Environmental Modelling and Software 109 (2018) 293–305

Contents lists available at ScienceDirect

Environmental Modelling & Software

journal homepage: www.elsevier.com/locate/envsoft

Geospatial uncertainty modeling using Stacked Gaussian Processes

Kareem Abdelfatah[a], Junshu Bao[b], Gabriel Terejanu[a,*]

[a] Dept. of Computer Science & Engineering at University of South Carolina, United States
[b] Dept. of Mathematics and Computer Science at Duquesne University, United States

ARTICLE INFO

Keywords:
Component-based modeling
Uncertainty propagation
Nonparametric hierarchical model
Cokriging
Data-driven emulators

ABSTRACT

A network of independently trained Gaussian processes (StackedGP) is introduced to obtain predictions of geospatial quantities of interest (model outputs) with quantified uncertainties. The uncertain nature of model outputs is due to model inadequacy, parametric uncertainty, and measurement noise. StackedGP framework supports component-based modeling in environmental science, enhances predictions of quantities of interest through a cascade of intermediate predictions usually addressed by cokriging, and propagates uncertainties through emulated dynamical systems driven by uncertain forcing variables. By using analytical first and second-order moments of a Gaussian process with uncertain inputs using squared exponential and polynomial kernels, approximated expectations of model outputs that require an arbitrary composition of functions can be obtained. The performance of the proposed nonparametric stacked model in model composition and cascading predictions is measured in a wildfire and mineral resource problem using real data, and its application to time-series prediction is demonstrated in a 2D puff advection problem.

Software Availability

An open source Python software package (StackedGP) is available as of January 2018 under GNU General Public License v2 and hosted by UQlab on BitBucket (https://bitbucket.org/uqlab/stackedgp) to create general StackedGP models, perform optimizations and calculate predictions. This software does not require specific hardware but it is dependent on the following Python packages: numpy, scipy, GPy (since 2012) and sklearn.preprocessing. The main developer is the first author of the paper Kareem Abdelfatah, who can be reached at krabea@email.sc.edu.

1. Introduction

Complex environmental models are modular and hierarchical (He et al. (2002); Grützner (1995); Letcher and Jakeman (2009); Jørgensen (2010)). As no one model can describe the entire behavior of a complex system, complex models requires coupling of submodels built using various sources of data. For example, component-based modeling is used in forest landscape modeling (He et al. (2002)), where fire and wind models are coupled with vegetation models to estimate the total burned area, and in crop modeling, where pest population models are coupled with biophysics models to estimate crop growth (Whish et al. (2015)). The central challenge with component-based modeling is that there is a compound effect of uncertainties coming from errors due to structural submodel inadequacies and noise in experimental data that need to be quantified and propagated to the model outputs. This model composition can be arbitrary and highly nested to capture the phenomenon of interest and can be used to make predictions for potentially unobserved quantities of interest.

This paper develops a general probabilistic modeling framework to address the above two challenges: component-based modeling under structural uncertainty and propagation of uncertainties to quantities of interest. One of the current challenges in component-based modeling arises from the fitted parametric nature of submodels with no information most of the time on the magnitude of the uncertainty of the parameters. While parametric uncertainty can be quantified in these cases and propagated to the quantities of interest using sampling methods, the uncertainty in model predictions may still be underestimated by ignoring model form uncertainty. In this paper we proposed to use a data-driven and nonparametric approach to build submodels and develop an expectation-based approach to propagate the uncertainty. The proposed model is based on a network of independently trained Gaussian processes accompanied by an approximate scheme to obtain expectations of quantities of interest that require model composition. Gaussian processes (GP) (Williams and

* Corresponding author.
E-mail addresses: krabea@email.sc.edu (K. Abdelfatah), baoj@duq.edu (J. Bao), terejanu@cec.sc.edu (G. Terejanu).

https://doi.org/10.1016/j.envsoft.2018.08.022
Received 21 May 2018; Received in revised form 9 August 2018; Accepted 22 August 2018
Available online 30 August 2018
1364-8152/

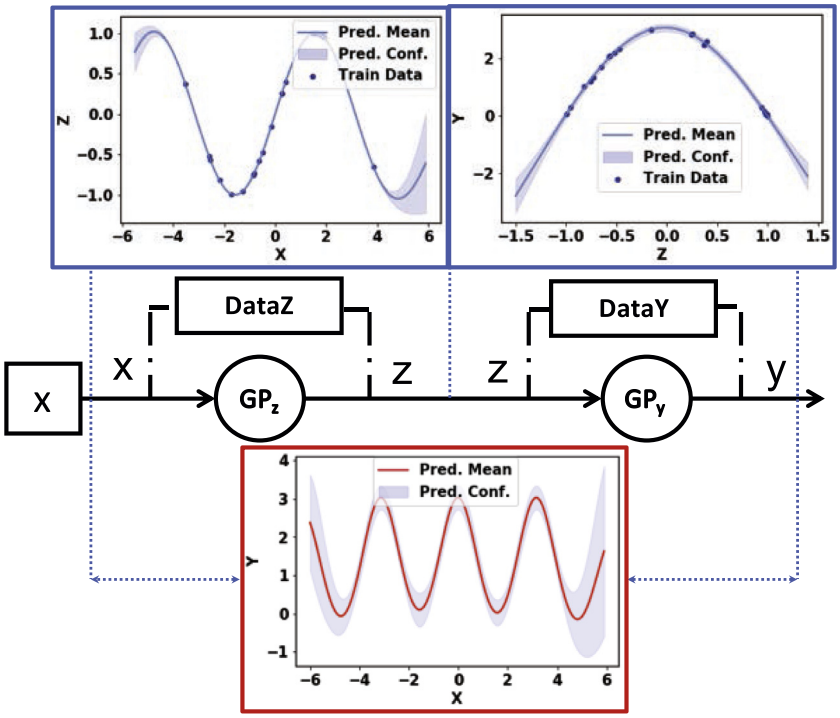

Fig. 1. StackedGP example - two chained Gaussian processes. Circles represent GP nodes and the square represents the input. *DataZ* and *DataY* are used to train the first and second GP, respectively. The figure shows the fitting of GP_z and GP_y with the training data in the blue frames. The final output of the StackedGP is obtained by integrating out the uncertainty in z. The predicted mean and confidence of y given x is shown in the red frame. Note that there is no training data to directly model y as a function of x. (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

Rasmussen (1996); Rasmussen (1997); Rasmussen and Williams (2005)) are nonparametric statistical models that compactly describe distributions over functions with continuous domains. This makes them ideal to quantify uncertainties for environmental subprocesses by modeling measurement noise and structure inadequacies that arise with usual parametric approaches. Since all environmental subprocesses/components are modeled using GPs, the resulted probabilistic model is a stacked Gaussian process (StackedGP). In this hierarchical setting, GPs modeling forcing variables govern the input space of GPs modeling environmental state variables.

To provide the intuition behind the motivation for StackedGP, consider the following simple environmental example of predicting fungal toxin production in corn (y - quantity of interest) at various spatial locations (x). Interpolation methods do not work directly as the quantity of interest is virtually unobservable over the spatial domain (Li et al. (2015)). As a result, this effort requires the derivation of a model composition where the toxin production is modeled as a function of temperature (z), which at its turn is easily observable, and can be estimated at any location via spatial interpolation models. Finding a parametrization for both models to obtain the fungal toxin production at different locations is a non trivial task. Furthermore, the uncertainties in these models as well as in the measurements need to be estimated and propagated to the quantity of interest. Fig. 1 shows a simple scenario that illustrate the motivation of this environmental example. It shows a proposed StackedGP comprising two Gaussian processes for both temperature interpolation and modeling toxin production as a function of temperature to predict with quantified uncertainties the toxin production at various spatial locations. The two GPs are built using two datasets: *DataZ* is obtained from field measurements and consists of temperature values (z) recorded at different locations (x), and *DataY* is obtained from wet-lab experiments and consists of fungal toxin production (y) at different temperatures (z). The final probabilistic prediction of toxin production (y) as a function of location (x) is obtained using StackedGP by integrating out the uncertainty in the temperature (z). The interactive Python code to generate this experiment can be found on our online StackedGP repository[1] and is sketched in Section 1.

The novelty of the paper is in the derivation of a novel approximate algorithm to propagate uncertainties through an arbitrary StackedGP to the quantities of interest using both squared exponential and polynomial kernels (Rasmussen and Williams (2005)). The second contribution is the application of StackedGP to several representative examples in environmental science (wildfire, geology of mineral resources, and atmospheric transport) by emphasizing improvements in modeling areas such as component-based modeling, cokriging, and emulation of dynamical systems. Finally, a Python package[2] is provided to build arbitrary StackedGP models and study uncertainty propagation using the proposed algorithm. All the environmental examples in this paper are included in our online code repository.

The proposed modeling framework is relevant in a number of environmental science problems such as wildfire and forest landscape modeling (Millington et al. (2009); He et al. (2002)), where component-based modeling is used to model various spatio-temporal subprocesses (vegetation, soil moisture, and solar radiation) to predict quantities of interest such as total burned area. In this paper, we study the applicability of the proposed method in predicting the total burned area by encoding the modular structure of the Canadian forest fire weather index (FWI) system (Taylor and Alexander (2006)) into the StackedGP.

In addition to supporting the component-based environmental modeling, StackedGP can be used to enhance predictions of quantities of interest using intermediate predictions of secondary variables, which is usually addressed using cokriging (Wackernagel (1996)). Enhanced predictions of quantities of interest can be obtained by stacking GPs for predicting intermediate secondary responses that govern the input space of GPs used to predict primary responses. Several examples can illustrate the idea such as estimating ozone concentrations (Singh et al.

[1] https://bitbucket.org/uqlab/stackedgp/src/master/Synthetic_Datasets.
[2] https://bitbucket.org/uqlab/stackedgp.

(2011)) using the results of chemical transport model simulation as secondary variables and predicting cadmium concentration using concentration of other metals as secondary variables in Swiss Jura (Goovaerts (1997); Wilson et al. (2012)). StackedGP is not designed to capture the correlations of response variables, however, StackedGP models can be constructed by stacking GPs for predicting intermediate secondary responses that govern the input space of GPs used to predict primary responses. This hierarchical framework outperforms other methods as described in the numerical results section, where Jura dataset (Wilson et al. (2012)) is used to assess the prediction accuracy of model with intermediate predictions.

In environmental sciences, uncertainty propagation through dynamical systems is also relevant when high-fidelity models are emulated (Castelletti et al. (2012); Bayarri et al. (2007); Conti and O'Hagan (2010); Bhattacharya (2007)). For example, propagating uncertainties through atmospheric dispersion models (Nielsen et al. (1999); Sykes et al. (2006)) can be tackled through emulation. In this case, emulators can be used to speed up the uncertainty propagation process and obtain estimates of quantities of interest with quantified uncertainties (Konda et al. (2010); Cheng and Sandu (2009); Conti et al. (2009)). This is pertinent in operational context when model predictions guide decision-making processes and uncertainty propagation and data assimilation (Terejanu et al. (2007, 2008)) need to be performed in real-time. StackedGP is especially applicable in the context of GP emulators driven by forcing variables predicted by other GP or StackedGP models.

```
from stackedGPNetwork import StackedGPNetwork

# create a StackedGP with a specific number of layers
stackedNetwork = StackedGPNetwork(nlayers = 2)

# to create a node specify the layer number and the input and
    output datasets used to train the GP node
stackedNetwork.createNewNode(layerIndx=0, inputdata = DataZ['x'
    ], outputdata = DataZ['z'])
stackedNetwork.createNewNode(layerIndx=1, inputdata = DataY['z'
    ], outputdata = DataY['y'])

# estimate the hyper-parameters of all the nodes in the network
stackedNetwork.optimize()

# final prediction for a particular test input
predMean, predVar = stackedNetwork.predict(x)
```

Code 1: Example StackedGP API.

A simple 2D puff advection example is provided to showcase StackedGP's applicability in uncertainty propagation using emulated dynamical systems.

This work unifies the approach of Girard et al. (2002) and Li et al. (2015). In Li et al. (2015), the authors introduced StackedGP to predict carcinogenic toxin concentrations using environmental conditions and Monte Carlo sampling was used to propagate the uncertainty through the stacked model and estimate the mean and variance of the quantity of interest. Since sampling requires a high computational cost, here, the uncertainty propagation through the network is achieved approximately by leveraging the exact moments for the predictive mean and variance derived by Girard et al. (2002) for a single GP with uncertain inputs and squared exponential kernel.

StackedGP is conceptually different from deep GPs (Damianou and Lawrence (2013)), where no data is available for the latent nodes and where the latent variable model requires to jointly infer the hyperparameters corresponding to the mappings between the layers. A model carrying the same name was introduced by Neumann et al. (2009), where a stacked Gaussian process was proposed to model pedestrian and public transit flows in urban areas. The model proposed by Neumann et al. (2009) is capable of capturing shared common causes using a joint Bayesian inference for multiple tasks. In our work, the inference is performed independently for each GP node and the uncertainty is approximately propagated through the network. StackedGP provides flexibility in kernel selection for intermediate nodes (RBF, polynomial as well as kernels obtained via their sum) and has no restriction in selecting a suitable kernel for input nodes. Since the GP nodes are independently trained using multiple datasets, the running time of the StackedGP grows linearly with the number of nodes and can be sped up through embarrassing parallel training of GPs.

The paper is organized as follows. Section 1 provides information about the StackedGP Python package. This is followed by a brief introduction to GP in Section 4. Section 5 re-derives the expectations of a GP with uncertain inputs for squared exponential kernel, and provides a novel derivation for the polynomial kernel. Section 6 generalizes the StackedGP to an arbitrary number of layers and nodes, and discusses the advantages and limitations of the proposed model. Three numerical results are presented in Section 7 and conclusions are given in Section 8.

2. Brief tutorial for StackedGP software package

The StackedGP software package hosted on BitBucket (https://bitbucket.org/uqlab/stackedgp) contains all the examples used in this paper to showcase the applicability of the StackedGP. The Python implementation using StackedGP of the hypothetical 1D example in the introduction, see Fig. 1, is provided below in Code 1.

3. Gaussian process background

Unlike parametric models, non-parametric models provide infinite dimensional parameters for modeling the distribution of the data. Gaussian processes are popular non-parametric models (Rasmussen and Williams (2005); Williams and Rasmussen (1996); Williams (1998); Reggente et al. (2014)) that have found various applications in the environmental modeling community. They are used as data-driven models capable to predict various quantities of interest with quantified uncertainties such as ultra fine particles (Reggente et al. (2014)), mean temperatures over North Atlantic Ocean (Higdon (1998)), wind speed (Hu and Wang (2015)), and monthly streamflow (Sun et al. (2014)), just to name a few. When the training data for GPs comes from simulators rather than field measurements, then GPs become computational efficient surrogate models or emulators of high-fidelity models (Kennedy et al. (2002); O'Hagan (2006); Conti and O'Hagan (2010)), with various applications in environmental modeling such as fire emissions (Katurji et al. (2015)), ocean and climate circulation (Tokmakian et al. (2012)), urban drainage (Machac et al. (2016)), and computational fluid dynamics (Moonen and Allegrini (2015)).

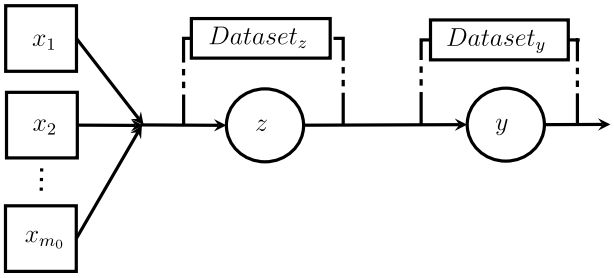

Fig. 2. Simple StackedGP - two chained Gaussian processes. Circles represent a GP node and squares represent the observable inputs. $Dataset_z$ and $Dataset_y$ are used to train the first and second GP, respectively.

Given $D = \{\boldsymbol{X}, \boldsymbol{z}\}$, a set of n data points, each consisting of d inputs ($\boldsymbol{X} \in \Re^{n\times d}$) and one output ($\boldsymbol{z} \in \Re^n$), the output of the ith data point, z_i, is modeled as follows:

$$z_i = g(\boldsymbol{x}_i) + \varepsilon_i^z \tag{1}$$

$$\varepsilon_i^z \sim N\left(0, \sigma_{\varepsilon_z}^2\right) \tag{2}$$

$$g \sim \mathrm{GP}(0, k_z(\cdot,\cdot)) \tag{3}$$

Here, g represents a latent function with zero mean Gaussian process prior and kernel or covariance function $k_z(\cdot,\cdot)$. The kernel measures the similarity between two inputs, $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. For example, the squared exponential or radial basis function (RBF) kernel is defined as follows.

$$k_z(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi \exp\{-\theta\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2\} \tag{4}$$

The hyperparemeters, $\sigma_{\varepsilon_z}^2$, and e.g. ϕ and θ corresponding to the RBF kernel, are estimated using the maximum likelihood approach, where the log-likelihood is given by,

$$\ln p = -\frac{1}{2}\boldsymbol{z}^T\left(\boldsymbol{K}_z + \sigma_{\varepsilon_z}^2 I\right)(\boldsymbol{z}|\boldsymbol{X}, \phi, \theta, \sigma_{\varepsilon_z}^2)^{-1}\boldsymbol{z} - \frac{1}{2}\ln\left|\boldsymbol{K}_z + \sigma_{\varepsilon_z}^2 I\right| - \frac{n}{2}\ln 2\pi\ , \tag{5}$$

and the covariance matrix $\boldsymbol{K}_z$ is an $n \times n$ Gram matrix with elements $K_{ij} = k_z(\boldsymbol{x}_i, \boldsymbol{x}_i)$.

Once the hyperparameters are estimated, the predictive distribution of z^* at a new testing input$\boldsymbol{x}^*$, is given by the following normal distribution.

$$z^* \sim N\left(\mu_{z^*}, \sigma_{z^*}^2\right) \tag{6}$$

$$\mu_{z^*} = \boldsymbol{k}_z^T \boldsymbol{C}_z^{-1}\boldsymbol{z} \tag{7}$$

$$\sigma_{z^*}^2 = k_z(\boldsymbol{x}^*, \boldsymbol{x}^*) + \sigma_z^2 - \boldsymbol{k}_z^T \boldsymbol{C}_z^{-1}\boldsymbol{k}_z \tag{8}$$

$$\boldsymbol{C}_z = \boldsymbol{K}_z + \sigma_{\varepsilon_z}^2 I \tag{9}$$

In the following section we provide the background for a simple StackedGP as an extension to GP with uncertain inputs as initially developed by Girard et al. (2002).

4. Simple StackedGP - two chained Gaussian processes

Consider the following simple StackedGP in Fig. 2 given by two chained GPs with their own training dataset. The input to the first GP is given by the vector $\boldsymbol{x}$. The output of the first GP, z governs the input to the second GP, and y is the final output of the StackedGP in Fig. 2.

The goal of this section is to introduce the mechanism of obtaining analytical expectations of two-layer StackedGPs for both RBF and polynomial kernels. Note that the predictive distribution of even a simple StackedGP as the one in Fig. 2 is non-Gaussian, however its mean and variance can be obtained analytically. In the next section we will generalize the approach to obtain the approximate expectations of StackedGPs with arbitrary number of layers and nodes per layer.

We start with providing analytical expressions for mean and variance for a general kernel, and follow with specific expressions for RBF kernel as initially derived by Girard et al. (2002), and then with a novel derivation for polynomial kernel.

The predicted mean of the StackedGP with input $\boldsymbol{x}^*$ is obtained using the law of total expectation by integrating out the intermediate variable z^*:

$$E[y^*|\boldsymbol{y}, \boldsymbol{x}^*] = \mathrm{E}_{z^*}[\mathrm{E}[y^*|\boldsymbol{y}, \boldsymbol{x}^*, z^*]] \tag{10}$$

Here, $\mathrm{E}[y^*|\boldsymbol{y}, \boldsymbol{x}^*, z^*] = \boldsymbol{k}_y^T \boldsymbol{C}_y^{-1}\boldsymbol{y}$ is the expectation of a standard GP with input z and output y, and it can be expanded as follows:

$$\mathrm{E}[y^*|\boldsymbol{y}, \boldsymbol{x}^*, z^*] = \boldsymbol{y}^T \sum_{i=1}^{n} \boldsymbol{C}_y^{-1}(,i) k_y(z^*, z_i)\ , \tag{11}$$

where $\boldsymbol{C}_y$ is the covariance matrix of the second GP and $k_y(z^*, z_i)$ is the kernel between the predicted variable z^* and the i^{th} training data point z_i, and n is the number of training points for the target node. The final predicted analytical mean of y^* can be written as

$$\mathrm{E}[y^*|\boldsymbol{y}, \boldsymbol{x}^*] = \boldsymbol{y}^T \sum_{i=1}^{n} \boldsymbol{C}_y^{-1}(,i) \underbrace{\mathrm{E}_{z^*}[k_y(z^*, z_i)]}_{\Delta_1}\ . \tag{12}$$

$\mathrm{E}_{z^*}[k_y(z^*, z_i)]$ is the key integration to obtain the analytical predicted mean. The expectation in Eq. (12) is with respect to a normal distribution with mean μ_{z^*} and variance $\sigma_{z^*}^2$ as obtained from the prediction of the first GP. The expectation can be obtained analytically for RBF and polynomial kernels as shown in the following two subsections.

The variance of the StackedGP can be obtained similarly using the law of total variance.

$$\begin{aligned}\mathrm{Var}(y^*|\boldsymbol{y}, \boldsymbol{x}^*) = &\ \mathrm{E}_{z^*}[\mathrm{Var}(y^*|\boldsymbol{y}, \boldsymbol{x}^*, z^*)] + \mathrm{Var}_{z^*}(\mathrm{E}[y^*|\boldsymbol{y}, \boldsymbol{x}^*, z^*]) \\ = &\ \mathrm{E}_{z^*}[k_y(z^*, z^*) + \sigma_y^2 - \boldsymbol{k}_y^T \boldsymbol{C}_y^{-1}\boldsymbol{k}_y] + \mathrm{Var}_{z^*}(\boldsymbol{k}_y^T \boldsymbol{C}_y^{-1}\boldsymbol{y}) \\ = &\ \sigma_{\varepsilon_y}^2 + \underbrace{\mathrm{E}_{z^*}[k_y(z^*, z^*)]}_{\Delta_2} - \mathrm{E}_{z^*}[\boldsymbol{k}_y^T \boldsymbol{C}_y^{-1}\boldsymbol{k}_y] + \mathrm{Var}_{z^*}(\boldsymbol{k}_y^T \boldsymbol{C}_y^{-1}\boldsymbol{y})\end{aligned} \tag{13}$$

Here, $\sigma_{\varepsilon_y}^2$ is the noise variance of the target GP and $\mathrm{E}_{z^*}[\boldsymbol{k}_y^T \boldsymbol{C}_y^{-1}\boldsymbol{k}_y]$ can be obtained using the following expansion.

$$\mathrm{E}_{z^*}[\boldsymbol{k}_y^T \boldsymbol{C}_y^{-1}\boldsymbol{k}_y] = \sum_{i=1}^{n}\sum_{j=1}^{n} \boldsymbol{C}_y^{-1}(i, j) \underbrace{\mathrm{E}_{z^*}[k_y(z^*, z_i)k_y(z^*, z_j)]}_{\Delta_3} \tag{14}$$

The last term in Eq. (13) is given by,

$$\mathrm{Var}_{z^*}(\boldsymbol{k}_y^T \boldsymbol{C}_y^{-1}\boldsymbol{y}) = \boldsymbol{y}^T \boldsymbol{C}_y^{-1}\boldsymbol{\Sigma}_k \boldsymbol{C}_y^{-1}\boldsymbol{y} \tag{15}$$

where, $\boldsymbol{\Sigma}_k = \mathrm{Var}_{z^*}(\boldsymbol{k}_y) \in \Re^{n\times n}$ can be expressed as

$$\boldsymbol{\Sigma}_k = \mathrm{E}_{z*}[\boldsymbol{k}_y \boldsymbol{k}_y^T] - \mathrm{E}_{z^*}[\boldsymbol{k}_y]\mathrm{E}_{z^*}[\boldsymbol{k}'_y]. \tag{16}$$

Note that $\boldsymbol{\Sigma}_k$ is computed using the two integrations of Δ_1 and Δ_3.

In the following two subsections, we will provide the analytical first and second moments of StackedGP for RBF and polynomial kernels.

4.1. RBF Kernel - simple case

Using the RBF kernel $k_y(z^*, z_i) = \phi\exp\{-\theta(z^* - z_i)^2\}$ to evaluate Δ_1 in Eq. (12) we obtain:

$$\Delta_1 = \phi\sqrt{\frac{(1/(2\theta))}{\sigma_{z^*}^2 + (1/(2\theta))}}\exp\left\{-\frac{(z_i - \mu_{z^*})^2}{2\left(\sigma_{z^*}^2 + 1/(2\theta)\right)}\right\}$$

$$\mathrm{E}[y^*|\boldsymbol{y}, \boldsymbol{x}^*] = \phi\boldsymbol{y}^T\sqrt{\frac{(1/(2\theta))}{\sigma_{z^*}^2 + (1/(2\theta))}} \times \sum_{i=1}^{n} \boldsymbol{C}_y^{-1}(,i)\exp\left\{-\frac{(z_i - \mu_{z^*})^2}{2\left(\sigma_{z^*}^2 + 1/(2\theta)\right)}\right\} \tag{17}$$

Here, θ is the corresponding length scale in the target node, ϕ is the kernel's variance, and $\boldsymbol{y}^T$ is the output training points that have been used during training of the target GP node.

For RBF kernel, $\Delta_2 = \phi$ and Δ_3 in Eq. (13) can be calculated using the following expression.

$$\Delta_3 = \phi^2 \sqrt{\frac{1/(4\theta)}{1/(4\theta) + \sigma_{z_*}^2}} \times \exp\left\{ -\frac{\theta(z_i - z_j)^2}{2} - \frac{[(z_i + z_j)/2 - \mu_{z_*}]^2}{2\left(1/(4\theta) + \sigma_{z_*}^2\right)} \right\}$$

Here, z_i is the i^{th} input training data point for the target node. Finally, the predicted variance is given by:

$$\mathrm{Var}(y^*|\boldsymbol{y}, \boldsymbol{x}^*) = \sigma_{\varepsilon_y}^2 + \phi + \boldsymbol{y}^T \boldsymbol{C}_y^{-1} \boldsymbol{\Sigma}_k \boldsymbol{C}_y^{-1} \boldsymbol{y} - \phi^2 \sum_{i=1}^{n} \sum_{j=1}^{n} \boldsymbol{C}_y^{-1}(i,j) \sqrt{\frac{1/(4\theta)}{1/(4\theta) + \sigma_{z_*}^2}} \times \exp\left\{ -\frac{\theta(z_i - z_j)^2}{2} - \frac{\left[(z_i + z_j)/2 - \mu_{z_*}\right]^2}{2\left(1/(4\theta) + \sigma_{z_*}^2\right)} \right\} \tag{18}$$

These analytical expressions corresponding to the RBF kernel coincide with those derived by Girard et al. (2002) and Candela et al. (2003). We have provided them here for completeness and to emphasize the role of uncertainty in the network as described in the following sections. In the next subsection we provide novel analytical expressions for the predicted mean and variance of StackedGP when using polynomial kernels.

4.2. Polynomial Kernel - simple case

Following the same simple StackedGP configuration and a d-order polynomial kernel at the target node $k_y(z^*, z_i) = (z^* * z_i)^d$, the predicted mean of Eq. (12) can be calculated as

$$\mathrm{E}[y^*|\boldsymbol{y}, \boldsymbol{x}^*] = \boldsymbol{y}^T \sum_{i=1}^{n} \boldsymbol{C}_y^{-1}(,i)(a_d z_i^d)$$

where $\Delta_1 = (a_d z_i^d)$ and a_d follows the non-central moments of the normal distribution, namely

$$a_d = \sum_{u=0}^{\left\lfloor \frac{d}{2} \right\rfloor} \binom{d}{2u} (2u-1)!! \sigma_{z_*}^{2u} \mu_{z_*}^{d-2u}. \tag{19}$$

The expression for the predicted variance in Eq. (13) is obtained by substituting $\Delta_2 = a_{2d}$ and $\Delta_3 = a_{2d} z_i^d z_j^d$ where a_{2d} is calculated using Eq. (19). Finally, the predicted variance in the case of polynomial kernel is given by,

$$\mathrm{Var}_{poly}(y^*|\boldsymbol{y}, \boldsymbol{x}^*) = \sigma_{\varepsilon_y}^2 + a_{2d} + \boldsymbol{y}^T \boldsymbol{C}_y^{-1} \boldsymbol{\Sigma}_k \boldsymbol{C}_y^{-1} \boldsymbol{y} - \sum_{i=1}^{n} \sum_{j=1}^{n} a_{2d} z_i^d z_j^d \boldsymbol{C}_y^{-1}(i,j).$$

5. Stacked Gaussian Process - generalization

The goal of this section is to extend the previous StackedGP to an arbitrary number of layers and nodes per layer. First, we start by presenting the analytical mean and variance of a two-layer StackedGP with arbitrary number of nodes in the first layer. Second, we provide a discussion on accommodating an arbitrary number of output nodes in the second layer. Finally, we present an algorithm to compute the approximate mean and variance of a generalized StackedGP, and discuss the advantages and limitations of the model.

5.1. Generalized number of nodes in the first layer of a two layer StackedGP

Consider an arbitrary number of nodes in the first layer as an extension of the simple two layer StackedGP in the previous section while keeping the single output, see Fig. 3. The analytical expectations

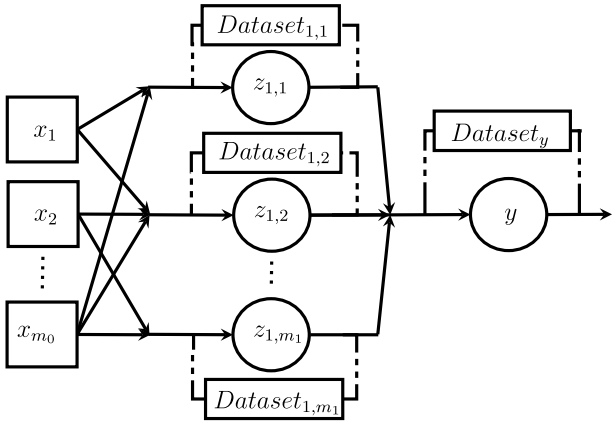

Fig. 3. StackedGP with multiple nodes in the first layer. Circles represent GP nodes and squares represent the observable inputs.

presented here will require the independence assumption for the input uncertainties in the target node. Namely, the outputs of the first layer, $\boldsymbol{z} = [z_1, z_2 ... z_{m_1}]^T$ are considered independent, see Eq. (20). In addition, the multidimensional kernel is assumed to be obtained as a product of 1D kernels. This can easily be extended to sum of kernels and sum of products of 1D kernels.

$$k_y(\boldsymbol{z}^*, \boldsymbol{z}) = \prod_{j=1}^{m_1} k_y(z_j^*, z_j) \tag{20}$$

Thus, the expectation of the kernel is factorized as follows:

$$\mathrm{E}_{\boldsymbol{z}^*}[k_y(\boldsymbol{z}^*, \boldsymbol{z})] = \prod_{j=1}^{m_1} \mathrm{E}_{z_j^*}[k_y(z_j^*, z_j)] \tag{21}$$

Eq. (12) for the mean is generalized as follows,

$$\mathrm{E}[y^*|\boldsymbol{y}, \boldsymbol{x}^*] = \boldsymbol{v}^T \boldsymbol{C}_y^{-1} \boldsymbol{y}\,, \tag{22}$$

where the elements of the vector $\boldsymbol{v} \in \Re^{n \times 1}$ correspond to the training data points $\boldsymbol{z}_i$ for $i = 1...n$ and act as kernels under the uncertain inputs.

$$v_i = \mathrm{E}_{\boldsymbol{z}^*}[k_y(\boldsymbol{z}^*, \boldsymbol{z}_i)] = \prod_{j=1}^{m_1} \mathrm{E}_{z_j^*}\left[k_y(z_j^*, z_{j_i})\right] \tag{23}$$

Similarly, given Eq. (14) the predicted variance of the target node can be generalized as follows:

$$var\,[y^*|\boldsymbol{y}, \boldsymbol{x}^*] = \sigma_{\varepsilon_y}^2 + \Delta_{2_g} + \underbrace{\boldsymbol{y}^T \boldsymbol{C}^{-1} \boldsymbol{\Sigma}_k \boldsymbol{C}^{-1} \boldsymbol{y}}_{\zeta} - \sum_{n,n} (\boldsymbol{C}^{-1} \odot \boldsymbol{H}) \tag{24}$$

where the symbol "$\odot$" is used for element-wise product or Hadamard product and the elements of $\boldsymbol{H} \in \Re^{n \times n}$ reflect integrations under the uncertain inputs of the product of two kernel functions as given in Eq. (20) and evaluated at different training data points.

$$\boldsymbol{H}_{i,j} = \mathrm{E}_{\boldsymbol{z}^*}[k_y(\boldsymbol{z}^*, \boldsymbol{z}_i) k_y(\boldsymbol{z}^*, \boldsymbol{z}_j)] \tag{25}$$

5.1.1. RBF kernel - generalized number of nodes in the first layer

The analytical mean in the case of the RBF kernel for the output node is obtained using the following elements of the $\boldsymbol{v}$ vector in Eq. (22).

$$v_i = w q_i \tag{26}$$

$$w = \prod_{j=1}^{m_1} \sqrt{\frac{1/(2\theta_j)}{\left(\left(1/\left(2\theta_j\right) + \sigma_{z_j}^2\right)\right.}} \tag{27}$$

$$q_i = \phi \exp\left\{ \sum_{j=1}^{m_1} - \frac{\left(z_{j_i} - \mu_{z_j}^*\right)^2}{2\left(\left(1/\left(2\theta_j\right) + \sigma_{z_j}^{2*}\right)\right)} \right\} \tag{28}$$

Here, $i = 1..n$, where n is the number of training data points for the output node, m_1 is the number of inputs to the output GP node, and z_{j_i} is the j^{th} element of the i^{th} training data point. Note that the predicted mean of the StackedGP has the same form as the standard GP but with two main differences. First, the kernel evaluations v_i measure the similarity between the i^{th} training data and the predicted mean $\boldsymbol{\mu}_{z^*}$ from the previous layer instead of the direct input. Second, the similarity is discounted based on the input uncertainty $\sigma_{z_j}^{2*}$. Note, that if we set $\sigma_{z_j}^{2*}$ to zero, we obtain a common product of RBF kernels corresponding to each node in the first layer. However, the larger the input uncertainty for a particular node the lower the similarity on that particular dimension.

To obtain the analytical variance for the RBF kernel in Eq. (24), we use the following relations: $\Delta_{2_g} = \phi$ and $\boldsymbol{H} = u\boldsymbol{P}$ where the scalar u and the elements of $\boldsymbol{P} \in \Re^{n\times n}$ are given by

$$u = \prod_{j=1}^{m_1} \sqrt{\frac{1/(4\theta_j)}{\left(\left(1/\left(4\theta_j\right) + \sigma_{z_j}^{2*}\right)\right)}} \tag{29}$$

$$\boldsymbol{P}_{a,b} = \phi^2 \exp\left\{ - \sum_{j=1}^{m_1} \left\{ \frac{\theta_j (z_{j_a} - z_{j_b})^2}{2} + \frac{\left[(z_{j_a} + z_{j_b})/2 - \mu_{z_j}^*\right]^2}{2\left(1/(4\theta_j) + \sigma_{z_j}^{2*}\right)} \right\} \right\}. \tag{30}$$

Using Eq. (16), we can get the following expression for $\boldsymbol{\Sigma}_k$:

$$\boldsymbol{\Sigma}_k = u\boldsymbol{P} - w^2\boldsymbol{T} \tag{31}$$

where the elements of the matrix $\boldsymbol{T} \in \Re^{n\times n}$ are defined as

$$\boldsymbol{T}_{a,b} = \phi^2 \exp\left\{ - \sum_{j=1}^{m_1} \frac{\left(z_{j_a} - \mu_{z_j}^*\right)^2 + \left(z_{j_b} - \mu_{z_j}^*\right)^2}{2\left(1/(2\theta_j) + \sigma_{z_j}^{2*}\right)} \right\}. \tag{32}$$

We emphasize the impact of input uncertainty on the predictive mean and variance, which is key in obtaining better predictions. Namely, the input uncertainty weighs the contributions of the particular input to the GP node's prediction. Note that if the uncertainty from the first layer $\sigma_{z_j}^{2*} = 0$ then we obtain the same standard variance of a Gaussian process. Namely, the scalers u and w become one and $\boldsymbol{P}_{a,b} = \boldsymbol{T}_{a,b} = \boldsymbol{k}_y\left(z_a, \mu_{z_j}^*\right)\boldsymbol{k}_y\left(z_b, \mu_{z_j}^*\right)^T$, which yields $\boldsymbol{\Sigma}_k = 0$ and thus $\zeta = 0$ in Eq. (24). As a result, in the case of certain inputs, the predicted variance of the StackedGP is similar to the standard GP, namely $\sigma_{\hat{y}}^2 + \phi - \boldsymbol{k}_y^T\boldsymbol{C}^{-1}\boldsymbol{k}_y$. Here, $\boldsymbol{k}_y$ is the kernel evaluated at the training point and the predicted mean of the first layer. In other words, if we have certain inputs, we get standard GP prediction. Otherwise, the uncertainty in the first layer is propagated to the second layer, increasing the predictive uncertainty of the StackedGP output.

In the next section we expand these derivations to polynomial kernels.

5.1.2. Polynomial kernel - generalized number of nodes in the first layer

The analytical mean in the case of polynomial kernel of order d for the output node is obtained using the following multinomial expansion for the i^{th} element of the $\boldsymbol{v}$ vector in Eq. (22).

$$\boldsymbol{v}_i = \sum_{p_1+p_2+\ldots p_{m_1}=d} \binom{d}{p_1, p_2, \ldots p_{m_1}} \prod_{1\le t\le m_1} \left[a_{p_t} z_{t_i}^{p_t}\right]. \tag{33}$$

Here, p_i indicates the power of the t^{th} input with $1 \le t \le m_1$. In additions, $\binom{d}{p_1, p_2, \ldots p_{m_1}} = \frac{d!}{p_1!p_2!\ldots p_{m_1}!}$, and the coefficient a_{p_t} follows the non-central moment of the normal distribution shown in Eq. (19). Note, that in the absence of input uncertainty, namely setting $\sigma_{z_j}^{2*} = 0$, we actually set all but the first term in Eq. (19) to zero, which results in the same formula for the mean of a standard GP with a polynomial kernel of order d.

To obtain the analytical variance for the polynomial kernel in Eq. (24), we use the following relations:

$$\Delta_{2_g} = \sum_{p_1+p_2+\ldots p_{m_1}=d} \binom{d}{p_1, p_2, \ldots p_{m_1}} \prod_{1\le t\le m_1} \left[a_{2p_t}\right] \tag{34}$$

$$a_{2p_t} = \sum_{u=0}^{\left\lfloor \frac{2*p_t}{2} \right\rfloor} \binom{2*p_t}{2u} (2u-1)!!\sigma_{z_t}^{2u}\mu_{z_t}^{2*p_t-2u}. \tag{35}$$

Using Eq. (16), we can get the expression for $\boldsymbol{\Sigma}_k$:

$$\boldsymbol{\Sigma}_k = \boldsymbol{H} - \boldsymbol{v}\boldsymbol{v}^T \tag{36}$$

where the elements of the matrix $\boldsymbol{H} \in \Re^{n\times n}$ are obtained using the following multinomial expansion,

$$\boldsymbol{H}_{i,j} = \sum_{p_1+\ldots p_{m_1}=d} \sum_{q_1+\ldots q_{m_1}=d} \binom{d}{p_1, \ldots p_{m_1}} \binom{d}{q_1, \ldots q_{m_1}} \prod_{1\le t\le m_1} \left[a_{p_t,q_t} z_{t_i}^{p_t} z_{t_j}^{q_t}\right] \tag{37}$$

$$a_{p_t,q_t} = \sum_{u=0}^{\left\lfloor \frac{p_t+q_t}{2} \right\rfloor} \binom{p_t+q_t}{2u} (2u-1)!!\sigma_{z_t}^{2u}\mu_{z_t}^{p_t+q_t-2u} \tag{38}$$

Similarly as in the RBF case, if there is no uncertainty coming from the first layer, namely $\sigma_{z_j}^{2*} = 0$, then $\boldsymbol{H} = \boldsymbol{v}\boldsymbol{v}^T$, which yields $\boldsymbol{\Sigma}_k = 0$ and thus $\zeta = 0$ in Eq. (24). Since $\boldsymbol{H}_{a,b} = \boldsymbol{k}_y\left(z_a, \mu_{z_j}^*\right)\boldsymbol{k}_y\left(z_b, \mu_{z_j}^*\right)^T$, this leads to a predicted variance of the StackedGP similar to the standard GP with polynomial kernel, $\sigma_{\hat{y}}^2 + \Delta_{2_g} - \boldsymbol{k}_y^T\boldsymbol{C}^{-1}\boldsymbol{k}_y$. Here, $\boldsymbol{k}_y$ is the polynomial kernel evaluated at the predicted mean of the first layer and the training points.

Note that the first two moments can be easily obtained also for kernels that involve sums of RBF and polynomial kernels. In the following section we discuss how we can expand the two-layer network to arbitrary number of outputs, and finally the assumptions needed to obtain approximate expectations in a StackedGP with arbitrary number of layers and nodes per layer.

5.2. StackedGP with arbitrary number of layers and nodes per layer

The only assumption in the previous sections is that the outputs of layers that propagate as inputs to the next layer are independent. This applies also to the extension of the previous StackedGP to an arbitrary number of outputs in the last layer. This assumption is for convenience as the derivations are significantly more involving, however the methodology can accommodate correlated inputs. For example, cokriging methods (Cressie (1992)) and dependent GPs (Boyle and Frean (2005)) provide an alternative formulation for obtaining coupled outputs. Any of these models might be used to generate correlated outputs for any layer, however these correlations need to be incorporated into the StackedGP expectations. In our numerical results, we have opted to pre-process the training data using independent component analysis (ICA) to obtain independent projections that are finally used to train the GPs. Note that this procedure does not include the deterministic input observations. We plan to extend the derivations to account for correlations in our next study.

The objective of this section is to build a StackedGP to model an m_l

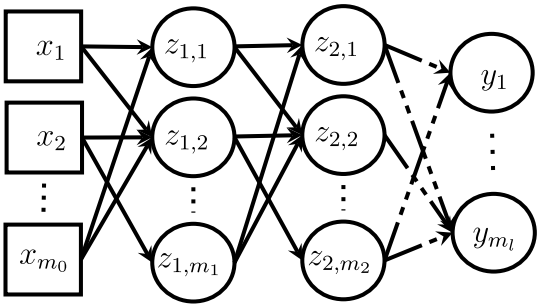

Fig. 4. Stacked Gaussian Process model. The output dimension of $\boldsymbol{y}(x)$ is m_l where the model has l stacked layers and each layer has m_i GP nodes (i refers to the index of the layer). Circles represent a GP node and squares represent the observable inputs.

dimensional function $\boldsymbol{y}(x)$ as shown in Fig. 4. The model has l stacked layers with each layer having m_i GP nodes (l refers to the index of the layer and the value of m_l can be different from layer to layer). We assume that we are given the following set of training datasets $D_{train} = \{D_1, D_2, ...D_Q\}$, where $Q = \sum_{i=1}^{l} m_i$ represents the total number of nodes in the model. In this stacked model each node is independently trained using its own available dataset D_q, where $q = 1..Q$. Thus, each node acts as a standalone standard GP, where the hyper-parameter optimization/inference is conducted using node specific datasets.

While for two-layer StackedGP the mean and the variance can be obtained analytical for both RBF and polynomial kernel, in the case of three or more layers the expectations are intractable for the RBF kernel, and in the case of polynomial kernels, they involve keeping track of large number of terms. We have opted to approximately propagate the uncertainty from layer to layer and approximate the expectations of the StackedGP. Note that even if we are able to obtain analytical expectations for a chain of two GPs, the underlying distribution is still non-Gaussian. As a result, in addition to the independence assumption for the outputs of each layer, we add another assumption which involves approximating the distribution of the output of each layer with a Gaussian distribution. Given the analytical mean and variance, we use the maximum entropy principle to obtain the Gaussian approximation (Shore and Johnson (1980); Trebicki and Sobczyk (1996)). The effect of this approximation is an increase in the uncertainty that is propagated through the network, resulting in conservative predictions.

In large networks or multi-step predictions this uncertainty inflation due to maximum entropy approach might have a significant impact. However, this impact is minimized in applications such as data assimilation, where frequent measurements can reduce the predicted uncertainty. Furthermore, a sensitivity analysis can be used to determine the nodes and the inputs that contribute the most to the final uncertainty of the quantity of interest. This way, one can allocate resources such as targeted data collection or kernel tuning to improve the GP model of the node with the highest uncertainty contribution.

Finally, Eqs. (22) and (24) provide the main mechanism to obtain the approximate mean and variance of a layer given the predictions of the previous layer. This process is applied sequentially until the mean and variance of the final quantities of interest are obtained. Algorithm 1 demonstrates how a general StackedGP is built and the steps required to obtain the desired expectations.

Algorithm 1. StackedGP - model building and uncertainty propagation

One limitation of the model is related to the matrix inversion required by the standard GP model, which takes $\mathcal{O}(n^3)$ operations, where n is the number of training data points for a particular node. Several approaches have been proposed to deal with the curse of dimensionality: kernel mixing (Higdon (1998)), sparse GP with pseudo-inputs (Snelson and Ghahramani (2006)), incremental local Gaussian regression (Meier et al. (2014)), and inversion free approaches (Anitescu et al. (2017)).

Require: $D_{train} = [D_1, D_2, ...D_Q]$. Q number of nodes in the StackedGP.
Require: $nodeLayerIdx = \{(l, n)_j\}_{j=1...Q}$. Q tuples of layer and node index for each node.
Require: *stackedStructure*: an array of Q lists, where each list $stackedStructure[node]$ has an arbitrary number of tuples to specify the inputs nodes to the current *node*.
Require: New observation $\boldsymbol{x}^*$
{# Create StackedGP}
for *i in range*$(1, Q)$: **do**
  kernel initialization (RBF, Polynomial, or RBF + Polynomial).
  if $nodeLayerIdx[i][1]! = 0$ **then**
    apply ICA on $D_{train}[i].X$.
  end if
  init *node* with inputs $D_{train}[i].X$ and outputs $D_{train}[i].Y$
  estimate hyperparameters for *node*.
  add *node* to StackedGP at location $nodeLayerIdx[i]$
end for
{# Uncertainty propagation}
for *i in range*(*number of layers*) **do**
  for *node in layer*$[i]$.*nodes* **do**
    extract mean and variance of all inputs from $stackedStructure[node]$
    {# Calculate the mean and variance for the current node}
    RBF kernel: mean (Eqs. 22, 28), and variance (Eqs. 24, 29, 30, 32).
    Polynomial kernel: mean (Eqs. 22, 33), variance (Eqs. 24, 34, 37).
  end for
end for

When the output of various layers is high-dimensional, then dimensionality reduction techniques can be added to pre-process the training data (Higdon et al. (2008)). Also, various operations in Algorithm 4 are easily parallelizable. Namely, the optimization for hyperparameter estimation of each node can be carried out in parallel as well as within layer propagation of information from the previous layer. Obviously, this computational efficiency over multi-output methods comes at a cost of properly accommodating for the correlation of the outputs.

6. Numerical results

In this section we provide three different examples to demonstrate the applicability of StackedGP. The first application corresponds to the Jura geological dataset, where the StackedGP is used to enhance the prediction of a primary response using intermediate predictions of secondary responses. In the second example, we use StackedGP to combine two real datasets to predict the burned area as part of a forest fire application. Finally, we demonstrate the use of StackedGP in the context of emulated dynamical systems for 2D puff advection driven by uncertain inputs for multi-step predictions.

6.1. Cascading predictions - Jura dataset

In this subsection we use Jura dataset collected by the Swiss Federal Institute of Technology at Lauasanne (Atteia et al. (1994); Webster et al. (1994)). The dataset contains concentration samples of several heavy metals at 359 different locations. Similar to previous experiments (Goovaerts (1997); Alvarez and Lawrence (2011a); Wilson et al. (2012)), we are interested in predicting cadmium concentrations, the primary response at 100 locations given 259 training measurement points. The training data contains location information and concentrations of various metals (Cd, Zn, Ni, Cr, Co, Pb and Cu) at the sampled sites. The primary response is the concentration of Cd, and the other metals are considered secondary responses.

Note that standard Gaussian processes model each response variable independently and thus knowledge of secondary responses cannot help in predicting the primary one (Teh et al. (2005)). In this case a standard Gaussian process (StandardGP) will use a training dataset with only locations as inputs and Cd measurements as target (Alvarez and Lawrence (2011a); Wilson et al. (2012)). Multi-output regression models such as co-kriging (Cressie (1992)) can use the correlation between secondary and primary response to improve the prediction of Cd. The StackedGP, while it does not model the correlation between primary and secondary responses, it can be used to enhance the prediction of the primary response using intermediate predictions of the secondary responses.

Table 1
Example 1 (cascading predictions) - Performance on modeling Cd using different two/three layers StackedGP structures with mean absolute error (MAE) as performance metric.

Model	MAE	STD
StackedGP	0.3860	6×10^{-3}
StackedGP(Co)	0.3617	3.3×10^{-6}
StackedGP(Cr)	0.3884	7.5×10^{-7}
StackedGP(Co,Cr)	0.3602	5.7×10^{-7}
GPRN(VB) Wilson et al. (2012)	0.4040	6×10^{-4}
SLFM(VB) Teh et al. (2005)	0.4247	4×10^{-4}
SLFM Teh et al. (2005)	0.4578	2.5×10^{-3}
ICM Goovaerts (1997)	0.4608	2.5×10^{-3}
CMOGP Boyle and Frean (2004)	0.4552	1.3×10^{-3}
Co-Kriging	0.51	
StandardGP(Zn,Ni)	0.3844	4×10^{-3}
StandardGP	0.5714	3×10^{-4}

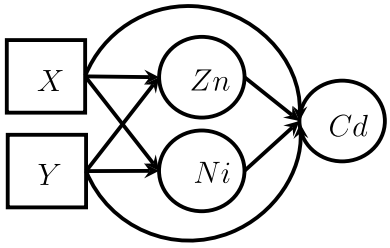

Fig. 5. Example 1 (cascading predictions) - StackedGP for predicting *Cd* based on estimated *Zn* and *Ni* at location of interest *X* and *Y*.

In the heterotopic case (Goovaerts (1997)), the primary target is undersampled relative to the secondary variables. This provides access to secondary information such as Ni and Zn at 100 locations being estimated. As a result a standard Gaussian process can be built to have Ni and Zn directly as inputs. Here we will denote it as StandardGP(Zn,Ni). This is also the case for comparing our results with other six multi-task regression models as reported by Wilson et al. (2012) and tabulated in Table 1.

Wilson et al. (2012) developed a Gaussian Process Regression Network (GPRN) to model the correlations between multiple outputs such as primary and secondary responses. The outputs are given by weighted linear combinations of latent functions where GP priors are defined over the weights, unlike similar studies for Semiparametric Latent Factor Model (SLFM) (Teh et al. (2005); Boyle and Frean (2005)) where the weights are considered fixed. As these models have no analytical solutions to learn its hyper-parameters, the authors use different approximation methods such as variational Bayes (VB) (Fox and Roberts (2012)). The SLFM has been motivated from intrinsic coregionalization model (ICM) (Goovaerts (1997)) in geostatistics. However unlike ICM, the SLFM includes Gaussian process hyper-parameters such as length-scales during the learning process. In additions, Convolution GP Model for Multiple Outputs (CMOGP) is another regression model where each output at each $x \in X$ is a mixture of latent Gaussian processes mixed across the whole input domain *X*. StackedGP is not designed to capture the correlations of response variables, however, StackedGP models can be constructed by stacking GPs for predicting intermediate secondary responses that govern the input space of GPs used to predict primary responses. This hierarchical framework outperforms other methods as shown in Table 1.

The first proposed StackedGP uses the first layer to model Zn and Ni based on locations and the second layer to model Cd based on the locations and the estimated output of the first layer, see Fig. 5. In the heterotopic case the StackedGP can use directly the available measurements of Ni and Zn instead of predictions by setting the uncertainty associated with these measurements to zero. In this case the StackedGP acts as the StandardGP(Zn,Ni).

Three other structures are proposed by using intermediate predictions of Co, Cr, and Co and Cr together.[3] In this case, we have a three layer StackedGP to model Cd, see Fig. 6. The first layer is the same as in the previous setup. The second layer models intermediate responses (Co, Cr, and Co and Cr). The third layer is used to model Cd based on the second layer predictions in additions to the input/output of the first layer, namely location and Zn and Ni. Fig. 6 also shows the predicted spatial fields for different metals. The predicted mean concentration of each metal is depicted as a heat map where x-axis and y-axis represent latitude and longitude respectively.

Table 1 shows the results of these stacked structures, StackedGP (Co), StackedGP(Cr) and StackedGP(Co,Cr). While measurements of Ni

[3] Interactive python for all stackedGP structures for Jura dataset can be found on https://bitbucket.org/uqlab/stackedgp/src/master/cadmium_prediction/demo.

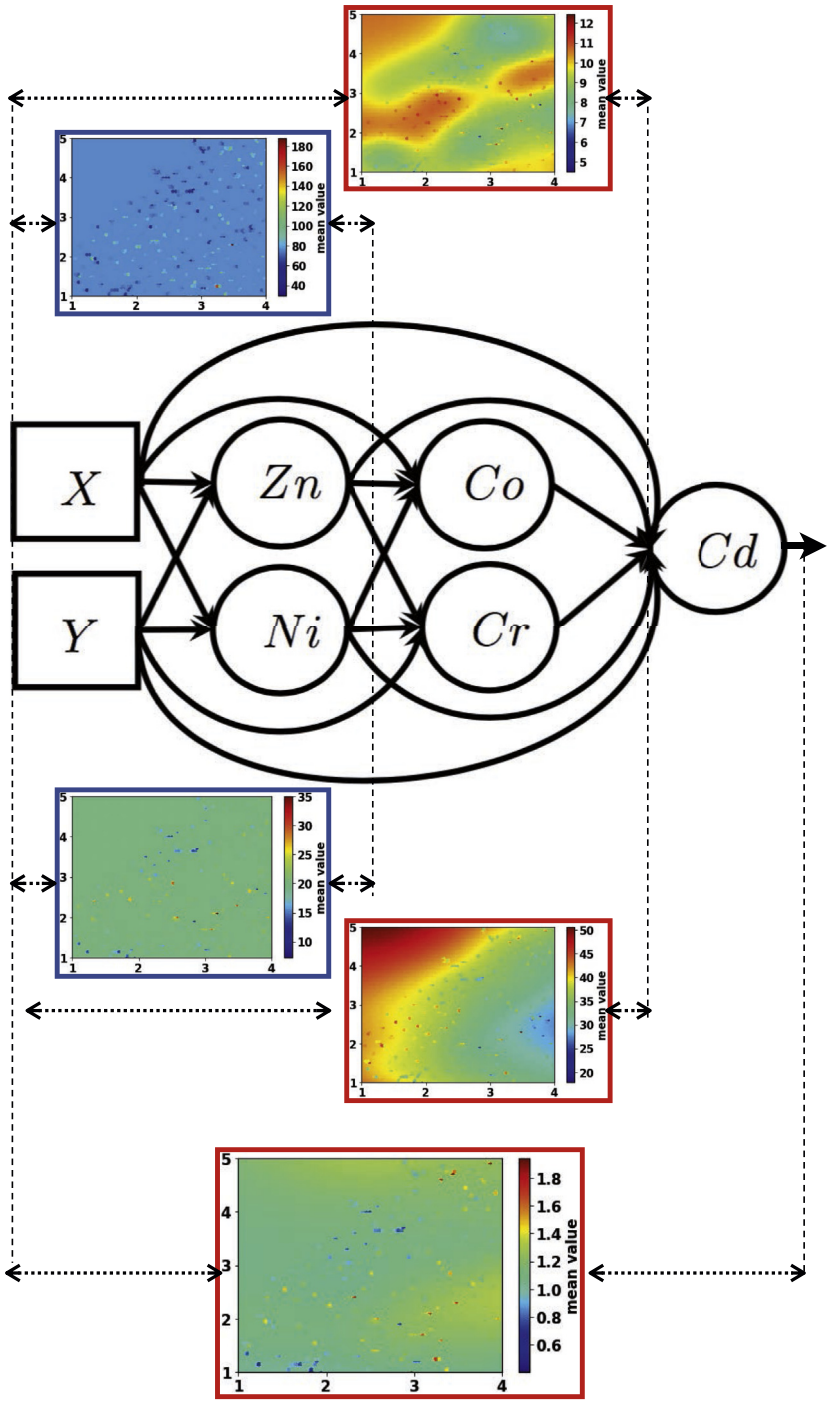

Fig. 6. Example 1 (cascading predictions) - StackedGP for predicting *Cd* based on estimated *Zn*, *Ni*, *Co*, and *Cr* at location of interest *X* and *Y*. The figure also shows the predicted spatial fields for Zn, Ni, Cr, Co, and Cd metals in Jura dataset.

and Zn are available in the testing scenarios, there are no measurements for Co and Cr during testing. Thus, Cd predictions of these three StackedGPs rely on predictions of Co and Cr using locations and Ni and Zn measurements at these locations.

The mean absolute error (MAE) between the true and estimated Cd is calculated at the 100 target locations. The experimental setup follows Alvarez and Lawrence (2011a) and Wilson et al. (2012) for which the simulation is restarted 10 times using different initializations of the parameters, namely the length scale for the RBF kernel in case of the StackedGP. The average and standard deviation of MAE over these 10 runs is reported in Table 1. Overall StackedGP gives better results as compared with the other models. Also, when Zn and Ni measurements are available as assumed by the other multi-output regression models (Wilson et al. (2012); Alvarez and Lawrence (2011a)), then a StandardGP(Ni,Zn) can provide a lower MAE than the other six multi-output regression models. However, StackedGP can provide a better performance over the Standard(Zn,Ni) by making use of intermediate predictions of secondary responses.

For all these experiments we found that the log transformation and normalization can lead to better results. For multi-responses in the middle layer, we used independent component analysis (ICA) to obtain independent projections of secondary responses. This is required as the

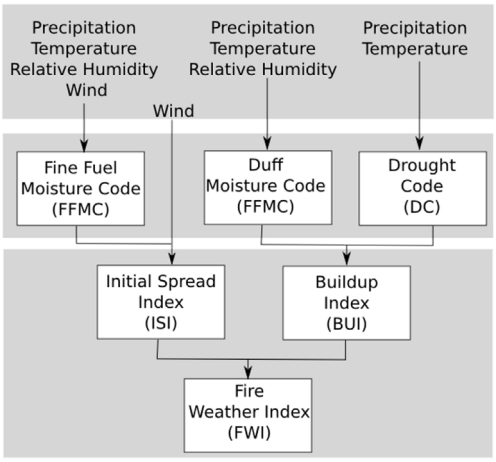

Fig. 7. Example 2 (forest fire) - Structure of the fire weather index (FWI) system module of the Canadian forest fire danger rating system (Taylor and Alexander (2006)).

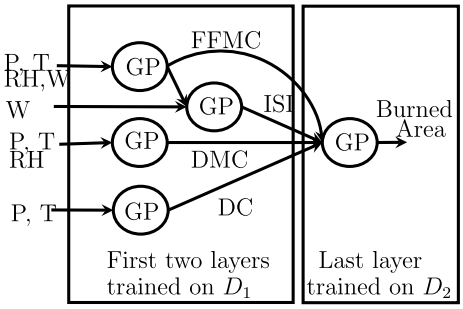

Fig. 8. Example 2 (forest fire) - StackedGP for predicting burned area based on estimated FWI indices. Letters P, T, RH, W stands for precipitation, temperature, relative humidity and wind respectively. Also, the first two layers are trained using dataset D_1 , while dataset D_2 is used to train the last layer.

Table 2
Example 2 (forest fire) - Predictive results using different models. The input for each model is *T* for meteorological features and *FWI* for fire indices. The results obtained with multiple regression (MR), decision trees (DT), random forests (RF), and neural networks (NN) have been reported by Cortez and Morais (2007).

Model	Input	MAE	RMSE
StackedGP	T	12.80	**46.0**
MR	FWI	13	64.5
DT	T	13.18	64.5
RF	T	12.98	64.4
NN	T	13.08	64.6
SVM	T	**12.71**	64.7

current derivation assumes that inputs to a GP node are independent.

The complexity of most of multi-task models (e.g. CMOGP, SLFM), is $O(N^3p^3)$ where N is size of the training dataset and p is the number of output responses (Alvarez and Lawrence (2011b); Wilson et al. (2012)). As GPRN depends on approximation methods such as variational Bayes, it needs several iterations to reach suitable hyper-parameters. A larger the number of iterations increases the time complexity of the model. Therefore, it may achieve lower complexity such as $O(pN^3)$ at the cost of obtaining a lower accuracy by decreasing the number of iterations. StackedGP scales linearly with the number of nodes in the structure because of the independent training of the nodes, which can be done in parallel. In the worst case StackedGP is $O(pN^3)$. Nonetheless, sparse approximation techniques can be used to further reduce this complexity in the case of large training datasets (Snelson (2007); Damianou et al. (2011)). Furthermore, StandardGP, Co-Kriging, and ICM have $O(N^3)$ complexity, but they achieve a lower accuracy as compared with the other mulit-task models.

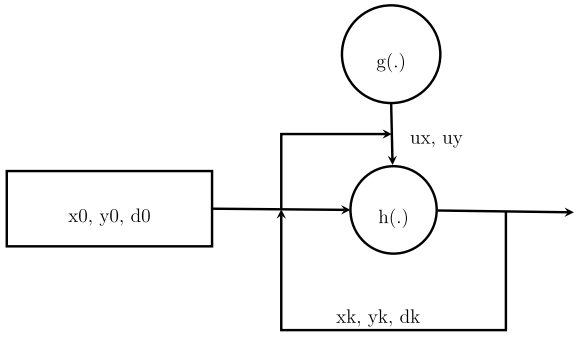

Fig. 9. Example 3 (uncertainty propagation) - StackedGP model for uncertainty propagation using emulated 2D puff advection driven by uncertain wind field.

6.2. Model composition - Forest Fire Dataset

The prediction of the burned area from forest fires has been discussed in different studies such as Cortez and Morais (2007) and Castelli et al. (2015). The burned area of forest fires has been predicted using meteorological conditions (e.g. temperature, wind) and/or several Canadian forest fire weather indices (Taylor and Alexander (2006)) for rating fire danger, namely fine fuel moisture code (FFMC), duff moisture code (DMC), drought code (DC), initial spread index (ISI), and buildup index (BUI), as shown in Fig. 7.

In this application we are interested in developing a StackedGP [4] by first modeling the fire indices using meteorological variables T from one dataset presented in Van Wagner et al. (1985) and then model the burned area based on fire indices using another dataset presented in Cortez and Morais (2007). The proposed StackedGP is depicted in Fig. 8. The GP nodes corresponding to the four fire indices (FFMC,DMC, DC, and ISI) are trained from data published in Van Wagner et al. (1985) according to the hierarchical structure shown in Fig. 7. While the second dataset (Cortez and Morais (2007)) contains meteorological conditions along with the fire indices and burned area, we assume that the meteorological conditions are missing in the training phase from this dataset and use only the fire indices and burned area data to train the GP node in the last layer of the StackedGP.

A 10-fold cross validation is applied to the dataset published by Cortez and Morais (2007) to train the burned area node and test the whole StackedGP model. Because of the skewed distribution of the burned area values and to ensure positive value for our predictions, instead of directly modeling the burned area using StackedGP, we have modeled the log of the burned area. As a result, the final mean and variance of the burned area $B[T]$ as a function of the meteorological conditions T is given by Eqs. (39) and (40) respectively. In additions, we have found that scaling the target variable to have zero mean and unit variance to be a beneficial preprocessing step.

$$E[B] = \left[e^{\sigma^2_{\ln B}} - 1\right] e^{2\mu_{\ln B} + \sigma^2_{\ln B}} \tag{39}$$

$$Var[B] = e^{\mu_{\ln B} + 0.5\sigma^2_{\ln B}} \tag{40}$$

Here, $\mu_{\ln B}$ and $\sigma_{\ln B}$ are the output of the probabilistic analytical StackedGP (Eqs. (22) and (24)) in the case of the RBF kernel, see Section 6.1.1.

The result of modeling the burned area using the StackedGP is shown in Table 2. The StackedGP model is compared with the results of

[4] Interactive python for stackedGP structure for Forest Fire Dataset can be found at https://bitbucket.org/uqlab/stackedgp/src/master/forestfire.

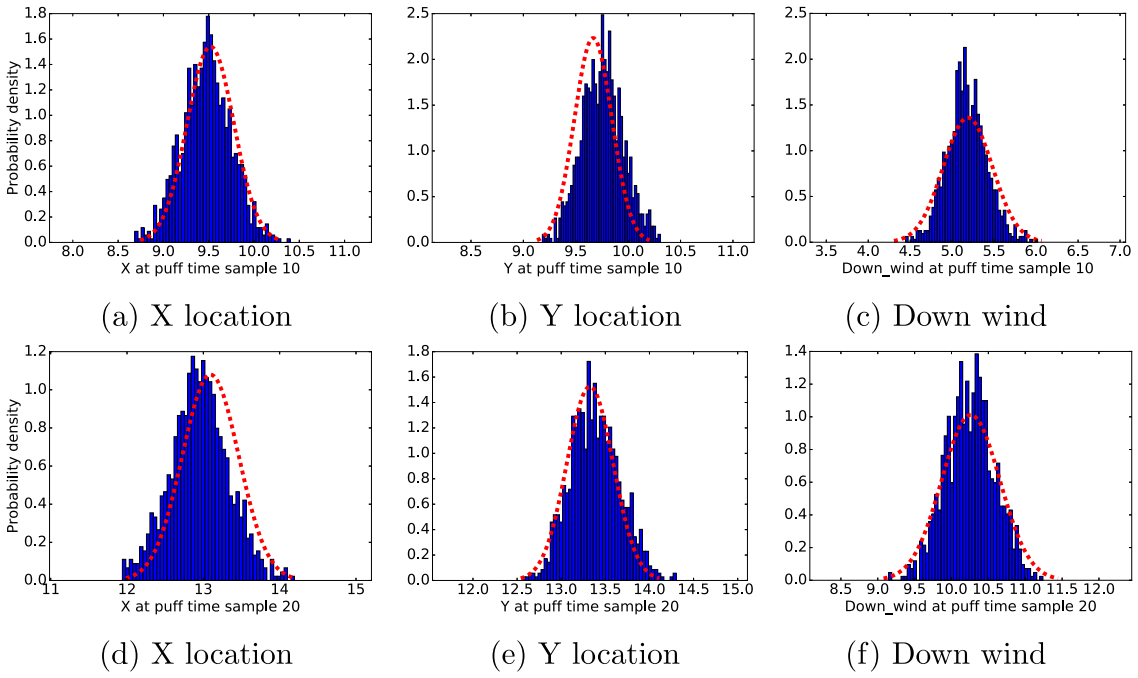

Fig. 10. Example 3 (uncertainty propagation) - Histogram of 1000 MC samples (blue) and the predicted StackedGP Gaussian distribution (red) at time step 10 for figures [a, b, and c] and time step 20 for figures [d, e, and f]. (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

Table 3
Example 3 (uncertainty propagation) - Predicted mean and standard deviation of puff states using proposed approximate and Monte Carlo propagation of uncertainty through StackedGP.

k	Approximate Propagation						Monte Carlo					
	x_k		y_k		d_k		x_k		y_k		d_k	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
5	7.76	0.19	7.83	0.13	2.61	0.23	7.74	0.19	7.89	0.14	2.59	0.16
10	9.52	0.26	9.67	0.18	5.19	0.29	9.47	0.27	9.76	0.2	5.17	0.24
15	11.3	0.32	11.5	0.22	7.74	0.34	11.22	0.33	11.59	0.24	7.72	0.29
20	13.09	0.37	13.33	0.26	10.26	0.39	12.95	0.38	13.38	0.28	10.25	0.33

5 other regression models reported by Cortez and Morais (2007). Because these regression models have been tested using different input spaces, Table 2 tabulates the best results achieved by each model as described in Cortez and Morais (2007). Even though the StackedGP predicts the burned area based on estimated indices from the first dataset and not the actual values as presented in the second dataset, it is still able to give comparable results with the other models that make use of meteorological conditions and/or fire indices available in the second dataset. This experiment emphasizes that the StackedGP is able to combine knowledge from multiple datasets with noticeable performance.

6.3. Uncertainty propagation - atmospheric transport

Gaussian processes with uncertain inputs have been previously used in multi-step time series predictions (Girard et al. (2003); Candela et al. (2003)). Modeling multi-step ahead predictions can be achieved by feeding back the predicted mean and variance at each time and propagating the uncertainty to the next time step. This idea has been used in different time-series applications such as electricity forecasting (Lourenço and Santos (2010)) and water demand forecasting (Wang et al. (2014)). Here we expand this concept by further driving the dynamical system using another GP for propagating uncertainty in an atmospheric transport problem. We consider a simple advection of a 2D Gaussian-shaped puff (Nielsen et al. (1999); Terejanu et al. (2007)). The states of the puff evolve using the following equations.

$$x_{k+1} = x_k + u_x(x_k)\Delta t \tag{41}$$

$$y_{k+1} = y_k + u_y(y_k)\Delta t \tag{42}$$

$$d_{k+1} = d_k + \sqrt{u_x^2(x_k) + u_y^2(y_k)}\,\Delta t \tag{43}$$

Here, (x_k, y_k) is the position of the center of the puff, and the downwind distance from the source d_k is used to compute the puff radius, $\sigma_k = pd_k^q$ in models such as RIMPUFF (Nielsen et al. (1999)) based on Karlsruhe-Jülich diffusion coefficients (Reddy et al. (2006)), (p, q).

The goal here is to build a GP emulator for the above dynamical system, knowing that the release location is fixed at ($x_0 = 0$ km, $y_0 = 0$ km) and the wind velocity is uncertain with normally distributed wind components (u_x, u_y).

$$u_x, u_y \sim \mathcal{N}(4\text{m/s}, 1\text{m/s}) \tag{44}$$

The GP emulator $h(\cdot)$ is constructed using 15 training trajectories that start at the same release location, but correspond to different wind fields that randomly sampled from the distribution in Eq. (44). The total simulation time is 30min with a time step $\Delta t = 90$ sec. As a result, k has range of 20 steps during the simulation time.

$$[x_{k+1}, y_{k+1}, d_{k+1}] = h(x_k, y_k, u_x(x_k), u_y(x_k)) \quad (45)$$

Another GP model is constructed to determine the wind field based on 16 wind sensors positioned 4 km apart in both directions. The wind sensor readings are just independent and identically distributed samples from Eq. (44).

$$[u_x(x), u_y(y)] = g(x, y) \quad (46)$$

Note, that in this particular case the wind velocity at different locations is correlated. Both emulators use RBF kernels, and they are stacked to build a recurrent StackedGP as shown in Fig. 9.

To assess the effect of the two assumptions in constructing the StackedGP[5] (independent inputs for each layer and Gaussian distribution approximation for the output of each layer), we compared the approximate mean and variance of the puff states from StackedGP using the proposed algorithm with those resulted from a Monte Carlo propagation of uncertainty through the StackedGP using 1000 samples.

Fig. 10 shows the approximate predicted Gaussian distribution of the states along with the histogram of the Monte Carlo samples propagated through the StackedGP. Table 3 lists the predicted mean and standard deviation of the puff states at different time steps.

Note that even though the state equations for the location of the puff are linear, because they are emulated using a GP, which at its turn is driven by a GP model for the wind field, the distribution of the StackedGP output may depart from the Gaussian distribution. The assumption of approximating the output with a Gaussian distribution may result in biasing the mean location. The statistical significant difference between the StackedGP approximate mean propagation and its Monte Carlo estimate confirms the impact of this approximation as shown in Table 3.

Furthermore, the assumption of ignoring the correlation structure between the outputs of StackedGP may result in an artificial inflation of the uncertainty. In our simple example, this is clearly manifested in larger standard deviations for the downwind using approximate propagation as compared with the Monte Carlo estimate. This impact on uncertainty propagation might be exacerbated when more nonlinear models are used, which limits the horizon of uncertainty propagation. Obviously, the gain in computational speed combined with field measurements in the context of data assimilation may position these stacked model as real contenders for real time applications. We plan to investigate in the future the application of StackedGP to data assimilation.

7. Conclusions

A stacked model of independently trained Gaussian processes, called StackedGP, is proposed as a modeling framework in the context of model composition. This is especially of interest in environmental modeling where, e.g., model composition is used to generate large scale predictions by combining geographical interpolation models with phenomenological models developed in the lab. An approximate approach is developed to obtain estimates of the quantities of interest with quantified uncertainties. This leverages the analytical moments of a Gaussian process with uncertain inputs when squared exponential and polynomial kernels are used. The StackedGP can be extended to any number of nodes and layers and has no restriction in selecting a suitable kernel for the input nodes.

The numerical results show the utility of using StackedGP to learn from multiple datasets and propagate the uncertainty to quantities of interest. While it is not specifically designed to model correlations between secondary and primary responses, StackedGP can be used to enhance the prediction of primary responses by creating an intermediate layer of predictions of secondary responses. This comes with a lower computational complexity as compared with multi-output methods - and can make use of off-the-shelves Gaussian processes. While in the current paper we assume that outputs of intermediate layers are independent and resolve this using independent component analysis preprocessing, we plan to extend our derivation to account for these correlations in the next study. This will allow multi-output models to act as nodes in the proposed StackedGP. Along with the independence assumption, the other drawback of the proposed uncertainty propagation algorithm is the Gaussian assumption of the predictive distribution. While this is motivated using maximum-entropy principle in a multi-step prediction setting it overestimates the predicted uncertainty.

[5] Interactive python for stackedGP structure for Atmospheric Transport experiment can be found on https://bitbucket.org/uqlab/stackedgp/src/master/Uncertainty_Propagation_Atmospheric_Transport.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grand No. 1504728 and 1632824. Dr. Terejanu has been supported by the National Institute of Food and Agriculture (NIFA)/USDA under Grand No. 2017-67017-26167.

Appendix A. Supplementary data

Supplementary data related to this article can be found at https://doi.org/10.1016/j.envsoft.2018.08.022.

References

Alvarez, M.A., Lawrence, N.D., 2011a. Computationally efficient convolved multiple output Gaussian processes. J. Mach. Learn. Res. 12 (May), 1459–1500.
Alvarez, M.A., Lawrence, N.D., 2011b. Computationally efficient convolved multiple output Gaussian processes. J. Mach. Learn. Res. 12 (May), 1459–1500.
Anitescu, M., Chen, J., Stein, M.L., 2017. An inversion-free estimating equations approach for Gaussian process models. J. Comput. Graph Stat. 26 (1), 98–107.
Atteia, O., Dubois, J.-P., Webster, R., 1994. Geostatistical analysis of soil contamination in the swiss jura. Environ. Pollut. 86 (3), 315–327.
Bayarri, M.J., Berger, J.O., Cafeo, J., Garcia-Donato, G., Liu, F., Palomo, J., Parthasarathy, R.J., Paulo, R., Sacks, J., Walsh, D., 2007. Computer model validation with functional output. Ann. Stat. 35 (5), 1874–1906.
Bhattacharya, S., 2007. A simulation approach to Bayesian emulation of complex dynamic computer models. Bayesian Anal 2 (4), 783–815.
Boyle, P., Frean, M., 2004. Dependent Gaussian processes. In: Advances in Neural Information Processing Systems, pp. 217–224.
Boyle, P., Frean, M., 2005. Dependent Gaussian processes. In: Advances in Neural Information Processing Systems, pp. 217–224.
Candela, J.Q., Girard, A., Larsen, J., Rasmussen, C.E., 2003. Propagation of uncertainty in Bayesian kernel models-application to multiple-step ahead forecasting. 2003 IEEE International Conference on. Vol. 2 In: Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). IEEE II–701.
Castelletti, A., Galelli, S., Ratto, M., Soncini-Sessa, R., Young, P.C., 2012. A general framework for Dynamic Emulation Modelling in environmental problems. Environ. Model. Software 34, 5–18.
Castelli, M., Vanneschi, L., Popovič, A., 2015. Predicting burned areas of forest fires: an artificial intelligence approach. Fire Ecology 11 (1), 106–118.
Cheng, H., Sandu, A., 2009. Uncertainty quantification and apportionment in air quality models using the polynomial chaos method. Environ. Model. Software 24 (8), 917–925.
Conti, S., Gosling, J.P., Oakley, J.E., O'Hagan, A., 2009. Gaussian process emulation of dynamic computer codes. Biometrika 96 (3), 663–676.
Conti, S., O'Hagan, A., 2010. Bayesian emulation of complex multi-output and dynamic computer models. J. Stat. Plann. Inference 140 (3), 640–651.
Cortez, P., Morais, A.d.J.R., 2007. A data mining approach to predict forest fires using meteorological data. In: 13th Portuguese Conference on Artificial Intelligence, pp. 512–523 (EPIA 2007).
Cressie, N., 1992. Statistics for spatial data. Terra. Nova 4 (5), 613–617.
Damianou, A., Titsias, M.K., Lawrence, N.D., 2011. Variational Gaussian process dynamical systems. In: Advances in Neural Information Processing Systems, pp. 2510–2518.
Damianou, A.C., Lawrence, N.D., 2013. Deep Gaussian processes. In: AISTATS, pp. 207–215.
Fox, C.W., Roberts, S.J., 2012. A tutorial on variational bayesian inference. Artif. Intell. Rev. 38 (2), 85–95.
Girard, A., Rasmussen, C.E., Candela, J.Q., Murray-Smith, R., 2003. Gaussian process priors with uncertain inputs-application to multiple-step ahead time series forecasting. Adv. Neural Inf. Process. Syst. 545–552.
Girard, A., Rasmussen, C.E., Murray-Smith, R., 2002. Gaussian Process Priors with Uncertain Inputs: Multiple-step Ahead Prediction. University of Glasgow.

Goovaerts, P., 1997. Geostatistics for Natural Resources Evaluation. Oxford University Press on Demand.
GPy, since, 2012. GPy: a Gaussian process framework in python. http://github.com/SheffieldML/GPy.
Grützner, R., 1995. Environmental Modeling and Simulation - Applications and Future Requirements. Chapman & Hall, pp. 113–122.
He, H.S., Larsen, D., Mladenoff, D.J., 2002. Exploring component-based approaches in forest landscape modeling. Environ. Model. Software 17 (6), 519–529.
Higdon, D., 1998. A process-convolution approach to modelling temperatures in the North Atlantic Ocean. Environ. Ecol. Stat. 5 (2), 173–190.
Higdon, D., Gattiker, J., Williams, B., Rightley, M., 2008. Computer model calibration using high-dimensional output. J. Am. Stat. Assoc. 103 (482), 570–583.
Hu, J., Wang, J., 2015. Short-term wind speed prediction using empirical wavelet transform and Gaussian process regression. Inside Energy 93 (Part 2), 1456–1466.
Jørgensen, S., 2010. Environmental models and simulations. In: Sydow, A. (Ed.), Environmental Systems. vol. II. Eolss Publishers, pp. 160–192.
Katurji, M., Nikolic, J., Zhong, S., Pratt, S., Yu, L., Heilman, W.E., 2015. Application of a statistical emulator to fire emission modeling. Environ. Model. Software 73, 254–259.
Kennedy, M.C., O'Hagan, A., Higgins, N., 2002. Bayesian analysis of computer code outputs. In: Anderson, C.W., Barnett, V., Chatwin, P.C., El-Shaarawi, A.H. (Eds.), Quantitative Methods for Current Environmental Issues. Springer London, London, pp. 227–243.
Konda, U., Singh, T., Singla, P., Scott, P., 2010. Uncertainty propagation in puff-based dispersion models using polynomial chaos. Environ. Model. Software 25 (12), 1608–1618.
Letcher, R.A., Jakeman, A.J., 2009. Types of environmental models. In: Marquette, C.M. (Ed.), Water and Development, Vol II. Eolss Publishers, pp. 131–154.
Li, H., Chowdhury, A., Terejanu, G., Chanda, A., Banerjee, S., November 2015. A stacked Gaussian process for predicting geographical incidence of aflatoxin with quantified uncertainties. In: International Conference on Advances in Geographic Information Systems ACM SIGSPATIAL. Seattle, Washington.
Lourenço, J., Santos, P., 2010. Short term load forecasting using Gaussian process models. In: Proceedings of Instituto de Engenharia de Sistemas e Computadores de Coimbra.
Machac, D., Reichert, P., Rieckermann, J., Albert, C., 2016. Fast mechanism-based emulator of a slow urban hydrodynamic drainage simulator. Environ. Model. Software 78, 54–67.
Meier, F., Hennig, P., Schaal, S., 2014. Incremental local Gaussian regression. In: Advances in Neural Information Processing Systems, pp. 972–980.
Millington, J.D., Wainwright, J., Perry, G.L., Romero-Calcerrada, R., Malamud, B.D., 2009. Modelling mediterranean landscape succession-disturbance dynamics: a landscape fire-succession model. Environ. Model. Software 24 (10), 1196–1208.
Moonen, P., Allegrini, J., 2015. Employing statistical model emulation as a surrogate for CFD. Environ. Model. Software 72, 77–91.
Neumann, M., Kersting, K., Xu, Z., Schulz, D., 2009. Stacked Gaussian process learning. In: 2009 Ninth IEEE International Conference on Data Mining. IEEE, pp. 387–396.
Nielsen, S., Deme, S., Mikkelsen, T., 1999. Description of the Atmospheric Dispersion Module RIMPUFF. Tech. Rep. RODOS(WG2)-TN(98)-02. Riso National Laboratory, Denmark P.O.Box 49, DK-4000 Roskilde.
O'Hagan, A., 2006. Bayesian analysis of computer code outputs: a tutorial. Reliab. Eng. Syst. Saf. 91 (10), 1290–1300.
Rasmussen, C.E., 1997. Evaluation of Gaussian Processes and Other Methods for Non-linear Regression. Ph.D. thesis, Toronto, Ont., Canada, Canada, aAINQ28300. .
Rasmussen, C.E., Williams, C.K.I., 2005. Gaussian Processes for Machine Learning. MIT press.
Reddy, K.V.U., Singh, T., Cheng, Y., Scott, P.D., July 2006. Data assimilation for dispersion models. In: 2006 9th International Conference on Information Fusion, pp. 1–8.
Reggente, M., Peters, J., Theunis, J., Van Poppel, M., Rademaker, M., Kumar, P., De Baets, B., 2014. Prediction of ultrafine particle number concentrations in urban environments by means of Gaussian process regression based on measurements of oxides of nitrogen. Environ. Model. Software 61, 135–150.
Shore, J., Johnson, R., 1980. Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy. IEEE Transactions on Information Theory 26 (1), 26–37.
Singh, V., Carnevale, C., Finzi, G., Pisoni, E., Volta, M., 2011. A cokriging based approach to reconstruct air pollution maps, processing measurement station concentrations and deterministic model simulations. Environ. Model. Software 26 (6), 778–786. http://www.sciencedirect.com/science/article/pii/S136481521000318X.
Snelson, E., Ghahramani, Z., 2006. Sparse Gaussian processes using pseudo-inputs. Adv. Neural Inf. Process. Syst. 18, 1257.
Snelson, E.L., 2007. Flexible and Efficient Gaussian Process Models for Machine Learning (Ph.D. thesis, Citeseer). .
Sun, A.Y., Wang, D., Xu, X., 2014. Monthly streamflow forecasting using Gaussian Process Regression. J. Hydrol. 511, 72–81.
Sykes, R., Parker, S., Henn, D., Chowdhury, B., 2006. SCIPUFF Version 2.2 Technical Documentation. Tech. Rep. 729, L-3. Titan Corporation.
Taylor, S.W., Alexander, M.E., 2006. Science, technology, and human factors in fire danger rating: the canadian experience. Int. J. Wildland Fire 15 (1), 121–135.
Teh, Y.W., Seeger, M., Jordan, M.I., 6–8 January 2005. Semiparametric latent factor models. In: Cowell, R.G., Ghahramani, Z. (Eds.), Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics,. Society for Artificial Intelligence and Statistics, Barbados, pp. 333–340.
Terejanu, G., Cheng, Y., Singh, T., Scott, P.D., November 2008. Comparison of SCIPUFF plume prediction with particle filter assimilated prediction for Dipole 26 data. In: Chemical and Biological Defense Physical Science and Technology Conference, New Orleans.
Terejanu, G., Singh, T., Scott, P.D., July 2007. Unscented Kalman filter/smoother for a CBRN puff-based dispersion model. In: 11th International Conference on Information Fusion, Quebec City, Canada.
Tokmakian, R., Challenor, P., Andrianakis, Y., 2012. On the use of emulators with extreme and highly nonlinear geophysical simulators. J. Atmos. Ocean. Technol. 29 (11), 1704–1715.
Trebicki, J., Sobczyk, K., 1996. Maximum entropy principle and non-stationary distributions of stochastic systems. Probabilist. Eng. Mech. 11 (3), 169–178.
Van Wagner, C., Pickett, T., et al., 1985. Equations and FORTRAN Program for the Canadian Forest Fire Weather Index System, vol. 33.
Wackernagel, H., 1996. Multivariate geostatistics: an introduction with applications. In: International Journal of Rock Mechanics and Mining Sciences and Geomechanics Abstracts. vol. 33 Springer 363A–363A.
Wang, Y., Ocampo-Martínez, C., Puig, V., Quevedo, J., 2014. Gaussian-process-based demand forecasting for predictive control of drinking water networks. In: International Conference on Critical Information Infrastructures Security. Springer, pp. 69–80.
Webster, R., Atteia, O., Dubois, J.-P., 1994. Coregionalization of trace metals in the soil in the swiss jura. Eur. J. Soil Sci. 45 (2), 205–218.
Whish, J.P., Herrmann, N.I., White, N.A., Moore, A.D., Kriticos, D.J., 2015. Integrating pest population models with biophysical crop models to better represent the farming system. Environ. Model. Software 72, 418–425.
Williams, C.K., 1998. Prediction with Gaussian processes: from linear regression to linear prediction and beyond. In: Learning in Graphical Models. Springer, pp. 599–621.
Williams, C.K.I., Rasmussen, C.E., 1996. Gaussian processes for regression. In: Advances in Neural Information Processing Systems. vol. 8. MIT press, pp. 514–520.
Wilson, A.G., Knowles, D.A., Ghahramani, Z., June, 2012. Gaussian process regression networks. In: Langford, J., Pineau, J. (Eds.), Proceedings of the 29th International Conference on Machine Learning (ICML). Omnipress, Edinburgh.