Ultra-fast Deep Mixtures of Gaussian Process Experts

Clement Etienam
Department of Mathematics
University of Manchester
clement.etienam@manchester.ac.uk

Kody Law
Department of Mathematics
University of Manchester
kody.law@manchester.ac.uk

Sara Wade
School of Mathematics
University of Edinburgh
sara.wade@ed.ac.uk

Abstract

Mixtures of experts have become an indispensable tool for flexible modelling in a supervised learning context, and sparse Gaussian processes (GP) have shown promise as a leading candidate for the experts in such models. In the present article, we propose to design the gating network for selecting the experts from such mixtures of sparse GPs using a deep neural network (DNN). This combination provides a flexible, robust, and efficient model which is able to significantly outperform competing models. We furthermore consider efficient approaches to computing maximum a posteriori (MAP) estimators of these models by iteratively maximizing the distribution of experts given allocations and allocations given experts. We also show that a recently introduced method called Cluster-Classify-Regress (CCR) is capable of providing a good approximation of the optimal solution extremely quickly. This approximation can then be further refined with the iterative algorithm.

1 Introduction

Gaussian processes (GPs) are key components of many statistical and machine learning models. In a Bayesian setting, they provide a probabilistic approach to model unknown functions, which can subsequently be used to quantify uncertainty in predictions. An introduction and overview of GPs in machine learning is given in [38].

In regression tasks, the GP is a popular prior for the unknown regression function, $f : x \to y$, due to its nonparametric nature and tractability. It assumes that the function evaluated at any finite set of inputs $(x_1, \ldots, x_N)$ is Gaussian distributed with mean vector $(\mu(x_1), \ldots, \mu(x_N))$ and covariance matrix with elements $K(x_i, x_j)$, where the mean function $\mu(\cdot)$ and the positive semi-definite covariance (or kernel) function $K(\cdot, \cdot)$ represent the parameters of the GP.

While GPs are flexible and have been successfully applied to various problems, limitations exist. First, typically parametric forms are specified for $\mu(\cdot)$ and $K(\cdot, \cdot)$, which crucially determine properties of the regression function, such as spatial correlation, smoothness, and periodicity. This limits the model's ability to recover changing behavior of the function, e.g. different smoothness levels across the input space. Second, GP models suffer from a high computational burden, due to the need to invert large or dense covariance matrices.

Mixtures of experts (MoEs) provide a framework to address both issues. First introduced in [19], MoEs probabilistically partition the input space into regions, and within each region, a local expert specifies the conditional model for the output y given the input x. A gating network is used to map

Preprint. Under review.

the experts to local regions of the input space. Thus, any simplifying assumptions of the experts need only hold locally within each region and scalability is enhanced as each expert only considers its local region. When employing GPs as experts, this allows the model to 1) infer different behaviors, such as smoothness and variability, within each region and 2) address the computational burden through local approximations that only require inversion of smaller matrices based on local subsets of the data.

In this paper, our contribution is threefold. First, we construct a novel MoE model that combines the expressive power of deep neural networks (DNNs) and the probabilistic nature of GPs. While powerful, DNNs lack the probabilistic framework and sound uncertainty quantification of GPs, and there has been increased interest in recent years in combining DNNs and GPs to benefit from the advantages and overcome the limitations of each method, see e.g. [17, 39, 18, 7] to name a few. Specifically, we use GP experts for smooth, probabilistic reconstructions of the unknown regression function within each local region, while employing DNNs for the gating network to flexibly determine the local regions. To further enhance scalability, we combine the local approximation of GPs through the MoE architecture with low-rank approximations using an inducing point strategy [35]. This combination leads to a robust and efficient model which is able to outperform competing models.

Second, we provide a connection between optimization algorithms commonly used to estimate MoEs and the recently introduced method called Cluster-Classify-Regress (CCR) [3]. Lastly, this novel connection is used to obtain an ultra-fast, accurate approximation of the proposed deep mixture of sparse GP experts.

2 Methodology

For independent and identically distributed data $(y_1, \dots y_N)$, mixture models are an extremely useful tool for flexible density estimation due to their attractive balance between smoothness and flexibility. When additional covariate information is present and the data consists of input-output pairs, $\{(x_i, y_i)\}_{i=1}^N$, MoEs extend mixtures by modelling the mixture parameters as functions of the inputs. This is achieved by defining the gating network, which probabilistically partitions the input space into regions, and by specifying the experts, which characterize the local relationship between x and y. This results in flexible framework which has been employed in numerous applications; for a recent overview, see [11].

Specifically, the MoE model assumes that outputs are independently generated as:

$$y_i|x_i \sim \sum_{l=1}^{L} g_l(x_i;\theta_c)\mathrm{N}(y_i \mid f_r(x_i;\theta_r^l), \sigma_r^{2\,l}) \quad \text{for } i = 1, \dots, N, \tag{1}$$

where $g_l(\cdot;\theta_c)$ is the gating network with parameters θ_c; $f_r(\cdot;\theta_r^l)$ is the local regression function with parameters θ_r^l; and L represents the number of experts. For simplicity, we focus on the case when $y \in \mathbb{R}$ and employ a Gaussian model for the experts, although this may generalized for other data types. MoEs can be augmented with a set of allocation variables $\mathbf{z} = (z_1, \dots, z_N)$, where $z_i = l$ if the i^{th} data point is generated from the l^{th} expert. Letting $\mathbf{y} = (y_1, \dots, y_N)$ and $\mathbf{x} = (x_1, \dots, x_N)$, the augmented model is

$$p(\mathbf{y}|\mathbf{x}, \mathbf{z}) = \prod_{i=1}^{N} \mathrm{N}(y_i \mid f_r(x_i;\theta_r^{z_i}), \sigma_r^{2\,z_i}) = \prod_{l=1}^{L} \prod_{i:z_i=l} \mathrm{N}(y_i \mid f_r(x_i;\theta_r^l), \sigma_r^{2\,l})\,,$$

$$p(\mathbf{z}|\mathbf{x}) = \prod_{i=1}^{N} g_{z_i}(x_i;\theta_c) = \prod_{l=1}^{L} \prod_{i:z_i=l} g_l(x_i;\theta_c)\,,$$

and (1) is recovered after marginalization of $\mathbf{z}$. Thus, the gating network $(g_1(\cdot;\theta_c), \dots, g_L(\cdot;\theta_c))$, which maps the input space $\mathcal{X} \subseteq \mathbb{R}^d$ to the $L-1$ dimensional simplex, is a classifier that reflects the relevance of each expert at any location $x \in \mathcal{X}$.

Various formulations have been proposed in literature for both the experts and gating networks, ranging from simple linear models to flexible non-linear approaches. Examples include (generalized) linear or semi-linear models [20, 40], splines [33], neural networks [4, 1], Gaussian processes [37,

31], tree-based classifiers [12], and others. In this work, we combine sparse GP experts with DNN gating networks to flexibly determine local regions and provide a probabilistic and nonparametric model of the unknown regression function.

2.1 Sparse Gaussian process experts

Mixtures of GP experts have proven to be very successful [37, 31, 27, 41, 29, 10]. In particular, they overcome limitations of stationary Gaussian process models by reducing the computational complexity through local approximations and allow different local properties of the unknown function to handle challenges, such as discontinuities, non-stationarity and non-normality. In this case, one assumes a GP prior on the local regression function with hyperparameters $\theta_r^l = (\mu^l, \psi^l)$:

$$f_r(\cdot; \theta_r^l) \sim \text{GP}(\mu^l, K_{\psi^l}),$$

where μ^l is the local mean function of the expert (for simplicity, it assumed to be constant) and ψ^l are the parameters of the covariance function K_{ψ^l}, whose chosen form and hyperparameters encapsulate properties of the local function such as the spatial correlation, smoothness, and periodicity. While GP experts are appealing due to their flexibility, intrepretability and probabilistic nature, they increase the computational cost of the model significantly. Indeed, given the allocation variables $\mathbf{z}$, the GP hyperparameters, which crucially determine the behavior of the unknown function, can be estimated by optimizing the log marginal likelihood:

$$\log(p(\mathbf{y}|\mathbf{x}, \mathbf{z})) = \sum_{l=1}^{L} \log\left(\text{N}(\mathbf{y}^l \mid \boldsymbol{\mu}^l, \mathbf{K}_{N_l}^l + \sigma_r^{2\,l}\mathbf{I}_{N_l})\right),$$

where $\mathbf{y}^l$ and $\mathbf{x}^l$ contain the outputs and inputs of the l^{th} cluster, i.e. $\mathbf{y}^l = \{y_i\}_{z_i=l}$ and $\mathbf{x}^l = \{x_i\}_{z_i=l}$; $\boldsymbol{\mu}^l$ is a vector with entries μ^l; $\mathbf{K}_{N_l}^l$ represents the $N_l \times N_l$ matrix obtained by evaluating the covariance function K_{ψ^l} at each pair of inputs in the l^{th} cluster; and N_l is number data points in the l^{th} cluster. This however requires inversion of $N_l \times N_l$ matrices, which scales $\mathcal{O}(\sum_{l=1}^{L} N_l^3)$. While this reduces the computational complexity compared with standard GP models which scale $\mathcal{O}(N^3)$, it can still be costly.

To improve scalability, one can resort to approximate methods for GPs, including sparse GPs based on a set of inducing points or pseudo inputs [35, 36, 5], the predictive process approach used in spatial statistics [2], basis function approximations [6], or sparse formulations of the precision matrix [25, 13, 8], among others (see [15] for a recent review of approaches in spatial statistics and [38, Chp. 8] for a review of approaches in machine learning). In the present work, we employ an inducing point strategy, assuming the local likelihood of the data points within each cluster factorizes given a set of $M_l < N_l$ pseudo-inputs $\tilde{\mathbf{x}}^l = (\tilde{x}_1^l, \ldots, \tilde{x}_{M_l}^l)$ and pseudo-targets $\tilde{\mathbf{f}}^l = (\tilde{f}_1^l, \ldots, \tilde{f}_{M_l}^l)$;

$$p(\mathbf{y}^l|\mathbf{x}^l, \tilde{\mathbf{x}}^l, \tilde{\mathbf{f}}^l) = \prod_{i:z_i=l} \text{N}(y_i|\widehat{\mu}_i^l, \widehat{\sigma}_i^{2\,l}), \tag{2}$$

where

$$\widehat{\mu}_i^l = \mu^l + (\mathbf{k}_{M_l,i}^l)^T(\mathbf{K}_{M_l}^l)^{-1}(\tilde{\mathbf{f}}^l - \boldsymbol{\mu}^l),$$
$$\widehat{\sigma}_i^{2\,l} = \sigma_r^{2\,l} + K_{\psi^l}(x_i, x_i) - (\mathbf{k}_{M_l,i}^l)^T(\mathbf{K}_{M_l}^l)^{-1}\mathbf{k}_{M_l,i}^l,$$

where $\mathbf{K}_{M_l}^l$ is the $M_l \times M_l$ matrix with elements $K_{\psi^l}(\tilde{x}_j^l, \tilde{x}_h^l)$ and $\mathbf{k}_{M_l,i}^l$ is the vector of length M_l with elements $K_{\psi^l}(\tilde{x}_j^l, x_i)$. This corresponds to the fully independent training conditional (FITC) approximation [30]. After marginalization of the pseudo-targets under the GP prior $\tilde{\mathbf{f}}^l \sim \text{N}(\boldsymbol{\mu}^l, \mathbf{K}_{M_l}^l)$, the pseudo-inputs $\tilde{\mathbf{x}}^l$ and hyperparameters (μ^l, ψ^l) can be estimated by optimizing the marginal likelihood:

$$\log(p(\mathbf{y}|\mathbf{x}, \mathbf{z}, \tilde{\mathbf{x}})) = \sum_{l=1}^{L} \log\left(\text{N}(\mathbf{y}^l \mid \boldsymbol{\mu}^l, (\mathbf{K}_{M_l N_l}^l)^T(\mathbf{K}_{M_l}^l)^{-1}\mathbf{K}_{M_l N_l}^l + \boldsymbol{\Lambda}^l + \sigma_r^{2\,l}\mathbf{I}_{N_l})\right), \tag{3}$$

where $\mathbf{K}_{M_l N_l}^l$ is the $M_l \times N_l$ matrix with columns $\mathbf{k}_{M_l,i}^l$ and $\boldsymbol{\Lambda}^l$ is the diagonal matrix with diagonal entries $K_{\psi^l}(x_i, x_i) - (\mathbf{k}_{M_l,i}^l)^T(\mathbf{K}_{M_l}^l)^{-1}\mathbf{k}_{M_l,i}^l$. This strategy allows us to reduce the complexity to $\mathcal{O}(\sum_{l=1}^{L} N_l M_l^2)$.

2.2 Deep neural gating networks

GP experts have been combined with different gating networks, including tree-based [12], naive Bayes [29], and GP [37] classifiers. In order to flexibly determine the local regions, especially in multivariate input spaces, while also retaining scalable inference, we employ the expressive power of DNNs. Specifically, we define the gating network by a feedforward DNN with a softmax output:

$$g_l(x;\theta_c) = \frac{\exp(h_l(x;\theta_c))}{\sum_{j=1}^{L}\exp(h_j(x;\theta_c))}\,,$$

where h_l is the l^{th} component of $h:\mathbb{R}^d \to \mathbb{R}^L$, defined by

$$h(\ \cdot\ ;\theta_c) = \eta_J(\eta_{J-1}(\cdots\eta_1(\ \cdot\ ;\theta_c^1)\cdots;\theta_c^{J-1});\theta_c^J)\,,$$

with $\eta_j:\mathbb{R}^{d_{j-1}}\to\mathbb{R}^{d_j}$ ($d_0=d$, $d_J=L$) the j^{th} layer of a neural network

$$\eta_j(\ \cdot\ ;\theta_c^j): x \mapsto \eta_j(x;\theta_c^j) = \mathrm{ReLU}(A_j x + b_j)\,,$$

where $\mathrm{ReLU}(x) = \max\{0,x\}$ is the element-wise rectifier, and $\theta_c^j = \{A_j, b_j\}$ comprises the weights $A_j \in \mathbb{R}^{d_j\times d_{j-1}}$ and biases $b_j\in\mathbb{R}^{d_j}$ for level $j=1,\ldots,J$.

Deep neural gating networks have been used in literature but are typically combined with DNN experts [4, 1]. The mixture density network [4] uses this gating network but parametrizes both the local regression function and variance of the Gaussian model in (1) by DNNs. This offers considerable flexibility beyond standard DNN regression, but significant valuable information can be gained with GP experts. Specifically, as the number of data points in each cluster is data-driven, DNN experts may overfit due to small cluster sizes. Instead, GP experts probabilistically model the local regression function, avoiding overfitting and providing uncertainty quantification.

3 Inference

We focus on maximum likelihood estimation (MLE) of the model parameters $(\theta_c, \theta_r, \sigma_r^2)$, which can be considered a special case of maximum a posteriori (MAP) estimation in the Bayesian model with vague priors $\pi(\theta_c,\theta_r,\sigma_r^2)\propto 1$. For GP experts, this is often called maximum marginal likelihood estimation or type II MLE of the GP hyperparameters $(\theta_r^l, \sigma_r^{2\,l})$, as the local GP regression functions are marginalized. For sparse GP experts, we have the additional parameters $(\tilde{\mathbf{x}}^l, \tilde{\mathbf{f}}^l)$. The pseduo-inputs $\tilde{\mathbf{x}}^l$ are treated as hyperparameters to be optimized, and while pseudo-targets $\tilde{\mathbf{f}}^l$ can be analytically integrated out, we also estimate $\tilde{\mathbf{f}}^l$ and discuss how this leads to faster inference.

First note that directly optimizing the log posterior is challenging due to identifiability issues of mixtures. While an expectation-maximization (EM) algorithm can be used, we instead focus on the faster maximization-maximization (MM) strategy [24] to optimise the augmented log posterior:

$$\begin{aligned}\log(\pi(\theta_c,\theta_r,\sigma_r^2,\mathbf{z},\tilde{\mathbf{f}}|\mathbf{y},\mathbf{x},\tilde{\mathbf{x}})) =& \text{const.} + \sum_{i=1}^{N}\log(g_{z_i}(x_i;\theta_c)) + \sum_{i=1}^{N}\log(\mathrm{N}(y_i \mid \widehat{\mu}_i^{z_i}, \widehat{\sigma}_i^{2\,z_i})) \\ &+ \sum_{l=1}^{L}\log(\mathrm{N}(\tilde{\mathbf{f}}^l \mid \boldsymbol{\mu}^l, \mathbf{K}^l_{M_l})).\end{aligned} \tag{4}$$

This is an iterative conditional modes algorithm [22] that alternates between optimizing the allocation variables $\mathbf{z}$ and the model parameters. It guaranteed to never decrease the log posterior of the augmented model and therefore will converge to a fixed point [24]. However, it is susceptible to local maxima, which can be alleviated with multiple restarts of random initializations.

3.1 Maximization-maximization

The MM algorithm iterates over the following two steps: 1) **optimize the allocation variables**:

$$\mathbf{z} = \operatorname*{argmax}_{\mathbf{l}\in\{1,\ldots,L\}^N} \log\pi(\mathbf{l}|\theta,\mathbf{x},\mathbf{y})\,, \tag{5}$$

and 2) **optimize the parameters:**

$$\theta = \underset{\theta \in \Theta}{\operatorname{argmax}} \log \pi(\theta | \mathbf{z}, \mathbf{x}, \mathbf{y}) \,, \tag{6}$$

where θ consists of all model parameters $\theta_c, \theta_r, \sigma_r^2, \tilde{\mathbf{f}}, \tilde{\mathbf{x}}$.

While the pseudo-targets $\tilde{\mathbf{f}}$ can be marginalized, in the first step, this would result in the allocation:

$$\mathbf{z} = \underset{\mathbf{l} \in \{1,\dots,L\}^N}{\operatorname{argmax}} \sum_{i=1}^{N} \log(g_{l_i}(x_i; \theta_c)) + \sum_{l=1}^{L} \log\left(\mathrm{N}(\mathbf{y}^l \mid \boldsymbol{\mu}^l, \boldsymbol{\Sigma}^l)\right) \,,$$

where $\boldsymbol{\Sigma}^l = (\mathbf{K}^l_{M_l N_l})^T (\mathbf{K}^l_{M_l})^{-1} \mathbf{K}^l_{M_l N_l} + \boldsymbol{\Lambda}^l + \sigma_r^{2\,l} \mathbf{I}_{N_l}$. As the local likelihood no longer factorizes, N sequential steps would be required to estimate $\mathbf{z}$ through an iterative conditional modes algorithm, allocating each data point based on the conditional Gaussian likelihood given the data points currently allocated to each cluster.

However, we can significantly reduce the computational cost by also estimating $\tilde{\mathbf{f}}$. Specifically, in the second step of the MM algorithm (6), we first estimate the GP hyperparameters and pseudo-inputs by optimizing the log marginal likelihood in (3) and then estimate $\tilde{\mathbf{f}}^l$ with its posterior mean:

$$\mathrm{E}[\tilde{\mathbf{f}}^l \mid \theta_r^l, \sigma_r^{2\,l}, \tilde{\mathbf{x}}^l, \mathbf{y}^l, \mathbf{x}^l] = \mathbf{K}^l_{M_l} (\mathbf{Q}^l_{M_l})^{-1} \left(\mathbf{K}^l_{M_l N_l} (\boldsymbol{\Lambda}^l + \sigma_r^{2\,l} \mathbf{I}_{N_l})^{-1} \mathbf{y}^l + \boldsymbol{\mu}^l\right) ,$$

where $\mathbf{Q}^l_{M_l} = \mathbf{K}^l_{M_l} + \mathbf{K}^l_{M_l N_l} (\boldsymbol{\Lambda}^l + \sigma_r^{2\,l} \mathbf{I}_{N_l})^{-1} (\mathbf{K}^l_{M_l N_l})^T$. Plugging this into the local likelihood (2), the first step of the MM algorithm is:

$$\mathbf{z} = \underset{\mathbf{l} \in \{1,\dots,L\}^N}{\operatorname{argmax}} \sum_{i=1}^{N} \log(g_{l_i}(x_i; \theta_c)) + \sum_{i=1}^{N} \log\left(\mathrm{N}(y_i \mid \widehat{f}_r(x_i; \theta_r^{l_i}), \lambda_i^{l_i} + \sigma_r^{2\,l_i})\right) \,,$$

where

$$\widehat{f}_r(x_i; \theta_r^l) = \mu^l + (\mathbf{k}^l_{M_l, i})^T \left[(\mathbf{Q}^l_{M_l})^{-1} \left(\mathbf{K}^l_{M_l N_l} (\boldsymbol{\Lambda}^l + \sigma_r^{2\,l} \mathbf{I}_{N_l})^{-1} \mathbf{y}^l + \boldsymbol{\mu}^l\right) - (\mathbf{K}^l_{M_l})^{-1} \boldsymbol{\mu}^l\right] , \tag{7}$$

$$\lambda_i^l = K_{\psi^l}(x_i, x_i) - (\mathbf{k}^l_{M_l, i})^T (\mathbf{K}^l_{M_l})^{-1} \mathbf{k}^l_{M_l, i} \,. \tag{8}$$

Thus, the allocation can be done in parallel across the N data points:

$$z_i = \underset{l \in \{1,\dots,L\}}{\operatorname{argmax}} \log(g_l(x_i; \theta_c)) + \log(\mathrm{N}(y_i | \widehat{f}_r(x_i; \theta_r^l), \lambda_i^l + \sigma_r^{2\,l}).$$

Optimization of the gating network and expert parameters can also be done in parallel, both between each other as well as across $l = 1, \dots, L$ for the experts. Specifically, the optimal gating network and expert parameters are respectively:

$$\theta_c = \underset{\theta \in \Theta_c}{\operatorname{argmax}} \sum_{l=1}^{L} \sum_{i : z_i = l} h_l(x_i; \theta_c) - \sum_{i=1}^{N} \log\left(\sum_{l=1}^{L} \exp(h_l(x_i; \theta_c))\right) ,$$

$$(\theta_r^l, \sigma_r^{2\,l}, \tilde{\mathbf{x}}^l) = \underset{\theta \in \Theta_r, \sigma^2 \in \mathbb{R}_+, \tilde{\mathbf{x}} \in \mathbb{R}^{M_l \times d}}{\operatorname{argmax}} \log\left(\mathrm{N}(\mathbf{y}^l \mid \boldsymbol{\mu}^l, (\mathbf{K}^l_{M_l N_l})^T (\mathbf{K}^l_{M_l})^{-1} \mathbf{K}^l_{M_l N_l} + \boldsymbol{\Lambda}^l + \sigma_r^{2\,l} \mathbf{I}_{N_l})\right) .$$

3.2 An ultra-fast approximation: CCR

The MM algorithm iterates between **clustering** (5) and in parallel **classification** and **regression** (6). This closely resembles the CCR algorithm recently introduced in [3]. The important differences are that 1) CCR is a one pass algorithm that does not iterate between the steps and 2) CCR approximates the clustering in the first step of the MM algorithm by a) re-scaling the data to emphasize the output y in relation to x and b) subsequently applying a fast clustering algorithm, e.g. k-means [14] or DB-scan [9], to jointly cluster the rescaled (y, x). We also note that the original formulation of CCR performs an additional clustering step so that the allocation variables used by the regression correspond to the prediction of the classifier; this is equivalent to the clustering step of the MM algorithm in (5) including only the term associated to the gating network.

This novel connection allows us to view CCR as a fast, one-pass approximation to the MM algorithm for MoEs and therefore construct an ultra-fast approximation of the proposed deep mixture

of sparse GP experts based on the CCR algorithm. As shown in the following section, CCR provides a good, fast approximation for many numerical examples. If extra computational resources are available, the CCR solution can be improved through additional MM iterations (i.e. it provides a good initialization for the MM algorithm). However, in our examples, we notice that the potential for further improvement is limited. We find that MM with random initialization also produces a reasonable estimate for this model after two iterations; we refer to this algorithm as MM2r. It is fast, but we will see that it takes approximately 2-3 times longer than CCR.

3.3 Complexity considerations

Suppose we parameterize the DNN with p_c parameters, and each of the sparse GP experts is approximated with M_l pseudo-inputs. The MM algorithm described in Section 3.1 incurs a cost per iteration of clustering the N points given the current set of parameters (5). This cost is $\mathcal{O}(N\sum_{l=1}^{L} M_l^2)$, and it is parallel in N. The algorithm also incurs a cost per iteration of classification with N points and L regressions using $N_1, \ldots, N_L$ points, where $N = \sum_{l=1}^{L} N_l$ (6). These operations can also be done in parallel. The cost for the classification is $\mathcal{O}(Np_c)$ assuming that the number of epochs for training is $\mathcal{O}(1)$. The cost for the regressions is $\mathcal{O}(\sum_{l=1}^{L} M_l^2 N_l)$. Hence the total cost for (6) is $\mathcal{O}(Np_c + \sum_{l=1}^{L} M_l^2 N_l)$, which can be roughly bounded by $\mathcal{O}(NP_{\max})$, where $P_{\max} = \max\{p_c, M_1^2, \ldots, M_L^2\}$. Randomly initialized MM cannot be expected to provide reasonable results after one pass, however with sparse GP models the first iteration provides a significant improvement which is also sometimes reasonable. Ignoring parallel considerations, the total cost for 2-pass MM (MM2r) is $\mathcal{O}(2Np_c + \sum_{l=1}^{L} M_l^2(2N_l + N))$.

For CCR, the cost is the same for the second step (6), while the first step is replaced with K-means, which incurs a cost of $\mathcal{O}(NL)$. The latter iterates between steps which can be parallelized in different ways. The total cost of CCR is hence $\mathcal{O}(N(p_c+L)+\sum_{l=1}^{L} M_l^2 N_l) = \mathcal{O}(NP_{\max})$. This is a *one pass* algorithm, which often provides acceptable results. The overhead for MM2r vs. CCR for our model is then roughly $\mathcal{O}((P_{\max} - L)N)$. More precisely it is $\mathcal{O}(N(p_c - L) + \sum_{l=1}^{L} M_l^2(N + N_l))$.

3.4 Prediction

There are two approaches that can be employed to predict y^* at a test value x^*. **Hard allocation based prediction** first allocates the test point based on the optimised classifier/gating network:

$$z^* = \underset{l \in \{1,\ldots,L\}}{\operatorname{argmax}} \log(g_l(x^*; \theta_c)), \tag{9}$$

and then predicts based on this regression/expert (given in (7)):

$$y^* = \widehat{f}_r(x^*; \theta_r^{z^*}).$$

Instead, **soft allocation based prediction** is based on a weighted combination of the regressions/experts with weights given by the classifier/gating network:

$$y^* = \sum_{l=1}^{L} g_l(x^*; \theta_c)\widehat{f}_r(x^*; \theta_r^l).$$

Soft-allocation may be preferred in cases when there is not a clear jump in the unknown function, thus allowing us to smooth the predictions in regions where the classifier is unsure. Similarly, the variance or density of the output or regression function can also be computed at any test location to obtain measures of uncertainty in our predictions. In cases when the density of the output may be multi-modal, looking at point predictions alone is not useful. In this setting, the soft density estimates are preferred, allowing one to capture and visualise the multi-modality.

4 Numerical Experiments

In this section, we perform a range of experiments to highlight the flexibility of our model and compare the accuracy and speed of the MM and CCR algorithms. For the sparse GP experts, we use the isotropic squared exponential covariance function with a variable number of inducing points M_l

Table 1: R^2 test accuracy (%) on the five datasets for our model with the CCR, MM, and MM2r algorithms and the FastGP and MDN benchmarks.

Model	CCR	MM	MM2r	FastGP	MDN
Motorcycle	75.44	**86.17**	75.64	75.02	73.13
Nasa	96.11	**96.88**	96.41	88.34	87.49
Higdon	99.94	**99.99**	99.97	99.88	95.31
Bernholdt	98.88	**99.99**	91.40	96.25	90.68
χ	94.67	**97.86**	97.71	93.66	91.87

Table 2: Wall-clock time (in seconds) on the five datasets for our model with the CCR, MM, and MM2r algorithms and the FastGP and MDN benchmarks. The number of iterations required for the MM algorithm is reported in the last column.

Model	CCR	MM	FastGP	MM2r	MDN
Motorcycle	**3.76**	198.46	29.15	21.69	62.72
Nasa	**37.09**	1854.89	455.57	87.11	304.22
Higdon	**32.98**	362.18	92.64	81.99	124.68
Bernholdt	**290.85**	2615.85	1490.83	491.62	1876.66
χ	**1956.47**	7826.47	2643.65	3982.64	9834.35

based on the cluster sizes. The pseudo-inputs $\tilde{\mathbf{x}}^l$ are initialized via K-means and the GP hyperparameters are initialized based on the scale of the data. The number of experts L is determined apriori using the elbow method to compare the K-means clustering solution of the rescaled (y, x) across different values of L. The DNN gating network has three hidden layers of 200, 40, and 30 neurons, and a quadratic regularization is used with a value of 0.001 for the penalization parameter. The adaptive stochastic gradient descent solver Adam [21] is used to optimize the model weights and biases, with a validation fraction of 0.1 and a maximum of 1000 epochs. The GPML toolbox [32] and the Deep Learning toolbox [26] are used to train the experts and gating network respectively. The code can be found at *commented Github link*, along with further results and experiments.

4.1 Datasets

Our experiments range from small to large datasets of varying dimension and complexity. First, the motorcycle dataset [34] consists of $N = 133$ measurements with $d = 1$. Second, the NASA dataset [12] comes from a computer simulator of a NASA rocket booster vehicle with $N = 3167$; we focus on modelling the lift force as a function of the speed (mach), the angle of attack (alpha), and the slide-slip angle (beta), i.e. $d = 3$. Our third experiment is the Higdon function [16, 12]; $N = 1000$ data points are generated by $x_i \sim \mathrm{U}[0, 20]$ and $y_i \sim \mathrm{N}(f(x_i), 0.1^2)$, with

$$f(x) = \left\{ \begin{array}{ll} \sin(\frac{\pi x}{5}) + 0.2\cos(\frac{4\pi x}{5}) & x < 10 \\ \frac{x}{10} - 1 & x \geq 10 \end{array} \right\}. \tag{10}$$

Next, $N = 1000$ points are generated from the Bernholdt function [3]; in this case, $\mathcal{X} = [-4, 10]^2$ and $f(x) = g(x_1)g(x_2)$, where $g(x)$ is the smooth piece-wise constant function studied in [28]. Lastly, the χ dataset comes from the critical gradient model of [3, 23] with $d = 10$ and $N = 300000$.

We split all datasets into train-test sets of 80% and 20%. The computer specifications are: Model = Dell Optiplex 790; Operating System = Windows 10 Enterprise 64-bit; Processor = Intel(R) Core(TM) i5-2400, CPU@ 3.10 GHz, 3101 Mhz, 4Cores(s), 4 Logical Processors; RAM = 16.00GB.

4.2 Results

For our model, we compare the MM algorithm with the ultra-fast CCR and MM2r approximations. The MM algorithm is initialized at the CCR solution and iterates until the reduction in the R^2 is below a threshold of 0.0001 or a maximum number of iterations is reached. Benchmarks include the mixture of GP experts [FastGP 29] and mixture density networks [MDN 4].

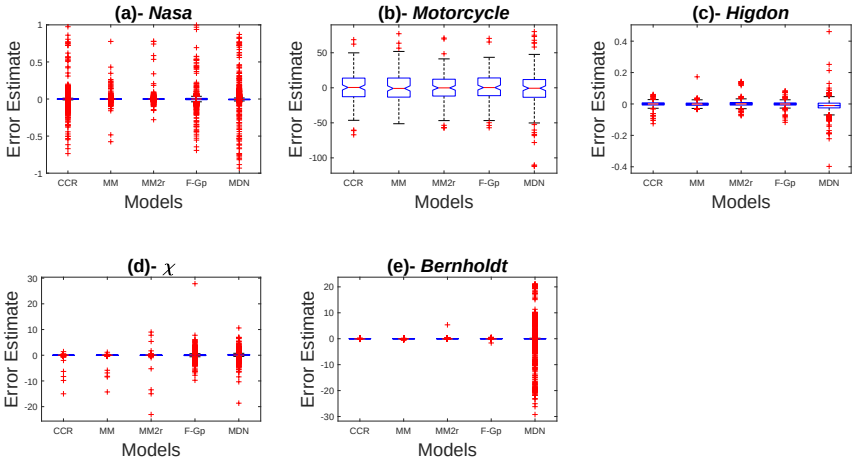

Figure 1: Box plots for the error between the observed output and predictions for all experiments.

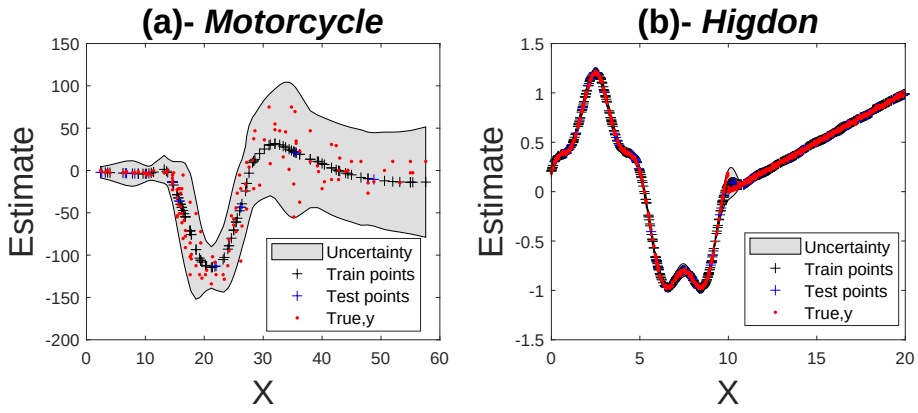

Figure 2: Soft allocation predictions with our model for the motorcycle and Higdon datasets, with the gray area representing two standard deviations from the predictions.

First, we observe that CCR and MM2r have similar test accuracy (Table 1), but CCR reduces wall-clock time (Table 2) by a factor of 2-3 for all experiments. The CCR solution can be further refined through MM iterations to improve test accuracy (Table 2); however this comes at a higher computational cost. Compared with state-of-the-art GP and neural network benchmarks, our model has the highest test accuracy in all experiments considered; this is further investigated in Figure 1, which highlights the more disperse errors of FastGP and MDN. While the test accuracy is reduced with the CCR approximation compared with the MM solution, CCR still offers improvements over the benchmarks and large gains in speed.

Lastly, we highlight that our model provides uncertainty quantification for both the unknown function and predictions. Figure 2 displays the soft-allocation predictions together with the standard deviation for the motorcycle and Higdon datasets. In both cases, the data (in red) is contained within gray region, suggesting that the model also provides good empirical coverage.

5 Conclusion

We have proposed a novel MoE, which combines powerful DNNs to flexibly determine the local regions and sparse GPs to probabilistically model the local regression functions. Through various experiments, we have demonstrated that this combination provides a flexible, robust model that is

able to recover challenging behaviors such as discontinuities, non-stationarity, and non-normality. In addition, we have established a novel connection between the maximization-maximization algorithm and the recently introduced CCR algorithm. This allows us to obtain an ultra-fast approximation that significantly outperforms competing methods. Moreover, in some cases, the solution can be further refined through additional MM iterations. While we focus on the proposed deep mixture of sparse GP experts, this connection can be generally applied to other MoE architectures for fast approximation.

Broader Impact

This new MoE model is applicable to general supervised learning tasks and thanks to the fast approximation, can be used when the computational budget is limited or the dataset is large. While we focus on regression tasks with real-valued inputs, this can be generalized to multivariate outputs or other input and output types, through appropriate specification of the local supervised learning model and choice of the kernel function. The sound uncertainty quantification provided by the method is becoming increasingly beneficial in many applications. For example, in health datasets, it is important to take into account not only point predictions but also uncertainty estimates when making decisions.

Acknowledgments and Disclosure of Funding

We gratefully acknowledge Lassi Roininen for the introduction which initiated discussions on this topic. C.E. and K.J.H.L. gratefully acknowledge the support of the U. S. Department of Energy, Office of Science, Office of Fusion Energy Sciences and Office of Advanced Scientific Computing Research through the Scientific Discovery through Advanced Computing (SciDAC) project on Advanced Tokomak Modeling (AToM) and Oak Ridge National Laboratory (ORNL). ORNL is managed by UT-Battelle, LLC for the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

References

[1] Luca Ambrogioni, Umut Güçlü, Marcel AJ van Gerven, and Eric Maris. The kernel mixture network: A nonparametric method for conditional density estimation of continuous random variables. *arXiv preprint arXiv:1705.07111*, 2017.

[2] Sudipto Banerjee, Alan E Gelfand, Andrew O Finley, and Huiyan Sang. Gaussian predictive process models for large spatial data sets. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(4):825–848, 2008.

[3] David E Bernholdt, Mark R Cianciosa, David L Green, Jin M Park, Kody JH Law, and Clement Etienam. Cluster, classify, regress: A general method for learning discontinuous functions. *Foundations of Data Science*, 1(2639-8001-2019-4-491):491, 2019.

[4] Christopher M Bishop. Mixture density networks. *Technical Report, Aston University*, 1994.

[5] Thang D Bui, Josiah Yan, and Richard E Turner. A unifying framework for sparse Gaussian process approximation using power expectation propagation. *Journal of Machine Learning Research*, 18:1–72, 2017.

[6] Noel Cressie and Gardar Johannesson. Fixed rank kriging for very large spatial data sets. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1):209–226, 2008.

[7] Constantinos Daskalakis, Petros Dellaportas, and Aristeidis Panos. Faster Gaussian processes via deep embeddings, 2020.

[8] Nicolas Durrande, Vincent Adam, Lucas Bordeaux, Stefanos Eleftheriadis, and James Hensman. Banded matrix operators for Gaussian Markov models in the automatic differentiation era. In *22nd International Conference on Artificial Intelligence and Statistics*, volume 89, pages 780–2789, 2019.

[9] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Knowledge Discovery in Databases*, volume 96, pages 226–231, 1996.

[10] Charles WL Gadd, Sara Wade, and Alexis Boukouvalas. Enriched mixtures of gaussian process experts. In *23rd International Conference on Artificial Intelligence and Statistics*, 2020.

[11] Isobel C. Gormley and Sylvia Frühwirth-Schnatter. *Mixtures of Experts Models*. Chapman and Hall/CRC, 2019.

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

[13] Alexander Grigorievskiy, Neil Lawrence, and Simo Särkkä. Parallelizable sparse inverse formulation Gaussian processes (SpInGP). In *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2017.

[14] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[15] Matthew J Heaton, Abhirup Datta, Andrew O Finley, Reinhard Furrer, Joseph Guinness, Rajarshi Guhaniyogi, Florian Gerber, Robert B Gramacy, Dorit Hammerling, Matthias Katzfuss, Finn Lindgren, Douglas W Nychka, Furong Sun, and Andrew Zammit-Mangion. A case study competition among methods for analyzing large spatial data. *Journal of Agricultural, Biological and Environmental Statistics*, 24(3):398–425, Sep 2019.

[16] Dave Higdon. Space and space-time modeling using process convolutions. In *Quantitative Methods for Current Environmental Issues*, pages 37–56. Springer, 2002.

[17] Wenbing Huang, Deli Zhao, Fuchun Sun, Huaping Liu, and Edward Chang. Scalable Gaussian process regression using deep neural networks. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[18] Tomoharu Iwata and Zoubin Ghahramani. Improving output uncertainty estimation and generalization in deep learning via neural network Gaussian processes. *arXiv preprint arXiv:1707.05922*, 2017.

[19] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

[20] Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214, 1994.

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[22] J Kittler and J Föglein. Contextual classification of multispectral pixel data. *Image and Vision Computing*, 2(1):13 – 29, 1984.

[23] M Kotschenreuther, W Dorland, MA Beer, and GW Hammett. Quantitative predictions of tokamak energy confinement from first-principles simulations with kinetic effects. *Physics of Plasmas*, 2(6):2381–2389, 1995.

[24] Kenichi Kurihara and Max Welling. Bayesian k-means as a “Maximization-Expectation" algorithm. *Neural Computation*, 21(4):1145–1172, 2009.

[25] Finn Lindgren, Håvard Rue, and Johan Lindström. An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(4):423–498, 2011.

[26] MathWorks. *Deep Learning Toolbox (R2019a)*. 2019.

[27] Edward Meeds and Simon Osindero. An alternative infinite mixture of Gaussian process experts. In *Advances in Neural Information Processing Systems*, pages 883–890, 2006.

[28] Karla Monterrubio-Gómez, Lassi Roininen, Sara Wade, Theodoros Damoulas, and Mark Girolami. Posterior inference for sparse hierarchical non-stationary models. *Computational Statistics & Data Analysis*, 2020.

[29] Trung Nguyen and Edwin Bonilla. Fast allocation of Gaussian process experts. In *International Conference on Machine Learning*, pages 145–153, 2014.

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

[31] Carl E Rasmussen and Zoubin Ghahramani. Infinite mixtures of Gaussian process experts. In *Advances in Neural Information Processing Systems*, pages 881–888, 2002.

[32] Carl E Rasmussen and Hannes Nickisch. Gaussian processes for machine learning (GPML) toolbox. *Journal of Machine Learning Research*, 11:3011–3015, 2010.

[33] Tommaso Rigon and Daniele Durante. Tractable Bayesian density regression via logit stick-breaking priors. *arXiv preprint arXiv:1701.02969*, 2017.

[34] Bernhard W Silverman. Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of the Royal Statistical Society: Series B (Methodological)*, 47(1):1–21, 1985.

[35] Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, pages 1257–1264, 2006.

[36] Michalis Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Artificial Intelligence and Statistics*, pages 567–574, 2009.

[37] Volker Tresp. Mixtures of Gaussian processes. In *Advances in Neural Information Processing Systems*, pages 654–660, 2001.

[38] Christopher KI Williams and Carl E Rasmussen. *Gaussian Processes for Machine Learning*. MIT press Cambridge, MA, 2006.

[39] Andrew G Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial Intelligence and Statistics*, pages 370–378, 2016.

[40] Lei Xu, Michael I Jordan, and Geoffrey E Hinton. An alternative model for mixtures of experts. In *Advances in Neural Information Processing Systems*, pages 633–640, 1995.

[41] Chao Yuan and Claus Neubauer. Variational mixture of Gaussian process experts. In *Advances in Neural Information Processing Systems*, pages 1897–1904, 2009.