistent with other papers. We therefore rejected datasets about which there is ambiguity over the correct features to use, or even which column to regress on or for which outlier rejection is required but undocumented elsewhere.

Referring to the datasets used in [8], we were able to locate the following:

- Song (`https://archive.ics.uci.edu/ml/machine-learning-databases/00203/YearPredictionMSD.txt.zip`)
- Bike (`https://archive.ics.uci.edu/ml/machine-learning-databases/00275/Bike-Sharing-Dataset.zip`)
- Poletele (`https://archive.ics.uci.edu/ml/machine-learning-databases/parkinsons/telemonitoring/parkinsons_updrs.data`)
- Keggdirected (`https://archive.ics.uci.edu/ml/machine-learning-databases/00220/Relation%20Network%20(Directed).data`)
- Keggundirected (`https://archive.ics.uci.edu/ml/machine-learning-databases/00221/Reaction%20Network%20(Undirected).data`)
- CTSlice (`https://archive.ics.uci.edu/ml/machine-learning-databases/00206/slice_localization_data.zip`)
- Road3d (`https://archive.ics.uci.edu/ml/machine-learning-databases/00246/3D_spatial_network.txt`)
- Protein (`https://archive.ics.uci.edu/ml/machine-learning-databases/00265/CASP.csv`)
- Buzz (`https://archive.ics.uci.edu/ml/machine-learning-databases/00248/regression.tar.gz`)
- HouseElectric (`https://archive-beta.ics.uci.edu/dataset/235/individual+household+electric+power+consumption`)

We were unable to find any documentation on the Kegg datasets to indicate which of the columns should be used as the independent variable (the regressor) and neither is this mentioned in any

[1] `https://archive-beta.ics.uci.edu`, accessed April 2023.

literature of which we are aware. Initial runs of standard exact GP training and prediction produced RMSEs much higher that reported in [8]. Combining these two observations, we chose to exclude both Kegg datasets. Likewise we faced problems with Buzz. An analysis of the y values revealed a small proportion of extremely large outliers that we found could unduly distort performance results (e.g. depending on whether these outliers appeared in the test set for some of the random splits). With the lack of documentation we were unable to identify an outlier rejection scheme that we were confident would be consistent with results quoted in other papers. For this reason we have excluded Buzz.

The choice of $(\boldsymbol{x}, y)$ value that we applied for each of the used datasets is as follows:

- Song. The first column is y, all remaining columns are $\boldsymbol{x}$.
- Bike. We use `hour.csv`. The y value is `cnt`. `dteday` (the date) is transformed to just be the integer representation of the day. `instant` is just an index so is dropped. `registered` and `casual` are dropped as `registered` + `casual` = `cnt`.
- Poletele. The y value is `total_UPDRS`. The columns `subject#` and `test_time` are not relevant to the problem so are dropped.
- CTSlice. y value is the final column. The first column is dropped as it is just an index. We additionally drop six columns which are constant over the majority of the dataset, namely columns 59, 69, 179, 189, 279 and 351.
- Road3d. y value is the final column. The first column is dropped as it is just an index.
- Protein. This dataset was processed as per `https://github.com/hughsalimbeni/bayesian_benchmarks`, whereafter we used our own random (seeded) train/test split.
- HouseElectric. y value is the column labelled "Global active power", rescaled by $1000/60$ and with "Sub metering 1,2,3" columns subtracted. We convert the date column into day-of-year/365 and the time column into time of day in minutes. Further, we remove any rows with null entries.

We note that although we are using a standard set of real-world datasets, it is not always clear exactly how others in the field have carried out their own preprocessing, limiting the ability to make direct comparisons to other results reported in the literature.

D Additional Implementation Details

D.1 Pre-whitening of Data

For all datasets covered in subsection 7.1 the following "whitening" preprocessing step is adopted: Let $\boldsymbol{y}$ be the vector of all regressor values in the *training* dataset only, and X the matrix of all regressands in the *training* dataset only, where each row of X is a feature. Let μ_y, σ_y^2 be the sample mean and variance of $\boldsymbol{y}$ respectively in the training dataset, then the whitened y values used in both the training and test set are simply $\sigma_y^{-1}(y - \mu_y)$. Let $\boldsymbol{\mu}_X, \Sigma_X$ be the sample mean and covariance matrix of X respectively . Let $\Sigma_X = MM^T$, then the whitened x values in both the training and test data are $\frac{1}{\sqrt{d}}M^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_X)$, where d is the feature dimension of X. **Note**: the performance metrics given in subsection 7.1 are expressed in terms of the whitened y values rather than the y values in their original form. This appears to be common practice in the literature and has no bearing on the *comparative* performance of the different methods within this paper.

D.2 Test-Set Batching

To prevent excessive memory consumption, we perform all predictions for the distributed and variational methods in batches of 1000 points at a time. Where this is not possible (e.g. for especially large datasets), we use smaller batches of 500 or 250 points, as appropriate.

D.3 Additional Implementation Details for SVGP

We use the sparse variational inducing point approach of [11], following the implementation provided by GPyTorch, which in particular uses a Choleksy decomposition to parameterise the covariance matrix of the variational prior. We broadly follow the SVGP implementation example provided by `https://docs.gpytorch.ai/en/stable/examples/04_Variational_and_Approximate_GPs/SVGP_Regression_CUDA.html`. In particular, we follow their example in using the Adam optimiser to train our model over 100 epochs with a minibatch size of 1024 and a learning rate of 0.01. We opt to use 1024 inducing points. All experiments under this method are run on a SageMaker ml.p3.2xlarge instance, consisting of a single Tesla V100 GPU with 16GB of memory.

D.4 Additional Implementation Details for Distributed methods

A good introduction to distributed methods for Gaussian process inference is [7]. Here we run the product-of-experts (PoE) [12], generalised product-of-experts (gPoE) [2], Bayesian committee machine (BCM) [24], robust Bayesian committee machine (rBCM) [7] and generalised robust Bayesian committee machine (GrBCM) [13] following the recommendation in [4] to aggregate in f-space. There are three components to any distributed method: the hyperparameter inference, the *partitioner* and the *aggregator*. Hyperparameter estimation is the same for all of the methods: we use the method in section 3.1 of [7], randomly partitioning the entire training set into subsets of size 625 (or as close as possible with equal-sized experts given that in general n is not a multiple of 625). A block diagonal approximation (with $n/625$ blocks) is then used to approximate to the full $n \times n$ gram kernel matrix. To recover hyperparameters with this we use Gaussian Process models with a zero prior mean and a scaled square-exponential kernel. Training is conducted using the Adam optimiser with a learning rate of 0.1 over 100 optimiser iterations. Once the hyperparameters are trained, we run our distributed prediction mechanism to evaluate performance against the test-set. The 625-sized partitioned blocks are referred to as "experts" and the shared hyperparameter values are distributed to each expert and held fixed thereafter. In the *aggregator*, or distributed prediction phase, each expert produces an individual predictive distribution and these are then aggregated to a final predictive mean and variance for each of our test points. GRBCM prediction is a little more complex than this as it makes use of an additional "communications" expert as explained in [13], aggregating in f-space as recommended in [4]. We provide timing statistics for training these models.

We use our own GPyTorch-based implementation of distributed GP approximations. All exact GP calculations are performed using GPyTorch using the default settings (so 20 Lanczos iterations throughout and a CG tolerance of 1 for hyperparameter inference, and 10^{-3} for posterior predictions). For all of our experiments, we utilise an AWS t3.2xlarge instance (consisting of 8 Intel Skylake Processors and 32 GB of RAM).

E Further simulation results

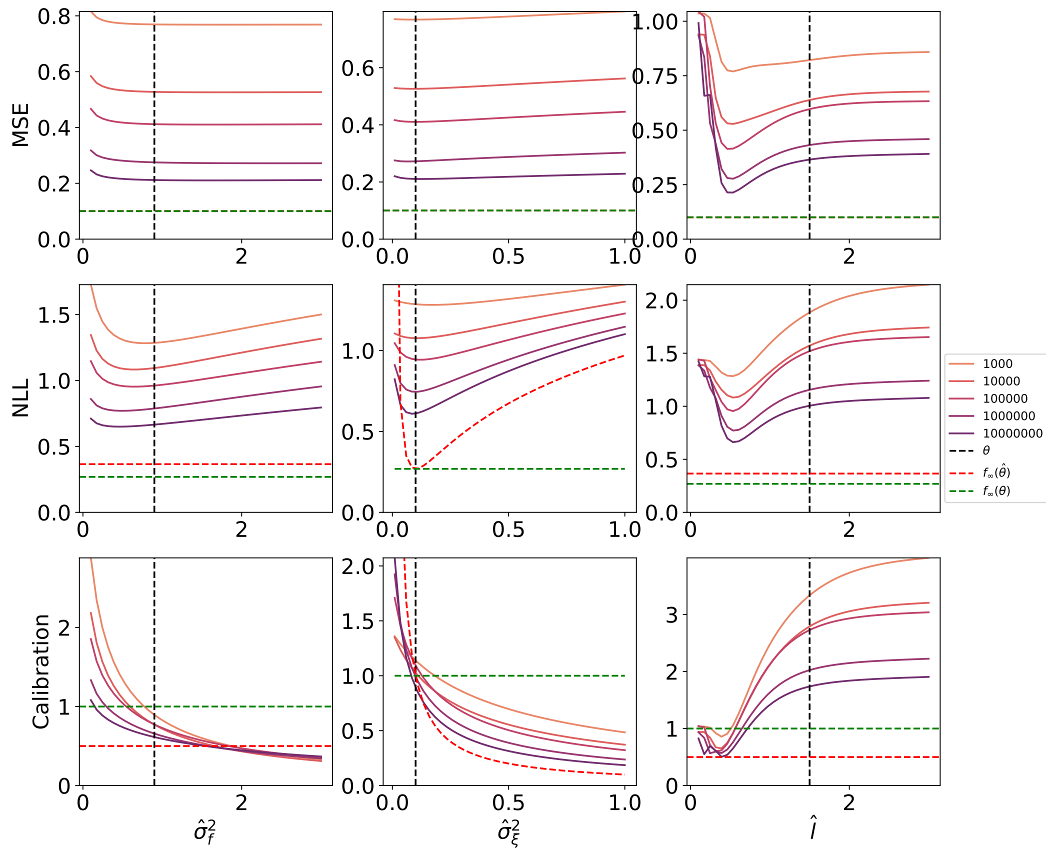

Figure 6: Behaviour of performance metrics as functions of kernel hyperparameters for increasing training set sizes n. The black dashed line denotes the true parameter value; the red dashed line shows the limiting behaviour as $n \to \infty$ and the green dashed line shows the limiting behaviour when the hyperparameters are correct. Simulations run with $d = 20, l = 0.5, \sigma_\xi^2 = 0.1, \sigma_f^2 = 0.9$. Assumed parameters when constant: $\hat{\sigma}_\xi^2 = 0.2, \hat{\sigma}_f^2 = 0.8, \hat{l} = 0.5$.

F Further Results on UCI Datasets

F.1 Results for all distributed methods

Table 3: Results for all methods on all metrics.

Dataset	n	d	Model	**Calibration**	**NLL**	**RMSE**
Bike	1.4e+04	13	BCM	1.02 ± 0.02	1.0 ± 0.0065	0.66 ± 0.0043
			GPOE	0.873 ± 0.012	1.03 ± 0.0069	0.664 ± 0.0054
			GRBCM	0.893 ± 0.014	0.977 ± 0.0057	0.634 ± 0.004
			OURS	0.974 ± 0.087	0.953 ± 0.013	0.624 ± 0.0079
			POE	1.03 ± 0.022	1.01 ± 0.0083	0.664 ± 0.0054
			RBCM	**1.01 ± 0.02**	1.0 ± 0.0065	0.659 ± 0.0043
			SVGP	0.898 ± 0.011	**0.93 ± 0.0043**	**0.606 ± 0.0033**
Ctslice	4.2e+04	378	BCM	5.04 ± 0.28	1.43 ± 0.13	0.311 ± 0.0052
			GPOE	0.435 ± 0.013	0.422 ± 0.0015	0.347 ± 0.0027
			GRBCM	1.13 ± 0.11	-0.159 ± 0.052	0.237 ± 0.012
			OURS	**1.04 ± 0.085**	**-1.26 ± 0.01**	**0.132 ± 0.00062**
			POE	6.39 ± 0.27	2.08 ± 0.12	0.347 ± 0.0027
			RBCM	4.16 ± 0.25	0.987 ± 0.11	0.28 ± 0.0048
			SVGP	0.865 ± 0.026	0.467 ± 0.016	0.384 ± 0.0064
Houseelectric	1.6e+06	8	BCM	1.27 ± 0.0046	-1.33 ± 0.0009	0.0634 ± 3.5e-05
			GPOE	0.908 ± 0.0065	-1.43 ± 0.0016	0.0638 ± 7.7e-05
			GRBCM	1.25 ± 0.011	-1.34 ± 0.0039	0.063 ± 0.00026
			OURS	**1.08 ± 0.21**	**-1.56 ± 0.0065**	**0.0506 ± 0.00072**
			POE	1.28 ± 0.006	-1.32 ± 0.0018	0.0638 ± 7.7e-05
			RBCM	1.24 ± 0.0054	-1.34 ± 0.0013	0.0626 ± 5.2e-05
			SVGP	0.911 ± 0.038	-1.46 ± 0.0046	0.0566 ± 0.00011
Poletele	4.6e+03	19	BCM	1.07 ± 0.029	0.00035 ± 0.019	0.243 ± 0.0048
			GPOE	0.917 ± 0.02	0.0344 ± 0.013	0.246 ± 0.0038
			GRBCM	0.872 ± 0.024	0.0091 ± 0.015	0.241 ± 0.0033
			OURS	**1.03 ± 0.073**	**-0.214 ± 0.019**	**0.195 ± 0.0042**
			POE	1.1 ± 0.036	0.00772 ± 0.016	0.246 ± 0.0038
			RBCM	1.08 ± 0.029	0.00309 ± 0.018	0.243 ± 0.0048
			SVGP	0.862 ± 0.035	-0.0667 ± 0.017	0.226 ± 0.0059
Protein	3.6e+04	9	BCM	1.04 ± 0.0097	1.14 ± 0.003	0.754 ± 0.0022
			GPOE	0.925 ± 0.007	1.15 ± 0.0035	0.763 ± 0.0024
			GRBCM	0.95 ± 0.012	1.11 ± 0.0051	0.733 ± 0.0038
			OURS	**0.991 ± 0.029**	**1.01 ± 0.0016**	**0.666 ± 0.0014**
			POE	1.07 ± 0.0088	1.15 ± 0.0033	0.763 ± 0.0024
			RBCM	1.03 ± 0.0096	1.13 ± 0.003	0.752 ± 0.0022
			SVGP	0.908 ± 0.016	1.05 ± 0.0059	0.688 ± 0.0043
Road3D	3.4e+05	2	BCM	1.01 ± 0.017	0.753 ± 0.007	0.514 ± 0.0035
			GPOE	0.756 ± 0.012	0.819 ± 0.0054	0.529 ± 0.0037
			GRBCM	0.873 ± 0.011	0.685 ± 0.0041	0.478 ± 0.0023
			OURS	**0.991 ± 0.041**	**0.371 ± 0.004**	**0.351 ± 0.0014**
			POE	1.07 ± 0.019	0.783 ± 0.0076	0.529 ± 0.0037
			RBCM	0.976 ± 0.016	0.735 ± 0.0066	0.505 ± 0.0034
			SVGP	0.9 ± 0.00094	0.608 ± 0.018	0.443 ± 0.008
Song	4.6e+05	90	BCM	1.56 ± 0.0063	1.32 ± 0.0012	0.851 ± 6.7e-05
			GPOE	0.926 ± 0.00049	1.27 ± 3.4e-05	0.864 ± 7.5e-05
			GRBCM	1.61 ± 0.11	1.46 ± 0.058	0.961 ± 0.035
			OURS	0.99 ± 0.037	**1.18 ± 0.0045**	**0.787 ± 0.0045**
			POE	1.61 ± 0.0067	1.34 ± 0.0013	0.864 ± 7.5e-05
			RBCM	1.56 ± 0.0062	1.31 ± 0.0011	0.851 ± 6.4e-05
			SVGP	**0.991 ± 0.02**	1.24 ± 0.0012	0.834 ± 0.0011

F.2 Performance of different kernels

Table 4: Results on the UCI datasets using different kernel choices for our method and demonstrating the apparent superiority of the exponential kernel in these cases.

Dataset	n	d	**Calibration** Distributed	Ours (Exp)	Ours (Matérn)	Ours (RBF)	SVGP
Poletele	4.6e+03	19	0.872 ± 0.024	**0.994 ± 0.15**	0.971 ± 0.13	1.03 ± 0.073	0.862 ± 0.035
Bike	1.4e+04	13	0.893 ± 0.014	**0.988 ± 0.098**	0.971 ± 0.086	0.974 ± 0.087	0.898 ± 0.011
Protein	3.6e+04	9	0.95 ± 0.012	**0.995 ± 0.038**	0.993 ± 0.031	0.991 ± 0.029	0.908 ± 0.016
Ctslice	4.2e+04	378	1.13 ± 0.11	0.912 ± 0.071	**1.04 ± 0.082**	1.04 ± 0.085	0.865 ± 0.026
Road3D	3.4e+05	2	0.873 ± 0.011	1.09 ± 0.065	**1.0 ± 0.054**	0.991 ± 0.041	0.9 ± 0.00094
Song	4.6e+05	90	1.56 ± 0.0063	**0.995 ± 0.033**	0.994 ± 0.035	0.99 ± 0.037	0.991 ± 0.02
Houseelectric	1.6e+06	8	1.24 ± 0.0054	1.11 ± 0.29	**1.08 ± 0.27**	1.08 ± 0.21	0.911 ± 0.038

Dataset	n	d	**RMSE** Distributed	Ours (Exp)	Ours (Matérn)	Ours (RBF)	SVGP
Poletele	4.6e+03	19	0.241 ± 0.0033	**0.169 ± 0.0076**	0.17 ± 0.0076	0.195 ± 0.0042	0.226 ± 0.0059
Bike	1.4e+04	13	0.634 ± 0.004	**0.565 ± 0.0036**	0.6 ± 0.0044	0.624 ± 0.0079	0.606 ± 0.0033
Protein	3.6e+04	9	0.733 ± 0.0038	**0.58 ± 0.0068**	0.629 ± 0.004	0.666 ± 0.0014	0.688 ± 0.0043
Ctslice	4.2e+04	378	0.237 ± 0.012	**0.123 ± 0.004**	0.126 ± 0.0024	0.132 ± 0.00062	0.384 ± 0.0064
Road3D	3.4e+05	2	0.478 ± 0.0023	**0.0976 ± 0.013**	0.27 ± 0.01	0.351 ± 0.0014	0.443 ± 0.008
Song	4.6e+05	90	0.851 ± 6.7e-05	**0.776 ± 0.004**	0.778 ± 0.0045	0.787 ± 0.0045	0.834 ± 0.0011
Houseelectric	1.6e+06	8	0.0626 ± 5.2e-05	**0.045 ± 0.00025**	0.0485 ± 0.0004	0.0506 ± 0.00072	0.0566 ± 0.0001

Dataset	n	d	**NLL** Distributed	Ours (Exp)	Ours (Matérn)	Ours (RBF)	SVGP
Poletele	4.6e+03	19	0.0091 ± 0.015	**-0.397 ± 0.028**	-0.346 ± 0.032	-0.214 ± 0.019	-0.0667 ± 0.017
Bike	1.4e+04	13	0.977 ± 0.0057	**0.854 ± 0.004**	0.915 ± 0.0077	0.953 ± 0.013	0.93 ± 0.0043
Protein	3.6e+04	9	1.11 ± 0.0051	**0.853 ± 0.013**	0.95 ± 0.0061	1.01 ± 0.0016	1.05 ± 0.0059
Ctslice	4.2e+04	378	-0.159 ± 0.052	-1.05 ± 0.027	**-1.31 ± 0.017**	-1.26 ± 0.01	0.467 ± 0.016
Road3D	3.4e+05	2	0.685 ± 0.0041	**-0.931 ± 0.14**	0.109 ± 0.039	0.371 ± 0.004	0.608 ± 0.018
Song	4.6e+05	90	1.32 ± 0.0012	**1.16 ± 0.0046**	1.17 ± 0.0051	1.18 ± 0.0045	1.24 ± 0.0012
Houseelectric	1.6e+06	8	-1.34 ± 0.0013	**-1.95 ± 0.028**	-1.62 ± 0.0095	-1.56 ± 0.0065	-1.46 ± 0.0046

G Overall Computational Expenditure

Our distributed and variational method experiments were conducted using cloud computing resources. Experiments using our own method have been carried out on an author's laptop. SVGP experiments were run using a SageMaker virtual machine on a single Nvidia Tesla V100 GPU with 16GB memory. Distributed method experiments were run using eight Intel Xeon Platinum 8000 CPU cores (t3.2xlarge EC2 instances).

Below we will attempt to give reasonable indications of the amount of computational work expended to obtain the results in this paper, though note that we are neglecting the work expended in the development and research stages that did not directly contribute to the runs in the paper. As such, the costs presented are representative of the costs of replicating our paper, not repeating the research from scratch. Instead of reporting costs in dollars, we will report approximate computing hours for each instance type. The reader can then estimate their own costs using the current instance costs in the region of their choice, or under other cloud providers or even using on-premise compute.

Dataset	**Billed hours (1 GPU, 3 runs)**	**Billed hours (8 CPUs, 3 runs of 5 methods)**
bike	0.027	0.222
ctslice	0.082	0.546
houseelectric	3.713	256.776
poletele	0.010	0.079
protein	0.068	0.680
road3d	0.635	31.674
song	0.904	12.237

This gives a total of around 5.4 hours of compute time on a 1 GPU VM and 302.2 hours on an 8 CPU VM.

References

[1] F. Bachoc, N. Durrande, D. Rullière, and C. Chevalier. Properties and Comparison of Some Kriging Sub-model Aggregation Methods. *Mathematical Geosciences*, 2022.

[2] Y. Cao and D. J. Fleet. Generalized product of experts for automatic and principled fusion of gaussian process predictions. *arXiv preprint arXiv:1410.7827*, 2014.

[3] K. Chalupka, C. K. Williams, and I. Murray. A framework for evaluating approximation methods for gaussian process regression. *Journal of Machine Learning Research*, 14:333–350, 2013.

[4] S. Cohen, R. Mbuvha, T. Marwala, and M. Deisenroth. Healing products of Gaussian process experts. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 2068–2077. PMLR, 2020-13.

[5] A. Datta, S. Banerjee, A. O. Finley, and A. E. Gelfand. Hierarchical Nearest-Neighbor Gaussian Process Models for Large Geostatistical Datasets. *Journal of the American Statistical Association*, 111(514):800–812, 2016.

[6] A. Datta, S. Banerjee, A. O. Finley, and A. E. Gelfand. On nearest-neighbor Gaussian process models for massive spatial data. *Wiley Interdisciplinary Reviews: Computational Statistics*, 8(5):162–171, 2016.

[7] M. Deisenroth and J. W. Ng. Distributed gaussian processes. In *International Conference on Machine Learning*, pages 1481–1490. PMLR, 2015.

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

[9] L. Györfi, M. Kohler, A. Krzyżak, and H. Walk. *A Distribution-Free Theory of Nonparametric Regression*. Springer Series in Statistics, 2010.

[10] K. Hayashi, M. Imaizumi, and Y. Yoshida. On random subsampling of gaussian process regression: A graphon-based analysis. In *International Conference on Artificial Intelligence and Statistics*, pages 2055–2065. PMLR, 2020.

[11] J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian processes for big data. *arXiv preprint arXiv:1309.6835*, 2013.

[12] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[13] H. Liu, J. Cai, Y. Wang, and Y. S. Ong. Generalized robust bayesian committee machine for large-scale gaussian process regression. In *International Conference on Machine Learning*, pages 3131–3140. PMLR, 2018.

[14] H. Liu, Y.-S. Ong, X. Shen, and J. Cai. When gaussian process meets big data: A review of scalable gps. *IEEE transactions on neural networks and learning systems*, 31(11):4405–4423, 2020.

[15] J. E. Oakley and A. O'Hagan. Probabilistic sensitivity analysis of complex models: A bayesian approach. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 66(3):751–769, 2004.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[17] B. Schölkopf. The Kernel Trick for Distances. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000.

[18] H. Song, T. Diethe, M. Kull, and P. Flach. Distribution calibration for regression. *36th International Conference on Machine Learning, ICML 2019*, 2019-June:10347–10356, 2019.

[19] M. L. Stein, Z. Chi, and L. J. Welty. Approximating likelihoods for large spatial data sets. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 66(2):275–296, 2004.

[20] A. Stephenson, R. Allison, and E. Pyzer-Knapp. Provably reliable large-scale sampling from gaussian processes. *arXiv preprint arXiv:2211.08036*, 2022.

[21] S. Surjanovic and D. Bingham. Virtual library of simulation experiments: Test functions and datasets. Retrieved May 5, 2023, from `http://www.sfu.ca/~ssurjano`.

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

[23] G.-L. Tran, D. Milios, P. Michiardi, and M. Filippone. Sparse within sparse gaussian processes using neighbor information. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10369–10378. PMLR, 2021-18.

[24] V. Tresp. A bayesian committee machine. *Neural computation*, 12(11):2719–2741, 2000.

[25] K. Wang, G. Pleiss, J. Gardner, S. Tyree, K. Q. Weinberger, and A. G. Wilson. Exact gaussian processes on a million data points. *Advances in Neural Information Processing Systems*, 32, 2019.

[26] C. K. Williams and C. E. Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[27] L. Wu, G. Pleiss, and J. Cunningham. Variational Nearest Neighbor Gaussian Process. *Proceedings of the 39th International Conference on Machine Learning*, 2022.