Fast Allocation of Gaussian Process Experts

Trung V. Nguyen VTNGUYEN@NICTA.COM.AU
ANU & NICTA

Edwin V. Bonilla EDWIN.BONILLA@NICTA.COM.AU
NICTA & ANU

Abstract

We propose a scalable nonparametric Bayesian regression model based on a mixture of Gaussian process (GP) experts and the inducing points formalism underpinning sparse GP approximations. Each expert is augmented with a set of inducing points, and the allocation of data points to experts is defined probabilistically based on their proximity to the experts. This allocation mechanism enables a fast variational inference procedure for learning of the inducing inputs and hyperparameters of the experts. When using K experts, our method can run K^2 times faster and use K^2 times less memory than popular sparse methods such as the FITC approximation. Furthermore, it is easy to parallelize and handles non-stationarity straightforwardly. Our experiments show that on medium-sized datasets (of around 10^4 training points) it trains up to 5 times faster than FITC while achieving comparable accuracy. On a large dataset of 10^5 training points, our method significantly outperforms six competitive baselines while requiring only a few hours of training.

1. Introduction

Gaussian processes (GPs) have become the prior of choice in nonparametric Bayesian regression approaches not only in the standard setting of iid Gaussian noise but also in more realistic scenarios that include heteroscedasticity (Kersting et al., 2007), non-stationarity (Paciorek & Schervish, 2004) and multi-task learning (Bonilla et al., 2008). However, the high computational cost in time and memory of GP-based methods, usually $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$ for N training data points respectively, have hindered their applicability to large scale problems. Consequently, a vast amount of research has tried to overcome these computational difficulties via the so-called sparse approximations (Lawrence et al., 2002; Seeger, 2003; Seeger et al., 2003; Smola & Bartlett, 2001; Snelson & Ghahramani, 2006; Williams & Seeger, 2001). In particular, the seminal work of Quiñonero-Candela & Rasmussen (2005) has provided the machine learning community with a better understanding of what these approximation methods are actually doing. The key insight of that work is that most GP approximations can be formulated within a single probabilistic framework, where the training data is augmented with *inducing points*, which are generated by the same latent function being learned. Conditioned on those points, the outputs become statistically independent. This leads to a lower complexity of $\mathcal{O}(NB^2)$ in computation and $\mathcal{O}(NB)$ in memory when $B \ll N$ inducing points are used.

Despite this encouraging progress to scale GPs via sparse approximations, time and memory complexities of $\mathcal{O}(NB^2)$ and $\mathcal{O}(NB)$ are still too costly to handle large datasets. Recent developments in stochastic variational inference for GPs under the inducing-point formalism (GPSVI, Hensman et al., 2013) have shown that these costs can be reduced further to $\mathcal{O}(B^3)$ and $\mathcal{O}(B^2)$ respectively. While these methods have made possible the application of GPs to large datasets, their main assumption of having *global* inducing points can be insufficient to capture the dependencies between the observations. In fact, as we shall see in our experiments in section 4, such an assumption comes at the expense of significant performance degradation, especially in high-dimensional spaces.

We believe that, as argued by Rasmussen & Ghahramani (2002), modeling large (and possibly high dimensional) datasets with a single GP is simply undesirable given that critical issues such as non-stationarity and locality become difficult to handle when having a small set of global inducing points. To address these issues, mixture-of-experts architectures (see e.g. Gramacy & Lee, 2008; Kim et al., 2005; Meeds & Osindero, 2006; Rasmussen & Ghahramani, 2002) have been proposed in the literature. In these

Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32. Copyright 2014 by the author(s).

approaches, the data is *partitioned* into local regions which are modeled by independent GP experts with their own hyperparameters. The functions generated by the experts can exhibit different smoothness properties, thereby allowing non-stationarity in the outputs to be captured.

However, a common drawback of these models is that the expert assignments must often be carried out via intensive MCMC sampling. As a result, although mixture of GP experts may be able to deal with non-stationarity, their poor scalability precludes any application on even moderate-sized problems. In fact, to the best of our knowledge, the largest experiment using the existing GP mixture models has been performed with no more than 2,000 data points (Kim et al., 2005).

In this paper we propose a *scalable nonparametric Bayesian regression* model based on a mixture of GP experts that leverages on the inducing points formalism underpinning sparse GP models. In the model, data points closer to the underlying inducing points of an expert are given higher probabilities to be assigned to that expert. This novel allocation allows each local input region to be accounted for by a different expert with its distinct *local* inducing set. This makes the model more powerful and flexible than traditional sparse methods whose global set of inducing points may fail to support some parts of the input space.

We derive a variational inference algorithm for this model and, by exploiting the locality generated by the expert allocation, we obtain a fast optimization-based algorithm to learn the inducing inputs and hyperparameters of the experts. When using K experts, each with M inducing points, the inference requires only $O(NM^2)$ in computation and $O(NM/K)$ in memory. This is a factor of K^2 improvement in time and storage complexity compared to the traditional sparse methods if the same total number of inducing points is used in both models, i.e. $B = M \times K$. Thus, our model can work with large datasets even on the moderate resources of a single desktop computer. Furthermore, our algorithm can be parallelized easily with multi-core or distributed computing.

Our experiments investigate non-stationarity on a small dataset where the model shows consistent behavior as seen in previous approaches. More importantly, empirical evaluation on 3 medium-sized datasets (of around 10^4 training points) confirms the K^2 speed-up factor by our method compared to traditional sparse GPs such as the FITC approximation (Quiñonero-Candela & Rasmussen, 2005), while achieving comparable accuracy. On the Million Song Dataset (Bertin-Mahieux et al., 2011) with 10^5 training points, our method significantly outperforms six competitive baselines, while requiring only a few hours of full training including learning of inducing inputs and hyper-parameters. Our results encourage the use and development of multiple experts model when dealing with very large datasets, which may not be adequately represented by a traditional global model or function.

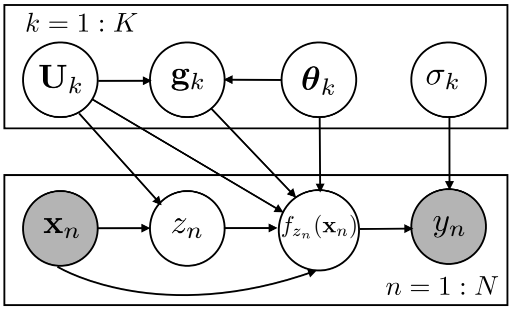


Figure 1. Graphical representation of the model where shaded nodes represent observed variables. Our model can be seen as a local mixture of sparse GP experts, where the expert assignment z_n is determined by the inducing inputs $\mathbf{U}$.

2. Model Specification

Let the data be a set of N observations: $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$ and $\mathbf{y} = \{y_n\}_{n=1}^N$, where $\mathbf{x}_n \in \mathcal{R}^d$ is a vector of d input features and $y_n \in \mathcal{R}$. The graphical representation of the model is shown in Figure 1. There are K independent experts in the model, each consists of a latent function $f_k(\mathbf{x}) \sim \mathcal{GP}(\mathbf{0}, \kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}_k))$, where $\kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}_k)$ is the kernel function of expert k which is parameterized by the hyperparameters $\boldsymbol{\theta}_k$[1]. Each expert is also augmented with a set of M *inducing inputs* $\mathbf{U}_k = \{\mathbf{u}_{k1}, \ldots, \mathbf{u}_{kM}\}$, whose latent values are denoted by $\mathbf{g}_k = (f_k(\mathbf{u}_{k1}), \ldots, f_k(\mathbf{u}_{kM}))^T$, which we shall refer to as the *inducing points*. We assume that each observation $\mathbf{x}_n$ and y_n is associated with only one expert identified by the indicator z_n. Under the standard Gaussian likelihood model, the output is $y_n = f_{z_n}(\mathbf{x}_n) + \epsilon_n$, i.e. the latent value at $\mathbf{x}_n$ by expert z_n corrupted by independent noise $\epsilon_n \sim \mathcal{N}(0, \sigma_{z_n}^2)$. Note that the inducing inputs $\mathbf{U}_k$ lie in the same input space as $\mathbf{X}$ and the inducing points $\mathbf{g}_k$ are from the same output space as the latent function $f_k(\mathbf{x})$.

We denote by $\mathbf{z}$, $\mathbf{g}, \mathbf{f}, \mathbf{U}, \boldsymbol{\theta}$, and $\boldsymbol{\sigma}$ the set of all z_n, $\mathbf{g}_k, \mathbf{f}_k, \mathbf{U}_k, \boldsymbol{\theta}_k$, and σ_k, respectively. The following auxiliary subsets of variables are defined for a given configuration assignment of experts:

$$\mathbf{f}_k = \{f_k(\mathbf{x}_n) | z_n = k\}, \quad \mathbf{X}_k = \{\mathbf{x}_n | z_n = k\}, \quad \text{and} \quad \mathbf{y}_k = \{y_n | z_n = k\}. \tag{1}$$

It is instructive to emphasize that $\mathbf{f}_k, \mathbf{X}_k, \mathbf{y}_k$ are defined given a particular configuration of $\mathbf{z}$.

[1] Here and henceforth we reserve the subscript $k = 1, \ldots, K$ to indicate associations with the expert k.

The full joint distribution of the model is given by:

$$p(\mathbf{y}, \mathbf{z}, \mathbf{f}, \mathbf{g}|\mathbf{U}, \boldsymbol{\theta}, \boldsymbol{\sigma}, \mathbf{X}) = p(\mathbf{y}|\mathbf{f}, \boldsymbol{\sigma})p(\mathbf{f}|\mathbf{z}, \mathbf{g}, \mathbf{U}, \boldsymbol{\theta}, \mathbf{X})p(\mathbf{g}|\mathbf{U}, \boldsymbol{\theta})p(\mathbf{z}|\mathbf{U}, \mathbf{X}), \quad (2)$$

where

$$p(\mathbf{y}|\mathbf{f}, \boldsymbol{\sigma}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{y}_k; \mathbf{f}_k, \sigma_k^2 \mathbf{I}), \quad (3)$$

$$p(\mathbf{g}|\mathbf{U}, \boldsymbol{\theta}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{g}_k; \mathbf{0}, \mathbf{K}(\mathbf{U}_k, \mathbf{U}_k)), \quad (4)$$

$$p(\mathbf{z}|\mathbf{U}, \mathbf{X}) = \prod_{n=1}^{N} p(z_n|\mathbf{x}_n, \mathbf{U}), \quad (5)$$

$$p(\mathbf{f}|\mathbf{z}, \mathbf{g}, \mathbf{U}, \boldsymbol{\theta}, \mathbf{X}) = \prod_{k=1}^{K} \mathcal{N}\left(\mathbf{f}_k; \bar{\mathbf{f}}_k(\mathbf{X}_k), \boldsymbol{\Lambda}_k(\mathbf{X}_k)\right), \quad (6)$$

with:

$$\bar{\mathbf{f}}_k(\mathbf{X}_k) = \mathbf{K}(\mathbf{X}_k, \mathbf{U}_k)\mathbf{K}(\mathbf{U}_k, \mathbf{U}_k)^{-1}\mathbf{g}_k, \text{ and} \quad (7)$$

$$\boldsymbol{\Lambda}_k(\mathbf{X}_k) = \operatorname{diag}\big(\mathbf{K}(\mathbf{X}_k, \mathbf{X}_k) - \mathbf{K}(\mathbf{X}_k, \mathbf{U}_k)\mathbf{K}(\mathbf{U}_k, \mathbf{U}_k)^{-1}\mathbf{K}(\mathbf{U}_k, \mathbf{X}_k)\big). \quad (8)$$

Here we have defined $\mathbf{K}(\mathbf{X}_k, \mathbf{U}_k)$ to be the covariance matrix evaluated at all pairs of points in $\mathbf{X}_k$ and $\mathbf{U}_k$ and similarly for the other covariance matrices. Additionally, the covariance matrices containing subscript k are parameterized by the kernel hyperparameters $\boldsymbol{\theta}_k$ of expert k.

Equation 3 trivially follows from the standard Gaussian likelihood as described above. Equation 4 is due to the independent GP priors over the experts. Equation 6 follows primarily from the *locality* of the experts, which means each expert only acts on the set of points assigned to it (by the indicators $\mathbf{z}$). This is similar to having multiple datasets each accounted for by one expert, although in our setting the assignment of points to experts is latent. Equation 6 also follows from the *sparsity* enforcement by each expert: the latent values are independent given the inducing points and for any $\mathbf{x}_i \in \mathbf{X}_k$ we have that:

$$p(f_k(\mathbf{x}_i)|\mathbf{x}_i, \mathbf{g}_k, \mathbf{U}_k, \boldsymbol{\theta}_k) = \mathcal{N}(f_k(\mathbf{x}_i); \bar{\mathbf{f}}_k(\mathbf{x}_i), \Lambda_k(\mathbf{x}_i)),$$

where:

$$\bar{\mathbf{f}}_k(\mathbf{x}_i) = \kappa(\mathbf{x}_i, \mathbf{U}_k)\mathbf{K}(\mathbf{U}_k, \mathbf{U}_k)^{-1}\mathbf{g}_k, \text{ and}$$

$$\Lambda_k(\mathbf{x}_i) = \kappa(\mathbf{x}_i, \mathbf{x}_i) - \kappa(\mathbf{x}_i, \mathbf{U}_k)\mathbf{K}(\mathbf{U}_k, \mathbf{U}_k)^{-1}\kappa(\mathbf{U}_k, \mathbf{x}_i).$$

Notice that this is exactly the predictive distribution of $f_k(\mathbf{x}_i)$ given the inducing points $\mathbf{g}_k$ and effectively corresponds to the fully independent training conditional (FITC, Quiñonero-Candela & Rasmussen, 2005).

We now turn the attention to the prior over the expert indicator variable z_n, which we define as

$$p(z_n = k|\mathbf{x}_n, \mathbf{U}) = \frac{\mathcal{N}(\mathbf{x}_n; \mathbf{m}_k, \mathbf{V})}{\sum_{j=1}^{K} \mathcal{N}(\mathbf{x}_n; \mathbf{m}_j, \mathbf{V})}, \quad (9)$$

where each mean $\mathbf{m}_k$ is termed the *centroid* of expert k and the covariance $\mathbf{V} = \operatorname{diag}(v_1, \ldots, v_d)$. These are given by:

$$\mathbf{m}_k = \frac{1}{M} \sum_{m=1}^{M} \mathbf{u}_{km},$$

$$v_j = \frac{1}{K(M-1)} \sum_{k=1}^{K} \sum_{m=1}^{M} (u_{kmj} - m_{kj})^2. \quad (10)$$

This novel allocation of a data point $\mathbf{x}_n$ to the experts based on its proximity to their centroids is reasonable since the closer the data point to the expert centroid, the more similar $\mathbf{x}_n$ is to its inducing inputs. Additionally, the formulation of the allocation probability as proportional to a Gaussian distribution is crucial to make learning analytically tractable via variational inference. Intuitively, one can imagine that $\mathbf{U}_k$ are generated by K multivariate Gaussians sharing a common covariance, in which case $\mathbf{m}_k$ is the empirical mean and $\mathbf{V}$ is the pooled empirical variance. Then Equation 9 can be interpreted as a probabilistic assignment of data point $\mathbf{x}_n$ to one of K clusters.

3. Inference

Our inference task is to compute the posterior distribution over the latent variables, $p(\mathbf{f}, \mathbf{g}, \mathbf{z}|\mathbf{y}, \mathbf{X}, \mathbf{U}, \boldsymbol{\theta}, \boldsymbol{\sigma})$. We treat $\{\mathbf{U}, \boldsymbol{\theta}, \boldsymbol{\sigma}\}$ as hyperparameters and learn their point estimates using maximum marginal likelihood. To this end, we use variational EM which iteratively optimizes one of the variational parameters and the hyperparameters while keeping the others fixed. We omit the hyperparameters from the conditioning set of the posterior (when unnecessary) for conciseness.

We begin by noticing that the latent values $\mathbf{f}$ can be integrated out from the full joint distribution in Equation 2 to give:

$$p(\mathbf{y}, \mathbf{z}, \mathbf{g}|\mathbf{U}, \mathbf{X}) = p(\mathbf{g}|\mathbf{U}, \boldsymbol{\theta})p(\mathbf{z}|\mathbf{U}, \mathbf{X}) \prod_{k=1}^{K} p(\mathbf{y}_k|\mathbf{g}_k), \quad (11)$$

where $p(\mathbf{y}_k|\mathbf{g}_k) = \mathcal{N}(\mathbf{y}_k; \bar{\mathbf{f}}_k, \boldsymbol{\Lambda}_k(\mathbf{X}_k) + \sigma_k^2 \mathbf{I})$. Furthermore, the posterior $p(\mathbf{f}|\mathbf{y})$ can always be retrieved by integrating over $\mathbf{g}$: $p(\mathbf{f}|\mathbf{y}) = \int p(\mathbf{f}|\mathbf{g})p(\mathbf{g}|\mathbf{y})\mathrm{d}\mathbf{g}$. It turns out that working with $\mathbf{g}$ directly is more convenient so we take this approach.

3.1. Variational Inference

We use variational inference to approximate the true posterior $p(\mathbf{g}, \mathbf{z}|\mathbf{y})$ with a tractable family of distributions $q(\mathbf{g}, \mathbf{z})$ that factorizes over $\mathbf{g}_k$ and $\mathbf{z}_n$:

$$q(\mathbf{g}, \mathbf{z}) = \prod_{k=1}^{K} q(\mathbf{g}_k) \prod_{n=1}^{N} q(\mathbf{z}_n). \tag{12}$$

Here we temporarily change the representation of $\mathbf{z}_n$ to the 1-of-K encoding, $\mathbf{z}_n = [z_{n1} \dots z_{nK}]^T$ where z_{nk} is binary and $\sum_{k=1}^{K} z_{nk} = 1$, and correspondingly for $\mathbf{z}$.

By minimizing the Kullback-Leibler divergence KL$[q||p]$ between $q(\mathbf{g}, \mathbf{z})$ and $p(\mathbf{g}, \mathbf{z}|\mathbf{y})$ we obtain the optimal distribution $q(\mathbf{z})$ as

$$q(\mathbf{z}) = \prod_{n=1}^{N} \prod_{k=1}^{K} r_{nk}^{z_{nk}}, \tag{13}$$

where $r_{nk} = \rho_{nk} / \sum_{j=1}^{K} \rho_{nj}$ is the *responsibility* of expert k to the point $\mathbf{x}_n$, and

$$\begin{aligned} \ln \rho_{nk} &= \text{const} + \ln \mathcal{N}(\mathbf{x}_n; \mathbf{m}_k, \mathbf{V}) \\ &+ \ln \mathcal{N}(y_n; \kappa(\mathbf{x}_n, \mathbf{U}_k)(\mathbf{K}_{\mathbf{u},\mathbf{u}}^{(k)})^{-1}\mathbb{E}[\mathbf{g}_k], \mathbf{\Lambda}_k(\mathbf{x}_n) + \sigma_k^2), \end{aligned} \tag{14}$$

where $\mathbf{K}_{\mathbf{u},\mathbf{u}}^{(k)} = \mathbf{K}(\mathbf{U}_k, \mathbf{U}_k)$ and the expectation is over $q(\mathbf{g}_k)$. The optimal distribution $q(\mathbf{g}_k)$ is given by:

$$q(\mathbf{g}_k) = \mathcal{N}\left(\mathbf{g}_k; \mathbf{K}_{\mathbf{u},\mathbf{u}}^{(k)} \mathbf{\Psi}_k^{-1} \mathbf{K}(\mathbf{U}_k, \mathbf{X}) \mathbf{\Gamma}_k \mathbf{y}, \mathbf{K}_{\mathbf{u},\mathbf{u}}^{(k)} \mathbf{\Psi}_k^{-1} \mathbf{K}_{\mathbf{u},\mathbf{u}}^{(k)}\right) \tag{15}$$

where

$$\mathbf{\Psi}_k = \mathbf{K}_{\mathbf{u},\mathbf{u}}^{(k)} + \mathbf{K}(\mathbf{U}_k, \mathbf{X}) \mathbf{\Gamma}_k \mathbf{K}(\mathbf{X}, \mathbf{U}_k), \tag{16}$$

$$\mathbf{\Gamma}_k = \text{diag}(\mathbf{r}_k) \bullet (\mathbf{\Lambda}_k(\mathbf{X}) + \sigma_k^2 \mathbf{I})^{-1}, \tag{17}$$

the symbol $\bullet$ denotes the Hadamard product; and $\mathbf{r}_k = [r_{1k}, \dots, r_{Nk}]^T$.

Since the mean and covariance of the posterior $q(\mathbf{g}_k)$ are completely determined by the responsibilities $\mathbf{r}_k$, there is no need to explicitly parameterize them during inference. This is very helpful as otherwise $\mathcal{O}(KM^2)$ parameters of covariance matrices must be optimized and stored.

The computational and memory demands to compute the responsibilities in Equation 14 are $\mathcal{O}(NM^2K)$ and $\mathcal{O}(NM)$ respectively, due to the computation of $\mathbf{\Psi}_k$. For large N (say $N \geq 10^5$), such complexity is still prohibitive. To make inference feasible for large datasets, we use the maximum a posteriori (MAP) assignment as described in the following section.

3.2. Approximate Inference with the MAP Assignment

To motivate the use of the MAP assignment, let us come back to the philosophy underlying this model. The experts are independent and specialize in disjoint subsets of the inputs, which encourages each data point to be explained mostly by one expert. Therefore, we can assign each point to *only* the expert of highest responsibility, which corresponds to the MAP assignment. That is we use:

$$\tilde{z}_n = \operatorname*{argmax}_k r_{nk}. \tag{18}$$

Note that the responsibilities are now $r_{nk} = 1$ if and only if $\tilde{z}_n = k$.

Once the data points are allocated to experts using the MAP assignment, the *posterior* of the inducing points become $q(\mathbf{g}_k) = \mathcal{N}(\mathbf{g}_k; \boldsymbol{\mu}_k, \mathbf{\Sigma}_k)$ where

$$\boldsymbol{\mu}_k = \mathbf{K}_{\mathbf{u},\mathbf{u}}^{(k)} \mathbf{\Psi}_k^{-1} \mathbf{K}(\mathbf{U}_k, \mathbf{X}_k) \left(\mathbf{\Lambda}_k(\mathbf{X}_k) + \sigma_k^2 \mathbf{I}\right)^{-1} \mathbf{y}_k, \tag{19}$$

$$\mathbf{\Sigma}_k = \mathbf{K}_{\mathbf{u},\mathbf{u}}^{(k)} \mathbf{\Psi}_k^{-1} \mathbf{K}_{\mathbf{u},\mathbf{u}}^{(k)}, \tag{20}$$

with $\mathbf{X}_k = \{\mathbf{x}_n | \tilde{z}_n = k\}$, $\mathbf{y}_k = \{y_n | \tilde{z}_n = k\}$ and $\mathbf{\Psi}_k = \mathbf{K}_{\mathbf{u},\mathbf{u}}^{(k)} + \mathbf{K}(\mathbf{U}_k, \mathbf{X}_k)(\mathbf{\Lambda}_k(\mathbf{X}_k) + \sigma_k^2 \mathbf{I})^{-1} \mathbf{K}(\mathbf{X}_k, \mathbf{U}_k)$. Compared to Equation 16, any point not belonging to expert k has disappeared from the kernel matrices; this is propagated from $\mathbf{\Gamma}_k$ (Equation 17) where many of r_{nk} are now zero.

The noise free *predictive* distribution for an unseen data point $\mathbf{x}_*$ is

$$\begin{aligned} p(f_* | \mathbf{x}_*, \mathbf{y}, \mathbf{U}) &\approx p(f_* | \mathbf{x}_*, \tilde{\mathbf{z}}, \mathbf{y}, \mathbf{U}) \\ &\propto \sum_{z_*=k}^{K} p(f_* | \mathbf{x}_*, z_*, \mathbf{y}_k, \mathbf{U}_k) p(z_* | \mathbf{x}_*, \mathbf{U}), \end{aligned} \tag{21}$$

where we used the approximation based on $\tilde{\mathbf{z}}$. The prediction by expert k is

$$p(f_* | \mathbf{x}_*, z_* = k, \mathbf{y}_k, \mathbf{U}_k) = \mathcal{N}(f_*; \mu_*, \sigma_*^2), \text{ where}$$

$$\mu_* = \kappa(\mathbf{x}_*, \mathbf{U}_k) \mathbf{\Psi}_k^{-1} \mathbf{K}(\mathbf{U}_k, \mathbf{X}_k)(\mathbf{\Lambda}_k(\mathbf{X}_k) + \sigma_k^2 \mathbf{I})^{-1} \mathbf{y}_k,$$

$$\sigma_*^2 = \kappa(\mathbf{x}_*, \mathbf{x}_*) - \kappa(\mathbf{x}_*, \mathbf{U}_k) \left((\mathbf{K}_{\mathbf{u},\mathbf{u}}^{(k)})^{-1} - \mathbf{\Psi}_k^{-1} \right) \kappa(\mathbf{U}_k, \mathbf{x}_*).$$

In practice, we find that the prediction by the nearest expert is better than the weighted prediction. This is reflective of the way we train the model: since we use MAP assignment of data points to experts, most experts are ignorant of the points not assigned to them. Hence prediction should be made by the localized experts to eliminate the poor contribution from the remote experts.

The MAP assignment is also used to compute the approximate *log marginal likelihood*:

$$\log\, p(\mathbf{y}|\tilde{\mathbf{z}}, \mathbf{U}, \boldsymbol{\theta}, \boldsymbol{\sigma}) = \\ -\frac{1}{2}\sum_{k=1}^{K} \log|\mathbf{Q}_k + \boldsymbol{\Lambda}_k(\mathbf{X}_k) + \sigma_k^2\mathbf{I}| \\ -\frac{1}{2}\sum_{k=1}^{K} \mathbf{y}_k^T \left(\mathbf{Q}_k + \boldsymbol{\Lambda}_k(\mathbf{X}_k) + \sigma_k^2\mathbf{I}\right)^{-1} \mathbf{y}_k, \quad (22)$$

where $\mathbf{Q}_k = \kappa(\mathbf{X}_k, \mathbf{U}_k)(\mathbf{K}_{\mathbf{u},\mathbf{u}}^{(k)})^{-1}\kappa(\mathbf{U}_k, \mathbf{X}_k)$.

The optimization proceeds iteratively as follows. In the E-step, we calculate the MAP assignment $\tilde{\mathbf{z}}$ based on Equation 14 and 18 using the new posterior mean $\mathbb{E}[\mathbf{g}_k] = \boldsymbol{\mu}_k$. The cost of this operation is $\mathcal{O}(NM^2S)$ in computation and $\mathcal{O}(NM/S)$ in memory, where we divide the dataset into S batches to fit in the memory (typically $S = K$). In the M-step, we optimize the log marginal likelihood wrt the hyperparameters using gradient-based methods. If each expert is assigned $\mathcal{O}(N/K)$ data points, this step requires $\mathcal{O}(K \times \frac{N}{K}M^2) = \mathcal{O}(NM^2)$ in computation and $\mathcal{O}(NM/K)$ in memory.

3.3. Fast Allocation for Large-scale Inference

The $\mathcal{O}(NM^2K)$ and $\mathcal{O}(NM/K)$ (taking $S = K$) complexity of the above inference procedure arises from the computation of the MAP assignment. However, observe that $\ln \rho_{nk}$ (Equation 14) comprises two terms which become larger as $\mathbf{x}_n$ gets closer to the expert centroid $\mathbf{m}_k$. The first term $\ln \mathcal{N}(\mathbf{x}_n; \mathbf{m}_k, \mathbf{V})$ increases as the (Mahalanobis) distance between $\mathbf{x}_n$ and $\mathbf{m}_k$ decreases. The second term measures the quality of prediction by expert k; this is better when $\mathbf{x}_n$ is similar to the inducing inputs of the expert (which is more likely when $\mathbf{x}_n$ is near $\mathbf{m}_k$). This motivates us to assign points to experts based on the inducing inputs, which means the MAP assignment $\tilde{\mathbf{z}}$ is replaced by:

$$\hat{z}_n = \underset{k}{\text{argmin}}(\mathbf{x}_n - \mathbf{m}_k)^T\mathbf{V}^{-1}(\mathbf{x}_n - \mathbf{m}_k). \quad (23)$$

As we shall see in our experiments in section 4, this seemingly coarse approximation works well in practice. One explanation is that the inducing inputs are optimized to increase the marginal likelihood in the M-step; thus, information from the likelihood is effectively propagated into the allocation step via the inducing inputs.

Using this assignment, we no longer need to compute the right-hand side term that dominates the computation of the MAP assignment in Equation 14. The time and memory complexity is thus reduced to $\mathcal{O}(NM^2)$ and $\mathcal{O}(NM/K)$ respectively. Compared to traditional sparse GPs with the same number of inducing points, i.e., $B = M \times K$, *our model can potentially run K^2 times faster using K^2 times less memory*. Indeed, our experiments in Section 4.2 empirically show that the model achieves the K^2 or higher computational speed-up while obtaining comparable predictive performance. The fast allocation enables the model to scale to very large datasets, such as the one we used in our experiments where $N = 10^5$ training points. By setting $K = 20$ and $M = 300$, a total number of 6,000 inducing points is easily afforded even on a standard desktop computer. In contrast, it is simply impossible to run either standard or traditional sparse GPs on problems of this size.

A further advantage of this approximation is the ease with which parallel or distributed computations can be implemented. This is because each term in the objective function (Equation 22 with $\hat{\mathbf{z}}$ replacing $\tilde{\mathbf{z}}$) can be computed independently without the need for communication or synchronization per iteration. We give an example of the computational speed-up obtained with parallelization in Section 4.3.

4. Experiments

We perform three different sets of experiments with datasets of varying size to demonstrate different aspects of the model. Our main focus, however, will be on the utility of this model when applied to large datasets. We use the squared exponential (SE) covariance function with automatic relevance determination (ARD) in all experiments. No transformation of the inputs and outputs is performed (except for the large dataset which we detail in Section 4.3). The initial inducing locations are randomly selected from the inputs and the hyperparameters are initialized based on the scale of the input features. The experiments are executed on an Intel(R) Core(TM) i7-2600 3.40GHz CPU with 8GB of RAM using Matlab R2012a. We measure the training time as the time to learn the hyperparameters and inducing inputs after initialization. Other costs such as initialization or prediction are negligible compared to the training time. We optimize the hyperparameters using the conjugate gradients code in the GPML package (Rasmussen & Nickisch, 2010) and limit the maximum number of function evaluations to 1000. The code for this paper is available at `http://trungngv.github.io/fagpe/`.

4.1. Toy dataset

First we evaluate the ability of the model to handle non-stationarity in the *motorcycle* dataset (Silverman, 1985). The dataset (Figure 2) contains 133 data points with input-dependent noise. We set $K = 2$ experts with $M = 20$ inducing inputs per expert. We run the experiment 5 times and select the configuration with the best (training) objective value. Our result is similar to that of Rasmussen & Ghahramani (2002) (also included in Figure 2) and Yuan & Neubauer (2009) at inputs ≤ 30. At inputs > 35, the

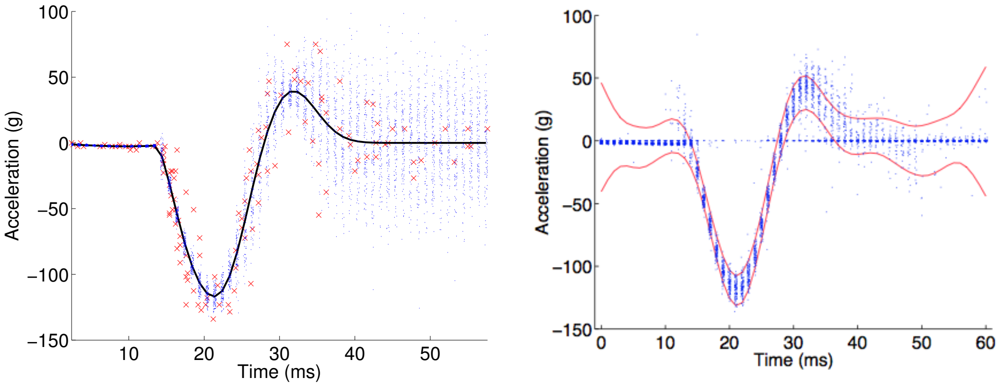


Figure 2. The left hand plot shows the training data (red crosses) and the predictive mean by the model (solid black line). The right hand plot is taken from Rasmussen & Ghahramani (2002) for comparison. The blue dots in both figures are samples (jittered for visibility) of the noise free predictive distribution evaluated at intervals of 1ms (100 samples per point). The two red lines in the right hand plot are the ± 2 std error confidence interval of prediction by a stationary GP.

result of Rasmussen & Ghahramani (2002) underestimates the noise, whereas our result and that of Yuan & Neubauer (2009) both capture the artifacts in that region. Training time for this dataset took 30 seconds, compared to one hour by Gibbs sampling in Rasmussen & Ghahramani (2002).

4.2. Medium-size Datasets

Next we evaluate the predictive performance and training time of our method on three datasets: *kin40k* (8 dimensions, 10000 training, 30000 testing), *pumadyn-32nm* (32 dimensions, 7168 training, 1024 testing), and *pole telecom* (26 dimensions, 10000 training, 5000 testing). We use the exact split as in Lázaro-Gredilla et al. (2010) and Snelson & Ghahramani (2006).

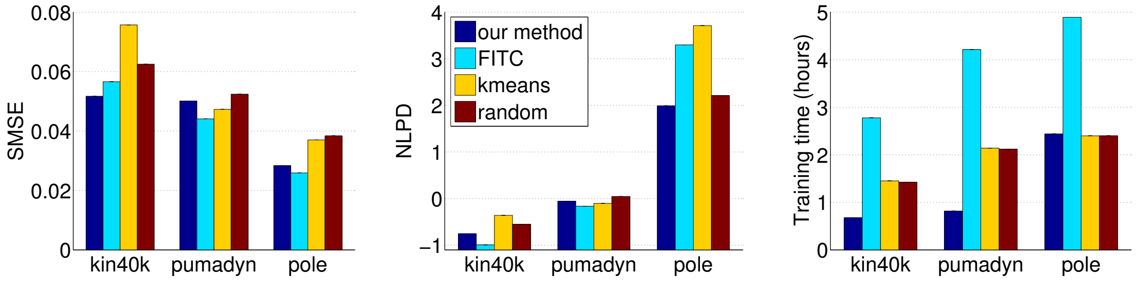
Our first baseline is the FITC approximation (Quiñonero-Candela & Rasmussen, 2005). Note that for $K = 1$, our model is identical to FITC. Our second baseline is *local* FITC which divides the training points into K clusters and runs FITC on each separate cluster. Various methods can be used to partition the dataset; here we employ k-means and random clustering, which we refer to as the *kmeans* and *random* method. We run each method 5 times with different initializations and select the best configuration according to the marginal likelihood to avoid bad local optima. For local FITC, the objective value is approximated as the sum of the marginals of all clusters. The total number of inducing inputs in all experiments is 1500. We set $K = 2$ experts and correspondingly $M = 750$ inducing inputs per expert (or cluster) for our method and also for local FITC.

The predictive performance and training time of all methods are shown in Figure 3. First we compare our method with global FITC and see that a speed-up factor of around $K^2 = 4$ is indeed achieved by our method. More importantly, this significant gain comes with comparable predictive performance. For the *pole* dataset, a small loss in accuracy is reversed by our method being more confident about its prediction. Here it may be instructive to remind that our model is equivalent to FITC when $K = 1$. When working at small or medium scales, we can use model selection to choose the best configuration (i.e. in terms of the number of experts and inducing inputs) given a dataset. In fact, the optimal setting based on the best marginal likelihood is $K = 1$ except for *pumadyn-32nm*, which has many bad local optima. However, when the computational budget is limited or the dataset is too big for global FITC to handle, the model we propose is valuable as it allows scalability and efficiency while achieving good performance. Other methods (*kmeans* and *random*) have their computational advantage over FITC countered by their prediction loss (see Figure 3). Our model is more accurate and faster than local FITCs because it sensibly assigns data points to the experts, based on their inducing points, which makes learning easier and also more effective.

4.3. Large-scale Experiments

In this section, we evaluate our model on the Million Song Dataset (Bertin-Mahieux et al., 2011). We extract the first 10^5 songs from this dataset for training and keep the original set of 51630 songs for testing. The goal is to predict the years in which songs are released based on their 90 acoustic features. We transform the outputs to have zero mean for all experiments. We set $K = 20$ experts and $M = 300$ inducing inputs per expert [2]. Initially the data points are assigned to experts uniformly at random or using recursive projection clustering (RPC), a clustering technique proposed in

[2] This choice is directed by the physical memory limit of the benchmark computer, which we expect to be the case when dealing with large datasets on a single machine.



(a) smse (b) nlpd (c) training time

Figure 3. Predictive performance and training time of our method compared to FITC and local FITC with kmeans and random clustering. The standardized mean square error (SMSE) and negative log predictive density (NLPD) averaged across all test points are reported; smaller is better.

Chalupka et al. (2012) that is more efficient than k-means and tends to give more balanced cluster sizes. We denote our model with the random and RPC initialization as FGP-RANDOM and FGP-RPC method respectively.

We evaluate our model against six other competitive baselines. The first baseline is the *local* FITC model described in the previous section, with random or RPC assignments of the data points to clusters. Analogous to our model, we refer to them as FITC-RANDOM and FITC-RPC. The second baseline (GPSVI2000) is FITC with stochastic variational inference (Hensman et al., 2013) training using $B = 2000$ inducing points. Note that GPSVI has quadratic storage complexity $\mathcal{O}(B^2)$ which limits the total number of inducing points that can be used. Unlike our model and local FITC, the inducing locations cannot be learned and must be selected on some ad hoc basis. In addition to random selection, we also clustered the dataset into partitions using RPC and k-means and used the centroids as the inducing inputs. We obtained essentially identical results with k-means selection so its results are reported here. The third baseline (SOD2000) is the standard GP regression model where a subset of 2000 data points is randomly sampled for training and the rest is discarded. For all of these GP-based methods, we repeat the experiments 5 times with different initialization of parameters in the corresponding models. We thus report their performance with means and standard deviations over the 5 runs.

The remaining baselines include CONSTANT, which predicts the mean of the outputs; nearest neighbors with $k = 1$ (NN1) and $k = 50$ (NN50) neighbors; and linear regression (LR) – these were used in Bertin-Mahieux et al. (2011). Table 1 shows the results of all methods in terms of the predictive accuracy (SMSE and MAE) and confidence (NLPD). The CONSTANT method (mean of the outputs is 1998) is quite effective, but uninteresting, due to the fact that a large portion of the the songs were released in the recent years. The nearest neighbors methods perform even worse than prediction using the constant mean. Linear regression does only slightly better than two of the GP-based methods namely FITC-RPC and SOD2000. Overall our model is significantly better than all of the competing methods. In particular, it is more accurate (for e.g. in terms of MAE) than all but GPSVI2000 by at least 0.27 year per song on average. This amounts to approximately 14,000 years in total, which is clearly a meaningful improvement. Furthermore there is noticeable difference in the log predictive density of FGP-RANDOM and FGP-RPC compared to the rest, which can be attributed to our model having localized experts. This encouraging result suggests the benefits of our model when dealing with very large datasets compared to a global function or model (like linear regression or FITC), which may not realistically capture the characteristics of the output space.

Table 1. Test performance of the models on the Million Song Dataset. MAE is the mean absolute error and SMSE and NLPD are as defined previously. All GP-based methods are reported with standard deviation over 5 runs. Our method (FGP-RANDOM and FGP-RPC) significantly outperforms all other baselines.

METHOD	SMSE	MAE	NLPD
FGP-RANDOM	**0.715 ± 0.003**	**6.47 ± 0.02**	3.59 ± 0.01
FGP-RPC	0.723 ± 0.003	6.48 ± 0.02	**3.58 ± 0.01**
FITC-RANDOM	0.761 ± 0.009	6.74 ± 0.07	3.63 ± 0.03
FITC-RPC	0.832 ± 0.027	7.11 ± 0.23	3.73 ± 0.07
GPSVI2000	0.724 ± 0.005	6.53 ± 0.04	3.64 ± 0.01
SOD2000	0.794 ± 0.011	6.94 ± 0.08	3.69 ± 0.01
LR	0.770	6.846	NA
CONSTANT	1.000	8.195	NA
NN1	1.683	9.900	NA
NN50	1.332	8.208	NA

It is also interesting to discuss the trade-off between time and accuracy of the non-trivial methods. The training time for linear regression is only 5 seconds – embarrassingly superior to the rest. Although both FGP-RANDOM and FGP-

RPC take around 8.5 hours to train, the performance gain is significant enough that it is worth the trade-off. Simpler methods such as FITC-RANDOM, FITC-RPC, and SOD2000 take 19.1, 8.8, and 3.7 hours respectively despite offering no improvement over linear regression. GPSVI2000 takes 1.6 hours to train but its predictive power is inferior to that of our method.

Finally, we remark the small variability in performance of both FGP-RANDOM and FGP-RPC as well as other FITC-based methods. This demonstrates the stability of this kind of models even under the presence of local optima. Experience tells us that local optima is generally not an issue when the data is abundant, as is the case with this large dataset.

4.3.1. Computational Speed-up with a Multicore Implementation

We have also implemented a parallel version for computation of the marginal likelihood and its derivatives, using the freely available *multicore* package[3]. Note that this parallelization package is crude: it distributes the computation of the objective and its derivatives of each expert to a local MATLAB session and the sessions communicate by writing to and reading from local disks. Preliminary experiments using 4 sessions (corresponding to 4 cores) give a speed-up factor of 1.5 to 2. The speed-up factor should increase if a better library/framework such as the MATLAB Parallel Toolbox (which unfortunately requires additional license) is used or if higher number of cores is available.

5. Related Work

Most related work has been described in the Introduction. Here we provide further details. We first discuss the existing mixture of GP experts models (Gramacy & Lee, 2008; Kim et al., 2005; Meeds & Osindero, 2006; Rasmussen & Ghahramani, 2002; Tresp, 2001; Yuan & Neubauer, 2009) and their computational limits. Common to these work is the assumption that individual experts are independent standard GP regression models, except for Yuan & Neubauer (2009) where each expert is a linear model that parameterizes a GP. In Gramacy & Lee (2008); Kim et al. (2005); Meeds & Osindero (2006); Rasmussen & Ghahramani (2002) the gating networks divide the data into disjoint subsets and each expert is responsible for *only one* subset. This constraint is also used in our model and it has the benefit of reducing the computation for each expert to depend only on the data points assigned to it. However, inference in these models is burdened by the intractability of the posterior of the latent assignments and learning of the hyperparameters. Most of these methods use intensive MCMC sampling which costs at least $\mathcal{O}(N^2)$ per iteration to obtain a sample. Thus, mixture of GP experts have largely been run on small problems to demonstrate the ability to capture non-stationarity rather than scalability.

The sparse approximations for GP regression (Lawrence et al., 2002; Seeger, 2003; Seeger et al., 2003; Smola & Bartlett, 2001; Snelson & Ghahramani, 2006; Williams & Seeger, 2001) are quite different from the mixture models in their philosophy. In particular, the inducing points give rise to the statistical independence in the data, directly determining the accuracy of the approximation. As such, the cardinality of the inducing set must grow with the size and dimension of the problems. However, due to the high time and storage complexity of traditional sparse GPs and stochastic variational inference (Hensman et al., 2013) as a function of the number of inducing points, the inducing set cardinality is very limited. Thus, when the data dimension grows, most data points will be supported by only a few or even no inducing points; in other words, most inducing points are irrelevant and thus wasteful to the observations remote from their neighborhoods. We call this phenomenon the *curse of inducing cardinality*. On the contrary, our model employs localized experts, each with its distinct set of inducing points, thereby ensuring that the data points in different local regions are sufficiently supported. This preserves the statistical independence assumption that is integral to the predictive performance of the approximation.

6. Discussion

We have presented a model underpinned by the mixture of experts and the inducing points frameworks. However, inference in our model is radically different from its mixture counterpart, as it can handle much larger datasets. It is also more flexible than traditional sparse GPs thanks to the localized experts in its composition that allow more complex patterns to be represented. The effectiveness of the model and its learning algorithm was empirically verified on 3 different sets of experiments, which demonstrated its scalability and predictive power. Our results encourage the use and development of multiple experts architectures when dealing with very large datasets, which may not be adequately modeled with a global function or process.

Acknowledgments

NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

[3] http://mathworks.com/matlabcentral/fileexchange/13775

References

Bertin-Mahieux, Thierry, Ellis, Daniel PW, Whitman, Brian, and Lamere, Paul. The million song dataset. In *Proceedings of the International Society for Music Information Retrieval Conference*, 2011.

Bonilla, Edwin V., Chai, Kian Ming A., and Williams, Christopher K. I. Multi-task Gaussian process prediction. In *Advances in Neural Information Processing Systems*. 2008.

Chalupka, Krzysztof, Williams, Christopher KI, and Murray, Iain. A framework for evaluating approximation methods for Gaussian process regression. *Journal of Machine Learning Research*, 14:333–350, 2012.

Gramacy, Robert B and Lee, Herbert KH. Bayesian treed gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, 103(483):1119–1130, 2008.

Hensman, James, Fusi, Nicolo, and Lawrence, Neil D. Gaussian processes for big data. In *Uncertainty in Artificial Intelligence*. 2013.

Kersting, Kristian, Plagemann, Christian, Pfaff, Patrick, and Burgard, Wolfram. Most likely heteroscedastic Gaussian process regression. In *International Conference on Machine Learning*, 2007.

Kim, Hyoung-Moon, Mallick, Bani K, and Holmes, CC. Analyzing nonstationary spatial data using piecewise Gaussian processes. *Journal of the American Statistical Association*, 100(470):653–668, 2005.

Lawrence, Neil D., Seeger, Matthias, and Herbrich, Ralf. Fast sparse Gaussian process methods: The informative vector machine. In *Advances in Neural Information Processing Systems*. 2002.

Lázaro-Gredilla, Miguel, Quiñonero-Candela, Joaquin, Rasmussen, Carl Edward, and Figueiras-Vidal, Aníbal R. Sparse spectrum Gaussian process regression. *The Journal of Machine Learning Research*, 99:1865–1881, 2010.

Meeds, Edward and Osindero, Simon. An alternative infinite mixture of Gaussian process experts. In *Advances in Neural Information Processing Systems*. 2006.

Paciorek, Christopher J. and Schervish, Mark J. Nonstationary covariance functions for Gaussian process regression. In *Advances in Neural Information Processing Systems*. 2004.

Quiñonero-Candela, Joaquin and Rasmussen, Carl Edward. A unifying view of sparse approximate Gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959, 2005.

Rasmussen, Carl Edward and Ghahramani, Zoubin. Infinite mixtures of Gaussian process experts. In *Advances in Neural Information Processing Systems*. 2002.

Rasmussen, Carl Edward and Nickisch, Hannes. Gaussian processes for machine learning (gpml) toolbox. *The Journal of Machine Learning Research*, 11:3011–3015, 2010.

Seeger, Matthias. *Bayesian Gaussian process models: PAC-Bayesian generalisation error bounds and sparse approximations*. PhD thesis, School of Informatics, University of Edinburgh, 2003.

Seeger, Matthias, Williams, Christopher KI, and Lawrence, Neil D. Fast forward selection to speed up sparse Gaussian process regression. In *Workshop on AI and Statistics*, 2003.

Silverman, Bernhard W. Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 1–52, 1985.

Smola, Alex J and Bartlett, Peter. Sparse greedy Gaussian process regression. In *Advances in Neural Information Processing Systems*, 2001.

Snelson, Ed and Ghahramani, Zoubin. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*. 2006.

Tresp, Volker. Mixtures of Gaussian processes. In *Advances in Neural Information Processing Systems*. 2001.

Williams, Christopher 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.