Distributed Gaussian Processes

Marc Peter Deisenroth M.DEISENROTH@IMPERIAL.AC.UK
Department of Computing, Imperial College London, United Kingdom

Jun Wei Ng JUNWEI.NG10@ALUMNI.IMPERIAL.AC.UK
Department of Computing, Imperial College London, United Kingdom

Abstract

To scale Gaussian processes (GPs) to large data sets we introduce the robust Bayesian Committee Machine (rBCM), a practical and scalable product-of-experts model for large-scale distributed GP regression. Unlike state-of-the-art sparse GP approximations, the rBCM is conceptually simple and does not rely on inducing or variational parameters. The key idea is to recursively distribute computations to independent computational units and, subsequently, recombine them to form an overall result. Efficient closed-form inference allows for straightforward parallelisation and distributed computations with a small memory footprint. The rBCM is independent of the computational graph and can be used on heterogeneous computing infrastructures, ranging from laptops to clusters. With sufficient computing resources our distributed GP model can handle arbitrarily large data sets.

1. Introduction

Gaussian processes (GPs) (Rasmussen & Williams, 2006) are the method of choice for probabilistic nonlinear regression: Their non-parametric nature allows for flexible modelling without specifying low-level assumptions (e.g., the degree of a polynomial) in advance. Inference can be performed in a principled way simply by applying Bayes' theorem. GPs have had substantial impact in geostatistics (Cressie, 1993), optimisation (Jones et al., 1998; Brochu et al., 2009), data visualisation (Lawrence, 2005), robotics and reinforcement learning (Deisenroth et al., 2015), spatio-temporal modelling (Luttinen & Ilin, 2012), and active learning (Krause et al., 2008). A strength of GPs is that they are a fairly reliable black-box function approximator, i.e., they produce reasonable predictions without manual parameter tuning. However, their inherent weakness is that they scale poorly with the size N of the data set: Training and predicting scale in $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$, respectively, which practically limits GPs to data sets of size $\mathcal{O}(10^4)$.

For large data sets (e.g., $N > 10^4$) sparse approximations are often used (Williams & Seeger, 2001; Quiñonero-Candela & Rasmussen, 2005; Hensman et al., 2013; Titsias, 2009; Lázaro-Gredilla et al., 2010; Shen et al., 2006; Gal et al., 2014). Typically, they lower the computational burden by implicitly (or explicitly) using a subset of the data, which scales (sparse) GPs to training set sizes $N \in \mathcal{O}(10^6)$. Recently, Gal et al. (2014) proposed an approach that scales variational sparse GPs (Titsias, 2009) further by exploiting distributed computations. In particular, they derive an exact re-parameterisation of the variational sparse GP by Titsias (2009) to update the variational parameters independently on different computing nodes. However, even with sparse approximations it is inconceivable to apply GPs to training set sizes of $N \geq \mathcal{O}(10^7)$.

An alternative to sparse approximations is to distribute the computations by using independent local "expert" models, which operate on subsets of the data. These local models typically require stationary kernels for a notion of "distance" and "locality". Shen et al. (2006) used KD-trees to recursively partition the data space into a multi-resolution tree data structure, which scale GPs to $\mathcal{O}(10^4)$ training points. However, no solutions for variance predictions are provided, and the approach is limited to stationary kernels. Along the lines of exploiting locality, mixture-of-experts (MoE) models (Jacobs et al., 1991) have been applied to GP regression (Rasmussen & Ghahramani, 2002; Meeds & Osindero, 2006; Yuan & Neubauer, 2009). However, these models have not primarily been used to speed up GP regression, but rather to increase the expressiveness of the model, i.e., allowing for heteroscedasticity and non-stationarity. Each local model possesses its own set of hyper-parameters to be optimised. Predictions are made by collecting the predictions of all local expert models, and

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

weighting them using the responsibilities assigned by the gating network. Closed-form inference in these models is intractable, and approximations typically require MCMC. Nguyen & Bonilla (2014) sidestep MCMC inference and speed the GP-MoE model up by (i) fixing the number of GP experts, (ii) combining it with the pseudo-input sparse approximation by Snelson & Ghahramani (2006). This approach assigns data points to expert probabilistically using proximity information provided by stationary kernels and scales GPs to $\mathcal{O}(10^5)$ data points.

Product-of-GP-experts models (PoEs) sidestep the weight assignment problem of mixture models: Since PoEs multiply predictions made by independent GP experts, the overall prediction naturally weights the contribution of each expert. However, the model tends to be overconfident (Ng & Deisenroth, 2014). Cao and Fleet (Cao & Fleet, 2014) recently proposed a generalised PoE-GP model in which the contribution of an expert in the overall prediction can weighted individually. This model is often too conservative, i.e., it over-estimates variances. Tresp's Bayesian Committee Machine (BCM) (Tresp, 2000) can be considered a PoE-GP model, which provides a consistent framework for combining independent estimators within the Bayesian framework, but it suffers from weak experts.

In this paper, we exploit the fact that the computations of PoE models can be distributed amongst individual computing units and propose the robust BCM (rBMC), a new family of hierarchical PoE-GP models that (i) includes the BCM (Tresp, 2000) and to some degree the generalised PoE-GP (Cao & Fleet, 2014) as special cases, (ii) provides consistent approximations of a full GP, (iii) scales to arbitrarily large data sets by parallelisation. Unlike sparse GPs our rBCM operates on the full data set but distributes the computational and memory load amongst a large set of independent computational units. The rBCM recursively recombines these independent computations to form an efficient distributed GP inference/training framework.

A key advantage of the rBCM is that all computations can be performed analytically, i.e., no sampling is required. With sufficient computing power our model can handle arbitrarily large data sets. We demonstrate that the rBCM can be applied to data sets of size $\mathcal{O}(10^7)$, which exceeds the typical data set sizes sparse GPs deal with by orders of magnitude. However, even with limited resources, our model is practical: A GP with a million data points can be trained in less than half an hour on a laptop.

2. Problem Set-up and Objective

We consider a regression problem $y = f(\boldsymbol{x}) + \epsilon \in \mathbb{R}$, where $\boldsymbol{x} \in \mathbb{R}^D$. The Gaussian likelihood $p(y|f(\boldsymbol{x})) = \mathcal{N}(f(\boldsymbol{x}), \sigma_\epsilon^2)$ accounts for the i.i.d. measurement noise $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$. The objective is to infer the latent function f from a training data set $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^N, \boldsymbol{y} = \{y_i\}_{i=1}^N$. For small data set sizes N, a Gaussian process (GP) is a method of choice for probabilistic non-parametric regression. A GP is defined as a collection of random variables, any finite number of which is Gaussian distributed. A GP is fully specified by a mean function m and a covariance function k (kernel) with hyper-parameters $\boldsymbol{\psi}$. Without loss of generality, we assume that the prior mean function is 0.

A GP is typically trained by finding hyper-parameters $\boldsymbol{\theta} = \{\boldsymbol{\psi}, \sigma_\epsilon\}$ that maximise the log-marginal likelihood

$$\begin{aligned}&\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) \\ &\doteq -\tfrac{1}{2}\left(\boldsymbol{y}^T(\boldsymbol{K} + \sigma_\epsilon^2\boldsymbol{I})^{-1}\boldsymbol{y} + \log|\boldsymbol{K} + \sigma_\varepsilon^2\boldsymbol{I}|\right), \qquad (1)\end{aligned}$$

where $\boldsymbol{K} = k(\boldsymbol{X}, \boldsymbol{X}) \in \mathbb{R}^{N\times N}$ is the kernel matrix.

For a given set of hyper-parameters $\boldsymbol{\theta}$, a training set $\boldsymbol{X}, \boldsymbol{y}$ and a test input $\boldsymbol{x}_* \in \mathbb{R}^D$, the GP posterior predictive distribution of the corresponding function value $f_* = f(\boldsymbol{x}_*)$ is Gaussian with mean and variance given by

$$\mathbb{E}[f_*] = m(\boldsymbol{x}_*) = \boldsymbol{k}_*^T(\boldsymbol{K} + \sigma_\varepsilon^2\boldsymbol{I})^{-1}\boldsymbol{y}\,, \qquad (2)$$

$$\mathrm{var}[f_*] = \sigma^2(\boldsymbol{x}_*) = k_{**} - \boldsymbol{k}_*^T(\boldsymbol{K} + \sigma_\varepsilon^2\boldsymbol{I})^{-1}\boldsymbol{k}_*\,, \qquad (3)$$

respectively, where $\boldsymbol{k}_* = k(\boldsymbol{X}, \boldsymbol{x}_*)$ and $k_{**} = k(\boldsymbol{x}_*, \boldsymbol{x}_*)$.

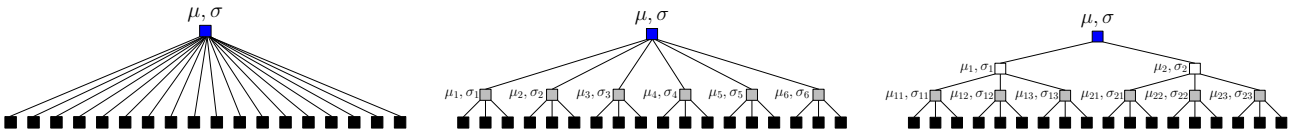
Training requires the inversion and the determinant of $\boldsymbol{K} + \sigma_\epsilon^2\boldsymbol{I}$ in (1), both of which scale in $\mathcal{O}(N^3)$ with a standard implementation. For predictions, we cache $(\boldsymbol{K} + \sigma_\epsilon^2\boldsymbol{I})^{-1}$, such that the mean and variance in (2) and (3) require $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$ computations, respectively. For $N > 10,000$ training and predicting become time-consuming procedures, which additionally require $\mathcal{O}(N^2 + ND)$ memory.

Throughout this paper, we assume that a standard GP is a good model for the latent function f. However, due to the data set size N the full GP is not applicable.

To scale GPs to large data sets with $N \gg \mathcal{O}(10^4)$, we address both the computational and the memory issues of full GPs by distributing the computational and memory loads to many independent computational units that only operate on subsets of the data. For this purpose, we devise a robust and scalable hierarchical product-of-GP-experts model.

3. Distributed Product-of-GP-Experts Models

Product-of-experts models (PoEs) are generally promising for parallelisation and distributed computing. In a PoE model, an overall computation is the product of many independent (smaller) computations, performed by "experts". In our case, every expert is a GP that accesses only a subset of the training data. In this paper, we consider a GP with a training data set $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{y}\}$. We partition the training



(a) 1-layer model. (b) 2-layer model. (c) 3-layer model.

Figure 1. Computational graphs of hierarchical PoE models. Main computations are at the leaf nodes (GP experts, black). All other nodes recombine computations from their direct children. The top node (blue) computes the overall prediction.

data into M sets $\mathcal{D}^{(k)} = \{\boldsymbol{X}^{(k)}, \boldsymbol{y}^{(k)}\}$, $k = 1, \ldots, M$, and use a GP on each of them as a (local) expert[1]. Each GP expert performs computations (e.g., mean/variance predictions) conditioned on their respective training data $\mathcal{D}^{(k)}$. These (local) predictions are recombined by a parent node (see Fig. 1(a)), which subsequently may play the role of an expert at the next level of the model architecture. Recursive application of these recombinations results in a multi-layered tree-structured computational graph, see Fig. 1(c).

The assumption of independent GP experts leads to a block-diagonal approximation of the kernel matrix, which (i) allows for efficient training and predicting (ii) can be computed efficiently (time and memory) by parallelisation.

3.1. Training

Due to the independence assumption, the marginal likelihood $p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})$ in a PoE model factorises into the product of M individual terms, such that

$$p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) \approx \prod_{k=1}^{M} p_k(\boldsymbol{y}^{(k)}|\boldsymbol{X}^{(k)}, \boldsymbol{\theta}) \,, \tag{4}$$

where each factor p_k is determined by the kth GP expert. For training the PoE model, we seek GP hyper-parameters $\boldsymbol{\theta}$ that maximise the corresponding log-marginal likelihood

$$\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) \approx \sum_{k=1}^{M} \log p_k(\boldsymbol{y}^{(k)}|\boldsymbol{X}^{(k)}, \boldsymbol{\theta}) \tag{5}$$

where M is the number of GP experts. The terms in (5) are independently computed and given by

$$\begin{aligned} \log p(\boldsymbol{y}^{(k)}|\boldsymbol{X}^{(k)}, \boldsymbol{\theta}) = &-\tfrac{1}{2}\boldsymbol{y}^{(k)}(\boldsymbol{K}_{\boldsymbol{\psi}}^{(k)} + \sigma_\epsilon^2 \boldsymbol{I})^{-1}\boldsymbol{y}^{(k)} \\ &- \tfrac{1}{2}\log|\boldsymbol{K}_{\boldsymbol{\psi}}^{(k)} + \sigma_\epsilon^2 \boldsymbol{I}| + \text{const} \,, \end{aligned} \tag{6}$$

where $\boldsymbol{K}_{\boldsymbol{\psi}}^{(k)} = k(\boldsymbol{X}^{(k)}, \boldsymbol{X}^{(k)})$ is an $n_k \times n_k$ matrix, and $n_k \ll N$ is the size of the data set associated with the kth GP expert. Since we assume that a standard GP is sufficient to model the latent function, all GP experts at the leaves of the tree-structured model are trained jointly and share a single set of hyper-parameters $\boldsymbol{\theta} = \{\boldsymbol{\psi}, \sigma_\epsilon\}$. Computing the log-marginal likelihood terms in (6) requires the inversion and determinant of $\boldsymbol{K}_{\boldsymbol{\psi}}^{(k)} + \sigma_\epsilon^2 \boldsymbol{I}$. These computations require $\mathcal{O}(n_k^3)$ time with a standard implementation. The memory consumption is $\mathcal{O}(n_k^2 + n_k D)$ for each GP expert.

In (5), the number of parameters $\boldsymbol{\theta}$ to be optimised is relatively small since we do not consider additional variational parameters or inducing inputs that we optimise. Training (i) allows for straightforward parallelisation, (ii) provides a significant speed-up compared to training a full GP, (iii) requires solving low-dimensional $\mathcal{O}(D)$ optimisation problem unlike sparse GPs, which additionally optimise inducing inputs or variational parameters.

3.2. Predictions

In the following, we assume that a set of M GP experts has been trained according to Section 3.1 and detail how the PoE (Ng & Deisenroth, 2014), the generalised PoE (Cao & Fleet, 2014) and the Bayesian Committee Machine (Tresp, 2000) combine predictions of the GP experts to form an overall prediction. Furthermore, we highlight strengths and weaknesses of these models, which motivates our robust Bayesian Committee Machine (rBCM). The rBCM unifies many other models while providing additional flexibility, which can address the shortcomings of the PoE, gPoE and the BCM. For illustration purposes, we focus on the model in Fig. 1(a), but many models generalise to an arbitrarily deep computational graph (see Section 4).

3.2.1. Product of GP Experts

The product-of-GP-experts model predicts a function value f_* at a corresponding test input $\boldsymbol{x}_*$ according to

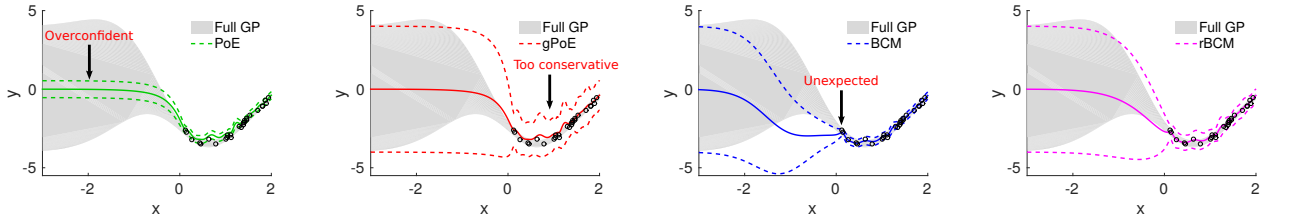
$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) = \prod_{k=1}^{M} p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)}) \,, \tag{7}$$

where M GP experts operate on different training data subsets $\mathcal{D}^{(k)}$. The M GP experts predict means $\mu_k(\boldsymbol{x}_*)$ and variances $\sigma_k^2(\boldsymbol{x}_*)$, $k = 1, \ldots, M$, independently. The joint prediction $p(f_*|\boldsymbol{x}_*, \mathcal{D})$ is obtained by the product of all experts' predictions. The product of these Gaussian predictions is proportional to a Gaussian with mean and precision

$$\mu_*^{\text{poe}} = (\sigma_*^{\text{poe}})^2 \sum\nolimits_k \sigma_k^{-2}(\boldsymbol{x}_*)\mu_k(\boldsymbol{x}_*) \,, \tag{8}$$

$$(\sigma_*^{\text{poe}})^{-2} = \sum\nolimits_k \sigma_k^{-2}(\boldsymbol{x}_*) \,, \tag{9}$$

[1] The notion of "locality" is misleading as our model does not require similarity measures induced by stationary kernels.



(a) Product of GP Experts. (b) Generalised Product of GP Experts. (c) Bayesian Committee Machine. (d) Robust Bayesian Committee Machine.

Figure 2. Predictions with four different product-of-experts models. The full GP to be approximated is shown by the shaded area, representing 95% of the GP confidence intervals. Training data is shown as black circles. Each GP expert was assigned two data points. (a): The standard PoE model does not fall back to the prior when leaving the training data. (b): The generalised PoE model falls back to the prior, but over-estimates the variances in the regime of the training data. (c): The BCM is close to the full GP in the range of the training data, but the predictive mean can suffer. (d): The rBCM is more robust to weak experts than the (g)PoE and the BCM and produces reasonable predictions.

respectively. For $k = 1$, this model is identical to the full GP we wish to approximate.

Strengths of the PoE model are (a) the overall prediction $p(f_*|\boldsymbol{x}_*, \mathcal{D})$ is straightforward to compute, (b) there are no free weight parameters to be assigned to each prediction (unlike in MoE models). A shortcoming of this model is that with an increasing number of GP experts the predictive variances vanish (the precisions add up, see (9)), which leads to overconfident predictions, especially in regions without data. Thus, the PoE model is inconsistent in the sense that it does not fall back to the prior, see Fig. 2(a).

3.2.2. Generalised Product of GP Experts

The generalised product-of-experts model (gPoE) by Cao & Fleet (2014) adds the flexibility of increasing/reducing the importance of experts. The predictive distribution is

$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) = \prod_{k=1}^{M} p_k^{\beta_k}(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)}) \,, \qquad (10)$$

where the β_k weight the contributions of the experts. The predictive mean and precision are, therefore,

$$\mu_*^{\text{gpoe}} = (\sigma_*^{\text{gpoe}})^2 \sum\nolimits_k \beta_k \sigma_k^{-2}(\boldsymbol{x}_*)\mu_k(\boldsymbol{x}_*) \,, \qquad (11)$$

$$(\sigma_*^{\text{gpoe}})^{-2} = \sum\nolimits_k \beta_k \sigma_k^{-2}(\boldsymbol{x}_*) \,, \qquad (12)$$

respectively. A strength of the gPoE is that with $\sum_k \beta_k = 1$ the model falls back to the prior outside the range of the data (and corresponds to a log-opinion-pool model (Heskes, 1998)). A weakness of the gPoE is that in the range of the data, it over-estimates the variances, i.e., the predictions are generally too conservative, especially with an increasing number of GP experts. Fig. 2(b) illustrates these two properties.

Cao & Fleet (2014) suggest to set β_k to the difference in the differential entropy between the prior and the posterior to determine the importance. In this paper, we do not consider this setting for the gPoE for two reasons: (i) $\sum_k \beta_k \neq 1$ leads to unreasonable error bars; (ii) Even with a normalisation, this setting does not allow for deep computational graphs. In fact, it only allows for a single-layer computational graph, such as Fig. 1(a). To allow for a general computational graph, we require an a-priori setting of the β_k. Thus, we set $\beta_k = 1/M$, where M is the number of GP experts. In this case, the predicted means in (8) and (11) are identical, but the precisions differ, see also Fig. 2(a) and 2(b) for a comparison.

3.2.3. Bayesian Committee Machine

A third model that falls in the category of PoE models is the Bayesian Committee Machine (BCM) proposed by Tresp (2000). Unlike the (g)PoE, the BCM explicitly incorporates the GP prior $p(f)$ when combining predictions (and not only at the leaves).

For two experts j, k and corresponding training data sets $\mathcal{D}^{(j)}, \mathcal{D}^{(k)}$, the predictive distribution is generally given by

$$p(f_*|\mathcal{D}^{(j)}, \mathcal{D}^{(k)}) \propto p(\mathcal{D}^{(j)}, \mathcal{D}^{(k)}|f_*)p(f_*) \,, \qquad (13)$$

where $p(f_*)$ is the GP prior over functions. The BCM makes the conditional independence assumption that $\mathcal{D}^{(j)} \perp\!\!\!\perp \mathcal{D}^{(k)}|f_*$. With (13) this yields

$$p(f_*|\mathcal{D}^{(j)}, \mathcal{D}^{(k)}) \overset{\text{BCM}}{\propto} p(\mathcal{D}^{(k)}|f_*)p(\mathcal{D}^{(j)}|f_*)p(f_*) \qquad (14)$$

$$= \frac{p(\mathcal{D}^{(k)}, f_*)p(\mathcal{D}^{(j)}, f_*)}{p(f_*)} \qquad (15)$$

$$\propto \frac{p_k(f_*|\mathcal{D}^{(k)})p_j(f_*|\mathcal{D}^{(j)})}{p(f_*)} \,, \qquad (16)$$

which is the PoE model in (7) divided by the GP prior. For M training data sets $\mathcal{D}^{(k)}$, $k = 1, \ldots, M$, the BCM applies the above approximation repeatedly, leading to the BCM's

posterior predictive distribution

$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) = \frac{\prod_{k=1}^M p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)})}{p^{M-1}(f_*|\boldsymbol{x}_*)}. \quad (17)$$

The $(M-1)$-fold division by the prior is the decisive difference between the BCM and the PoE model in (7) and leads to the BCM's predictive mean and precision

$$\mu_*^{\text{bcm}} = (\sigma_*^{\text{bcm}})^2 \sum_{k=1}^M \sigma_k^{-2}(\boldsymbol{x}_*)\mu_k(\boldsymbol{x}_*) \quad (18)$$

$$(\sigma_*^{\text{bcm}})^{-2} = \sum_{k=1}^M \sigma_k^{-2}(\boldsymbol{x}_*) + (1-M)\sigma_{**}^{-2}, \quad (19)$$

respectively, where σ_{**}^{-2} is the prior precision of $p(f_*)$.

The repeated application of Bayes's theorem leads to an $(M-1)$-fold division by the prior in (17), which plays the role of a "correction" term in (19) that ensures a consistent model that falls back to the prior.[2] The error bars of the BCM within the range of the data are usually good, but it is possible to "break" the BCM when only few data points are assigned to each GP expert. In Fig. 2(c), we see that the posterior mean suffers from weak experts when leaving the data (around $x = 0$).[3]

3.2.4. Robust Bayesian Committee Machine

In this section, we propose the robust Bayesian Committee Machine (rBCM), a unified model that (a) includes the gPoE and BCM as special cases, (b) yields consistent predictions, (c) can be implemented on a distributed computing architecture.

Inspired by the gPoE in (10), the rBCM is a BCM with the added flexibility of increasing/decreasing an expert's importance. The rBCM's predictive distribution is

$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) = \frac{\prod_{k=1}^M p_k^{\beta_k}(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)})}{p^{-1+\sum_k \beta_k}(f_*|\boldsymbol{x}_*)}, \quad (20)$$

where the predictive mean and precision are given as

$$\mu_*^{\text{rbcm}} = (\sigma_*^{\text{rbcm}})^2 \sum_k \beta_k \sigma_k^{-2}(\boldsymbol{x}_*)\mu_k(\boldsymbol{x}_*) \quad (21)$$

$$(\sigma_*^{\text{rbcm}})^{-2} = \sum_{k=1}^M \beta_k \sigma_k^{-2}(\boldsymbol{x}_*) + (1 - \sum_{k=1}^M \beta_k)\sigma_{**}^{-2}, \quad (22)$$

respectively. The derivation of the rBCM in (20) is analogous to the BCM's derivation in (14)–(16).

The rBCM combines the flexibility of the generalised PoE with the appropriate Bayesian treatment of the BCM, which leads to the correction term $(1 - \sum_k \beta_k)\sigma_{**}^{-2}$ in the the precision in (22). This correction term ensures that the predictive variance falls back to the prior when leaving the data. Note that we no longer require $\sum_k \beta_k = 1$ to ensure this (see also (Bordley, 1982)), which will facilitate computational graphs with multiple layers. The gPoE from Section 3.2.2 and the BCM from Section 3.2.3 are recovered for $\beta_k = 1/M$ and $\beta_k = 1$, respectively. For $\beta_k = 0$, the rBCM is identical to the GP prior, and for $\beta_k = 1$ but without the correction, the rBCM recovers the PoE from Section 3.2.1

The parameters β_k control not only the importance of the individual experts, but they also control how strong the influence of the prior is. Assuming each GP expert is a good predictive model, we would set $\beta_k = 1$ for all k, such that we retain the BCM. If the quality of the GP experts is weak, e.g., data is noisy and the experts' data sets $\mathcal{D}^{(k)}$ are small, β_k allows us to weaken the experts' votes and to robustify the predictive model by putting (relatively) more weight on the prior. Therefore, we follow the suggestion by Cao & Fleet (2014) and choose β_k according to the predictive power of each expert at $\boldsymbol{x}_*$. Specifically, we use the difference in differential entropy between the prior $p(f_*|\boldsymbol{x}_*)$ and the posterior $p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)})$. This quantity can be computed efficiently and is given as $\beta_k = \frac{1}{2}(\log \sigma_{**}^2 - \log \sigma_k^2(\boldsymbol{x}_*))$, where σ_{**}^2 is the prior variance and $\sigma_k^2(\boldsymbol{x}_*)$ is the predictive variance of the kth expert.

Fig. 2(d) shows that for this choice of β_k the rBCM expresses more uncertainty about the learned model than the BCM: Due to the adaptive influence of the prior in (21)–(22), the variances within the range of the data (black circles) are slightly larger, but the predictive mean no longer suffers from the dominant "kink" at around $x = 0$ compared to the BCM in Fig. 2(c). Overall, the rBCM provides more reasonable predictions than any other model in Fig. 2.

4. Distributed Computations

In the following, we show that for a given number M of GP experts, the rBCM can be implemented in different computational graphs while providing identical predictions. For instance, with 32 experts, we show that a single-layer computational graph with 32 experts and one central node, see Fig. 1(a), is equivalent to a two-layer computational graph with 32 experts, 8 parent nodes (each of which is responsible for 4 GP experts), and one central node. This property can be exploited in distributed systems or to balance the communication between computing units.

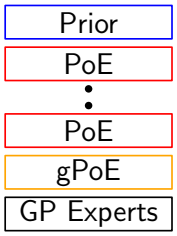


Figure 3. General architecture for the rBCM with arbitrarily many layers.

In a single-layer model as shown in Fig. 1(a) the rBCM

[2] The BCM predictions fall into the category of normalised group odds (Bordley, 1982).

[3] Here, each expert was assigned only two data points.

predictions in (20) can be constructed by a gPoE (numerator) combining predictions of GP experts, followed by a correction via the prior (denominator). Let us consider a two-layer model with M GP experts, L nodes at level 1 and one central node at level 2, see Fig. 1(b). Each grey node at level 1 is responsible for L_k GP experts (black nodes). All M GP experts k_i, compute weighted predictive distributions $p_{k_i}^{\beta_{k_i}}(f_*|\mathcal{D}^{(k_i)})$. These predictions are then multiplied by the corresponding L parent nodes (grey) to $p_k^{\beta_k}(f_*|\mathcal{D}^{(k)})$, where $\beta_k = \sum_i \beta_{k_i}$ is the overall weight of the subtree following the kth node L_k and $\mathcal{D}^{(k)} = \bigcup_i \mathcal{D}^{(k_i)}$. The overall prediction at the top node (blue in Fig. 1) is

$$\frac{\prod_{k=1}^{M} p_k^{\beta_k}(f_*|\mathcal{D}^{(k)})}{p^{\sum_k \beta_k - 1}(f_*)} = \frac{\prod_{k=1}^{L} \prod_{i=1}^{L_k} p_{k_i}^{\beta_{k_i}}(f_*|\mathcal{D}^{(k_i)})}{p^{\sum_k \beta_k - 1}(f_*)} \quad (23)$$

where we accounted for the $(\sum_k \beta_k - 1)$-fold correction by the prior. This computation can be obtained by a gPoE (black), followed by a PoE (red) and a prior correction (blue) in (23). Hence, for a given number of GP experts, the rBCM predictions can be equivalently realised in a single and two-layer computational graph, see Fig. 1.

This can be generalised further to an arbitrarily deep computational graph, whose general implementation structure is shown in Fig. 3. The GP experts at the leaves compute their individual means, variances and confidence values β_i. The next layer consists of gPoE models, which compute the weighted means and variances according to (8) and (12), respectively (plus their overall weights $\beta_k = \sum_{i \in \text{children}} \beta_{k_i}$, which are passed on to the next-higher level). The gPoE is followed by an arbitrary number of PoE models, which compute means and precisions according to (8) and (9), respectively (plus their overall weights $\beta_k = \sum_{j \in \text{children}} \beta_{k_j}$, which are passed on to the next-higher level). The top layer accounts for the prior (blue term in (23)), which uses all the β_k from the subtrees starting at its children to compute the overall mean and precision according to (21) and (22), respectively.

Hence, for a given number of GP experts, there are many equivalent computational graphs for the rBCM (and the other distributed GPs discussed in Section 3). For instance, all computational graphs in Fig. 1 return identical predictive distributions. This allows us to choose the computational graph that works best with the computing infrastructure available: Shallow graphs minimise overall traffic. However, they are more vulnerable to communication bottlenecks at the central node since it has a large number of connections. Deeper computational graphs cause more overall communication, but the tree has a smaller branching factor. In practice, for a set of computational graphs we took the time it requires to compute the gradient of the marginal likelihood and chose the "fastest" architecture.

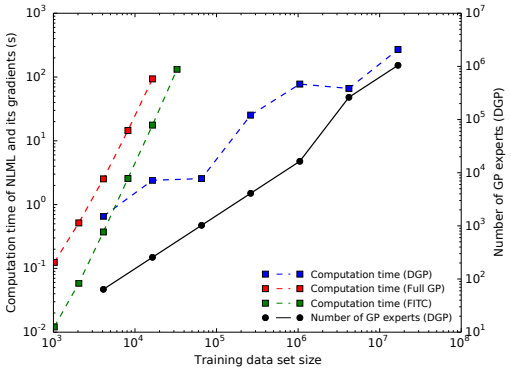


Figure 4. Computation time for the log-marginal likelihood and its gradient with respect to the kernel hyper-parameters as a function of the size of the training data. The distributed GPs (DGP), i.e., (g)PoE and (r)BCM, scale favourably to large-scale data sets.

5. Experiments

We empirically assess three aspects of all distribute GP models: (1) The required training time, (2) the approximation quality, (3) a comparison with state-of-the-art sparse GP methods. In all experiments, we chose the standard squared exponential kernel with automatic relevance determination and a Gaussian likelihood. Moreover, we assigned training data to experts *randomly* for two reasons: First, we demonstrate that our models do not need locality information; second, random assignment is very fast compared to clustering methods, e.g., KD-trees.

5.1. Training Time for Large Data Sets

To evaluate the training time for distributed GPs (DGP)[4], we measured the time required to compute the log-marginal likelihood and its gradient with respect to the kernel hyper-parameters. Since the model is trained using LBFGS, the overall training time is proportional to the time it takes to compute the log-marginal likelihood and its gradient. For this evaluation, we chose a computer architecture of 64 nodes with four cores each. Furthermore, we chose a three-layer computational graph with varying branching factors. For data sets of $\leq 2^{20}$ data points each GP expert possessed 512 data points, for data set sizes of $> 2^{20}$, we chose the number of data points per node to be 128.

Fig. 4 shows the time required for computing the,log-marginal likelihood and its gradient with respect to the hyper-parameters. The horizontal axis shows the size of the training set (logarithmic scale), the left vertical axis shows the computation time in seconds (logarithmic scale) for the DGP (blue-dashed), a full GP (red-dashed) and a sparse GP (FITC) with inducing inputs (Snelson & Ghahramani, 2006) (green-dashed). For the sparse GP model, we chose

[4]This comprises the (g)PoE and (r)BCM for which the training time is identical.

the number M of inducing inputs to be 10% of the size of the training set, i.e., the computation time is of the order of $\mathcal{O}(NM^2) = \mathcal{O}(N^3/100)$, which offsets the curve of the full GP. The right vertical axis shows the number of GP experts (black-solid) amongst which we distribute the computation. While the training time of the full GP becomes impractical at data set sizes of about 20,000, the sparse GP model can be reasonably trained up to 50,000 data points.[5] The computational time required for the DGP to compute the marginal likelihood and gradients is significantly lower than that of the full GP, and we scaled it up to $2^{24} \approx 1.7 \times 10^7$ training data points, which required about the same amount of time ($\approx$ 230 s) for training a full GP with $2^{14} \approx 1.6 \times 10^4$ and a sparse GP with $2^{15} \approx 3.2 \times 10^4$ data points. The figure shows that for any problem size we can find a computational graph that allows us to train the model within a reasonable amount of time.

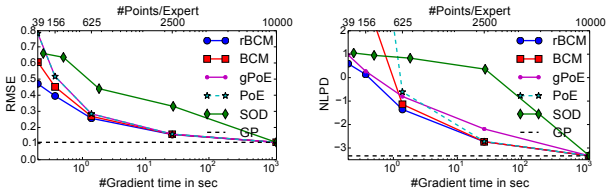
Even if a big computing infrastructure is not available our model is useful in practice: We performed a full training cycle of the DGP with 10^6 data points on a standard laptop in about 20 minutes. This is also a clear indicator that the memory consumption of the DGP is relatively small.

5.2. Empirical Approximation Errors

We evaluate the predictive quality of the DGP models introduced in Section 3 on the Kin40K data set. The data set represents the forward dynamics of an 8-link all-revolute robot arm. The goal is to predict the distance of the end-effector from a target, given the joint angles of the eight links as features. The Kin40K data set consists of 10,000 training points and 30,000 test points. We use the same split into training and test data as Seeger et al. (2003), Lázaro-Gredilla et al. (2010), and Nguyen & Bonilla (2014).

We considered two baselines: a full (ground truth) GP and the subset-of-data (SOD) approximation, which uses a random subset of the full training data set to train a sparse GP. Taking training time into account, Chalupka et al. (2013) identified the SOD method as a good and robust sparse GP approximation. All approximate models (PoE, gPoE, BCM, rBCM, SOD) used the hyper-parameters from the full GP to eliminate issues with local optima. For every model, we took the time for computing the gradient of the marginal likelihood (training is proportional to this amount of time). We selected the number of data points for SOD, such that the gradient computation time approximately matches the one of the DGPs.

Experiments were repeated 10 times to average out the effect of the random assignment of data points to experts and the selection of the subset of training data for the SOD approximation. For this experiment, we used a Virtual Machine with 16 3 GHz cores and 8 GB RAM.



(a) RMSE. (b) NLPD.

Figure 5. Performance as a function of training time (lower horizontal axes) and the number of data points per expert (upper horizontal axes). (a) RMSE, (b) NLPD. Standard errors (not displayed) are smaller than 10^{-2}.

Fig. 5 shows the average performance (RMSE and negative log predictive density (NLPD) per data point) of all models as a function of (a) the time to compute the gradient of the marginal likelihood and (b) the number of training data points per GP expert. The full GP (black, dashed) shows the ground-truth performance, but requires 1162 seconds for the gradient computation. The performances of the (r)BCM and the (g)POE have been evaluated using 256, 64, 16, 4, and 1 expert with 39, 156, 625, 2500 and 10000 data points per expert, respectively. In Fig. 5(a), the rBCM consistently outperforms all other methods, where SOD is substantially worse than all DGPs. The NLPD in Fig. 5(b) allows us to make some more conclusive statements: While the rBCM again outperforms all other methods, the BCM and the PoE's performances suffer when only a small number of data points is assigned to the GP experts. The PoE suffers from variance underestimation (see Fig. 2(a)) whereas the BCM suffers from "weak" experts (see Fig. 2(c)). SOD does not work well, even with 2500 data points. Overall, the rBCM provides an enormous training speed-up compared to a full GP, with a significantly better predictive performance than SOD.

5.3. Airline Delays (US Flight Data)

We assess the performance of the distributed GPs (PoE, gPoE, BCM and rBCM) on a large-scale non-stationary data set reporting flight arrival and departure times for every commercial fight in the US from January to April 2008[6]. This data set contains information about almost 6 million flights. We followed the procedure described by Hensman et al. (2013)[7] to predict the flight delay (in minutes) at arrival: We selected the first P data points to train the model and the following 100,000 to test it. We chose the same eight input variables $\boldsymbol{x}$ as Hensman et al. (2013): age of the aircraft, distance that needs to be covered, airtime,

[5] We did not include any computational overhead for learning the inducing inputs, which is often time consuming.

[6] http://stat-computing.org/dataexpo/2009/

[7] Thanks to J Hensman for the pre-processing script.

Table 1. US Flight Data Set. SVIGP and Dist-VGP results are reported from the respective papers. Results with $^+$ are the best solution found. Good and bad performances are highlighted in blue and red, respectively.

	700K/100K		2M/100K		5M/100K	
	RMSE	NLPD	RMSE	NLPD	RMSE	NLPD
SVIGP	**33.0**	—	—	—	—	—
Dist-VGP$^+$	**33.0**	—	35.3	—	—	—
rBCM	**27.1**	9.1	**34.4**	**8.4**	**35.5**	**8.8**
BCM	**33.5**	**14.7**	**55.8**	**17.0**	**55.3**	**19.5**
gPoE	28.7	**8.1**	35.5	**8.6**	37.3	**8.7**
PoE	28.7	**14.1**	35.5	**26.3**	37.3	**17.7**
SOD	$> 10^3$	**22.6**	$> 10^3$	**17.2**	$> 10^3$	**16.4**

departure and arrival times, day of the week and month, month. This data set has been evaluated by Hensman et al. (2013) and Gal et al. (2014), who use either Stochastic Variational Inference GP (SVIGP) or Distributed Variational GPs (Dist-VGP) to deal with this training set size.

Using a workstation with 12 3.5 GHz cores and 32 GB RAM, we conducted 10 experiments with $P = 7 \times 10^5$, $P = 2 \times 10^6$ and $P = 5 \times 10^6$ where we chose 4096, 8192 and 32768 experts corresponding to 170, 244 and 152 training data points per expert, respectively. The computation of the marginal likelihoods required 13, 39, and 90 seconds, respectively.[8] The computational graphs were (16-16-16), (8192), and (32-32-32), respectively, where each number denotes the branching factor at the corresponding level in the tree. After normalising the data every single experiment consisted of an independent full training/test cycle, with random assignments of data points to GP experts. Training the DGPs normally required 30–100 line searches to converge. Table 1 reports the performance (RMSE and NLPD) of various large-scale GP methods for the flight data set. The results for SVIGP and Dist-VGP are taken from Hensman et al. (2013) and Gal et al. (2014), respectively. Since SVIGP and Dist-VGP are difficult to optimise due to the large number of variational parameters, Gal et al. (2014) report only their best results, whereas we report an average of *all* experiments conducted.

The standard errors (not shown in Table 1) of the rBCM and gPoE are consistently below 0.3, whereas the BCM and the PoE suffered from a few outliers, which is also indicated by the relatively large NLPD values. Compared to the Dist-VGP and SVIGP on the 700K data set, the rBCM, gPoE and PoE perform significantly better in RMSE. The table highlights the weaknesses of the PoE (under-estimation of the variance) and the BCM (problems with weak experts) very clearly. The property of the gPoE (too conservative) is a bit hidden: Although the RMSE of the gPoE is consistently worse than that of the rBCM, its NLPD tends to be a bit lower. The NLPD values of the rBCM and the gPoE are fairly consistent across all three experiments.

The data set exhibits the property that the 700K/100K data set is more stationary than the 2M/100K and 5M/100K data sets. Therefore, we observe a decreasing performance although we include more training data. This effect has already been reported by Gal et al. (2014).

[8] All experiments can be conducted on a MacBook Air (2012) with 8 GB RAM.

6. Discussion and Conclusion

The distributed GP models discussed in this paper exhibit the appealing property that they are conceptually simple: They split the data set into small pieces, train GP experts jointly, and subsequently combine individual predictions to an overall computation. Compared to sparse GP models, which are based on inducing or variational parameters, the distributed GPs possess only the normal GP hyper-parameters, i.e., it is less likely to end up in local optima.

In this paper, all experts share the same hyper-parameters, which leads to automatic regularisation: The overall gradient is an average of the experts' marginal likelihood gradients, i.e., overfitting of individual experts is not favoured.

We believe that distributed computations are promising to scale GPs to truly large data sets. Sparse methods using inducing inputs are naturally limited by the number of inducing variables. In practice, we see $\mathcal{O}(10^2)$ many inducing variables (optimisation is hard), although their theoretical limit might be at $\mathcal{O}(10^4)$, if we assume that this is the limit up to which we can train a full GP. If we assume that a single inducing variable summarises 100 data points, the practical limit of sparse methods are data sets of size $\mathcal{O}(10^6)$. This can be extended by either a higher compression rate or parallelisation (Hensman et al., 2013; Gal et al., 2014).

We introduced the robust Bayesian Committee Machine (rBCM), a conceptually straightforward, but effective, product-of-GP-experts model that scales GPs to (in principle) arbitrarily large data sets. The rBCM distributes computations amongst independent computing units, allowing for straightforward parallelisation. Recursive and closed-form recombinations of these independent computations yield a practical model that is both computationally and memory efficient. The rBCM addresses shortcomings of other distributed models by appropriately incorporating the GP prior when combining predictions. The rBCM is independent of the computational graph and can be equivalently used on heterogeneous computing infrastructures, ranging from laptops to large clusters. Training an rBCM with a million data points takes less than 30 minutes on a laptop. With more computing power training the rBCM with 10^7 data points can be done in a few hours. Compared to state-of-the-art sparse GPs, our model is conceptually simple, performs well, learns fast, requires little memory and does not require high-dimensional optimisation.

Acknowledgements

MPD has been supported by an Imperial College Junior Research Fellowship.

References

Bordley, Robert F. A Multiplicative Formula for Aggregating Probability Assessments. *Management Science*, 28 (10):1137–1148, 1982.

Brochu, Eric, Cora, Vlad M., and de Freitas, Nando. A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. Technical Report TR-2009-023, Department of Computer Science, University of British Columbia, 2009. URL http://haikufactory.com/files/bayopt.pdf.

Cao, Yanshuai and Fleet, David J. Generalized Product of Experts for Automatic and Principled Fusion of Gaussian Process Predictions. http://arxiv.org/abs/1410.7827, October 2014.

Chalupka, Krzysztof, Williams, Christopher K. I., and Murray, Iain. A Framework for Evaluating Approximate Methods for Gaussian Process Regression. *Journal of Machine Learning Research*, 14:333–350, February 2013. URL http://jmlr.org/papers/v14/chalupka13a.html.

Cressie, Noel A. C. *Statistics for Spatial Data*. Wiley-Interscience, 1993.

Deisenroth, Marc P., Fox, Dieter, and Rasmussen, Carl E. Gaussian Processes for Data-Efficient Learning in Robotics and Control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):408–423, February 2015. doi: 10.1109/TPAMI.2013.218.

Gal, Yarin, van der Wilk, Mark, and Rasmussen, Carl E. Distributed Variational Inference in Sparse Gaussian Process Regression and Latent Variable Models. In *Advances in Neural Information Processing Systems*. 2014.

Hensman, James, Fusi, Nicolò, and Lawrence, Neil D. Gaussian Processes for Big Data. In Nicholson, A. and Smyth, P. (eds.), *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2013. URL http://auai.org/uai2013/prints/papers/244.pdf.

Heskes, Tom. Selecting Weighting Factors in Logarithmic Opinion Pools. In *Advances in Neural Information Processing Systems*, pp. 266–272. Morgan Kaufman, 1998.

Jacobs, Robert A., Jordan, Michael I., Nowlan, Steven J., and Hinton, Geoffrey E. Adaptive Mixtures of Local Experts. *Neural Computation*, 3:79–87, 1991. URL http://www.cs.toronto.edu/~hinton/absps/jjnh91.pdf.

Jones, Donald R., Schonlau, Matthias, and Welch, William J. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13(4):455–492, December 1998. doi: 10.1023/A:1008306431147.

Krause, Andreas, Singh, Ajit, and Guestrin, Carlos. Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies. *Journal of Machine Learning Research*, 9:235–284, February 2008. URL http://www.jmlr.org/papers/volume9/krause08a/krause08a.pdf.

Lawrence, Neil. Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models. *Journal of Machine Learning Research*, 6:1783–1816, November 2005. URL http://www.jmlr.org/papers/volume6/lawrence05a/lawrence05a.pdf.

Lázaro-Gredilla, Miguel, Quionero-Candela, Joaquin, Rasmussen, Carl E., and Figueiras-Vidal, Aníbal R. Sparse Spectrum Gaussian Process Regression. *Journal of Machine Learning Research*, 11:1865–1881, June 2010. URL http://jmlr.csail.mit.edu/papers/v11/lazaro-gredilla10a.html.

Luttinen, Jaakko and Ilin, Alexander. Efficient Gaussian Process Inference for Short-Scale Spatio-Temporal Modeling. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 22 of *JMLR W&CP*, 2012. URL http://jmlr.org/proceedings/papers/v22/luttinen12/luttinen12.pdf.

Meeds, Edward and Osindero, Simon. An Alternative Infinite Mixture of Gaussian Process Experts. In *Advances in Neural Information Processing Systems*, 2006.

Ng, Jun W. and Deisenroth, Marc P. Hierarchical Mixture-of-Experts Model for Large-Scale Gaussian Process Regression. http://arxiv.org/abs/1412.3078, December 2014.

Nguyen, Trung V. and Bonilla, Edwin V. Fast Allocation of Gaussian Process Experts. In *Proceedings of the International Conference on Machine Learning*, 2014. URL http://jmlr.org/proceedings/papers/v32/nguyena14.pdf.

Quiñonero-Candela, Joaquin and Rasmussen, Carl E. A Unifying View of Sparse Approximate Gaussian Process Regression. *Journal of*

Machine Learning Research, 6(2):1939–1960, 2005. URL http://jmlr.csail.mit.edu/papers/volume6/quinonero-candela05a/quinonero-candela05a.pdf.

Rasmussen, Carl E. and Ghahramani, Zoubin. Infinite Mixtures of Gaussian Process Experts. In *Advances in Neural Information Processing Systems*, 2002.

Rasmussen, Carl E. and Williams, Christopher K. I. *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, MA, USA, 2006. URL http://www.gaussianprocess.org/gpml/chapters/.

Seeger, Matthias, Williams, Christopher K. I., and Lawrence, Neil D. Fast Forward Selection to Speed up Sparse Gaussian Process Regression. In *Ninth International Workshop on Artificial Intelligence and Statistics*. Society for Artificial Intelligence and Statistics, 2003. URL http://eprints.kfupm.edu.sa/40653/.

Shen, Yirong, Ng, Andrew Y., and Seeger, Matthias. Fast Gaussian Process Regression Using KD-Trees. In *Advances in Neural Information Processing Systems*, 2006.

Snelson, Edward and Ghahramani, Zoubin. Sparse Gaussian Processes using Pseudo-Inputs. In *Advances in Neural Information Processing Systems*, 2006. URL http://books.nips.cc/papers/files/nips18/NIPS2005_0543.pdf.

Titsias, Michalis K. Variational Learning of Inducing Variables in Sparse Gaussian Processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2009.

Tresp, Volker. A Bayesian Committee Machine. *Neural Computation*, 12(11):2719–2741, 2000. URL http://www.dbs.ifi.lmu.de/~tresp/papers/bcm6.pdf.

Williams, Christopher K.I. and Seeger, Matthias. Using the Nyström Method to Speed up Kernel Machines. In *Advances in Neural Information Processing Systems*, 2001.

Yuan, Chao and Neubauer, Claus. Variational Mixture of Gaussian Process Experts. In *Advances in Neural Information Processing Systems*, 2009.