# Deep Structured Mixtures of Gaussian Processes

**Martin Trapp**
TU Graz & OFAI
SPSC Lab

**Robert Peharz**
TU Eindhoven
Information Systems WSK&I

**Franz Pernkopf**
TU Graz
SPSC Lab

**Carl Edward Rasmussen**
University of Cambridge
CBL Lab

## Abstract

Gaussian Processes (GPs) are powerful non-parametric Bayesian regression models that allow exact posterior inference, but exhibit high computational and memory costs. In order to improve scalability of GPs, approximate posterior inference is frequently employed, where a prominent class of approximation techniques is based on local GP experts. However, local-expert techniques proposed so far are either not well-principled, come with limited approximation guarantees, or lead to intractable models. In this paper, we introduce deep structured mixtures of GP experts, a stochastic process model which i) allows *exact* posterior inference, ii) has attractive computational and memory costs, and iii) when used as GP approximation, captures predictive uncertainties consistently better than previous expert-based approximations. In a variety of experiments, we show that deep structured mixtures have a low approximation error and often perform competitive or outperform prior work.

## 1 INTRODUCTION

Gaussian Processes (GPs) are powerful and versatile models for probabilistic non-linear regression that can capture complex non-linear relationships in data. GPs allow for exact inference, that is, computing the posterior mean and covariance of a GP given $N$ observations with $D$ dimensions. However, the computational and memory costs scale as $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2 + ND)$ [Rasmussen and Williams, 2006], respectively, which limits their use to small data domains or require approximation schemes for big datasets. The most common approaches to overcome these limitations are variational approximations to the GP posterior and methods based on local GP experts [Liu et al., 2018].

The first approach is undoubtedly the more dominant one as it allows for straightforward implementation using differential programming [Wang et al., 2018]. In this case, the posterior of a GP is represented with $Q$ *inducing points* which are treated as variational parameters and learned by minimising the KL divergence between approximate and full posterior. Variational approximations reduce the computational burden to $\mathcal{O}(NQ^2)$ [Titsias, 2009]. As shown by [Burt et al., 2019], the number of *inducing points* has to increase with $Q = \mathcal{O}(\log^D N)$ in order to guarantee convergence with high probability. In the non-asymptotic regime, this may imply that inducing points struggle in producing a good sparse approximation.

Approximations based on local experts, on the other hand, use a *divide-and-conquer* strategy and partition the covariate space (or the data set) into subsets, each modelled with an individual GP expert. For $K$ experts, each with $M << N$ observations, the computational and memory costs are typically reduced to $\mathcal{O}(KM^3)$ and $\mathcal{O}(K(M^2 + MD))$, respectively. Prominent examples include the Naive-Local-Experts model (NLE) [Kim et al., 2005, Vasudevan et al., 2009], which naively models each partition of the covariate space with an independent GP, Products-of-Experts (PoE) [Tresp, 2000a, Cao and Fleet, 2014], which aggregate predictive distributions from experts using a product operation, and the Mixture-of-Experts (MoE) [Tresp, 2000b, Rasmussen and Ghahramani, 2001], which dynamically distribute observations to experts.

All these local-expert approaches have different advantages and disadvantages. The NLE model allows exact posterior inference, which reduces to independent GP inference at each expert, but introduces hard discontinuities in the covariate space. Thus, leading to high generalisation errors [Liu et al., 2018] if the partitioning is not well-supported by the data. PoE approaches have

Proceedings of the 23rdInternational Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108. 

been shown to result in sub-optimal rates of the posterior contraction [Szabó and van Zanten, 2019] and the combination of local experts using product aggregation is known to be Kolmogorov inconsistent [Samo and Roberts, 2016]. Even in the case of the Bayesian committee machine (BCM) [Tresp, 2000a], where the PoE approach is justified as approximation to Bayesian posterior inference, the introduced approximation error is hard to analyse. Finally, while MoE models specify a sound stochastic process model, they do not permit tractable posterior inference and rely on approximate inference techniques.

In this paper, we introduce Deep Structured Mixtures of GPs (DSMGPs)[1] as an attractive alternative to previous local-expert approaches. Our model is based on a natural combination of Sum-Product Networks (SPNs) [Darwiche, 2003, Poon and Domingos, 2011] and GPs. SPNs, in a nutshell, are a deep generalisation of classical mixture models, and recursively model a distribution using i) user-provided distributions (leaves), ii) factorisations (products), and iii) mixtures (sums), whose arrangement is captured by an acyclic directed graph. See Section 2 for details on SPNs. A key advantage of SPNs is that – akin to GPs – many inference scenarios can be computed *exactly*.

So far, SPNs have solely been used as density representations for finitely many random variables. DSMGPs, introduced in this paper, can be understood as an extension of SPNs to the stochastic process case, by equipping SPNs with Gaussian measures (corresponding to GPs [Rajput and Cambanis, 1972]) as leaves. Equivalently, we can also interpret our model as an hierarchically structured mixture over a large number of NLEs. In particular, the posterior of DSMGPs can be naturally understood as Bayesian model averaging over an *exponentially large mixture* of NLEs, i.e. combinatorial in the states of latent SPN variables [Zhao et al., 2016, Peharz et al., 2017]. The crucial key advantage of DSMGPs is that posterior inference can be computed *exact* and *efficiently*, i.e. they inherit tractable inference from SPNs and GPs.

We further show that the structure of DSMGPs can be exploited to speed up computations, by sharing Cholesky decompositions among GP leaves, and to model non-stationary time-series, by locally adapting hyperparameters. In a variety of experiments we show that our approach captures uncertainties consistently better than previous experts-based approximations, is competitive to state-of-the-art, and has competitive running times compared to state-of-the-art.

[1] `https://tinyurl.com/dsmgp-jl`

## 2 RELATED WORK

While our proposed DSMGP is a process model on its own right, our main motivation in this paper is to use it as an approximation to a full GP, following a *divide-and-conquer* approach. In this sense, the most related approaches are expert-based approaches, which we review in this section.

The probably simplest approach are Naive-Local-Experts [Kim et al., 2005], and subsequent approaches [Gramacy and Lee, 2008, Vasudevan et al., 2009]. NLEs use a pre-defined, sometimes nested, partition of the covariate space and model each subspace using an independent GP expert. Due to the independence assumptions, NLEs introduce hard discontinuities in the modelled functions. Recent approaches [Park and Huang, 2016] try to ameliorate this effect by imposing continuity constraints onto the local experts using patched GPs. However, this approach suffers from inconsistent variances and does not scale well with the number of boundaries and, consequently, the dimensionality of the covariate space. In contrast to NLEs and patched GPs, our model does not rely on a single partition, but rather performs posterior inference over a large set of partitions, and thus effectively selects partitions which are well supported by the data.

Product-of-Expert (PoE) approaches, generalised PoE (gPoE) [Cao and Fleet, 2014], the Bayesian Committee Machine (BCM) [Tresp, 2000a] and the robust Bayesian Committee Machine (rBCM) [Deisenroth and Ng, 2015] distribute subsets of the data to local experts and aggregate their predictive distributions using a product operation – weighted by some adaptive or non-adaptive scale factors. The key motivation in these approaches is that a product of Gaussians is still Gaussian. The major drawback of these methods is that they are somewhat heuristic, as PoEs typically do not correspond to inference in some well-defined statistical model. BCMs justify PoEs as approximation to posterior inference in GPs, but the introduced approximation error is hard to analyse. Moreover, the product aggregation of expert predictions is Kolmogorov inconsistent [Samo and Roberts, 2016], and PoEs are known to have sub-optimal rates of the posterior contraction, and therefore uncalibrated predictive uncertainties [Szabó and van Zanten, 2019]. In contrast to PoE approaches, our model is a well-defined stochastic process and adequately captures predictive uncertainties.

The MoE model [Tresp, 2000b] is a sound probabilistic model, defined as a mixture of GP experts and a so-called gating network which dynamically assigns data to GPs. One of the most prominent variants is the infi-

nite MoE model [Rasmussen and Ghahramani, 2001], which removes the i.i.d. assumption of the MoE and uses a Dirichlet process as gating network. Alternative formulations and improvements of the infinite MoE model can be found in [Meeds and Osindero, 2005, Gadd et al., 2019]. However, while MoE models are designed to capture multi-modality and non-stationarity, they usually lack tractable inference. Consequently, they inherently rely on approximate posterior inference, which hampers their application to large data domains. In contrast to MoE models, our approach does not use a gating network, but performs inference over a large set of pre-determined partitions of the covariate space. Crucially, and unlike as in MoE models, posterior inference in our model can be performed exactly and efficiently. Note that the approach by [Zhang and Williamson, 2019], which was published around the time of this paper, is similar in spirit but does not utilise exact posterior inference.

# 3 BACKGROUND

## 3.1 Gaussian Process Regression

A Gaussian Process (GP) is defined as a collection of random variables (RVs) $\mathbf{F}$ indexed by an arbitrary covariate space $\mathcal{X}$, where any finite subset of $\mathbf{F}$ is Gaussian distributed, and of which any two overlapping finite sets are marginally consistent [Rasmussen and Williams, 2006]. In that way, GPs can naturally be interpreted as distributions over functions $f\colon \mathcal{X} \to \mathbb{R}$. A GP is uniquely specified by a *mean-function* $m\colon \mathcal{X} \to \mathbb{R}$ and a *covariance function* $k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Given a training set of $N$ observations $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$ with $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$ and $\mathbf{y} = \{y_n\}_{n=1}^N$, let $k_{\mathbf{X},\mathbf{X}}$ be the $N \times N$ covariance matrix defined by $[k_{\mathbf{X},\mathbf{X}}]_{n,m} = k(\mathbf{x}_n, \mathbf{x}_m)$ and let $m_{\mathbf{X}}$ be the respective mean values, i.e., $[m_{\mathbf{X}}]_n = m(\mathbf{x}_n)$.

In GP regression, we aim to model noisy observed output $y_n \in \mathbb{R}$ given input locations $\mathbf{x}_n \in \mathcal{X}$, i.e.,

$$f \sim \mathrm{GP}(m_{\mathbf{X}}, k_{\mathbf{X},\mathbf{X}})\,, \tag{1}$$

$$y_n \mid f, \mathbf{x}_n \overset{\text{iid}}{\sim} \mathcal{N}(f(\mathbf{x}_n), \sigma^2)\,, \tag{2}$$

where $\sigma^2$ is the noise variance. The posterior of a GP conditioned on $\mathcal{D}$ can be obtained by computing the posterior mean, $m_{\mathcal{D}}(\mathbf{x}^*) = k_{\mathbf{x}^*,\mathbf{X}}[k_{\mathbf{X},\mathbf{X}} + \sigma^2 \boldsymbol{I}]^{-1}\mathbf{y}$, and the posterior variance, $V_{\mathcal{D}}(\mathbf{x}^*) = k_{\mathbf{x}^*,\mathbf{x}^*} - k_{\mathbf{x}^*,\mathbf{X}}[k_{\mathbf{X},\mathbf{X}} + \sigma^2 \boldsymbol{I}]^{-1} k_{\mathbf{X},\mathbf{x}^*}$. The main challenge is the inversion of $[k_{\mathbf{X},\mathbf{X}} + \sigma^2 \boldsymbol{I}]$, which is frequently realised via the Cholesky decomposition [Press, 2002].

Note that there is an intimidate relationship between GPs, whose function draws are almost surely from a certain function space, and Gaussian measures defined on the same function space. In particular, this relationship is one-to-one for the space of continuously differentiable functions on any real interval, and for $L_2$-spaces defined on arbitrary measurable spaces [Rajput and Cambanis, 1972]. We will take use of this equivalence, and describe our model as a hierarchical mixture, realised as a sum-product network, over Gaussian measures.

## 3.2 Sum-Product Networks

Sum-Product Networks (SPNs) [Darwiche, 2003, Poon and Domingos, 2011] are a prominent type of tractable deep probabilistic model, which allow fast and exact inference in high-dimensional data domains.

**Definition 1** (Sum-Product Network)**.** *A sum-product network over a finite set of RVs* $\mathbf{F} = \{F_1, \ldots, F_D\}$ *is a 4-tuple* $\mathcal{S} = (\mathcal{G}, \psi, \boldsymbol{w}, \theta)$*, where* $\mathcal{G}$ *is a* computational graph*,* $\psi$ *is a* scope-function*,* $\boldsymbol{w}$ *denotes a set of sum-weights, and* $\theta$ *is a set of leaf parameters.*

*The* computational graph $\mathcal{G}$ *is a connected acyclic directed graph, containing three types of nodes: sums* ($\mathsf{S}$)*, products* ($\mathsf{P}$) *and leaves* ($\mathsf{L}$) *(nodes without children). We use* $\mathsf{N}$ *to denote a generic node, and* $\mathbf{N}$ *is the set of all SPN nodes. The set of children of node* $\mathsf{N}$ *is denoted as* $\mathbf{ch}(\mathsf{N})$*.*

*The* scope function *is a function* $\psi\colon \mathbf{N} \mapsto 2^{\mathbf{F}}$*, assigning each node in* $\mathcal{G}$ *a subset of* $\mathbf{F}$*, where* $2^{\mathbf{F}}$ *denotes the power set of* $\mathbf{F}$*. It has the following properties: i) If* $\mathsf{N}$ *is the root node, then* $\psi(\mathsf{N}) = \mathbf{F}$*; ii) If* $\mathsf{N}$ *is a sum or product, then* $\psi(\mathsf{N}) = \bigcup_{\mathsf{N}' \in \mathbf{ch}(\mathsf{N})} \psi(\mathsf{N}')$*; iii) For each sum node* $\mathsf{S}$ *we have* $\forall \mathsf{N}, \mathsf{N}' \in \mathbf{ch}(\mathsf{S})\colon \psi(\mathsf{N}) = \psi(\mathsf{N}')$ (completeness)*; iv) For each product node* $\mathsf{P}$ *we have* $\forall \mathsf{N}, \mathsf{N}' \in \mathbf{ch}(\mathsf{P})\colon \psi(\mathsf{N}) \cap \psi(\mathsf{N}') = \emptyset$ (decomposability)*.*

In an SPN, each node $\mathsf{N}$ in $\mathcal{G}$ represents a distribution over RVs $\psi(\mathsf{N})$. In particular, each $\mathsf{L}$ computes a distribution over its scope parameterised by $\theta_{\mathsf{L}}$. A sum node $\mathsf{S}$ computes a weighted sum $\mathsf{S} = \sum_{\mathsf{N} \in \mathbf{ch}(\mathsf{S})} w_{\mathsf{S},\mathsf{N}}\, \mathsf{N}$ where $w_{\mathsf{S},\mathsf{N}} \geq 0$. Note that w.l.o.g. we assume that all sum nodes are normalised, i.e., $\sum_{\mathsf{N} \in \mathbf{ch}(\mathsf{S})} w_{\mathsf{S},\mathsf{N}} = 1$ [Peharz et al., 2015, Zhao et al., 2015]. Finally, a product node $\mathsf{P}$ computes a factorisation over its children, i.e. $\mathsf{P} = \prod_{\mathsf{N} \in \mathbf{ch}(\mathsf{P})} \mathsf{N}$. It can be shown, that the conditions *completeness* and *decomposability* guarantee that many inference scenarios, e.g. marginalisation, can be performed in linear time of the network size [Darwiche, 2003, Poon and Domingos, 2011, Peharz et al., 2015].

As shown in [Zhao et al., 2015, Zhao et al., 2016], SPNs can be interpreted as deep structured mixture models, using the notion of *induced trees*.

**Definition 2** ([Zhao et al., 2016])**.** *Given an SPN graph* $\mathcal{G}$*, a sub-graph* $\mathcal{T} = (\mathcal{T}_V, \mathcal{T}_E)$ *of* $\mathcal{G}$ *is called an* induced tree *if i) the root of* $\mathcal{G}$ *is in* $\mathcal{T}$*; ii) if* $\mathsf{N} \in \mathcal{T}_V$ *is*

*a sum node, then exactly one child of* $\mathsf{N}$ *in* $\mathcal{S}$ *is in* $\mathcal{T}_V$*, and the corresponding edge is in* $\mathcal{T}_E$*; iii) if* $\mathsf{N} \in \mathcal{T}_V$ *is a product node, then all the children of* $\mathsf{N}$ *in* $\mathcal{S}$ *are in* $\mathcal{T}_V$*, and the corresponding edges are in* $\mathcal{T}_E$*.*

Using the notion of induced trees, it can be shown that the distribution of an SPN, denoted as $\mathcal{S}(\mathbf{x})$, can be expressed as a mixture whose components correspond to induced trees [Zhao et al., 2016], i.e.,

$$\mathcal{S}(\mathbf{x}) = \sum_{i=1}^{K} \underbrace{\prod_{(\mathsf{S},\mathsf{N}) \in \mathcal{T}_{i,E}} w_{\mathsf{S},\mathsf{N}}}_{=p(\mathcal{T}_i)} \prod_{\mathsf{L} \in \mathcal{T}_{i,V}} p(\mathbf{x} \,|\, \theta_{\mathsf{L}}) \,, \qquad (3)$$

where $K$ denotes the (exponentially large) number of induced trees.

To the best of our knowledge, SPNs have been previously defined only over finitely many RVs. In the next section, we extend SPNs to stochastic process models, i.e. extending SPNs to infinitely many RVs, by equipping them with GP leaves.

## 4 DEEP STRUCTURED MIXTURE OF GAUSSIAN PROCESSES

Intuitively, a Deep Structured Mixture of GPs (DSMGPs) can be though of as an "SPN over GPs." Formally, this is most naturally defined via the one-to-one correspondence of Gaussian measures on a function space of interest and GPs which almost surely realise in this function space [Rajput and Cambanis, 1972].

**Definition 3** (Deep Structured Mixture of GPs)**.** *Given a measurable covariate space* $(\mathcal{X}, \Sigma)$*, let* $(\mathcal{F}, \Sigma_{\mathcal{F}})$ *be a measurable function space of real-value functions defined on* $\mathcal{X}$*, i.e.,* $\mathcal{F} \subset \mathbb{R}^{\mathcal{X}}$ *equipped with a suitable sigma algebra* $\Sigma_{\mathcal{F}}$*. Then a Deep Structured Mixture of GPs (DSMGP) is defined as an SPN* $(\mathcal{G}, \psi, \boldsymbol{w}, \theta)$*, where* $\mathcal{G}$ *is a computational graph (as in Definition 1),* $\psi$ *is a scope function* $\psi\colon \mathsf{N} \mapsto \Sigma$*,* $\boldsymbol{w}$ *is a set of sum weights, and* $\theta$ *is a set of GP parameters. When* $\mathsf{N}$ *is the root of* $\mathcal{G}$*, then* $\psi(\mathsf{N}) = \mathcal{X}$*; additionally,* $\psi$ *satisfies the conditions ii-iv) in Definition 1. Furthermore:*

1. *A leaf* $\mathsf{L} \in \mathcal{G}$ *computes a Gaussian measure, corresponding to a GP on* $\psi(\mathsf{L})$*, parametrised by* $\theta_{\mathsf{L}}$*.*
2. *A product node* $\mathsf{P} \in \mathcal{G}$ *computes a product measure of its children.*
3. *A sum nodes* $\mathsf{S} \in \mathcal{G}$ *computes a convex combination (determined by its sum-weights) of the measures computed by its children.*

Definition 3 is mathematically elegant as it replaces the usual definition of an SPN leaf – involving densities over finitely many RVs – to Gaussian measures, corresponding to GPs. On the other hand, this definition might obscure how to work with DSMGPs in practice. Therefore, recall that a Gaussian measure evaluated (projected onto) on finitely many data points yields a multivariate Gaussian and similarly a NLE yields a multivariate Gaussian with block-diagonal covariance-structure. Consequently, a DSMGP evaluated on finitely many data points yields a finite – albeit large – mixture of Gaussians with block-diagonal covariance-structure, for which covariance-structure is determined by the scope function $\psi$. Therefore, our model yields a "normal" SPN with Gaussian leaves, when evaluated on finitely many data points.

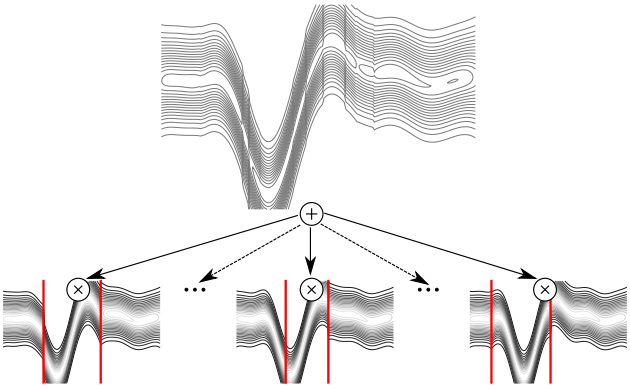


Figure 1: Illustration of a DSMGP (depth 1). Vertical lines (red) represent hypotheses of split-points in the input space, i.e., independence assumptions.

The structure $(\mathcal{G}, \psi)$ of a DSMGP is either pre-defined or learned using posterior inference [Trapp et al., 2019a]. For simplicity, we assume that $\mathcal{G}$ is tree-shaped (i.e. each node has at most one parent), and pre-specify $\psi$ by fixing a random partition of the covariate space at each product node. An algorithm to construct a DSMGPs is described in detail in the supplement. When using DSMGPs as a prior over functions, we assume all sum node weights to be uniform, i.e. $w_{\mathsf{S},\mathsf{N}} = 1/K_{\mathsf{S}}$ where $K_{\mathsf{S}}$ is the number of children under $\mathsf{S}$. Note that in the course of exact posterior inference, these weights will be automatically updated. Intuitively, each sum node represents a prior over hypotheses of split-points in the input space, were a split-point marks statistical independence. Split-points are selected in a hierarchical manner, following the same hierarchy as sum nodes in the DSMGP. This mechanism is illustrated in Figure 1. Therefore, a DSMGP is particularly well suited when it can be expected that certain regions of the input space are approximately independent. The respective split-points will be automatically inferred (among a rich set of choices) through exact posterior inference.

Because DSMGPs naturally have overlapping local GPs, leaves share parts of their kernel matrix. Thus, making it possible to speed up computations of the Cholesky decompositions. We refer to Section 0.1 in the supplement for a detail discussion on sharing Cholesky decompositions in DSMGPs.

### 4.1 Exact Posterior Inference

Posterior inference in DSMGPs combines exact inference in GP experts, defined over a subspace of $\mathcal{X}$, with tractable computations in SPNs. This is a crucial advantage over PoE approaches, which do not define a sound probabilistic model, and over MoE approaches, which are inherently intractable.

**Theorem 1.** *Let $\mathcal{S} = (\mathcal{G}, \psi, w, \theta)$ be a DSMGP on the measurable space $(\mathcal{X}, \Sigma)$, with $\mathcal{X}$ being a covariate space and $\Sigma$ a $\sigma$-algebra over $\mathcal{X}$. Then, computing the unnormalised posterior distribution of $\mathcal{S}$ simplifies to tractable posterior inference at the leaves.*

*Proof.* Under the usual iid. assumption, given a training set $\mathcal{D}$ the unnormalised posterior is

$$p(\boldsymbol{f} \mid \mathcal{D}) \propto \prod_{(\mathbf{x}_n, y_n) \in \mathcal{D}} \underbrace{p(y_n \mid f_n)}_{\text{likelihood}} \underbrace{p(f_n \mid \mathbf{x}_n)}_{\text{prior}} . \tag{4}$$

If the DSMGP is a leaf $\mathsf{L}$, i.e. it is a Gaussian measure induced by the GP at $\mathsf{L}$, then the computation of the posterior follows the standard computations [Rasmussen and Williams, 2006, Eq. 2.7].

In case the DSMGP is a sum node $\mathsf{S}$, the likelihood terms can be "pulled" over the sum, i.e.,

$$\begin{aligned} &p_{\mathsf{S}}(\boldsymbol{f} \mid \mathcal{D}) \\ &\quad \propto \prod_{(\mathbf{x}_n, y_n) \in \mathcal{D}} p(y_n \mid f_n) \sum_{\mathsf{N} \in \mathbf{ch}(\mathsf{S})} w_{\mathsf{S},\mathsf{N}}\, p_{\mathsf{N}}(f_n \mid \mathbf{x}_n) \\ &\quad = \sum_{\mathsf{N} \in \mathbf{ch}(\mathsf{S})} w_{\mathsf{S},\mathsf{N}} \prod_{(\mathbf{x}_n, y_n) \in \mathcal{D}} p(y_n \mid f_n)\, p_{\mathsf{N}}(f_n \mid \mathbf{x}_n) , \end{aligned} \tag{5}$$

simplifying inference to inference at the children.

Finally, in case the DSMGP is a product node $\mathsf{P}$, we can swap the product over observations with the product over children and "pull" the likelihood terms down to the respective children, i.e.,

$$\begin{aligned} p_{\mathsf{P}}(\boldsymbol{f} \mid \mathcal{D}) &\propto \prod_{(\mathbf{x}_n, y_n) \in \mathcal{D}} p(y_n \mid f_n) \prod_{\mathsf{N} \in \mathbf{ch}(\mathsf{P})} p_{\mathsf{N}}(f_n \mid \mathbf{x}_n) \\ &= \prod_{\mathsf{N} \in \mathbf{ch}(\mathsf{P})} \Bigg( \prod_{(\mathbf{x}_n, y_n) \in \mathcal{D}_{(\mathsf{N})}} p(y_n \mid f_n)\, p_{\mathsf{N}}(f_n \mid \mathbf{x}_n) \Bigg) , \end{aligned} \tag{6}$$

$$\bigcup_{\mathsf{N} \in \mathbf{ch}(\mathsf{P})} \mathcal{D}_{(\mathsf{N})} = \mathcal{D}$$

where $\mathcal{D}_{(\mathsf{N})}$ denotes the subset of observations node $\mathsf{N}$ is responsible for and $\cap_{\mathsf{N} \in \mathbf{ch}(\mathsf{P})} \mathcal{D}_{(\mathsf{N})} = \emptyset$. Therefore, posterior inference simplifies to inference at the children of the product node $\mathsf{P}$ using sub-sets of $\mathcal{D}$.

Inductively repeating this argument for all internal nodes, we see that we obtain the unnormalised posterior by multiplying each leaf with its local likelihood. Therefore, the unnormalised posterior of a DSMGP is obtained by performing inference on the leaves, which can be done exactly [Rasmussen and Williams, 2006, Eq. 2.7]. □

Finally, we can obtain the normalised posterior, i.e. $p(\boldsymbol{f} \mid \mathcal{D}) = \frac{p(\mathbf{y} \mid \boldsymbol{f})\, p(\boldsymbol{f} \mid \mathbf{X})}{p(\mathbf{y} \mid \mathbf{X})}$, by re-normalising the unnormalised posterior of the DSMGP using a bottom-up propagation of the marginal likelihood of each expert. In this paper we use [Peharz et al., 2015, Alg. 1], which scales linear in the number of nodes, for this purpose, c.f. algorithm 2 in the supplement for a pseudocode implementation. Note that normalising the posterior can be understood as updating our belief over split-points, i.e., independence assumptions in $\mathcal{X}$.

### 4.2 Predictions

The predictive posterior distribution of a DSMGP for an unseen datum $\mathbf{x}^*$ is naturally a mixture distribution and, therefore, can be multimodal. For practical reasons, it is, therefore, useful to project the posterior of a DSMGP to the closest GP, i.e., the GP with minimal KL divergence from the DSMGP. This can be done by computing the first and second moments of the resulting mixture distribution, see [Rasmussen and Williams, 2006, Eq. A.24]. Let $\mathbf{L}$ be the set of all GP leaves in a DSMGP. Then, given a function $\tau_i \colon \mathcal{X} \to \mathbf{L}$ which maps an unseen datum at location $\mathbf{x}^*$ to a leaf $\mathsf{L}$ for each induced tree $\mathcal{T}_k$, we can write the mean (first moment) as

$$m_{\mathcal{D}}(\mathbf{x}^*) = \sum_{i=1}^{K} \prod_{(\mathsf{S},\mathsf{N}) \in \mathcal{T}_{i,E}} w_{\mathsf{S},\mathsf{N}}\, m_{\tau_i(\mathbf{x}^*)}(\mathbf{x}^*) , \tag{7}$$

and the variance (second moment) as

$$\begin{aligned} V_{\mathcal{D}}(\mathbf{x}^*) = \sum_{i=1}^{K} \prod_{(\mathsf{S},\mathsf{N}) \in \mathcal{T}_{i,E}} & w_{\mathsf{S},\mathsf{N}} (m^2_{\tau_i(\mathbf{x}^*)}(\mathbf{x}^*) \\ & + V_{\tau_i(\mathbf{x}^*)}(\mathbf{x}^*) - m^2_{\mathcal{D}}(\mathbf{x}^*)) , \end{aligned} \tag{8}$$

where we use $m_{\tau_i(\mathbf{x}^*)}(\mathbf{x}^*)$ and $V_{\tau_i(\mathbf{x}^*)}(\mathbf{x}^*)$ as short-hand notation for the mean and variance of the predictive distribution of the GP allocated at leaf $\tau_i(\mathbf{x}^*)$. Both moments can be computed efficiently in DSMGPs.

### 4.3 Hyperparameter Optimisation

We can optimise the hyperparameters, i.e. noise variance and kernel parameters, of a DSMGP by max-

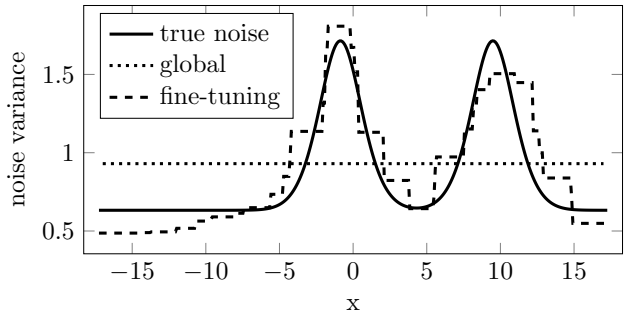



Figure 2: Noise parameter of DSMGP after global hyperparameter optimisation (global) and fine-tuning on a dataset with heteroscedastic noise.

imising the *log marginal likelihood* of the data $\mathcal{D}$. Assuming a zero *mean-function*, the log marginal likelihood of a GP at leaf $\mathsf{L}$ is computed only for the observations that fall into the subspace $\mathcal{X}_\mathsf{L}$. Let $\mathcal{D}_{(\mathsf{L})} = \{(\mathbf{x}_n, y_n) \in \mathcal{D} \,|\, x_n \in \mathcal{X}_\mathsf{L}\}$ denote the respective observations and let $\mathbf{X}_{(\mathsf{L})}$ and $\mathbf{y}_{(\mathsf{L})}$ be the inputs/covariates and the observed outputs contained in $\mathcal{D}_{(\mathsf{L})}$. Then the log marginal likelihood is given as

$$\log p(\mathbf{y}_{(\mathsf{L})} \,|\, \mathbf{X}_{(\mathsf{L})}) = -\frac{1}{2}\left((\mathbf{y}_{(\mathsf{L})}{}^T C^{-1} \mathbf{y}_{(\mathsf{L})}) + \log|C| + N \log 2\pi\right), \tag{9}$$

where $C = k_{\mathbf{X}_{(\mathsf{L})},\mathbf{X}_{(\mathsf{L})}} + \sigma^2 \boldsymbol{I}$ and $\log|C|$ denotes the log determinant of $C$. Consequently, because the DSMGP is a mixture of Gaussian measures, the log marginal likelihood is

$$\log p(\mathbf{y} \,|\, \mathbf{X}) = \mathrm{L}\sum_{i=1}^{K}\mathrm{E}\left(\log \underbrace{p(\mathcal{T}_i)}_{\text{c.f. Eq. 3}} + \sum_{\mathsf{L}\in\mathcal{T}_{i,V}} \underbrace{\log p(\mathbf{y}_{(\mathsf{L})} \,|\, \mathbf{X}_{(\mathsf{L})})}_{\text{Eq. 9}}\right), \tag{10}$$

where $p(\mathcal{T}_i)$ is the probability of the $i^{\text{th}}$ induced tree and $\mathrm{L}\sum_{i=1}^{K}\mathrm{E}$ denotes the log-sum-exp operation. Note that Eq. (10) can be computed efficiently using a single upward-pass through the model.

To optimise the hyperparameters we perform gradient-based optimisation according to the partial derivatives of $\theta$, i.e.,

$$\frac{\partial \log p(\mathbf{y} \,|\, \mathbf{X})}{\partial \theta} = \sum_{\mathsf{L}\in\mathcal{S}} \nabla_\mathsf{L} \frac{\partial \log p(\mathbf{y}_{(\mathsf{L})} \,|\, \mathbf{X}_{(\mathsf{L})})}{\partial \theta}, \tag{11}$$

where $\nabla_\mathsf{L} = \frac{p(\mathsf{L}) p(\mathbf{y}_{(\mathsf{L})} \,|\, \mathbf{X}_{(\mathsf{L})})}{p(\mathbf{y} \,|\, \mathbf{X})} \frac{\partial p(\mathbf{y} \,|\, \mathbf{X})}{\partial \mathsf{L}}$ denotes the gradient for leaf $\mathsf{L}$ and $p(\mathsf{L})$ is the probability of selecting $\mathsf{L}$, c.f. [Poon and Domingos, 2011]. Note that $\nabla_\mathsf{L}$ can be computed by applying the chain-rule. We refer to [Poon and Domingos, 2011, Trapp et al., 2019b] for details on the gradient computation in SPNs.

In case of non-stationary data, we can optionally fine-tune the hyperparameters of each expert. For this purpose, let $\#\mathbf{L}$ denote the cardinality of $\mathbf{L}$ and let $S \in \mathbb{R}^{\#\mathbf{L}\times\#\mathbf{L}}$ be a similarity matrix. Further, let $S$ contain similarity values, i.e. $0 \leq [S]_{i,j} \leq 1$ and $[S]_{i,i} = 1$, between all pairs of leaves $(\mathsf{L}_i, \mathsf{L}_j)$, with $\mathsf{L}_i \in \mathbf{L}, \mathsf{L}_j \in \mathbf{L}$. A natural choice for $S$ is a matrix of normalised overlap values, i.e. $[S]_{i,j} = \sum_{\mathbf{x}_n \in \mathcal{D}_{(\mathsf{L}_i)}} \mathbb{1}\{\mathbf{x}_n \in \mathcal{D}_{(\mathsf{L}_j)}\} / \#\mathcal{D}_{(\mathsf{L}_i)}$ where $\#\mathcal{D}_{(\mathsf{L}_i)}$ is the cardinality of $\mathcal{D}_{(\mathsf{L}_i)}$.

Given a similarity matrix $S$, we can compute the gradients for $\theta_{\mathsf{L}_i}$ of leaf $\mathsf{L}_i$ as

$$\frac{1}{\partial\theta_{\mathsf{L}_i}} \partial \log p(\mathbf{y} \,|\, \mathbf{X}) = \sum_{\mathsf{L}_j\in\mathcal{S}} S_{i,j} \nabla_{\mathsf{L}_j} \frac{\partial \log p(\mathbf{y}_{(\mathsf{L}_j)} \,|\, \mathbf{X}_{(\mathsf{L}_j)}, \theta_{\mathsf{L}_j} = \theta_{\mathsf{L}_i})}{\partial \theta_{\mathsf{L}_j}}. \tag{12}$$

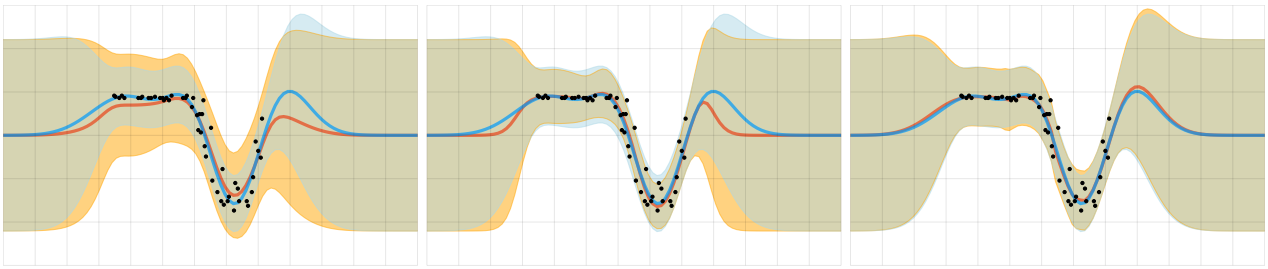
Therefore, $S$ constraints hyperparameters of similar leaves to similar values. Note that Eq. (12) reduces to (11) if $S$ is a matrix of ones.

Figure 2 illustrates the effects of fine-tuning on a synthetic dataset with heteroscedastic noise [Tolvanen et al., 2014]. In contrast to global hyperparameter optimisation (Eq. (11)), fine-tuning allows to capture heteroscedasticity by obtaining an individual noise parameter for each leaf.

## 5 EXPERIMENTS

To assess the performance of DSMGPs, we first compare the approximation error of our model against existing approaches in Section 5.1. Subsequently, we evaluate the predictive performance of DSMGPs against state-of-the-art on various benchmark datasets in Section 5.2.

To construct the DSMGP structure for each experiment, we used algorithm 1 in the supplement. In short, we construct a hierarchical structure consisting of sum nodes, with $K_\mathsf{S}$ children and $w_{\mathsf{S},\mathsf{C}} = \frac{1}{K_\mathsf{S}}$, and product nodes, with $K_\mathsf{P}$ children, by alternating between sum and product nodes. This process terminates and constructs a leaf node once we reached $R$ many repetitions – consecutive sum and product nodes – or the number of observations in the subspace is smaller than a predefined minimum $M$. Finally, we equip each GP leaf with a Squared Exponential (SE) covariance function with Automatic Relevance Detection (ARD) and a zero mean-function. Note that we use the same covariance- and mean-function for all other methods. To obtain suitable hyperparameters, we perform global hyperparameter optimisation for each model using RMSprop (over $1k$ iterations) and in case of DSMGPs refrain



(a) generalized PoE (b) robust BCM (c) DSMGP (our work)

Figure 3: Comparison of generalized PoE, robust BCM and DSMGP (orange) against an exact GP (blue).

from local fine-tuning in favour of a fair comparison.

### 5.1 Approximation Error

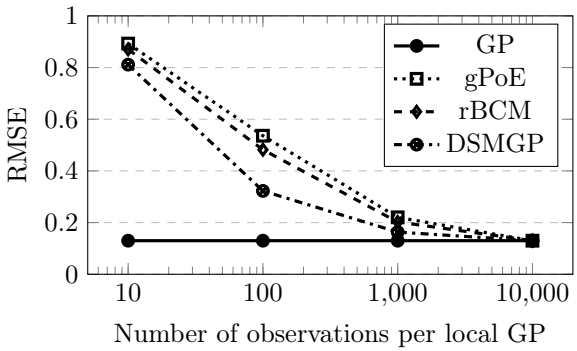



Figure 4: Approximation error on Kin40k dataset.

We use the motorcycle dataset [Silverman, 1985] to compare the approximation error of DSMGPs against popular expert-based approaches. Figure 3 shows the posterior distribution a gPoE, a rBCM and our DSMGP overlain by the posterior of an exact GP. All models use the same SE covariance-function as the exact GP and distribute the covariate space/data set onto local experts with $M = 7$ observations. We see that the gPoE and the rBCM algorithms result in over-conservative predictions and wrong estimates of the mean in regions without observations. On the other hand, our model provides an accurate representation of the uncertainties and mean in regions with and without observed data, when used as an approximation to a GP. Note that DSMGPs do not suffer from severe discontinuities and can exploit discontinuities in data when appropriate, e.g. [Cornford et al., 1998]. We want emphasise that we selected the number of observations $M$ in favour of the gPoE and the rBCM as both degenerate with less observations.

Figure 4 quantitatively compares the approximation error on the Kin40k dataset [Seeger et al., 2003], in terms of the Root Mean Squared Error (RMSE). Note that the DSMGP was constructed using $K_{\mathsf{S}} = 4$, $R = 2$ and $K_{\mathsf{P}} = \sqrt[R]{\frac{N}{M}}$. DSMGPs consistently obtain a lower approximation error than existing approaches, independently of the number of observations per expert.

Table 1: Average runtime (seconds) of an iteration of hyperparameter opt. on an i7-6900k CPU @ 3.2 GHz.

| Dataset | GP | gPoE | rBCM | Ours |
|---|---|---|---|---|
| Airfoil | 0.28 | 0.05 | 0.05 | 0.06 |
| Parkin. | 42.61 | 1.21 | 1.30 | 1.27 |
| Kin40k | 107.65 | 0.86 | 0.87 | 0.89 |
| House | NA | 2.55 | 2.55 | 2.59 |
| Protein | NA | 2.69 | 2.70 | 2.53 |
| Year | NA | 28.82 | 28.90 | 22.17 |

### 5.2 Quantitative Evaluation

To compare the performance of the DSMGP against state-of-the-art, we assess the predictive performance of an exact GP, linear regression (LR), constant regression (Conts.), gPoE, rBCM[2], sparse variational GPs (SVGPs)[3] [Gal et al., 2014] and structured kernel interpolation (KISS)[3] [Wilson and Nickisch, 2015] on various benchmark dataset. Statistics and details on the benchmark datasets are described in the supplementary. The experiments use $Q = 100$ inducing points and consistently use $M = 100$ observations per expert for each expert-based approach and $K_{\mathsf{S}} = 4$ for the DSMGP. For the structured kernel interpolation (KISS) we chose the grid size according to the number of data points and used an additive kernel decomposition as KISS GPs scale exponentially with the dimensionality of the covariate space. Note, that we obtained the hyperparameters for DSMGPs using surrogate DSMGP with $K_{\mathsf{S}} = 1$. The results for DSMGPs are likely to improve if hyperparameter optimisation is performed with $K_{\mathsf{S}} = 4$.

[2] `https://github.com/jopago/GPyBCM`
[3] `https://gpytorch.ai`

Table 2: Mean Absolute Error (MAE) and Negative Log Predictive Density (NLPD) of state-of-the-art approaches and DSMGPs (our work) on benchmark datasets with $1.5K$ to $500K$ observations. Smaller values are better.

| Dataset | | Const. | LR | GP | SVGP | KISS | gPoE | rBCM | **DSMGP** |
|---|---|---|---|---|---|---|---|---|---|
| **Airfoil** | MAE | 0.82 | 0.53 | 0.50 | **0.32** | 0.51 | 0.35 | 0.34 | **0.32** |
| | NLPD | 1.43 | 1.05 | 0.99 | 0.59 | 1.00 | 0.72 | 3.21 | **0.57** |
| **Parkin.** | MAE | 0.85 | 0.82 | 0.78 | **0.68** | 0.78 | 0.84 | 0.80 | 0.74 |
| | NLPD | 2.88 | 2.79 | 2.73 | **2.52** | 2.73 | 4.49 | 3.80 | 2.66 |
| **Kin40K** | MAE | 0.81 | 0.81 | 0.79 | **0.25** | 0.79 | 0.80 | 0.43 | 0.78 |
| | NLPD | 1.42 | 1.42 | 1.39 | **0.37** | 1.39 | 2.68 | 4.14 | 1.38 |
| **House** | MAE | 0.62 | 0.49 | NA | **0.39** | 0.43 | 0.50 | 0.40 | **0.39** |
| | NLPD | 1.45 | 1.30 | NA | **1.06** | 1.10 | 4.61 | 4.58 | 1.11 |
| **Protein** | MAE | 0.89 | 0.71 | NA | 0.57 | 0.64 | 0.82 | 0.70 | **0.55** |
| | NLPD | 1.41 | 1.25 | NA | **1.11** | 1.19 | 2.38 | 4.57 | **1.11** |
| **Year** | MAE | 0.74 | 0.73 | NA | **0.57** | NA | 0.74 | 0.74 | 0.72 |
| | NLPD | 1.41 | 1.39 | NA | **1.21** | NA | 3.78 | 1.49 | 1.38 |
| **Flight** | MAE | 0.56 | **0.54** | NA | **0.54** | NA | 0.56 | 0.56 | **0.54** |
| | NLPD | 2.87 | 2.85 | NA | **2.80** | NA | 8.05 | 11.51 | 2.84 |

Table 2[4] reports the Mean Absolute Error (MAE) and the Negative Log Predictive Density (NLPD) on each dataset, see supplement for details on the preprocessing and an extended results table. Note that NLPDs for *LR* and *Const* are computed using the inferred noise as the variance of the predictive distribution. We see that DSMGPs consistently outperform other expert-based approaches and often perform competitive or outperform SVGPs[5]. Further, our model consistently captures predictive uncertainties better than previous expert-based approaches resulting in low NLPDs. Note that DSMGPs often have a lower approximation error, compared to exact GPs, then SVGPs.

Additionally, we computed the effective number of mixture components of the DSMGPs and measured the average runtime of a single hyperparameter optimisation step on an i7-6900k CPU @ 3.2 GHz. The effective mixture sizes are: airfoil: $5.44 \times 10^2$, parkinsons: $1.41 \times 10^3$, kin40k: $6.71 \times 10^7$, house: $1.68 \times 10^7$, protein: $7.21 \times 10^{16}$, and year: $4.30 \times 10^{18}$. We want to emphasis that these mixtures are not explicitly constructed but rather implicitly encoded through the structure of the DSMGP. Table 1 lists the resulting runtimes for hyperparameter optimisation, indicating that optimising DSMGPs is competitive to prior work when trained as described above. Finally, we empirically evaluated the performance gains through sharing solutions of the Cholesky decompositions, see supplement Section 0.1. Through exploitation of the structure of DSMGPs we gain a speed-up by a factor of two, allowing us to explore twice as many partitions of the input space.

[4]We refer to the extended arXiv version of this paper for additional experiments and discussion.

[5]We reran the experiments due to errors in the software.

# 6 CONCLUSION

In this paper, we have introduced Deep Structured Mixtures of GPs (DSMGPs), which combine Sum-Product Networks (SPNs) with Gaussian Processes (GPs) as sub-modules, i.e., leaf distributions. For this, we first introduced a measure-theoretic perspective on DSMGPs, extending the standard definition of SPNs. Subsequently, we showed that DSMGPs enable *efficient and exact posterior inference* and have attractive computation costs for hyperparameter optimisation. We discussed that DSMGPs can be understood to perform exact Bayesian model averaging over a large set of naive-local-experts (NLE) models and showed that the structure can be exploited to speed-up computations and model non-stationary data.

Finally, we showed, in a variety of experiments, that DSMGPs provide low approximation errors and capture predictive uncertainties consistently better than existing expert-based approximations. Future directions include, more advanced structure learning techniques, advanced techniques to distributed the load of the individual experts, approaches to reduce the memory requirements and combinations with sparse variational GPs and deep GPs.

### Acknowledgments

We want to thank Mark van der Wilk for the insightful discussion. This work was funded by the Austrian Science Fund (FWF): I2706-N31 and received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 797223 – HYBSPN.

## References


[Burt et al., 2019] Burt, D. R., Rasmussen, C. E., and van der Wilk, M. (2019). Rates of convergence for sparse variational gaussian process regression. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 862–871.

[Cao and Fleet, 2014] Cao, Y. and Fleet, D. J. (2014). Generalized product of experts for automatic and principled fusion of gaussian process predictions. *CoRR*, abs/1410.7827.

[Cornford et al., 1998] Cornford, D., Nabney, I. T., and Williams, C. K. I. (1998). Adding constrained discontinuities to gaussian process models of wind fields. In *In proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pages 861–867.

[Darwiche, 2003] Darwiche, A. (2003). A differential approach to inference in Bayesian networks. *Journal of the ACM*, 50(3):280–305.

[Deisenroth and Ng, 2015] Deisenroth, M. P. and Ng, J. W. (2015). Distributed gaussian processes. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1481–1490.

[Gadd et al., 2019] Gadd, C. W. L., Wade, S., and Boukouvalas, A. (2019). Enriched mixtures of gaussian process experts. *CoRR*, abs/1905.12969.

[Gal et al., 2014] Gal, Y., van der Wilk, M., and Rasmussen, C. E. (2014). Distributed variational inference in sparse gaussian process regression and latent variable models. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pages 3257–3265.

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

[Kim et al., 2005] Kim, H.-M., Mallick, B. K., and Holmes, C. C. (2005). Analyzing nonstationary spatial data using piecewise gaussian processes. *Journal of the American Statistical Association*, 100(470):653–668.

[Liu et al., 2018] Liu, H., Ong, Y., Shen, X., and Cai, J. (2018). When gaussian process meets big data: A review of scalable gps. *CoRR*, abs/1807.01065.

[Meeds and Osindero, 2005] Meeds, E. and Osindero, S. (2005). An alternative infinite mixture of gaussian process experts. In *In proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pages 883–890.

[Park and Huang, 2016] Park, C. and Huang, J. Z. (2016). Efficient computation of gaussian process regression for large spatial data sets by patching local gaussian processes. *Journal of Machine Learning Research (JMLR)*, 17:174:1–174:29.

[Peharz et al., 2017] Peharz, R., Gens, R., Pernkopf, F., and Domingos, P. (2017). On the latent variable interpretation in sum-product networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 39(10):2030–2044.

[Peharz et al., 2015] Peharz, R., Tschiatschek, S., Pernkopf, F., and Domingos, P. M. (2015). On theoretical properties of sum-product networks. In *Proceedings of the Conference on Artificial Intelligence and Statistics (AISTATS)*.

[Poon and Domingos, 2011] Poon, H. and Domingos, P. M. (2011). Sum-product networks: A new deep architecture. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 337–346.

[Press, 2002] Press, W. H. (2002). *Numerical recipes in C++: the art of scientific computing (second ed.)*. Cambridge University Press.

[Rajput and Cambanis, 1972] Rajput, B. S. and Cambanis, S. (1972). Gaussian processes and gaussian measures. *The Annals of Mathematical Statistics*, pages 1944–1952.

[Rasmussen and Ghahramani, 2001] Rasmussen, C. E. and Ghahramani, Z. (2001). Infinite mixtures of gaussian process experts. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pages 881–888.

[Rasmussen and Williams, 2006] Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press.

[Samo and Roberts, 2016] Samo, Y. K. and Roberts, S. J. (2016). String and membrane gaussian processes. *Journal of Machine Learning Research (JMLR)*, 17:131:1–131:87.

[Seeger et al., 2003] Seeger, M. W., Williams, C. K. I., and Lawrence, N. D. (2003). Fast forward selection to speed up sparse gaussian process regression. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*.

[Silverman, 1985] Silverman, B. W. (1985). Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of the Royal Statistical Society*, 47(1):1–52.

[Szabó and van Zanten, 2019] Szabó, B. and van Zanten, H. (2019). An asymptotic analysis of distributed nonparametric methods. *Journal of Machine Learning Research*, 20:87:1–87:30.

[Titsias, 2009] Titsias, M. K. (2009). Variational learning of inducing variables in sparse gaussian processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 567–574.

[Tolvanen et al., 2014] Tolvanen, V., Jylänki, P., and Vehtari, A. (2014). Expectation propagation for nonstationary heteroscedastic gaussian process regression. In *In proceedings of the IEEE International Workshop on Machine Learning for Signal Processing*, pages 1–6.

[Trapp et al., 2019a] Trapp, M., Peharz, R., Ge, H., Pernkopf, F., and Ghahramani, Z. (2019a). Bayesian learning of sum-product networks. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*.

[Trapp et al., 2019b] Trapp, M., Peharz, R., and Pernkopf, F. (2019b). Optimisation of overparametrized sum-product networks. *CoRR*, abs/1905.08196.

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

[Tresp, 2000b] Tresp, V. (2000b). Mixtures of gaussian processes. In *In proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pages 654–660.

[Vasudevan et al., 2009] Vasudevan, S., Ramos, F. T., Nettleton, E., Durrant-Whyte, H. F., and Blair, A. (2009). Gaussian process modeling of large scale terrain. In *In proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1047–1053.

[Wang et al., 2018] Wang, F., Decker, J. M., Wu, X., Essertel, G. M., and Rompf, T. (2018). Backpropagation with callbacks: Foundations for efficient and expressive differentiable programming. In *In proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pages 10201–10212.

[Wilson and Nickisch, 2015] Wilson, A. G. and Nickisch, H. (2015). Kernel interpolation for scalable structured gaussian processes (KISS-GP). In Bach, F. R. and Blei, D. M., editors, *In Proceedings of the International Conference on Machine Learning (ICML)*, volume 37, pages 1775–1784.

[Zhang and Williamson, 2019] Zhang, M. M. and Williamson, S. A. (2019). Embarrassingly parallel inference for gaussian processes. *Journal of Machine Learning Research*, 20:1 – 26.

[Zhao et al., 2016] Zhao, H., Adel, T., Gordon, G. J., and Amos, B. (2016). Collapsed variational inference for sum-product networks. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 1310–1318.

[Zhao et al., 2015] Zhao, H., Melibari, M., and Poupart, P. (2015). On the relationship between sum-product networks and Bayesian networks. In *Proceedings of the International Conference on Machine Learning (ICML)*.