Collaborative Multi-output Gaussian Processes

Trung V. Nguyen
ANU & NICTA
Canberra, Australia

Edwin V. Bonilla
NICTA & ANU
Sydney, Australia

Abstract

We introduce the collaborative multi-output Gaussian process (GP) model for learning dependent tasks with very large datasets. The model fosters task correlations by mixing sparse processes and sharing multiple sets of inducing points. This facilitates the application of variational inference and the derivation of an evidence lower bound that decomposes across inputs and outputs. We learn all the parameters of the model in a single stochastic optimization framework that scales to a large number of observations per output and a large number of outputs. We demonstrate our approach on a toy problem, two medium-sized datasets and a large dataset. The model achieves superior performance compared to single output learning and previous multi-output GP models, confirming the benefits of correlating sparsity structure of the outputs via the inducing points.

1 INTRODUCTION

Gaussian process models (GPs, Rasmussen and Williams, 2006) are a popular choice in Bayesian regression due to their ability to capture complex dependencies and non-linearities in data. In particular, when having multiple outputs or tasks they have proved effective in modeling the dependencies between the tasks, outperforming competitive baselines and single output learners (Bonilla et al., 2008; Teh et al., 2005; Alvarez and Lawrence, 2009; Wilson et al., 2012). However, the prohibitive cost of performing exact inference in GP models severely hinders their application to large scale multi-output problems. For example, naïve inference in a fully coupled Gaussian process model over P outputs and N data points can have a complexity of $\mathcal{O}(N^3P^3)$ and $\mathcal{O}(N^2P^2)$ in time and memory, respectively.

A motivating example of a large scale multi-output application is the tracking of movements of a robot arm using 2 or more joint torques. If one of the robot motors malfunctions and fails to record a torque, data collected from the other motors may be used to infer the missing torque values. However, taking 100 measurements per second already results in a total of over 40,000 data points per torque in just 7 minutes. Clearly this problem is well beyond the capabilities of conventional multiple output GPs. Building multi-output GP models that can learn correlated tasks at the scale of these types of problems is thus the main focus of this paper.

In the single output setting previous attempts to scale up GP inference resort to approximate inference. Most approximation methods can be understood within a single probabilistic framework that uses a set of *inducing points* in order to obtain an approximate process (or a low-rank approximate covariance) over which inference can be performed more efficiently (Quiñonero-Candela and Rasmussen, 2005). These models have been referred to in the literature as sparse models. Nevertheless, straightforward application of such approximate techniques will yield a computational cost of at least $\mathcal{O}(PNM^2)$ in time and $\mathcal{O}(PNM)$ in memory, where M is the number of inducing points. This high complexity still prevents us from applying GP models to large scale multi-output problems.

In this work we approach the challenge of building scalable multi-output Gaussian process models based on the following observations. Firstly, inducing variables are the key catalyst for achieving sparsity and dealing with large scale problems in Gaussian process models. In particular, they capture the *sufficient statistics* of a dataset allowing the construction of sparse processes that can approximate arbitrarily well the exact GP model (Titsias, 2009). Secondly, the use of global latent variables (such as the inducing points) allows us to induce dependencies in a highly correlated model efficiently. This observation is exploited in Hensman et al. (2013) for single output GP regression models where, by explicitly representing a distribution over the inducing variables, stochastic variational inference can be used to work with millions of data points. Finally, the key to multi-output and multi-task learning is to model dependencies between the outputs based on realistic assumptions of what can be shared across the tasks. It turns out that sharing "sparsity structure" can not only be a reasonable assumption but also a crucial component when modeling dependencies between different related tasks.

Based on these observations, we propose the collaborative multi-output Gaussian Process (COGP) model where latent processes are mixed to generate dependent outputs. Each process is sparse and characterized by its own set of inducing points. The sparsity structure enabling output correlations is thus created via the shared inducing sets. To learn this structure, we maintain an explicit representation of the posterior over the inducing points which in turn allows inference to be carried out efficiently. In particular, we obtain a variational lower bound of the model evidence that decomposes across inputs and outputs. This decomposition makes possible the application of stochastic variational inference, thus allowing the model to handle a large number of observations per output and a large number of outputs. Furthermore, learning of all the parameters in the model, including kernel hyperparameters and inducing inputs, can be done in a single stochastic optimization framework.

We analyze our multi-out model on a toy problem where the inducing variables are shown to be conducive to the sharing of information between two related tasks. Additionally, we evaluate our model on two moderate-sized datasets in which we show that it can outperform previous non-scalable multi-output approaches as well as single output baselines. Finally, on a large scale experiment regarding the learning of robot inverse dynamics we show the substantial benefits of collaborative learning provided by our model.

Related work. Most GP-based multi-output models create correlated outputs by mixing a set of independent latent processes. The mixing can be a linear combination with fixed coefficients (see e.g. Teh et al., 2005; Bonilla et al., 2008). This is known in the geostatistics community as the "linear model of coregionalization" (Goovaerts, 1997). Such models may also be reformulated in a common Bayesian framework, for example by placing a spike and slab prior over the coefficients (Titsias and Lázaro-Gredilla, 2011). More complex dependencies can be induced by using input-dependent coefficients (Wilson et al., 2012; Nguyen and Bonilla, 2013) or convolving processes (Boyle and Frean, 2005; Alvarez and Lawrence, 2009; Álvarez et al., 2010).

While we also use the mixing construction, the key difference in our model is the role of inducing variables. In particular, when used in previous models to reduce the computational costs (see e.g. Alvarez and Lawrence, 2009; Álvarez et al., 2010), the inducing points are integrated out or collapsed. In contrast, our model maintains an explicit representation of the posterior over the inducing variables that is learned using data from all outputs. This explicit representation facilitates scalable learning in a similar fashion to the approach in Hensman et al. (2013), making it applicable to very large datasets.

2 MODEL SPECIFICATION

Before diving into technical details of the model specification, we discuss the modeling philosophy behind our collaborative multi-output Gaussian processes. To learn the outputs jointly, we need a mechanism through which information can be transferred among the outputs. This is achieved in the model by allowing the outputs to share multiple sets of inducing variables, each of which captures a different pattern common to the outputs. These variables play a double pivotal role in the model: they collaboratively share information across the outputs and provide sufficient statistics so as to induce sparse processes.

Consider the joint regression of P tasks with inputs $\mathbf{X} = \{\mathbf{x}_n \in \mathcal{R}^D\}_{n=1}^N$ and outputs $\mathbf{y} = \{\mathbf{y}_i\}_{i=1}^P$ where $\mathbf{y}_i = \{y_{in}\}_{n=1}^N$. We model each output as a weighted combination of Q shared latent functions $\{g_j\}_{j=1}^Q$, plus an individual latent function $\{h_i\}_{i=1}^P$ unique to that output for greater flexibility. The Q shared functions have independent Gaussian process priors $g_j(\mathbf{x}) \sim \mathcal{GP}(0, k_j(\cdot,\cdot))$. Similarly, each individual function of an output also has a GP prior, i.e. $h_i(\mathbf{x}) \sim \mathcal{GP}(0, k_i^h(\cdot,\cdot))$.

As we want to sparsify these processes, we introduce a set of *shared inducing variables* $\mathbf{u}_j$ for each $g_j(\mathbf{x})$, i.e. $\mathbf{u}_j$ contains the values of $g_j(\mathbf{x})$ at the inducing inputs $\mathbf{Z}_j$. Likewise, we have individual inducing variables corresponding to each $h_i(\mathbf{x})$, which we denote with $\mathbf{v}_i$ and their corresponding inducing inputs $\mathbf{Z}_i^h$. The inducing inputs lie in the same space as the inputs $\mathbf{X}$. For convenience, we assume all processes have the same number of inducing points, M. However we emphasize that this is not imposed in practice.

We denote the collective variables: $\mathbf{g} = \{\mathbf{g}_j\}$, $\mathbf{h} = \{\mathbf{h}_i\}$, $\mathbf{u} = \{\mathbf{u}_j\}$, $\mathbf{v} = \{\mathbf{v}_i\}$, $\mathbf{Z} = \{\mathbf{Z}_j\}$, and $\mathbf{Z}^h = \{\mathbf{Z}_i^h\}$

where $\mathbf{g}_j = \{g_j(\mathbf{x}_n)\}$, $\mathbf{h}_i = \{h_i(\mathbf{x}_n)\}$. Note that we reserve subscript i for indexing the outputs and their corresponding individual processes ($i = 1 \ldots P$), j for the shared latent processes ($j = 1 \ldots Q$), and n for the inputs ($n = 1 \ldots N$).

2.1 PRIOR MODEL

From the definition of the GPs and the independence of the processes, the *prior* of the multi-output model can be written as:

$$p(\mathbf{g}|\mathbf{u}) = \prod_{j=1}^{Q} p(\mathbf{g}_j|\mathbf{u}_j) = \prod_{j=1}^{Q} \mathcal{N}(\mathbf{g}_j; \boldsymbol{\mu}_j, \tilde{\mathbf{K}}_j) \quad (1)$$

$$p(\mathbf{u}) = \prod_{j=1}^{Q} p(\mathbf{u}_j) = \prod_{j=1}^{Q} \mathcal{N}(\mathbf{u}_j; \mathbf{0}, k(\mathbf{Z}_j, \mathbf{Z}_j)) \quad (2)$$

$$p(\mathbf{h}|\mathbf{v}) = \prod_{i=1}^{P} p(\mathbf{h}_i|\mathbf{v}_i) = \prod_{i=1}^{P} \mathcal{N}(\mathbf{h}_i; \boldsymbol{\mu}_i^h, \tilde{\mathbf{K}}_i^h) \quad (3)$$

$$p(\mathbf{v}) = \prod_{i=1}^{P} p(\mathbf{v}_i) = \prod_{i=1}^{P} \mathcal{N}(\mathbf{v}_i; \mathbf{0}, k(\mathbf{Z}_i^h, \mathbf{Z}_i^h)), \quad (4)$$

where the corresponding means and covariances of the Gaussians are given by:

$$\boldsymbol{\mu}_j = k(\mathbf{X}, \mathbf{Z}_j) k(\mathbf{Z}_j, \mathbf{Z}_j)^{-1} \mathbf{u}_j \quad (5)$$

$$\boldsymbol{\mu}_i^h = k(\mathbf{X}, \mathbf{Z}_i^h) k(\mathbf{Z}_i^h, \mathbf{Z}_i^h)^{-1} \mathbf{v}_i \quad (6)$$

$$\tilde{\mathbf{K}}_j = k_j(\mathbf{X}, \mathbf{X}) - k(\mathbf{X}, \mathbf{Z}_j) k(\mathbf{Z}_j, \mathbf{Z}_j)^{-1} k(\mathbf{Z}_j, \mathbf{X}) \quad (7)$$

$$\tilde{\mathbf{K}}_i^h = k_i^h(\mathbf{X}, \mathbf{X}) - k(\mathbf{X}, \mathbf{Z}_i^h) k(\mathbf{Z}_i^h, \mathbf{Z}_i^h)^{-1} k(\mathbf{Z}_i^h, \mathbf{X}). \quad (8)$$

In the equations and hereafter, we omit the subscripts j, h, i from the kernels $k_j(\cdot, \cdot)$ and $k_i^h(\cdot, \cdot)$ when it is clear from the parameters inside the parentheses which covariance function is in action.

Equations (2) and (4) follow directly from the properties of GPs, while the expressions for $p(\mathbf{g}|\mathbf{u})$ and $p(\mathbf{h}|\mathbf{v})$ (Equations (1) and (3)) come from the conditionals of the multivariate Gaussian distributions. Instead of writing the joint priors $p(\mathbf{g}, \mathbf{u})$ and $p(\mathbf{h}, \mathbf{v})$, the above equivalent equations are given to emphasize the sufficient statistics role of $\mathbf{u}$ and $\mathbf{v}$ in the model. Here by sufficient statistics we mean, for any sparse process (say g_j), any other set of function values is independent of $\mathbf{g}_j$ given the inducing variables $\mathbf{u}_j$.

2.2 LIKELIHOOD MODEL

As mentioned above, we assume that observations for each output are (noisy) linear combinations of the Q latent functions $g_j(\mathbf{x})$ plus an independent function $h_i(\mathbf{x})$. Hence we have that the likelihood with standard iid Gaussian noise is given by:

$$p(\mathbf{y}|\mathbf{g}, \mathbf{h}) = \prod_{i=1}^{P} \prod_{n=1}^{N} \mathcal{N}(y_{in}; \sum_{j=1}^{Q} w_{ij} g_j(\mathbf{x}_n) + h_i(\mathbf{x}_n), \beta_i^{-1}), \quad (9)$$

where w_{ij} are the corresponding weights and β_i is the precision of each Gaussian. As the latent values $\mathbf{g}$ are specified conditioned on the inducing variables $\mathbf{u}$, this construction implies that each output is a weighted combination of the inducing values. We note that if $\mathbf{u}$ and $\mathbf{v}$ are marginalized out, we obtain the semiparametric latent factor model (Teh et al., 2005). However, doing so is against the purpose of our model which encourages sharing of outputs via the inducing variables. Furthermore, as we shall see in the next section, explicit representation of these variables is fundamental to scalable inference of the model.

3 INFERENCE

We approximate the posterior over the latent variables $\mathbf{g}, \mathbf{h}, \mathbf{u}, \mathbf{v}$ given observations $\mathbf{y}$ using variational inference (Jordan et al., 1999). In section 3.1 we derive a lower bound of the marginal likelihood which has the key property of factorizing over the data points and outputs. Section 3.2 takes advantage of this factorization to derive stochastic variational inference, allowing the model to scale to very large datasets. Section 3.3 compares the complexity of the model with previous multi-output methods.

3.1 VARIATIONAL LOWER BOUND

In variational inference, we find the "closest" approximate distribution to the true posterior in terms of the KL divergence. We first observe that the true posterior distribution can be written as:

$$p(\mathbf{g}, \mathbf{h}, \mathbf{u}, \mathbf{v}|\mathbf{y}) = p(\mathbf{g}|\mathbf{u}, \mathbf{y}) p(\mathbf{h}|\mathbf{v}, \mathbf{y}) p(\mathbf{u}, \mathbf{v}|\mathbf{y}). \quad (10)$$

Here we recall the modeling assumption that each set of inducing variables is the sufficient statistics of the corresponding latent process. This motivates replacing the true posteriors over $\mathbf{g}$ and $\mathbf{h}$ with their conditional distributions given the inducing variables, leading to a distribution of the form:

$$q(\mathbf{g}, \mathbf{h}, \mathbf{u}, \mathbf{v}|\mathbf{y}) = p(\mathbf{g}|\mathbf{u}) p(\mathbf{h}|\mathbf{v}) q(\mathbf{u}, \mathbf{v}), \quad (11)$$

with

$$q(\mathbf{u}, \mathbf{v}) = \prod_{j=1}^{Q} \underbrace{\mathcal{N}(\mathbf{u}_j; \mathbf{m}_j, \mathbf{S}_j)}_{q(\mathbf{u}_j)} \prod_{i=1}^{P} \underbrace{\mathcal{N}(\mathbf{v}_i; \mathbf{m}_i^h, \mathbf{S}_i^h)}_{q(\mathbf{v}_i)}. \quad (12)$$

This technique has been used by Titsias (2009) and Hensman et al. (2013) to derive variational inference algorithms for the single output case. Since the conditionals $p(\mathbf{g}|\mathbf{u})$ and $p(\mathbf{h}|\mathbf{v})$ are known (Equations (1) and (3)), we only need to learn $q(\mathbf{u}, \mathbf{v})$ so as to minimize the divergence between the approximate posterior and the true posteriors. The quality of approximation depends entirely on the posterior over the inducing variables, thus underlining their pivotal role in the model as previously discussed.

To find the best $q(\mathbf{u}, \mathbf{v})$, we optimize the evidence lower bound (ELBO) of the log marginal:

$$\begin{aligned}\log p(\mathbf{y}) \geq & \int q(\mathbf{u}, \mathbf{v}) \log p(\mathbf{y}|\mathbf{u}, \mathbf{v}) \mathrm{d}\mathbf{u}\mathrm{d}\mathbf{v} \\ & - \sum_{j=1}^{Q} \mathrm{KL}[q(\mathbf{u}_j)||p(\mathbf{u}_j)] - \sum_{i=1}^{P} \mathrm{KL}[q(\mathbf{v}_i)||p(\mathbf{v}_i)], \quad (13)\end{aligned}$$

which is derived using Jensen's inequality and the fact that both of $q(\mathbf{u}, \mathbf{v})$ and $p(\mathbf{u}, \mathbf{v})$ fully factorize. Since $q(\mathbf{u}_j), q(\mathbf{v}_i), p(\mathbf{u}_j), p(\mathbf{v}_i)$ are all multivariate Gaussian distributions, the KL divergence terms are analytically tractable. To compute the expected likelihood term in the ELBO we first see that

$$\begin{aligned}\log\ p(\mathbf{y}|\mathbf{u}, \mathbf{v}) &\geq \langle \log p(\mathbf{y}|\mathbf{g}, \mathbf{h}) \rangle_{p(\mathbf{g}, \mathbf{h}|\mathbf{u}, \mathbf{v})} \\ &= \sum_{i=1}^{P} \sum_{n=1}^{N} \langle \log p(y_{in}|\mathbf{g}_n, h_{in}) \rangle_{p(\mathbf{g}|\mathbf{u})p(\mathbf{h}_i|\mathbf{v}_i)} \quad (14)\end{aligned}$$

where $\mathbf{g}_n = \{g_{jn} = (\mathbf{g}_j)_n\}_{j=1}^{Q}$. The inequality is due to Jensen's inequality and the equality is due to the factorization of the likelihood.

The ELBO can be computed by first solving for the individual expectations $\langle \log p(y_{in}|\mathbf{g}_n, h_{in}) \rangle$ over $p(\mathbf{g}|\mathbf{u})p(\mathbf{h}_i|\mathbf{v}_i)$ and then substituting these into Equation (13) (see the supplementary material for details). Hence the resulting lower bound is given by:

$$\begin{aligned}\mathcal{L} = & \sum_{i,n} \Bigg(\log \mathcal{N}(y_{in}; \tilde{\mu}_{in}, \beta_i^{-1}) - \frac{1}{2}\beta_i \sum_{j=1}^{Q} w_{ij}^2 \tilde{k}_{jnn} \\ & - \frac{1}{2}\beta_i \tilde{k}_{inn}^h - \frac{1}{2}\beta_i \sum_{j=1}^{Q} \text{ tr } w_{ij}^2 \mathbf{S}_j \mathbf{\Lambda}_{jn} - \beta_i \frac{1}{2} \text{ tr } \mathbf{S}_i^h \mathbf{\Lambda}_{in} \Bigg) \\ & - \sum_{j=1}^{Q} \Bigg(\frac{1}{2} \log |\mathbf{K}_{jzz} \mathbf{S}_j^{-1}| + \frac{1}{2} \text{ tr } \mathbf{K}_{jzz}^{-1} \left(\mathbf{m}_j \mathbf{m}_j^T + \mathbf{S}_j \right) \Bigg) \\ & - \sum_{i=1}^{P} \Bigg(\frac{1}{2} \log |\mathbf{K}_{izz} (\mathbf{S}_i^h)^{-1}| \\ & \qquad + \frac{1}{2} \text{ tr } \mathbf{K}_{izz}^{-1} \left(\mathbf{m}_i^h (\mathbf{m}_i^h)^T + \mathbf{S}_i^h \right) \Bigg), \quad (15)\end{aligned}$$

where $\mathbf{K}_{jzz} = k(\mathbf{Z}_j, \mathbf{Z}_j)$, $\mathbf{K}_{izz} = k(\mathbf{Z}_i^h, \mathbf{Z}_i^h)$, and:

$$\tilde{\mu}_{in} = \sum_{j=1}^{Q} w_{ij} \mathbf{A}_j(n, :) \mathbf{m}_j + \mathbf{A}_i^h(n, :) \mathbf{m}_i^h, \quad (16)$$

$$\mathbf{\Lambda}_{jn} = \mathbf{A}_j(n, :)^T \mathbf{A}_j(n, :), \quad (17)$$

$$\mathbf{\Lambda}_{in} = \mathbf{A}_i^h(n, :)^T \mathbf{A}_i^h(n, :), \quad (18)$$

with $\tilde{k}_{jnn} = (\tilde{\mathbf{K}}_j)_{nn}$; $\tilde{k}_{inn}^h = (\tilde{\mathbf{K}}_i^h)_{nn}$; $\mu_{jn} = (\boldsymbol{\mu}_j)_n$; $\mu_{in}^h = (\boldsymbol{\mu}_i^h)_n$; and we have defined the auxiliary matrices $\mathbf{A}_j = k(\mathbf{X}, \mathbf{Z}_j)\mathbf{K}_{jzz}^{-1}$ and $\mathbf{A}_i^h = k(\mathbf{X}_i, \mathbf{Z}_i^h)\mathbf{K}_{izz}^{-1}$ and used $\mathbf{A}_j(n, :)$ to denote the n-th row vector of $\mathbf{A}_j$. Notice that this ELBO generalizes the bound for standard GP regression derived in Hensman et al. (2013), which can be recovered by setting $P = Q = 1$, $w_{ij} = 1$ and $h_i(\mathbf{x}) = 0$.

The novelty of the variational lower bound in Equation (15) is that it decomposes across both inputs and outputs. This enables the use of stochastic optimization methods, which allow the model to handle very large datasets for which existing GP-based multi-output models are simply impractical.

3.2 STOCHASTIC VARIATIONAL INFERENCE

So far in the description of the model and inference we have implicitly assumed that every output has full observations at all inputs $\mathbf{X}$. To discern where learning occurs for each output, we make the missing data scenario more explicit. Specifically, each output i can have observations at a different set of inputs $\mathbf{X}_i$. We shall use $\mathbf{o}_i$ to denote the indices of $\mathbf{X}_i$ (in the set $\mathbf{X}$) and use the indexing operator $\mathbf{B}(\mathbf{o}_i)$ to select the rows corresponding to $\mathbf{o}_i$ from any arbitrary matrix $\mathbf{B}$. We also overload $\mathbf{y}_i$ as the observed targets of output i.

3.2.1 Learning the Parameters of the Variational Distribution

We can obtain the derivatives of the ELBO in Equation (15) wrt the variational parameters for optimization. The derivatives of $\mathcal{L}$ wrt the parameters of $q(\mathbf{u}_j)$ are given by:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \mathbf{m}_j} = & \sum_{i=1}^{P} \beta_i w_{ij} \mathbf{A}_j(\mathbf{o}_i)^T \mathbf{y}_i^{\backslash j} \quad (19) \\ & - \Bigg[\mathbf{K}_{jzz}^{-1} + \sum_{i=1}^{P} \beta_i w_{ij}^2 \mathbf{A}_j(\mathbf{o}_i)^T \mathbf{A}_j(\mathbf{o}_i) \Bigg] \mathbf{m}_j, \\ \frac{\partial \mathcal{L}}{\partial \mathbf{S}_j} = & \frac{1}{2}\mathbf{S}_j^{-1} - \frac{1}{2} \Bigg[\mathbf{K}_{jzz}^{-1} + \sum_{i=1}^{P} \beta_i w_{ij}^2 \mathbf{A}_j(\mathbf{o}_i)^T \mathbf{A}_j(\mathbf{o}_i) \Bigg], \quad (20)\end{aligned}$$

where $\mathbf{y}_i^{\backslash j} = \mathbf{y}_i - \mathbf{A}_i^h(\mathbf{o}_i)\mathbf{m}_i^h - \sum_{j' \neq j} w_{ij'} \mathbf{A}_{j'}(\mathbf{o}_i)\mathbf{m}_{j'}$.

The derivatives of $\mathcal{L}$ wrt the parameters of $q(\mathbf{v}_i)$ are given by:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \mathbf{m}_i^h} =& \beta_i \mathbf{A}_i^h(\mathbf{o}_i)^T \mathbf{y}_i^{\backslash h} \\ & - \left[\mathbf{K}_{izz}^{-1} + \beta_i \mathbf{A}_i^h(\mathbf{o}_i)^T \mathbf{A}_i^h(\mathbf{o}_i)\right]\mathbf{m}_i, \qquad (21)\\ \frac{\partial \mathcal{L}}{\partial \mathbf{S}_i^h} =& \frac{1}{2}\mathbf{S}_i^{-1} - \frac{1}{2}\left[\mathbf{K}_{izz}^{-1} + \beta_i \mathbf{A}_i^h(\mathbf{o}_i)^T \mathbf{A}_i^h(\mathbf{o}_i)\right], \qquad (22)\end{aligned}$$

where $\mathbf{y}_i^{\backslash h} = \mathbf{y}_i - \sum_{j=1}^{Q} w_{ij}\mathbf{A}_j(\mathbf{o}_i, :)\mathbf{m}_j$.

It can be seen that the derivatives of the parameters of $q(\mathbf{v}_i)$ only involve the observations of the output i. The derivatives of the parameters of $q(\mathbf{u}_j)$ involve the observations of all outputs but is a sum of contributions from individual outputs. Computation of the derivatives can therefore be easily distributed or parallelized.

Since the optimal distributions $q(\mathbf{u}_j)$ and $q(\mathbf{v}_i)$ are in the exponential family, it is more convenient to use stochastic variational inference (Hensman et al., 2012, 2013) to perform update of their canonical parameters. This works by taking a step of length l in the direction of the natural gradient approximated by mini-batches of the data. For instance, consider $q(\mathbf{u}_j)$ whose canonical parameters are $\Phi_1 = \mathbf{S}_j^{-1}\mathbf{m}_j$ and $\Phi_2 = -\frac{1}{2}\mathbf{S}_j^{-1}$. Their stochastic update equations at time $t+1$ are given by:

$$\begin{aligned}\Phi_{1(t+1)} =& \mathbf{S}_{j(t)}^{-1}\mathbf{m}_{j(t)} \\ & + l\left(\sum_{i=1}^{P} \beta_i w_{ij} \mathbf{A}_j(\mathbf{o}_i)^T \mathbf{y}_i^{\backslash j} - \mathbf{S}_{j(t)}^{-1}\mathbf{m}_{j(t)}\right) \qquad (23)\\ \Phi_{2(t+1)} =& -\frac{1}{2}\mathbf{S}_{j(t)}^{-1} + l\left(\frac{1}{2}\mathbf{S}_{j(t)}^{-1} - \frac{1}{2}\mathbf{\Lambda}\right), \qquad (24)\end{aligned}$$

where $\mathbf{\Lambda} = \mathbf{K}_{jzz}^{-1} + \sum_{i=1}^{P} \beta_i w_{ij}^2 \mathbf{A}_j(\mathbf{o}_i)^T \mathbf{A}_j(\mathbf{o}_i)$.

3.2.2 Inducing Inputs and Hyper-parameters

To learn the hyperparameters, which in this model include the mixing weights, the covariance hyperparameters of the latent processes, and the noise precision of each output, we follow standard practice in GP inference. For this model this involves taking derivatives of the ELBO and applying standard stochastic gradient descent in alternative steps with the variational parameters, much like a variational EM algorithm. The derivatives are given in the supplementary material.

Learning of the inducing inputs, which was not considered in the single output case in Hensman et al. (2013), is also possible in our stochastic optimization approach. In the supplementary material, we show that the additional cost of computing the derivatives of the lower bound wrt the inducing inputs is not significantly higher than the cost of updating the variational parameters. This makes optimizing the inducing locations a practical option, which can be critical in high-dimensional problems. Indeed, our experiments on a large scale multi-output problem show that automatic learning of the inducing inputs can lead to significant performance gain with little overhead in computation.

Table 1: Comparison of the time and storage complexity of approximate inference of multi-output GP models. A&W, 2009 refers to Alvarez and Lawrence (2009). COGP is the only method with complexity *independent* of the number of inputs N and outputs P, thus it can scale to very large datasets.

METHOD	**TIME**	**STORAGE**
COGP, this paper	$\mathcal{O}(M^3)$	$\mathcal{O}(M^2)$
SLFM, (Teh et al., 2005)	$\mathcal{O}(QNM_t^2)$	$\mathcal{O}(QNM_t)$
MTGP, (Bonilla et al., 2008)	$\mathcal{O}(PNM_t^2)$	$\mathcal{O}(PNM_t)$
CGP-FITC (A&W, 2009)	$\mathcal{O}(PNM_t^2)$	$\mathcal{O}(PNM_t)$
GPRN, (Wilson et al., 2012)	$\mathcal{O}(PQN^3)$	$\mathcal{O}(PQN^2)$

3.3 COMPLEXITY ANALYSIS

In this section we analyze the complexity of the model and compare it to existing multi-output approaches. For consistency, we first unify common notations used for all models. We use P as the number of outputs; N as the number of inputs; Q as the number of shared latent processes; and M_t as the *total* number of inducing inputs. It is worth noting that $M_t = (P+Q) \times M$ in our COGP model, assuming that each sparse process has equal number of inducing points. Also, COGP has P additional individual processes, one for each output.

The complexity of COGP can be read off by inspecting the ELBO in Equation (15), with the key observation that it contains a sum over the outputs as well as over the inputs. This means a mini-batch containing a small subset of the inputs and outputs can be used for stochastic optimization. Technically, the cost is $\mathcal{O}(M^3)$ or $\mathcal{O}(N_b M^2)$, where N_b is the size of the mini-batches, depending on which is larger between M and N_b. In practice, we may use $N_b > M$ as a large batch size (e.g. $N_b = 1000$) helps reduce stochasticity of the optimization. However, here we use $\mathcal{O}(M^3)$ for easier comparison with other models whose time and storage demands are given in Table 1. We see that COGP has a computational complexity that is independent of the size of the inputs and outputs, which makes it the only method capable of handling large scale problems.

3.4 PREDICTION

The predictive distribution of the i-th output for a test input $\mathbf{x}_*$ is given by:

$$p(f_*|\mathbf{y}, \mathbf{x}_*) = \mathcal{N}(f_*; \sum_{j=1}^{Q} w_{ij}\mu_{j*} + \mu_{i*}^h, w_{ij}^2 s_{j*} + s_{i*}^h), \tag{25}$$

where μ_{j*} and s_{j*} are the mean and variance of the prediction for $g_{j*} = g_j(\mathbf{x}_*)$, i.e. $p(g_{j*}|\mathbf{y}, \mathbf{x}_*) = \mathcal{N}(g_{j*}; \mu_{j*}, s_{j*})$. Likewise, μ_{i*}^h and s_{i*}^h are the mean and variance of the prediction for $h_{i*} = h_i(\mathbf{x}_*)$, $p(h_{i*}|\mathbf{y}, \mathbf{x}_*) = \mathcal{N}(h_{i*}; \mu_{i*}^h, s_{i*}^h)$. These predictive means and variances are given by:

$$\mu_{j*} = \mathbf{k}_{j*z}\mathbf{K}_{jzz}^{-1}\mathbf{m}_j, \tag{26}$$

$$s_{j*} = k_{j**} - \mathbf{k}_{j*z}\left(\mathbf{K}_{jzz}^{-1} - \mathbf{K}_{jzz}^{-1}\mathbf{S}_j\mathbf{K}_{jzz}^{-1}\right)\mathbf{k}_{j*z}^T, \tag{27}$$

$$\mu_{i*}^h = \mathbf{k}_{i*z}\mathbf{K}_{izz}^{-1}\mathbf{m}_i^h, \tag{28}$$

$$s_{i*}^h = k_{i**} - \mathbf{k}_{i*z}\left(\mathbf{K}_{izz}^{-1} - \mathbf{K}_{izz}^{-1}\mathbf{S}_i\mathbf{K}_{izz}^{-1}\right)\mathbf{k}_{i*z}^T, \tag{29}$$

where $k_{j**} = k_j(\mathbf{x}_*, \mathbf{x}_*)$, $k_{i**} = k_i^h(\mathbf{x}_*, \mathbf{x}_*)$, $\mathbf{k}_{j*z}$ is the covariance between $\mathbf{x}_*$ and $\mathbf{Z}_j$, and $\mathbf{k}_{i*z}$ is the covariance between $\mathbf{x}_*$ and $\mathbf{Z}_i^h$.

4 EXPERIMENTS

We evaluate the proposed approach with four experiments. A toy problem is first used to study the transfer of learning between two related processes via the shared inducing points. We then compare the model with existing multi-output models on the tasks of predicting foreign exchange rate and air temperature. In the final experiment, we show that joint learning under sparsity can yield significant performance gain on a large scale dataset of inverse dynamics of a robot arm.

Since we are using stochastic optimization, the learning rates need to be chosen carefully. We found that the rates used in Hensman et al. (2013) also work well for our model. Specifically, we used the learning rates of 0.01 for the variational parameters, 1×10^{-5} for the covariance hyperparameters, and 1×10^{-4} for the weights, noise precisions, and inducing inputs. We also included a momentum term of 0.9 for all of the parameters except the variational parameters and the inducing inputs. All of the experiments are executed on an Intel(R) Core(TM) i7-2600 3.40GHz CPU with 8GB of RAM using Matlab R2012a.

4.1 TOY PROBLEM

In this toy problem, two related outputs are simulated from the same latent function $sin(x)$ and corrupted by independent noise: $y_1(x) = sin(x) + \epsilon$ and $y_2(x) = -sin(x) + \epsilon$, $\epsilon \sim \mathcal{N}(0, 0.01)$. Each output is given 200 observations with missing values in the $(-7, -3)$ interval for the first output and the $(4, 8)$ interval for the second output. We used $Q = 1$ latent sparse process with squared exponential kernel, $h_1(x) = h_2(x) = 0$, and $M = 15$ inducing inputs for our model.

Figure 1 shows the predictive distributions by our model (COGP) and independent GPs with stochastic variational inference (SVIGP, one for each output). The locations of the inducing inputs are fixed and identical for both methods. It is apparent from the figure that the independent GPs fail to predict the functions in the unobserved regions, especially for output 1. In contrast, by using information from the observed intervals of one output to interpolate the missing signal of the other, COGP makes perfect prediction for both outputs. This confirms the effectiveness of collaborative learning of sparse processes via the shared inducing variables. Additionally, we note that the inference procedure learned that the weights are $w_{11} = 1.07$ and $w_{21} = -1.06$ which accurately reflects the correlation between the two outputs.

4.2 FOREIGN EXCHANGE RATE PREDICTION

The first real world application we consider is to predict the foreign exchange rate w.r.t the US dollar of the top 10 international currencies (CAD, EUR, JPY, GBP, CHF, AUD, HKD, NZD, KRW, and MXN) and 3 precious metals (gold, silver, and platinum)[1]. The setting of our experiment described here is identical to that in Álvarez et al. (2010). The dataset consists of all the data available for the 251 working days in the year of 2007. There are 9, 8, and 42 days of missing values for gold, silver, and platinum, respectively. We remove from the data the exchange rate of CAD on days 50–100, JPY on day 100–150, and AUD on day 150–200. Note that these 3 currencies are from very different geographical locations, making the problem more interesting. The 153 points are used for testing, and the remaining 3051 data points are used for training. Since the missing data corresponds to long contiguous sections, the objective here is to evaluate the capacity of the model to impute the missing currency values based on other currencies.

For preprocessing we normalized the outputs to have zero mean and unit variance. Since the exchange rates are driven by a small number of latent market forces (see e.g. Álvarez et al., 2010), we tried different values of $Q = 1, 2, 3$ and selected $Q = 2$ which gave the best model evidence (ELBO). We used the squared-

[1] Data is available at http://fx.sauder.ubc.ca

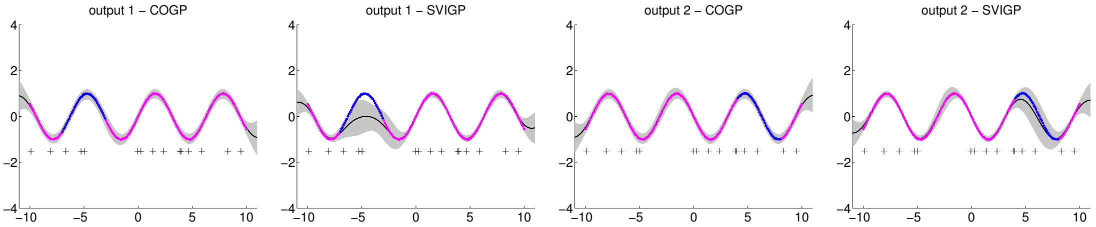

Figure 1: Simulated data and predictive distributions of by COGP (first and third figure) and independent GPs using stochastic variational inference (second and last figure) for the toy problem. Solid black line: predictive mean; grey bar: two standard deviations; magenta dots: real observations; blue dots: missing data. The black crosses show the locations of the inducing inputs. By sharing inducing points across the outputs, COGP accurately interpolates the missing function values.

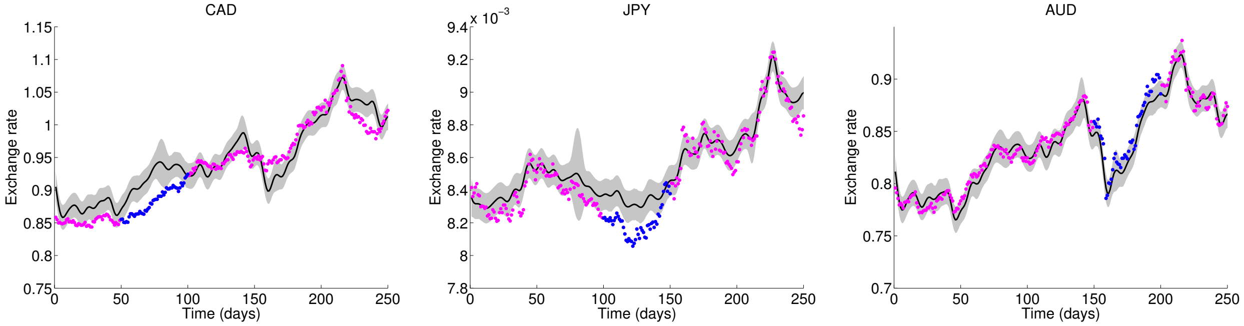

Figure 2: Real observations and predictive distributions for CAD (left), JPY (middle), and AUD (right). The model used information from other currencies to effectively extrapolate the exchange rates of AUD. The color coding scheme is the same as in Figure 1.

Table 2: Performance comparison on the foreign exchange rate dataset. Results are averages of the 3 outputs over 5 repetitions. Smaller figures are better.

METHOD	SMSE	NLPD
COGP	**0.2125**	-0.8394
CGP	0.2427	**-2.9474**
IGP	0.5996	0.4082

exponential covariance function for the shared processes and the noise covariance function for the individual process of each output. $M = 100$ inducing inputs (per sparse process) were randomly selected from the training data and fixed throughout training.

The real data and predictive distributions by our model are shown in Figure 2. They exhibit similar behaviors to those by the convolved model with inducing kernels in Álvarez et al. (2010). In particular, both models perform better at capturing the strong depreciation of the AUD than the fluctuations of the CAD and JPY currency. Further analysis of the dataset found that 4 other currencies (GBP, NZD, KRW, and MXN) also experienced the same trend during the days 150 – 200. This information from these currencies was effectively used by the model to extrapolate the values of the AUD.

We also report in Table 2 the predictive performance of our model compared to the convolved GPs model with exact inference (CGP, Alvarez and Lawrence, 2009) and independent GPs (IGP, one for each output). Our model outperforms both of CGP and IGP in terms of the standardized mean squared error (SMSE). CGP has lower negative log predictive density (NLPD), mainly due to the less conservative predictive variance of the exact CGP for the CAD currency. For reference, the convolved GPs with approximation via the variational inducing kernels (CGPVAR, Álvarez et al., 2010) has an SMSE of 0.2795 while the NLPD was not provided. Training took only 10 minutes for our model compared to 1.4 hours for the full CGP model.

4.3 AIR TEMPERATURE PREDICTION

Next we consider the task of predicting air temperature at 4 different locations in the south coast of England. The air temperatures are recorded by a network of weather sensors (named Bramblemet, Sotonmet, Cambermet, and Chimet) during the period from

Table 3: Performance comparison on the air temperature dataset. Results are averages of 2 outputs over 5 repetitions.

METHOD	SMSE	NLPD
COGP	**0.1077**	**2.1712**
CGP	0.1125	2.2219
IGP	0.8944	12.5319

July 10 to July 15, 2013. Measurements were taken every 5 minutes, resulting in a maximum of 4320 observations. There are missing data for Bramblemet (100 points), Chimet (15 points), and Sotonmet (1002 points), possibly due to network outages or hardware failures. We further simulated failure of the sensors by removing the observations from the time periods [10.2 - 10.8] for Cambermet and [13.5 - 14.2] for Chimet. The removed data comprises 375 data points and is used for testing. The remaining data consisting of 15,788 points is used for training. Similar to the previous experiment, the objective is to evaluate the ability of the model to use the signals from the functioning sensors to extrapolate the missing signals.

We normalized the outputs to have zero mean and unit variance. We used $Q = 2$ sparse processes with the squared exponential covariance function and individual processes with the noise covariance function. $M = 200$ inducing inputs were randomly selected from the training set and fixed throughout training.

The real data and the predictive distributions by our model, CGP with exact inference (Alvarez and Lawrence, 2009), and independent GPs are shown in Figure 3. It is clear that the independent GP model is clueless in the test regions and thus simply uses the average temperature as its prediction. For Cambermet, both COGP and CGP can capture the rising in temperature from the morning until the afternoon and the fall afterwards. The performance of the models are summarized in Table 3, which shows that our model outperforms CGP in terms of both SMSE and NLPD. It took 5 minutes on average to train our model compared to 3 hours of CGP with exact inference.

It is also worth noting the characteristics of the sparse processes learned by our model as they correspond to different patterns in the data. In particular, one process has an inverse lengthscale of 136 which captures the global increase in temperature during the training period while the other has an inverse lengthscale of 0.5 to model the local variations within a single day.

Table 4: Performance comparison on the robot inverse dynamics dataset. In the last two lines, standard GP is applied to output 1 and the other method is applied to output 2. Results are averaged over 5 repetitions.

	OUTPUT 1		OUTPUT 2	
METHOD	SMSE	NLPD	SMSE	NLPD
COGP, learn z	**0.2631**	**3.0600**	0.0127	**0.8302**
COGP, fix z	0.2821	3.2281	0.0131	0.8685
GP, SVIGP	0.3119	3.2198	**0.0101**	1.1914
GP, SOD	0.3119	3.2198	0.0104	1.9407

4.4 ROBOT INVERSE DYNAMICS

Our last experiment is with a dataset relating to an inverse dynamics model of a 7-degree-of-freedom anthropomorphic robot arm (Vijayakumar and Schaal, 2000). The data consists of 48,933 datapoints mapping from a 21-dimensional input space (7 joints positions, 7 joint velocities, 7 joint accelerations) to the corresponding 7 joint torques. It has been used in previous work (see e.g. Rasmussen and Williams, 2006; Vijayakumar and Schaal, 2000) but only for single task learning. Chai et al. (2008) considered multitask learning of robot inverse dynamics but on a different and much smaller dataset.

Here we consider joint learning for the 4th and 7th torques, where the former has 2,000 points while the latter has 44,484 points for training. The test set consists of 8,898 observations equally divided between the two outputs.

Since none of the existing multi-output models are applicable to problems of this scale, we compare with independent models that learn each output separately. Standard GP is applied to the first output as it has only 2,000 observations for training. For the second output, we used two baselines. The first is the subset of data (SOD) approach where 2,000 data points are randomly selected for training with a standard GP model. The second is the sparse GP with stochastic variational inference (SVIGP) using 500 inducing inputs and a batch size of 1,000. In case of COGP, we also used a batch size of 1,000 and 500 inducing points for the shared process ($Q = 1$) and each of the individual processes.

The performance of all methods in terms of SMSE and NLPD is given in Table 4. The benefits of learning the two outputs jointly are evident, as can be seen by the significantly lower SMSE and NLPD of COGP compared to the full GP for the first output (4th torque). While the SMSE of the second output is essentially the same for all methods, the NLPD of COGP is sub-

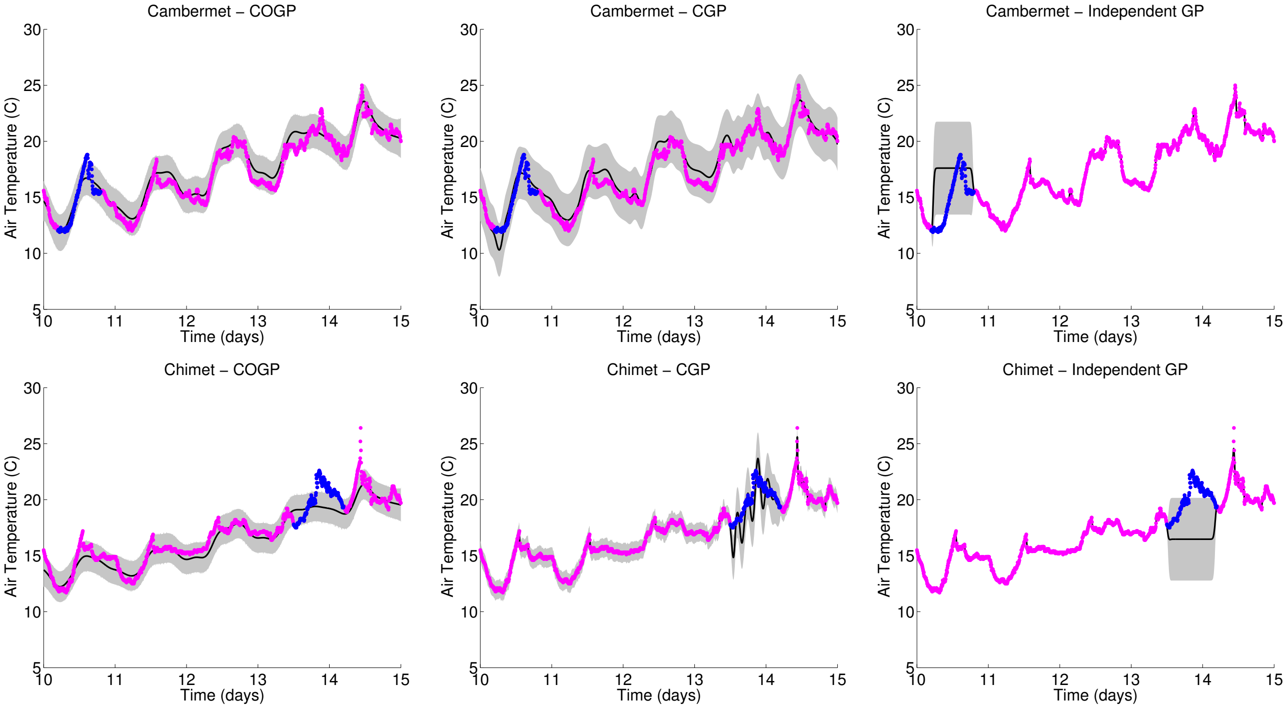

Figure 3: Real data and predictive distributions by our method (COGP, left figures), the convolved GP method with exact inference (CGP, middle figures), and full independent GPs (right figures) for the air temperature problem. The coding color scheme is the same as in Figure 1.

stantially better than that of the independent SVIGP model which has the same amount of training data for this torque. These results validate the impact of collaborative learning under sparsity assumptions, opening up new opportunities for improvement over single task learning with independent sparse processes.

Finally, we see on Table 4 that optimizing the inducing inputs can yield better performance than fixing them. More importantly, the overhead in computation is small, as demonstrated by the training times shown in Figure 4. For instance, the total training time is only 1.9 hours when learning with 500 inducing inputs compared to 1.6 hours when fixing them. As this dataset is 21-dimensional, this small difference in training time confirms that learning of the inducing inputs is a practical option even when dealing with problems of high dimensions.

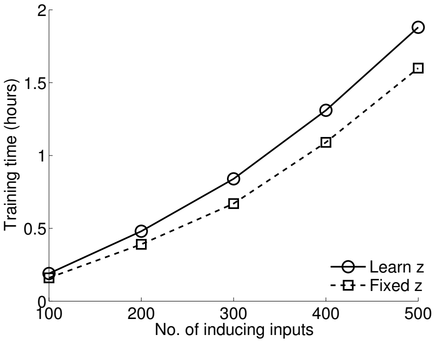

Figure 4: Learning of the inducing inputs is a practical option as the overhead in training time is small.

5 DISCUSSION

We have presented scalable multi-output GPs for learning of correlated functions. The formulation around the inducing variables was shown to be conducive to effective and scalable joint learning under sparsity. We note that although our large scale experiments were done with over 40,000 observations – the largest publicly available multi-output dataset found, the model can easily handle much bigger datasets.

Acknowledgements

EVB thanks Stephen Hardy for early discussions on inducing-point sharing for multi-output GPs. NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

References

Alvarez, M. and Lawrence, N. D. (2009). Sparse convolved Gaussian processes for multi-output regression. In *NIPS*.

Álvarez, M. A., Luengo, D., Titsias, M. K., and Lawrence, N. D. (2010). Efficient multioutput Gaussian processes through variational inducing kernels. In *AISTATS*.

Bonilla, E. V., Chai, K. M. A., and Williams, C. K. I. (2008). Multi-task Gaussian process prediction. In *NIPS*.

Boyle, P. and Frean, M. (2005). Dependent Gaussian processes. In *NIPS*.

Chai, K. M. A., Williams, C. K., Klanke, S., and Vijayakumar, S. (2008). Multi-task gaussian process learning of robot inverse dynamics. In *NIPS*.

Goovaerts, P. (1997). *Geostatistics for natural resources evaluation*. Oxford University Press.

Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for big data. In *Uncertainty in Artificial Intelligence*.

Hensman, J., Rattray, M., and Lawrence, N. D. (2012). Fast variational inference in the conjugate exponential family. In *NIPS*.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine Learning*, 37:183–233.

Nguyen, T. V. and Bonilla, E. V. (2013). Efficient variational inference for gaussian process regression networks. In *AISTATS*.

Quiñonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate Gaussian process regression. *The Journal of Machine Learning Research*.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

Teh, Y. W., Seeger, M., and Jordan, M. I. (2005). Semiparametric latent factor models. In *AISTATS*.

Titsias, M. (2009). Variational learning of inducing variables in sparse Gaussian processes. In *AISTATS*.

Titsias, M. K. and Lázaro-Gredilla, M. (2011). Spike and slab variational inference for multi-task and multiple kernel learning. In *NIPS*.

Vijayakumar, S. and Schaal, S. (2000). Locally weighted projection regression: An O(n) algorithm for incremental real time learning in high dimensional space. In *ICML*.

Wilson, A. G., Knowles, D. A., and Ghahramani, Z. (2012). Gaussian process regression networks. In *ICML*.