Contents lists available at ScienceDirect

Neurocomputing

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

A conditional one-output likelihood formulation for multitask Gaussian processes

Óscar García-Hinde [a,*], Manel Martínez-Ramón [b], Vanessa Gómez-Verdejo [a,*]

[a] Department of Signal Processing and Communications, Universidad Carlos III de Madrid Leganés, 28911, Spain
[b] Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM, 8711, USA

ARTICLE INFO

ABSTRACT

Multitask Gaussian processes (MTGP) are the Gaussian process (GP) framework's solution for multioutput regression problems in which the T elements of the regressors cannot be considered conditionally independent given the observations. Standard MTGP models assume that there exist both a multitask covariance matrix as a function of an intertask matrix, and a noise covariance matrix. These matrices need to be approximated by a low rank simplification of order P in order to reduce the number of parameters to be learnt from T^2 to TP. Here we introduce a novel approach that simplifies the multitask learning process by reducing it to a set of conditioned univariate GPs without the need for any low rank approximations, therefore completely eliminating the need to select an adequate value for hyperparameter P. At the same time, by extending this approach with both a hierarchical and an approximate model, the proposed extensions are capable of recovering the multitask covariance and noise matrices after learning only $2T$ parameters, avoiding the validation of any model hyperparameter and reducing the overall complexity of the model as well as the risk of overfitting. Experimental results over synthetic and real problems confirm the advantages of this inference approach in its ability to accurately recover the original noise and signal matrices, as well as the achieved performance improvement in comparison to other state of art MTGP approaches. We have also integrated the model with standard GP toolboxes, showing that it is computationally competitive with state of the art options.

1. Introduction

Gaussian processes (GP) [1] can be considered state of the art in nonlinear regression, among other reasons, because they provide a natural way of implementing a predictive posterior distribution. This distribution has a clear advantage over non-Bayesian models since it is a surrogate statistical model that provides not only the predictive target means, but also a relevant measure of uncertainty in the form of a predictive covariance function. While standard GPs were initially designed to handle single scalar outputs, it is becoming more and more common to have to face multi-task (MT) or multidimensional output problems in which each individual output cannot be considered conditionally independent from the rest given the predictors. This can be found in many application examples, such as medical applications [2,3], air quality forecasting [4], product design [5] and, particularly with multioutput GP models, in multioutput time-series analysis [6], manufacturing applica-

tions [7], detection of damages in structures [8], forecast of multi-scale solar radiation with application in photovoltaics [9], or COVID-19 outbreak detection [10]. See also the extensive survey on multi-output learning in [11]. In these cases, the use of adequate approaches which are able to model the relationships among the tasks can offer significant advantages [12].

The general MTGP formulation proposed in [13] can be considered the reference model and we will therefore refer to it in this paper as the standard MTGP (Std-MTGP). Indeed, this model is the one chosen in all the above mentioned MTGP applications. This model assumes that the MT covariance matrix is expressed as the Kroneker product of an inter-task matrix $\mathbf{C}$ and the input kernel matrix. Specifying a full rank $\mathbf{C}$ requires a computational cost of $\mathcal{O}(N^3T^3)$ and the inference of $T(T-1)/2$ parameters, which becomes computationally unwieldy when T is large. In order to circumvent these problems, the authors of the Std-MTGP use a low-rank approximation of order P of $\mathbf{C}$, $\mathbf{C} \approx \mathbf{U}\lambda\mathbf{U}^\top + \sigma^2\mathbf{I}$, so that the number of parameters to be learnt is reduced from $\mathcal{O}(T^2)$ to $\mathcal{O}(TP)$ and the computational burden of the method is reduced from $\mathcal{O}(N^3T^3)$ to $\mathcal{O}(N^3T^2P)$. This model is reformulated as a linear

* Corresponding author.
E-mail addresses: oghinde@tsc.uc3m.es (Ó. García-Hinde), vanessa@tsc.uc3m.es (V. Gómez-Verdejo).

model of coregionalization (LMC) in [14,15], where the model outputs are expressed as a linear combination of P latent functions and, therefore, the multitask kernel function can be also expressed as a linear combination of several covariance functions. In both cases, parameter P must be cross validated in order to obtain matrices that are representative of the process to be modelled, but authors usually choose a low value for parameter P in order to keep the model's complexity low. In the particular case where $P = 1$, this model simplifies into the well-known intrinsic coregionalization model (ICM) [16], which results in significant computational savings. To obtain further computational savings in the general MTGP formulation, [17] proposes an efficient inversion of the MT covariance matrix by combining properties of the singular value decomposition (SVD) and the Kronecker product, reducing the computational cost from $\mathcal{O}(N^3 T^2 P)$ to $\mathcal{O}(N^3 + T^3)$.

All the approaches cited so far consider a noise model that is independent and identically distributed across tasks, i.e., their MT noise covariance matrices are of the form $\sigma^2 \mathbf{I}$. The approach in [18] offers a more general solution by introducing a noise covariance matrix which models inter-task noise dependencies. This results in a more realistic model with improved performance compared to the aforementioned alternatives. However, all of these methods have an important drawback in the number of parameters to be inferred. To mitigate this, the models presented in [13,17,18] reduce the effective number of parameters of the inter-task covariance matrix by approximating it with a sum of P rank one matrices that are further regularized in the GP model by the noise covariance term.

Additionally, several convolutional models [19–22] have emerged, establishing a more sophisticated formulation that is able to model blurred relationships between tasks by the generalization of the MT kernel matrix through a convolution. However, adequate usage requires careful selection of the convolutional kernel in order to make the integral tractable, and the number of parameters must be limited to balance the model's flexibility against its complexity to avoid overfitting issues. Furthermore, this complex design limits the model's interpretability since the inter-task covariance matrices are not explicitly estimated. Efficient versions of these models [22] introduce sparse GP formulations able to select M inducing points to reduce the computational cost down to $\mathcal{O}(M^3 T)$.

The main challenge for all these approaches (sse also [23–25]), lies in the fact that they have to fit a large number of parameters to model all the task relationships and, despite the fact that we can find many ad hoc implementations that use very accurate optimizers [26,27], all these approaches are prone to fail in local minima, therefore resulting in suboptimal performance.

This work has been inspired by the MTGP probabilistic model featured in [18], where the intertask and the noise covariances are explicitly modelled. Our proposal however presents a number of innovations and advantages that are summarized below.

First, we rely on a new formulation based on a decomposition of the likelihood function into a set of conditional one-output GPs, combined with a hierarchical extension of the conditioned GPs. To our knowledge, this innovation has not been introduced before, and it leads to the following advantages. In the first place, as opposed to previous approaches, the one presented here does not need to use a low rank approximation of the inter-task covariance matrix. This avoids the need to select the corresponding hyperparameter (i.e: approximation rank parameter P in [18] and others).

This also means that the present methodology uses a full rank expression of the intertask covariance matrix, with $T(T-1)/2$

parameters. Despite this fact, the number of parameters to be learnt is reduced to $2T$, which is sufficient to recover the full noise and intertask covariance matrices, while previous approaches require PT parameters to recover an intertask covariance of rank P.

The complexity of our model is also reduced, which leads to a reduction of the risk of overfitting and, hence, to an improvement in overall predictive performance.

Secondly, this learning approach can be easily adapted to leverage efficient GP libraries, such as Pytorch [26] or MOGPTK [27], allowing our method to be run on graphical processing units (GPUs) with a computational burden of $\mathcal{O}(TN^2)$. In fact, this implementation is publicly available at https://github.com/OGHinde/Cool_MTGP. Additionally, in this case, the model also admits an embarrasingly parallel implementation over the tasks to get a computational cost per process of $\mathcal{O}(N^2)$.

2. Introduction to the Multitask Gaussian Processes

Given the set of observations $\mathbf{x}_i \in \mathbb{R}^D, i = 1, \cdots, N$ and a transformation $\phi(\cdot)$ into a reproducing kernel Hilbert space $\mathscr{H}$ [28], the general expression for the MT regression problem of estimating T regressors or tasks, $\mathbf{y}_{1:T,i} = [y_{1,i} \cdots y_{T,i}]^\top, y_{t,i} \in \mathbb{R}$ from $\phi(\mathbf{x}_i) = \phi_i$, results in the model

$$\mathbf{y}_{1:T,i} = \mathbf{W}_{1:T}^\top \phi_i + \mathbf{e}_i, \tag{1}$$

$\mathbf{W}_{1:T} = [\mathbf{w}_1, \cdots, \mathbf{w}_T]$ being a mixing matrix where $\mathbf{w}_t \in \mathscr{H}$, and $\mathbf{e}_i \in \mathbb{R}^T$ representing the model noise. To complete this probabilistic model, the following inter-task noise distribution and weight prior are assumed

$$\begin{aligned} p(\mathbf{e}_i) &= \mathscr{N}(\mathbf{e}_i|\mathbf{0}, \mathbf{\Sigma}_{1:T,1:T}) \\ p(\text{vect}(\mathbf{W}_{1:T})) &= \mathscr{N}(\text{vect}(\mathbf{W}_{1:T})|\mathbf{0}, \mathbf{C}_{1:T,1:T} \otimes \mathbf{\Sigma}_p), \end{aligned} \tag{2}$$

where vect$(\cdot)$ is a column-wise vectorization operator and $\otimes$ is the Kronecker product. Matrix $\mathbf{\Sigma}_p$ models the covariances between the elements of $\mathbf{w}_t$ and it is common for all the tasks. This considers that correlation between the noise elements of different tasks is represented through the noise covariance $\mathbf{\Sigma}_{1:T,1:T}$, and relationships between tasks are modelled with the intertask covariance $\mathbf{C}_{1:T,1:T}$.

In order to do Bayesian inference we define the multitask likelihood

$$\begin{aligned} p(\text{vect}(\mathbf{Y}_{1:T})|\mathbf{\Phi}, \mathbf{W}_{1:T}) &= \prod_{i=1}^N \mathscr{N}(\mathbf{y}_{1:T,i}|\bar{\mathbf{y}}_{1:T,i}, \mathbf{\Sigma}_{1:T,1:T}) \\ &= \mathscr{N}(\text{vect}(\mathbf{Y}_{1:T})|\text{vect}(\bar{\mathbf{Y}}_{1:T}), \mathbf{\Sigma}_{1:T,1:T} \otimes \mathbf{I}) \end{aligned} \tag{3}$$

where $\mathbf{\Phi} = [\phi_1, \cdots, \phi_N], \mathbf{Y}_{1:T} = [\mathbf{y}_{1:T,1}, \cdots, \mathbf{y}_{1:T,N}]$, and the bar notation $\bar{\mathbf{u}}$ denotes the expectation of any random variable $\mathbf{u}$; in particular, $\bar{\mathbf{Y}}_{1:T} = \mathbf{W}_{1:T}^\top \mathbf{\Phi}$. Now, we can obtain the marginal likelihood (marginalizing the influence of $\mathbf{W}$):

$$\begin{aligned} p(\text{vect}(\mathbf{Y}_{1:T})|\mathbf{\Phi}, \mathbf{C}_{1:T,1:T}, \mathbf{\Sigma}_{1:T,1:T}) = \mathscr{N}(\text{vect}(\mathbf{Y}_{1:T})|\mathbf{0}, \mathbf{C}_{1:T,1:T} \\ \otimes \mathbf{\Phi}^\top \mathbf{\Sigma}_p \mathbf{\Phi} + \mathbf{\Sigma}_{1:T,1:T} \otimes \mathbf{I}) \end{aligned} \tag{4}$$

The estimation of matrices $\mathbf{\Sigma}_{1:T,1:T}$ and $\mathbf{C}_{1:T,1:T}$ is obtained through the maximization of (4). Finally, the predictive posterior, $\mathbf{f}_{1:T}^* = [f_1^* \cdots f_T^*]^\top$, for test sample ϕ^* is constructed as:

$$\begin{aligned} p(\mathbf{f}_{1:T}^*|\phi^*, \mathbf{y}_{1:T}, \mathbf{\Phi}) &= \mathscr{N}(\mathbf{f}_{1:T}^*|\bar{\mathbf{f}}_{1:T}^*, \mathbf{C}^*) \\ \bar{\mathbf{f}}_{1:T}^* &= \left(\mathbf{C}_{1:T,1:T} \otimes \mathbf{k}_*^\top\right)(\mathbf{C}_{1:T,1:T} \otimes \mathbf{K} + \mathbf{\Sigma}_{1:T,1:T} \otimes \mathbf{I})^{-1} \text{vect}(\mathbf{Y}_{1:T}) \\ \mathbf{C}^* &= \mathbf{C}_{1:T,1:T} \otimes k_{**} - \left(\mathbf{C}_{1:T,1:T} \otimes \mathbf{k}_*^\top\right)(\mathbf{C}_{1:T,1:T} \otimes \mathbf{K} + \mathbf{\Sigma}_{1:T,1:T} \otimes \mathbf{I})^{-1}(\mathbf{C}_{1:T,1:T} \otimes \mathbf{k}_*), \end{aligned} \tag{5}$$

where vector $\mathbf{k}_* = \mathbf{\Phi}^\top \mathbf{\Sigma}_p \phi^*$ contains the dot products between the test sample ϕ^* and the training dataset $\mathbf{\Phi}$, $\mathbf{K} = \mathbf{\Phi}^\top \mathbf{\Sigma}_p \mathbf{\Phi}$ is the matrix of dot products between data, and $k_{**} = \phi^{*\top} \mathbf{\Sigma}_p \phi^*$.

So far, this formulation extends the model of [13,17] and is formally identical to [18], which introduces the noise covariance. The underlying limitation of these works lies in the fact that the number of parameters to be optimized grows with T^2, and therefore a rank-P approximation of the form $\sum_p \lambda_p \mathbf{u}_p \mathbf{u}_p^\top + s^2 \mathbf{I}$ is used to model matrices $\mathbf{C}_{1:T,1:T}$ and $\mathbf{\Sigma}_{1:T,1:T}$. This reduces the number of parameters to $2TP$, but imposes the need of selecting a suitable value for parameter P.

3. Parameter learning through conditional one-output likelihood for MTGPs

Here, we introduce a novel methodology based on a conditional one-output likelihood MTGP (Cool-MTGP) where the previous MTGP formulation is decomposed into a set of T conditioned one-output GPs. This methodology reduces the number of parameters to be learnt to twice the number of tasks T without assuming any low rank approximation and the adjustment of additional hyperparameters.

3.1. MT likelihood as a product of conditional one-output distributions

Let us consider a model whose output corresponding to input ϕ_i in the t-th task is given by a linear combination of both the input data and the output of the previous tasks, $\mathbf{y}_{1:t-1,i}$, i.e.,

$$y_{t,i} = \phi_i^\top \mathbf{w}_{\mathbf{x},t} + \mathbf{y}_{1:t-1,i}^\top \mathbf{w}_{\mathbf{y},t} + \epsilon_{t,i} \tag{6}$$

where $\epsilon_{i,t} \sim \mathcal{N}(0, \sigma_t^2)$ is the noise process for task t and the weight of each factorized task is split into two components: weight $\mathbf{w}_{\mathbf{x},t} \in \mathcal{H}$ for the input data and weight $\mathbf{w}_{\mathbf{y},t} \in \mathbb{R}^{t-1}$ for the previous tasks. This model is closely related to the original MTGP described in Section 2 since, given the values of $\mathbf{w}_{\mathbf{x},t}$ and $\mathbf{w}_{\mathbf{y},t}$, one can recursively recover the original weights $\mathbf{w}_t$ as:

$$\mathbf{w}_t = \mathbf{w}_{\mathbf{x},t} + \mathbf{W}_{1:t-1} \mathbf{w}_{\mathbf{y},t}. \tag{7}$$

We can now apply the chain rule of probability to the original joint MT likelihood to factorize it into a set of conditional probabilities, each associated to a conditional one-output GP:

$$p(\text{vect}(\mathbf{Y}_{1:T}) | \mathbf{\Phi}, \mathbf{W}_{1:T}) = p(\mathbf{y}_T | \mathbf{Y}_{1:T-1}, \mathbf{\Phi}, \mathbf{w}_{\mathbf{x},T}, \mathbf{w}_{\mathbf{y},T}) \cdot$$
$$\cdot p(\mathbf{y}_{T-1} | \mathbf{Y}_{1:T-2}, \mathbf{\Phi}, \mathbf{w}_{\mathbf{x},T-1}, \mathbf{w}_{\mathbf{y},T-1}) \cdots p(\mathbf{y}_2 |, \mathbf{y}_1, \mathbf{\Phi}, \mathbf{w}_{\mathbf{x},2}, \mathbf{w}_{\mathbf{y},2}) \cdot \tag{8}$$
$$\cdot p(\mathbf{y}_1 |, \mathbf{\Phi}, \mathbf{w}_{\mathbf{x},1}),$$

where the likelihood for each of these conditioned GPs is given by:

$$p(\mathbf{y}_t | \mathbf{Y}_{1:t-1}, \mathbf{\Phi}, \mathbf{w}_{\mathbf{x},t}, \mathbf{w}_{\mathbf{y},t}) = \mathcal{N}(\mathbf{y}_t | \mathbf{\Phi}^\top \mathbf{w}_{\mathbf{x},t} + \mathbf{Y}_{1:t-1}^\top \mathbf{w}_{\mathbf{y},t}, \sigma_t^2 \mathbf{I}). \tag{9}$$

And the prior over the input weight components $\mathbf{w}_{\mathbf{x},t}$ is defined as:

$$p(\mathbf{w}_{\mathbf{x},t}) = \mathcal{N}(\mathbf{w}_{\mathbf{x},t} | \mathbf{0}, b_t \mathbf{\Sigma}_p), \tag{10}$$

where $\mathbf{\Sigma}_p$ assumes a common prior for all tasks scaled by a task-dependent factor b_t.

This approach assumes that $\mathbf{w}_{\mathbf{x},t}$ are latent variables modelled with a prior distribution, whereas previous task weights $\mathbf{w}_{\mathbf{y},t}$ are defined as model parameters (see the graphical model in Fig. 1 (a)); this way, for each task we generate a conditioned one-output likelihood GP (Cool-GP) with mean $\mathbf{w}_{\mathbf{y},t}^\top \mathbf{y}_{1:t-1}$ and covariance $b_t \mathbf{K}$:

$$\text{Cool} - \text{GP}_t \sim \mathcal{GP}\left(\mathbf{w}_{\mathbf{y},t}^\top \mathbf{y}_{1:t-1}, b_t \mathbf{K}\right) \tag{11}$$

This guarantees that the model remains Gaussian, allowing us to recover the original MTGP joint likelihood from the set of Cool-GP likelihoods, as defined in (9).

3.2. Parameter learning and model inference

In order to optimize the model in Fig. 1(a), we need to learn the input prior factors $\mathbf{b}_{1:T} = [b_1, \ldots, b_T]$, the noise covariances $\mathbf{\sigma}_{1:T}^2 = [\sigma_1^2, \ldots, \sigma_T^2]$, the common kernel parameters θ and, additionally, the weights of the previous tasks $\mathbf{w}_{\mathbf{y},1}, \ldots, \mathbf{w}_{\mathbf{y},T}$. To reduce the number of parameters to be learnt we propose two approaches to infer the values of $\mathbf{w}_{\mathbf{y},1}, \ldots, \mathbf{w}_{\mathbf{y},T}$ with a closed expression instead of having to learn their values with gradient descent approaches; in this way, the model complexity is reduced to $2T$ parameters plus the kernel parameters to be learnt.

3.2.1. A hierarchical approach for Cool-MTGP learning

This first approach proposes to solve the Cool-GP, depicted in Fig. 1(a), by means of a hierarchical methodology based on a two step learning process.
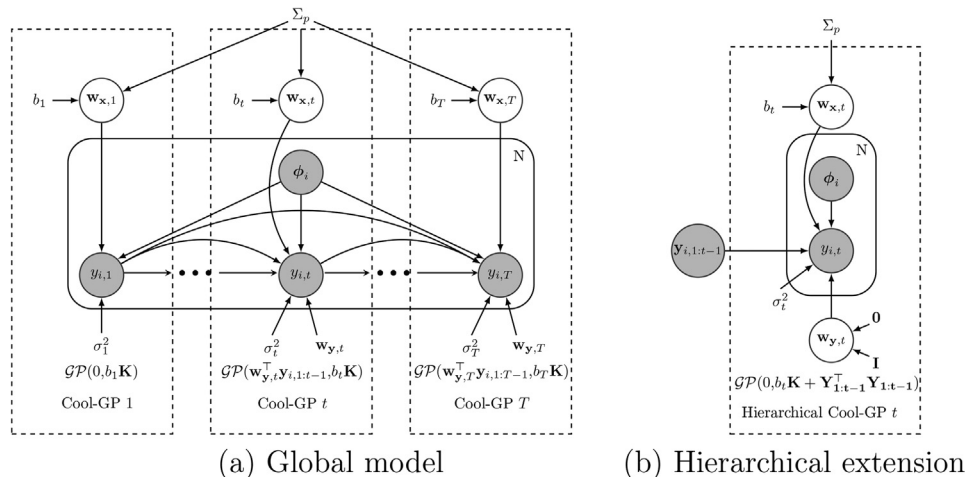

Fig. 1. Graphical model for the conditional one-output likelihood MTGP model.

(a) Global model   (b) Hierarchical extension

In the first step, we consider the model introduced in Fig. 1b) and define a prior distribution for each $\mathbf{w}_{\mathbf{y},t}$ with the form

$$p(\mathbf{w}_{\mathbf{y},t}) = \mathcal{N}(\mathbf{w}_{\mathbf{y},t}|\mathbf{0},\mathbf{I}). \tag{12}$$

that will be used, together with the independent prior assigned to $\mathbf{w}_{\mathbf{x},t}$,

$$p(\mathbf{w}_{\mathbf{x},t}) = \mathcal{N}(\mathbf{w}_{\mathbf{x},t}|\mathbf{0}, b_t\mathbf{\Sigma}_p), \tag{13}$$

to infer a joint posterior probability of both sets of parameters. This joint posterior distribution can be found as an extension of the inference applied to the standard GP [1] as

$$p([\mathbf{w}_{\mathbf{x},t}, \mathbf{w}_{\mathbf{y},t}]|\mathbf{\Phi}, \mathbf{Y}_{1:t}) = \mathcal{N}\left([\mathbf{w}_{\mathbf{x},t}, \mathbf{w}_{\mathbf{y},t}]|[\bar{\mathbf{w}}_{\mathbf{x},t}, \bar{\mathbf{w}}_{\mathbf{y},t}], \mathbf{A}_t^{-1}\right) \tag{14}$$

where the inverse $\mathbf{A}_t$ of the posterior covariance is

$$\mathbf{A}_t = \sigma_t^{-2}\begin{bmatrix}\mathbf{\Phi}\\\mathbf{Y}_{1:t-1}\end{bmatrix}\begin{bmatrix}\mathbf{\Phi}\\\mathbf{Y}_{1:t-1}\end{bmatrix}^\top + \begin{bmatrix}b_t\mathbf{\Sigma}_p & \mathbf{0}\\\mathbf{0} & \mathbf{I}\end{bmatrix}^{-1} \tag{15}$$

Its mean value provides the MAP estimation of the weight vector $\mathbf{w}_{\mathbf{y},t}$, given by

$$\bar{\mathbf{w}}_{\mathbf{y},t} = \sigma_t^{-2}\mathbf{A}_t^{-1}\mathbf{Y}_{1:t-1}\mathbf{y}_t. \tag{16}$$

Alternatively, the solution for the dual vector of $\mathbf{w}_{\mathbf{y},t}$ has the same formal expression as those of a standard GP [1], but using a composed kernel matrix $\mathbf{K}_{\mathbf{xy},t} = b_t\mathbf{K} + \mathbf{Y}_{1:t-1}^\top\mathbf{Y}_{1:t-1} + \sigma_t^2\mathbf{I}$:

$$\boldsymbol{\alpha}_{\mathbf{y},t} = \mathbf{K}_{\mathbf{xy},t}^{-1}\mathbf{y}_t, \tag{17}$$

Eq. (16) can be redefined as:

$$\bar{\mathbf{w}}_{\mathbf{y},t} = \mathbf{Y}_{1:t-1}\boldsymbol{\alpha}_{\mathbf{y},t}. \tag{18}$$

Note that this process is equivalent to training a GP with zero mean and covariance matrix $b_t\mathbf{K} + \mathbf{Y}_{1:t-1}^\top\mathbf{Y}_{1:t-1}$ (as depicted in the model in Fig. 1(b)); this way $\bar{\mathbf{w}}_{\mathbf{y},t}$ has the formal expression of a standard GP [1], where the input data kernel $\mathbf{K}$ matrix is rescaled by factor b_t plus a linear kernel matrix for the previous tasks outputs.

In the second step of the hierarchical model, we learn the remaining parameter values ($\mathbf{b}_{1:T}$, $\boldsymbol{\sigma}_{1:T}^2$ and $\boldsymbol{\theta}$) by going back to the original cool-GP of Eq. (11) and Fig. 1(a), where $\mathbf{w}_{\mathbf{y},t}$ is substituted by its MAP estimation.

Then, we can learn the model parameters ($\mathbf{b}_{1:T}$, $\boldsymbol{\sigma}_{1:T}^2$, $\boldsymbol{\theta}$) by maximizing the joint log-likelihood over all the tasks, given by

$$\log p(\mathbf{Y}_{1:T}|\mathbf{\Phi}, \bar{\mathbf{w}}_{\mathbf{y},1}, \dots, \bar{\mathbf{w}}_{\mathbf{y},T}, \mathbf{b}_{1:T}, \boldsymbol{\sigma}_{1:T}^2, \boldsymbol{\theta})$$
$$= \sum_{t=1}^{T} \log p(\mathbf{y}_t|\mathbf{\Phi}, \mathbf{Y}_{1:t-1}, \bar{\mathbf{w}}_{\mathbf{y},t}, b_t, \sigma_t^2, \boldsymbol{\theta}) \tag{19}$$

To obtain each one of the conditional one-output likelihoods at the right side of Eq. (19), we consider the prior $\mathcal{N}(\mathbf{w}_{\mathbf{x},t}|\mathbf{0}, b_t\mathbf{\Sigma}_p)$ for parameters $\mathbf{w}_{\mathbf{x},t}$ and marginalize the likelihood with respect to $\mathbf{w}_{\mathbf{x},t}$, resulting in a Gaussian distribution with the expression

$$p(\tilde{\mathbf{y}}_t|\mathbf{\Phi}, b_t, \sigma_t^2, \boldsymbol{\theta}) = \mathcal{N}(\tilde{\mathbf{y}}_t|\mathbf{0}, b_t\mathbf{\Phi}^\top\mathbf{\Sigma}_p\mathbf{\Phi} + \sigma_t^2\mathbf{I}) = \mathcal{N}(\tilde{\mathbf{y}}_t|\mathbf{0}, \mathbf{K}_{\mathbf{x},t}), \tag{20}$$

where we make use of the notation $\tilde{\mathbf{y}}_t = \mathbf{y}_t - \bar{\mathbf{w}}_{\mathbf{y},t}^\top\mathbf{Y}_{1:t-1}$ and $\mathbf{K}_{\mathbf{x},t} = b_t\mathbf{K} + \sigma_t^2\mathbf{I}$, which takes advantage of the fact that term $\bar{\mathbf{w}}_{\mathbf{y},t}^\top\mathbf{Y}_{1:t-1}$ is considered constant and can then be used as a mean subtracted from the random variable $\mathbf{y}_t$.

Replacing this expression into (19), we obtain the joint log-likelihood over all the tasks

$$\log p(\mathbf{Y}_{1:T}|\mathbf{\Phi}, \bar{\mathbf{w}}_{\mathbf{y},1}, \dots, \bar{\mathbf{w}}_{\mathbf{y},T}, \mathbf{b}_{1:T}, \boldsymbol{\sigma}_{1:T}^2, \boldsymbol{\theta})$$
$$= \sum_{t=1}^{T} \log p(\tilde{\mathbf{y}}_t|\mathbf{\Phi}, b_t, \sigma_t^2, \boldsymbol{\theta}) =$$
$$= \sum_{t=1}^{T} -\frac{1}{2}\tilde{\mathbf{y}}_t^\top\mathbf{K}_{\mathbf{x},t}^{-1}\tilde{\mathbf{y}}_t - \frac{1}{2}\log|\mathbf{K}_{\mathbf{x},t}| - \frac{N}{2}\log 2\pi \tag{21}$$

The criterion for the inference of the parameters consists of the maximization of this log likelihood through gradient ascent with respect to them. The derivatives with respect to each one of the common parameters $\boldsymbol{\theta}$ are

$$\frac{\partial}{\partial\theta_j}\log p(\mathbf{Y}_{1:T}|\mathbf{\Phi}, \bar{\mathbf{w}}_{\mathbf{y},1}, \dots, \bar{\mathbf{w}}_{\mathbf{y},T}, \mathbf{b}_{1:T}, \boldsymbol{\sigma}_{1:T}^2, \boldsymbol{\theta})$$
$$= \sum_{t=1}^{T}\left(-\frac{1}{2}\tilde{\mathbf{y}}_t^\top\mathbf{K}_{\mathbf{x},t}^{-1}\frac{\partial\mathbf{K}_{\mathbf{x},t}}{\partial\theta_j}\mathbf{K}_{\mathbf{x},t}^{-1}\tilde{\mathbf{y}}_t - \frac{1}{2}\text{trace}\left(\mathbf{K}_{\mathbf{x},t}^{-1}\frac{\partial\mathbf{K}_{\mathbf{x},t}}{\partial\theta_j}\right)\right) \tag{22}$$

For the task dependent parameters, $\mathbf{b}_{1:T}$ and $\boldsymbol{\sigma}_{1:T}^2$, these derivatives are

$$\frac{\partial}{\partial b_t}\log p(\mathbf{Y}_{1:T}|\mathbf{\Phi}, \bar{\mathbf{w}}_{\mathbf{y},1}, \dots, \bar{\mathbf{w}}_{\mathbf{y},T}, \mathbf{b}_{1:T}, \boldsymbol{\sigma}_{1:T}^2, \boldsymbol{\theta}) =$$
$$= -\frac{1}{2}\tilde{\mathbf{y}}_t^\top\mathbf{K}_{\mathbf{x},t}^{-1}\frac{\partial\mathbf{K}_{\mathbf{x},t}}{\partial b_t}\mathbf{K}_{\mathbf{x},t}^{-1}\tilde{\mathbf{y}}_t - \frac{1}{2}\text{trace}\left(\mathbf{K}_{\mathbf{x},t}^{-1}\frac{\partial\mathbf{K}_{\mathbf{x},t}}{\partial b_t}\right) \tag{23}$$

$$\frac{\partial}{\partial\sigma_t^2}\log p(\mathbf{Y}_{1:T}|\mathbf{\Phi}, \bar{\mathbf{w}}_{\mathbf{y},1}, \dots, \bar{\mathbf{w}}_{\mathbf{y},T}, \mathbf{b}_{1:T}, \boldsymbol{\sigma}_{1:T}^2, \boldsymbol{\theta})$$
$$= -\frac{1}{2}\tilde{\mathbf{y}}_t^\top\mathbf{K}_{\mathbf{x},t}^{-1}\frac{\partial\mathbf{K}_{\mathbf{x},t}}{\partial\sigma_t^2}\mathbf{K}_{\mathbf{x},t}^{-1}\tilde{\mathbf{y}}_t - \frac{1}{2}\text{trace}\left(\mathbf{K}_{\mathbf{x},t}^{-1}\frac{\partial\mathbf{K}_{\mathbf{x},t}}{\partial\sigma_t^2}\right) \tag{24}$$

Note that the derivatives of the common parameters $\boldsymbol{\theta}$ are the sum of the partial derivatives for each task, which implies maximizing the joint multitask likelihood; whereas the derivatives of task dependent parameters, $\mathbf{b}_{1:T}$ and $\boldsymbol{\sigma}_{1:T}^2$, only depend on their associated factorized likelihoods. The algorithm for the inference of the parameters consists of the estimation of parameters $\boldsymbol{\alpha}_{\mathbf{y},t}$ and $\mathbf{w}_{\mathbf{y},t}$ with Eqs. (17) and (18) and, later, the optimization of the log likelihood in Eq. (21) with respect to parameters $\boldsymbol{\theta}, \mathbf{b}_{1:T}, \boldsymbol{\sigma}_{1:T}^2$ through gradient ascent with the use of gradients ()()() (22)–(24). The process must be repeated until some convergence criterion has been reached.

Finally, the dual parameters corresponding to the MAP estimation of the input related parameter vector can be computed as

$$\bar{\mathbf{w}}_{\mathbf{x},t} = \mathbf{\Phi}\boldsymbol{\alpha}_t \tag{25}$$

where $\boldsymbol{\alpha}_t = \mathbf{K}_{\mathbf{x},t}^{-1}\left(\mathbf{y}_t - \bar{\mathbf{w}}_{\mathbf{y},t}^\top\mathbf{Y}_{1:t-1}\right)$. This process is summarized in Algorithm 1.

Algorithm 1: Hierarchical_Cool_GPs

$\boldsymbol{\sigma}_{1:T}^2, \mathbf{b}_{1:T}, \boldsymbol{\theta}, \boldsymbol{\alpha}_{1:T}, \mathbf{w}_{\mathbf{y},1}, \ldots, \mathbf{w}_{\mathbf{y},T} = \text{Hierarchical\_Cool\_GPs}(\mathbf{X}, \mathbf{Y}_{1:T}, K)$

Data: $\mathbf{X}$ (inputs), $\mathbf{Y}_{1:T}$ (multi-output targets), K (covariance function)

// Randomly initialize parameters

$\boldsymbol{\sigma}_{1:T}^2, \mathbf{b}_{1:T}, \boldsymbol{\theta}$

// Compute kernel matrix of the input data

$\mathbf{K} = K(\mathbf{X}, \mathbf{X}; \boldsymbol{\theta})$

// Infer parameters of factorized GPs

while *Likelihood maximum is not reached* **do**

$\quad \Delta(\boldsymbol{\theta}) = 0$

$\quad$ **for** $t \leftarrow 1$ **to** T **do**

$\quad\quad$ // Step 1: Obtain MAP estimation of $\mathbf{w}_{\mathbf{y},t}$

$\quad\quad \boldsymbol{\alpha}_{\mathbf{y},t} = \left(b_t\mathbf{K} + \mathbf{Y}_{1:t-1}^\top\mathbf{Y}_{1:t-1} + \sigma_t^2\mathbf{I}\right)^{-1}\mathbf{y}_t$

$\quad\quad \bar{\mathbf{w}}_{\mathbf{y},t} = \mathbf{Y}_{1:t-1}\boldsymbol{\alpha}_{\mathbf{y},t}$

$\quad\quad$ // Step 2.1: Update independent parameters

$\quad\quad \sigma_t^2,\ b_t \leftarrow \sigma_t^2,\ b_t + \mu\dfrac{\partial}{\partial(\sigma_t^2, b_t)}\log p\left(\mathbf{y}_t|\boldsymbol{\Phi}, \mathbf{Y}_{1:t-1}, \bar{\mathbf{w}}_{\mathbf{y},t}, b_t, \sigma_t^2, \boldsymbol{\theta},\right)$

$\quad\quad$ // Step 2.2: Update gradient for common parameters

$\quad\quad \Delta(\boldsymbol{\theta}) \leftarrow \Delta(\boldsymbol{\theta}) + \mu\dfrac{\partial}{\partial\boldsymbol{\theta}}\log p\left(\mathbf{y}_t|\boldsymbol{\Phi}, \mathbf{Y}_{1:t-1}, \bar{\mathbf{w}}_{\mathbf{y},t}, b_t, \sigma_t^2, \boldsymbol{\theta}\right)$

$\quad$ **end**

$\quad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \mu\Delta(\boldsymbol{\theta})$

$\quad \mathbf{K} = K(\mathbf{X}, \mathbf{X}; \boldsymbol{\theta})$

end

// With inferred parameters, obtain final values of the dual variables

for $t \leftarrow 1$ **to** T **do**

$\quad \boldsymbol{\alpha}_t = \left(b_t\mathbf{K} + \sigma_t^2\mathbf{I}\right)^{-1}\left(\mathbf{y}_t - \bar{\mathbf{w}}_{\mathbf{y},t}^\top\mathbf{Y}_{1:t-1}\right)$

end

Result: $\boldsymbol{\sigma}_{1:T}^2$ (noise variances of factorized GPs) , $\mathbf{b}_{1:T}$ (prior amplitudes), $\boldsymbol{\theta}$ (kernel or covariance function parameters) , $\boldsymbol{\alpha}_{1:T}$ (dual variables), $\bar{\mathbf{w}}_{\mathbf{y},1}, \ldots, \bar{\mathbf{w}}_{\mathbf{y},T}$ (MAP estimation previous task weights)

The algorithm developed above needs a matrix inversion for the computation of $\boldsymbol{\alpha}_{\mathbf{y},t}$ and another inversion of $\mathbf{K}_{\mathbf{x},t}$ for the gradient descent and the computation of $\boldsymbol{\alpha}_t$. To reduce this computational cost, in the next section we introduce an approximated inference approach that only uses a single learning step, requiring a single matrix to be inverted.

3.2.2. An approximate approach for Cool-MTGP learning

To simplify the hierarchical learning approach described above, we can assume a prior for both $\mathbf{w}_{\mathbf{x},t}$ and $\mathbf{w}_{\mathbf{y},t}$ to learn the model parameters and infer their variables $\mathbf{w}_{\mathbf{x},t}$ and $\mathbf{w}_{\mathbf{y},t}$ with a common model. In fact, considering that $\mathbf{w}_{\mathbf{y},t}$ and $\mathbf{w}_{\mathbf{x},t}$ are given by (10) and (12) the model for each Cool-MTGP would consist of a Gaussian Process with zero mean and covariance $b_t\mathbf{K} + \mathbf{Y}_{1:t-1}^\top\mathbf{Y}_{1:t-1}$ (the same GP defined by the first step of the hierarchical model as it is shown in Fig. 1(b)).

So, with this scheme, the joint marginalized MT likelihood would be approximated by the following set of conditional one-output likelihoods[1]

$$\sum_{t=1}^{T}\log p(\mathbf{y}_t|\boldsymbol{\Phi}, \mathbf{Y}_{1:t-1}, b_t, \sigma_t^2, \boldsymbol{\theta})$$

$$= -\sum_{t=1}^{T}\left(\frac{1}{2}\mathbf{y}_t^\top\mathbf{K}_{\mathbf{xy},t}^{-1}\mathbf{y}_t + \frac{1}{2}\log|\mathbf{K}_{\mathbf{xy},t}| + \frac{N}{2}\log 2\pi\right) \quad (26)$$

where $\mathbf{K}_{\mathbf{xy},t}$ is the full kernel matrix that combines the information of the input data with that of the previous tasks. We can learn the model parameters by maximizing them using the following derivatives with respect to the common parameters θ:

$$\frac{\partial}{\partial\theta_j}\sum_{t=1}^{T}\log p(\mathbf{y}_t|\boldsymbol{\Phi}, \mathbf{Y}_{1:t-1}, b_t, \sigma_t^2, \boldsymbol{\theta})$$

$$= \sum_{t=1}^{T}\left(-\frac{1}{2}\mathbf{y}_t^\top\mathbf{K}_{\mathbf{xy},t}^{-1}\frac{\partial\mathbf{K}_{\mathbf{xy},t}}{\partial\theta_j}\mathbf{K}_{\mathbf{xy},t}^{-1}\mathbf{y}_t - \frac{1}{2}\text{trace}\left(\mathbf{K}_{\mathbf{xy},t}^{-1}\frac{\partial\mathbf{K}_{\mathbf{xy},t}}{\partial\theta_j}\right)\right) \quad (27)$$

and for the task dependent parameters:

$$\frac{\partial}{\partial b_t}\sum_{t=1}^{T}\log p(\mathbf{y}_t|\boldsymbol{\Phi}, \mathbf{Y}_{1:t-1}, b_t, \sigma_t^2, \boldsymbol{\theta})$$

$$= \frac{\partial}{\partial\vartheta_j}\log p(\mathbf{y}_t|\boldsymbol{\Phi}, \mathbf{Y}_{1:t-1}, b_t, \sigma_t^2, \boldsymbol{\theta})$$

$$= -\frac{1}{2}\mathbf{y}_t^\top\mathbf{K}_{\mathbf{xy},t}^{-1}\frac{\partial\mathbf{K}_{\mathbf{xy},t}}{\partial b_t}\mathbf{K}_{\mathbf{xy},t}^{-1}\mathbf{y}_t - \frac{1}{2}\text{trace}\left(\mathbf{K}_{\mathbf{xy},t}^{-1}\frac{\partial\mathbf{K}_{\mathbf{xy},t}}{\partial b_t}\right) \quad (28)$$

---

[1] Note that each cool likelihood is Gaussian, but their covariance is depending on $\mathbf{Y}_{1:t-1}$, so their products do not return the equivalent marginalized MT likelihood Gaussian distribution but an approximation to it.

$$\frac{\partial}{\partial \sigma_t^2} \sum_{t=1}^{T} \log p(\mathbf{y}_t | \mathbf{\Phi}, \mathbf{Y}_{1:t-1}, b_t, \sigma_t^2, \boldsymbol{\theta})$$

$$= \frac{\partial}{\partial \sigma_t^2} \log p(\mathbf{y}_t | \mathbf{\Phi}, \mathbf{Y}_{1:t-1}, b_t, \sigma_t^2, \boldsymbol{\theta})$$

$$= -\frac{1}{2} \mathbf{y}_t^\top \mathbf{K}_{\mathbf{xy},t}^{-1} \frac{\partial \mathbf{K}_{\mathbf{xy},t}}{\partial \sigma_t^2} \mathbf{K}_{\mathbf{xy},t}^{-1} \mathbf{y}_t - \frac{1}{2} \text{trace} \left(\mathbf{K}_{\mathbf{xy},t}^{-1} \frac{\partial \mathbf{K}_{\mathbf{xy},t}}{\partial \sigma_t^2} \right) \quad (29)$$

Once the model parameters are learnt, we can use the joint posterior distribution of $\mathbf{w}_{\mathbf{x},t}$ and $\mathbf{w}_{\mathbf{y},t}$ (see (14) and (15)) to obtain their MAPs estimations as:

$$\bar{\mathbf{w}}_{\mathbf{x},t} = \mathbf{\Phi} \boldsymbol{\alpha}_t \quad (30)$$

$$\bar{\mathbf{w}}_{\mathbf{y},t} = \mathbf{Y}_{1:t-1} \boldsymbol{\alpha}_t \quad (31)$$

where the dual variables $\boldsymbol{\alpha}_t$ are

$$\boldsymbol{\alpha}_t = \mathbf{K}_{\mathbf{xy},t}^{-1} \mathbf{y}_t \quad (32)$$

The difference between this approach and the hierarchical approach is how $\mathbf{w}_{\mathbf{y},t}$ is treated. In the hierarchical model two inference steps are used, one to estimate the mean of $\mathbf{w}_{\mathbf{y},t}$ (see Eq. (17)–(18)) and another to obtain the cool-GP; but in the approximate Cool-MTGP, each cool-GP models $\mathbf{w}_{\mathbf{y},t}$ as a random variable with a single inference step. The disadvantage of this approach relies in the fact that the conditional likelihoods are no longer Gaussian and therefore we cannot ensure that the original marginalized MTGP likelihood can be exactly recovered. For this reason, this approach is called *approximate*.

However, assuming that this criterion is valid, we get several advantages. First, this optimization only requires the inversion of matrix $\mathbf{K}_{\mathbf{xy},t}$ in both parts of the hierarchic process described in the paper, whereas the exact procedure requires the additional inversion of matrix $\mathbf{K}_{\mathbf{x},t}$ in the first level of the hierarchic process. Besides, if the kernel is linear, the approximate procedure can be implemented with the use of standard univariate GP libraries, and a more sophisticated wrapper that modifies the common parameter inference or a simple cross validation can be used for the common parameters. Both procedures have been compared in the experiments, showing similar results, albeit slightly better for the exact process. Algorithm 2 summarizes the main steps of this process.

Algorithm 2: Approximate_Cool_GPs

$\boldsymbol{\sigma}_{1:T}^2, \mathbf{b}_{1:T}, \boldsymbol{\theta}, \boldsymbol{\alpha}_{1:T}, \mathbf{w}_{\mathbf{y},1}, \dots, \mathbf{w}_{\mathbf{y},T} = \text{Approximate\_Cool\_GPs}(\mathbf{X}, \mathbf{Y}_{1:T}, K)$

Data: $\mathbf{X}$ (inputs), $\mathbf{Y}_{1:T}$ (multi-output targets), K (covariance function)

// Randomly initialize hyperparameters

$\boldsymbol{\sigma}_{1:T}^2, \mathbf{b}_{1:T}, \boldsymbol{\theta}$

// Compute kernel matrix of the input data:

$\mathbf{K} = K(\mathbf{X}, \mathbf{X}; \boldsymbol{\theta})$

// Infer parameters of factorized GPs

while *Likelihood maximum is not reached* **do**

    $\Delta(\boldsymbol{\theta}) = 0$

    for $t \leftarrow 1$ **to** T **do**

        // Update independent parameters

        $\sigma_t^2, \ b_t \leftarrow \sigma_t^2, \ b_t + \mu \dfrac{\partial}{\partial (\sigma_t^2, b_t)} \log p \left(\mathbf{y}_t | \mathbf{\Phi}, b_t, \sigma_t^2, \boldsymbol{\theta} \right)$

        // Update gradient for common parameters

        $\Delta(\boldsymbol{\theta}) \leftarrow \Delta(\boldsymbol{\theta}) + \mu \dfrac{\partial}{\partial \boldsymbol{\theta}} \log p \left(\mathbf{y}_t | \mathbf{\Phi}, b_t, \sigma_t^2, \boldsymbol{\theta} \right)$

    end

    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \mu \Delta(\boldsymbol{\theta})$

    $\mathbf{K} = K(\mathbf{X}, \mathbf{X}; \boldsymbol{\theta})$

end

// With inferred parameters, obtain final values of the dual
    variables and MAP estimation of previous task weights

for $t \leftarrow 1$ **to** T **do**

    $\boldsymbol{\alpha}_t = \mathbf{K}_{\mathbf{xy},t}^{-1} \mathbf{y}_t$

    $\bar{\mathbf{w}}_{\mathbf{y},t} = \mathbf{Y}_{1:t-1} \boldsymbol{\alpha}_t$

end

Result: $\boldsymbol{\sigma}_{1:T}^2$ (noise variances of factorized GPs) , $\mathbf{b}_{1:T}$ (prior amplitudes), $\boldsymbol{\theta}$
        (kernel or covariance function parameters) , $\boldsymbol{\alpha}_T$ (dual variables),
        $\bar{\mathbf{w}}_{\mathbf{y},1}, \dots, \bar{\mathbf{w}}_{\mathbf{y},T}$ (MAP estimation previous task weights)

3.3. Recovering the multitask model

Despite the fact that intertask and noise covariance matrices $\mathbf{C}_{1:T,1:T}$ and $\Sigma_{1:T,1:T}$ are not explicit in the conditioned model, they can be recovered from parameters $\mathbf{b}_{1:T}, \sigma^2_{1:T}$ and the MAP estimation of the weights, $\bar{\mathbf{w}}_{\mathbf{y},1}, \ldots, \bar{\mathbf{w}}_{\mathbf{y},T}$ and $\mathbf{W}_{\mathbf{x},1:T} = [\bar{\mathbf{w}}_{\mathbf{x},1}, \ldots, \bar{\mathbf{w}}_{\mathbf{x},T}]$. The general MTGP formulation considers that the joint MT likelihood, which contains the noise matrix $\Sigma_{1:T,1:T}$, is given by

$$p(\text{vect}(\mathbf{Y}_{1:T})|\mathbf{\Phi}, \mathbf{W}_{1:T}) = \mathcal{N}(\mathbf{y}_{1:T}|\bar{\mathbf{y}}_{1:T}, \Sigma_{1:T,1:T} \otimes \mathbf{I}). \tag{33}$$

Considering the factorization given by (8), the factorized likelihoods of (9), and applying the properties of the products of conditional Gaussians (see, e.g. [29]), we can recursively recover the mean $\bar{\mathbf{y}}_{1:T}$ and the covariance terms, which leads us directly to the MT noise covariance $\Sigma_{1:T,1:T}$, as:

$$\begin{aligned} \bar{\mathbf{y}}_t &= \mathbf{\Phi}^\top \bar{\mathbf{w}}_t = \mathbf{\Phi}^\top \mathbf{w}_{\mathbf{x},t} + \bar{\mathbf{y}}^\top_{1:t-1} \\ \Sigma_{t,t} &= \sigma^2_t + \bar{\mathbf{w}}^\top_{\mathbf{y},t} \Sigma_{1:t-1,1:t-1} \bar{\mathbf{w}}_{\mathbf{y},t} \\ \Sigma_{t,1:t} &= \Sigma^\top_{1:t,t} = \bar{\mathbf{w}}^\top_{\mathbf{y},t} \Sigma_{1:t-1,1:t-1} \end{aligned} \tag{34}$$

where it is assumed that $\Sigma_{1,1} = \sigma^2_1$ and $\bar{\mathbf{y}}_1 = \mathbf{\Phi}^\top \mathbf{w}_{\mathbf{x},1}$.

For the purpose of recovering covariance matrix $\mathbf{C}_{1:T,1:T}$, two elements are needed: (1) An expression relating the multitask weights $\mathbf{W}_{1:T}$ to the input related weights $\mathbf{w}_{\mathbf{x},t}$; and, (2) the joint prior of all $\mathbf{w}_{\mathbf{x},t}$ parameters. We assume here that the values of $\mathbf{b}_{1:T}$ and the MAP value of the previous task weights $\bar{\mathbf{w}}_{\mathbf{y},1}, \ldots, \bar{\mathbf{w}}_{\mathbf{y},T}$ have been already learnt using any of the MTGP approaches proposed above.

To obtain an expression of $\mathbf{W}_{1:T}$ as a function of the input parameters $\mathbf{w}_{\mathbf{x},t}$, we use

$$\bar{\mathbf{y}}_t = \mathbf{\Phi}^\top \bar{\mathbf{w}}_t = \mathbf{\Phi}^\top \mathbf{w}_{\mathbf{x},t} + \bar{\mathbf{y}}^\top_{1:t-1} \bar{\mathbf{w}}_{\mathbf{y},t} \tag{35}$$

to get:

$$\mathbf{w}_t = \mathbf{w}_{\mathbf{x},t} + \mathbf{W}^\top_{1:t-1} \bar{\mathbf{w}}_{\mathbf{y},t}, \tag{36}$$

that extended for $t = 1, \ldots, T$, leads to the following equation system

$$\mathbf{W}^\top_{1:T} = (\mathbf{I} - \bar{\mathbf{W}}_{\mathbf{y}})^{-1} \mathbf{W}^\top_{\mathbf{x},1:T} \tag{37}$$

where $\mathbf{W}_{\mathbf{x},1:T} = [\mathbf{w}_{\mathbf{x},1}, \cdots, \mathbf{w}_{\mathbf{x},T}]$ and

$$\bar{\mathbf{W}}_{\mathbf{y}} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ \bar{\mathbf{w}}_{\mathbf{y},2}[1] & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \bar{\mathbf{w}}_{\mathbf{y},T-1}[1] & \bar{\mathbf{w}}_{\mathbf{y},T-1}[2] & \cdots & 0 & 0 \\ \bar{\mathbf{w}}_{\mathbf{y},T}[1] & \bar{\mathbf{w}}_{\mathbf{y},T}[3] & \cdots & \bar{\mathbf{w}}_{\mathbf{y},T}[T-1] & 0 \end{bmatrix}, \tag{38}$$

where $\bar{\mathbf{w}}_{\mathbf{y},t}[k]$ is component k of vector $\bar{\mathbf{w}}_{\mathbf{y},t}$.

In order to identify the joint prior of all $\mathbf{w}_{\mathbf{x},t}$ parameters, we formulate it as the following distribution

$$p(\text{vect}(\mathbf{W}_{\mathbf{x},1:T})) = \mathcal{N}(\text{vect}(\mathbf{W}_{\mathbf{x},1:T})|\mathbf{0}, \mathbf{B}_{1:T,1:T} \otimes \Sigma_p) \tag{39}$$

with matrix $\mathbf{B}_{1:T,1:T}$ being a cross correlation between tasks. Taking into account that the diagonal terms of $\mathbf{B}_{1:T,1:T}$ are $b_{tt} = b_t$ (they have been already learnt – see Sections 3.2.1 and 3.2.2), the remaining cross terms can be expressed through the correlation coefficient

$$b_{tt'} = \sqrt{b_t b_{t'}} \hat{\rho}_{t,t'} \tag{40}$$

where $\hat{\rho}_{t,t'}$ can be estimated as:

$$\hat{\rho}_{t,t'} = \frac{\mathbf{w}^\top_{\mathbf{x},t} \mathbf{w}_{\mathbf{x},t'}}{|| \mathbf{w}_{\mathbf{x},t} |||| \mathbf{w}_{\mathbf{x},t'} ||} \tag{41}$$

Since this computation depends on variables $\mathbf{w}_{\mathbf{x},t}$ and $\mathbf{w}_{\mathbf{x},t'}$, we can carry it out by either generating samples from their posterior distribution and obtaining the values of $\mathbf{B}_{1:T,1:T}$ with a Monte Carlo approximation, or by directly considering that $\mathbf{W}_{\mathbf{x},1:T}$ are given by their MAP estimations. In fact, if we consider the latter approach, the calculation of the terms of $\mathbf{B}_{1:T,1:T}$ can be expressed in a more compact form as:

$$b_{tt'} = \sqrt{b_t b_{t'}} \hat{\rho}_{t,t'} = \frac{\sqrt{b_t b_{t'}} \boldsymbol{\alpha}^\top_t \mathbf{K} \boldsymbol{\alpha}_{t'}}{\sqrt{\boldsymbol{\alpha}^\top_t \mathbf{K} \boldsymbol{\alpha}_t \boldsymbol{\alpha}^\top_{t'} \mathbf{K} \boldsymbol{\alpha}_{t'}}}. \tag{42}$$

where $\boldsymbol{\alpha}_t = \mathbf{K}^{-1}_{\mathbf{x},t}(\mathbf{y}_t - \bar{\mathbf{w}}^\top_{\mathbf{y},t} \mathbf{Y}_{1:t-1})$ with $\bar{\mathbf{w}}_{\mathbf{y},t}$ given by (16) in the hierarchical Cool-MTGP method, or $\boldsymbol{\alpha}_t = \mathbf{K}^{-1}_{\mathbf{xy},t} \mathbf{y}_t$ in the approximate approach.

Finally, the knowledge of matrix $\mathbf{B}_{1:T,1:T}$ together with Eq. (37) leads to the expression of the multitask covariance matrix

$$\begin{aligned} \mathbf{C}_{1:T,1:T} \otimes \Sigma_p &= \mathbb{E}\{\text{vect}(\mathbf{W}_{1:T})\text{vect}(\mathbf{W}_{1:T})^\top\} \\ &= \mathbb{E}\left\{\text{vect}\left((\mathbf{I} - \mathbf{W}_{\mathbf{y}})^{-1} \mathbf{W}^\top_{\mathbf{x},1:T}\right)\text{vect}\left((\mathbf{I} - \mathbf{W}_{\mathbf{y}})^{-1} \mathbf{W}^\top_{\mathbf{x},1:T}\right)^\top\right\} \\ &= (\mathbf{I} - \mathbf{W}_{\mathbf{y}})^{-1} \mathbb{E}\left\{\text{vect}(\mathbf{W}^\top_{\mathbf{x},1:T})\text{vect}(\mathbf{W}_{\mathbf{x},1:T})\right\}\left((\mathbf{I} - \mathbf{W}_{\mathbf{y}})^{-1}\right)^\top \end{aligned} \tag{43}$$

where, by Eq. (39), we see that

$$\mathbf{C}_{1:T,1:T} \otimes \Sigma_p = (\mathbf{I} - \mathbf{W}_{\mathbf{y}})^{-1}(\mathbf{B}_{1:T,1:T} \otimes \Sigma_p)\left((\mathbf{I} - \mathbf{W}_{\mathbf{y}})^{-1}\right)^\top \tag{44}$$

and, finally, using the Kronecker product properties, it reduces to

$$\mathbf{C}_{1:T,1:T} = (\mathbf{I} - \bar{\mathbf{W}}_{\mathbf{y}})^{-1} \mathbf{B}_{1:T,1:T}\left((\mathbf{I} - \bar{\mathbf{W}}_{\mathbf{y}})^{-1}\right)^\top \tag{45}$$

The summary of the process is in Algorithm 3.

Algorithm 3: Cool_MTGP_Learning

$\boldsymbol{\Sigma}_{1:T,1:T}, \mathbf{C}_{1:T,1:T}, \boldsymbol{\theta} = \text{Cool\_MTGP\_Learning}(\mathbf{X}, \mathbf{Y}_{1:T}, K, \text{Type})$

Data: $\mathbf{X}$ (inputs), $\mathbf{Y}_{1:T}$ (multi-output targets), K (covariance function), Type (indicator for hierarchical or approximate Cool-GPs approach)

// Train a model of Cool-GPs

if *Type $==$ hierarchical* **then**

$\quad | \quad \boldsymbol{\sigma}^2_{1:T}, \mathbf{b}_{1:T}, \boldsymbol{\theta}, \boldsymbol{\alpha}_{1:T}, \bar{\mathbf{w}}_{\mathbf{y},1}, \ldots, \bar{\mathbf{w}}_{\mathbf{y},T} = \text{Hierarchical\_Cool\_GPs}(\mathbf{X}, \mathbf{Y}_{1:T}, K)$

end

if *Type $==$ approximate* **then**

$\quad | \quad \boldsymbol{\sigma}^2_{1:T}, \mathbf{b}_{1:T}, \boldsymbol{\theta}, \boldsymbol{\alpha}_{1:T}, \bar{\mathbf{w}}_{\mathbf{y},1}, \ldots, \bar{\mathbf{w}}_{\mathbf{y},T} = \text{Approximate\_Cool\_GPs}(\mathbf{X}, \mathbf{Y}_{1:T}, K)$

end

// Compute multitask covariance of noise

$\boldsymbol{\Sigma}_{1,1} = \sigma^2_1$

for $t \leftarrow 2$ **to** T **do**

$\quad | \quad \boldsymbol{\Sigma}_{t,t} = \sigma^2_t + \bar{\mathbf{w}}^\top_{\mathbf{y},t} \boldsymbol{\Sigma}_{1:t-1,1:t-1} \bar{\mathbf{w}}_{\mathbf{y},t}$

$\quad | \quad \boldsymbol{\Sigma}_{t,1:t} = \boldsymbol{\Sigma}^\top_{1:t,t} = \bar{\mathbf{w}}^\top_{\mathbf{y},t} \boldsymbol{\Sigma}_{1:t-1,1:t-1}$

end

// Compute multitask weights of the prior

$\mathbf{C}_{1:T,1:T} = \left(\mathbf{I} - \bar{\mathbf{W}}_\mathbf{y}\right)^{-1} \mathbf{B}_{1:T,1:T} \left(\left(\mathbf{I} - \bar{\mathbf{W}}_\mathbf{y}\right)^{-1}\right)^\top$

where $\bar{\mathbf{W}}_\mathbf{y}$ is given by (38), $\mathbf{K} = K(\mathbf{X}, \mathbf{X}; \boldsymbol{\theta})$ and entries t, t' of $\mathbf{B}_{1:T,1:T}$ are given by

$b_{tt'} = \dfrac{\sqrt{b_t b_{t'}} \boldsymbol{\alpha}^\top_t \mathbf{K} \boldsymbol{\alpha}_{t'}}{\sqrt{\boldsymbol{\alpha}^\top_t \mathbf{K} \boldsymbol{\alpha}_t \boldsymbol{\alpha}^\top_{t'} \mathbf{K} \boldsymbol{\alpha}_{t'}}}$,

Result: $\boldsymbol{\Sigma}_{1:T,1:T}$ (noise covariance matrix) and $\mathbf{C}_{1:T,1:T}$ (intertask covariance matrix)

One might be concerned that this reconstruction process, and therefore the overall performance of the model depends on the order in which the tasks are assigned to the Cool-GPs. However, experimental results show that the reconstruction of the matrices is consistent for any random permutation of the tasks, confirming that task order shows no impact.

4. Predictive multitask model

Once the noise $\boldsymbol{\Sigma}_{1:T,1:T}$ and intertask $\mathbf{C}_{1:T,1:T}$ covariance matrices have been obtained using any of the described methods, we are ready to apply the general predictive model of the multitask Gaussian process from Eqs. (5) (see Section 2) to obtain the predictive model as describes the Algorithm 4.

Algorithm 4: Cool_MTGP_Predictive

$\bar{\mathbf{f}}^*_{1:T}, \text{cov}(\mathbf{f}^*_{1:T}) = \text{Cool\_MTGP\_Predictive}(\mathbf{X}, \mathbf{Y}_{1:T}, K, Type, \mathbf{x}^*)$

Data: $\mathbf{X}$ (inputs), $\mathbf{Y}_{1:T}$ (multi-output targets), K (covariance function), Type (indicator for hierarchical or approximate Cool-GPs approach), $\mathbf{x}^*$ (test input)

// Inference over MTGP model

$\boldsymbol{\Sigma}_{1:T,1:T}, \mathbf{C}_{1:T,1:T}, \boldsymbol{\theta} = \text{Cool\_MTGP\_Learning}(\mathbf{X}, \mathbf{Y}_{1:T}, K, \text{Type})$

// Compute kernel matrices

$\mathbf{K} = K(\mathbf{X}, \mathbf{X}; \boldsymbol{\theta})$

$\mathbf{k}_* = K(\mathbf{X}, \mathbf{x}^*; \boldsymbol{\theta})$

$\mathbf{k}_{**} = K(\mathbf{x}^*, \mathbf{x}^*; \boldsymbol{\theta})$

// Predictive Mean and Covariance

$\bar{\mathbf{f}}^*_{1:T} = \left(\mathbf{C}_{1:T,1:T} \otimes \mathbf{k}^\top_*\right) \left(\mathbf{C}_{1:T,1:T} \otimes \mathbf{K}) + \boldsymbol{\Sigma}^{-1}_{1:T,1:T} \otimes \mathbf{I}\right)^{-1} \text{vect}(\mathbf{Y}_{1:T})$

$\text{cov}(\mathbf{f}^*_{1:T}) = (\mathbf{C}_{1:T,1:T} \otimes \mathbf{k}_{**}) -$
$\quad \left(\mathbf{C}_{1:T,1:T} \otimes \mathbf{k}^\top_*\right) \left(\mathbf{C}_{1:T,1:T} \otimes \mathbf{K} + \boldsymbol{\Sigma}^{-1}_{1:T,1:T} \otimes \mathbf{I}\right)^{-1} (\mathbf{C}_{1:T,1:T} \otimes \mathbf{k}_*)$

Result: $\bar{\mathbf{f}}^*_{1:T}$ (predictive mean), $\text{cov}(\mathbf{f}^*_{1:T})$ (predictive covariance)

---

5. Experimental Results

5.1. Synthetic benchmark

We have carried out a synthetic data experiment to compare the performance of all the considered MTGP algorithms against a ground truth model. Following the general model of Section 2, a data set has been drawn from likelihood function $p(\mathbf{y}|\mathbf{X}) = \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{C}_{1:T,1:T} \otimes \mathbf{K} + \Sigma_{1:T,1:T} \otimes \mathbf{I})$, where $\mathbf{K} = \mathbf{X}^\top\mathbf{X}$ and the intertask and noise covariance matrices follow the low rank form $\mathbf{C}_{1:T,1:T} = \sum_{r=1}^{R} \mathbf{c}_r \mathbf{c}_r^\top$ (and similarly for $\Sigma_{1:T,1:T}$). We have created datasets for three scenarios in which the $\mathbf{C}_{1:T,1:T}$ and $\Sigma_{1:T,1:T}$ matrices were generated with R values of $= 5, 10$ and 15. In all cases, $T = 15$ tasks, $N = 200$ samples and 10 iterations were run with randomly split training and test partitions with 100 samples each.

We compare both the hierarchical (HCool-MT) and approximate ($\sim$Cool-MT) versions of the proposed model to the standard MTGP (Std-MT) of [13] and the extension introduced in [18], which includes a noise matrix (Σ-MT). Since these methods require the selection of parameter P, we have analyzed three values: one equal to R (the ideal case), a value of P smaller than R and, where possible, a value of P greater than R. Additionally, a ground-truth model that uses the true intertask and noise covariance matrices is included, as well as a set of T independent GPs. Predictive performance is measured with the mean square error (MSE) averaged over all the tasks.

Fig. 2 shows that Σ-MT has the highest sensitivity to the choice of P, and its performance degrades when the scenario complexity (defined by matrix rank R) increases. Std-MT shows more robustness with respect to both the selection of P and the value of R. As

was expected, both models show their best performances when $P = R$, and thus a cross validation of P is paramount for these methods to perform optimally. This sensitivity depends on the number of the parameters to be inferred, which is $TP + 1$ in the case of Std-MT and $2TP$ for Σ-MT, while it is only $2T$ for both Cool-MTs. This allows our model to perform closer to the ground truth independently of the scenario complexity R.

Experimentally, it can be seen in Fig. 3 that, while all models are capable of inferring the intertask covariance, the noise matrix is not properly inferred by the Σ-MT when scenario complexity (R value) is high. Besides, comparing both versions of the Cool-MT model, the hierarchical approach is unsurprisingly slightly better at estimating the true parameters and reconstructing the noise matrix, leading to a higher consistency in its predictions as was already shown in Fig. 2.

Finally we address the possibility that the model is sensitive to task ordering during the training of the Cool-MTs. As can be seen in Fig. 4, the model is also consistent with regard to the order of the tasks.

5.2. Predictive variance assessment

One of the most important benefits of using a GP model is the ability to obtain the confidence for each prediction from the predictive distribution as well as the predicted value. In this section we study the integrity of these predictive distributions for the different MTGP models under study.

For this purpose, we have generated a simple benchmark in which a synthetic dataset is generated with the following likelihood
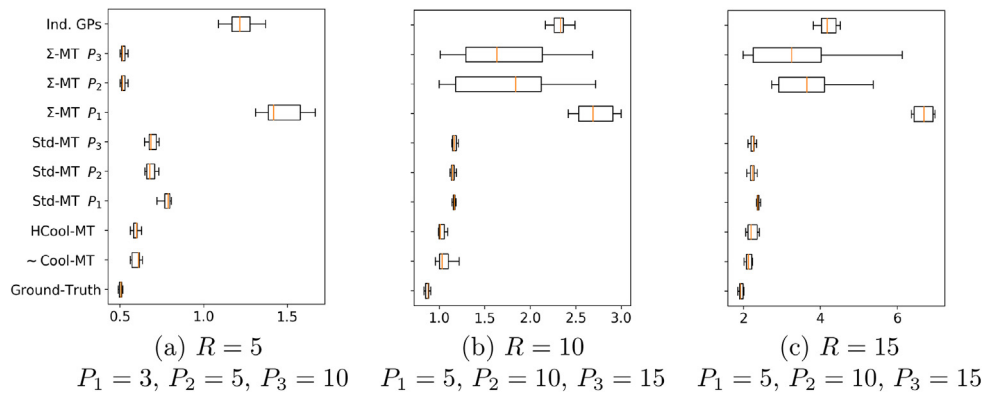

(a) $R = 5$ (b) $R = 10$ (c) $R = 15$

$P_1 = 3, P_2 = 5, P_3 = 10$ $P_1 = 5, P_2 = 10, P_3 = 15$ $P_1 = 5, P_2 = 10, P_3 = 15$

Fig. 2. MSE for all models of the synthetic experiment and different values of the matrix rank R of the generative model and parameter P for Std-MT and Σ-MT.
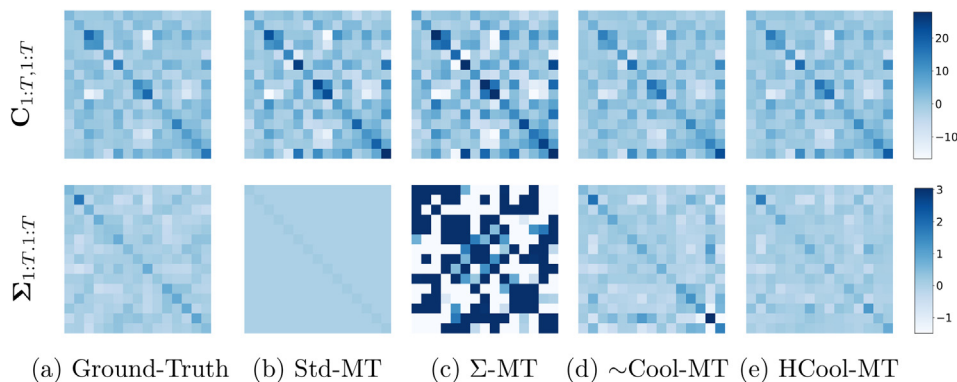

(a) Ground-Truth   (b) Std-MT   (c) Σ-MT   (d) $\sim$Cool-MT   (e) HCool-MT

Fig. 3. Estimated intertask, $\mathbf{C}_{1:T,1:T}$, and noise, $\Sigma_{1:T,1:T}$, covariance matrices vs. true ones (Ground-Truth) when R = 10. Std-MT and Σ-MT were trained for $P = R = 10$.
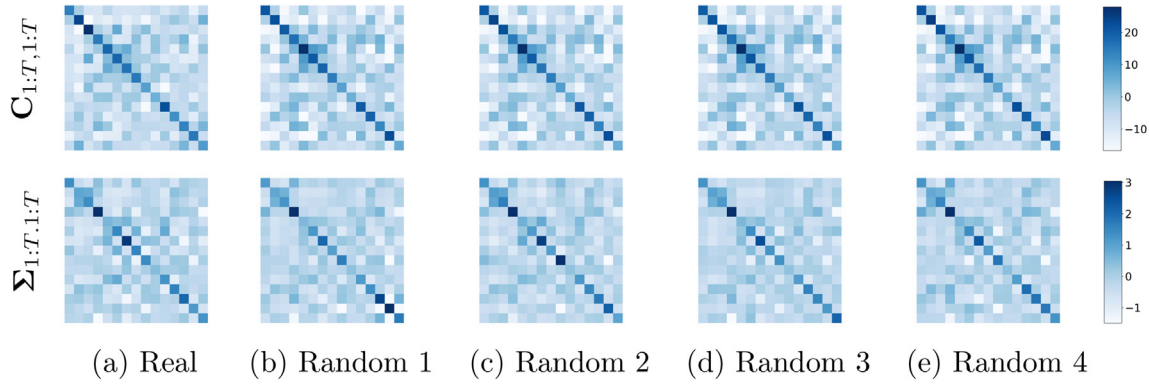
Fig. 4. Estimated $\mathbf{C}_{1:T,1:T}$ and $\mathbf{\Sigma}_{1:T,1:T}$ covariances for four random permutations of the task order for 15 tasks with R = 15.
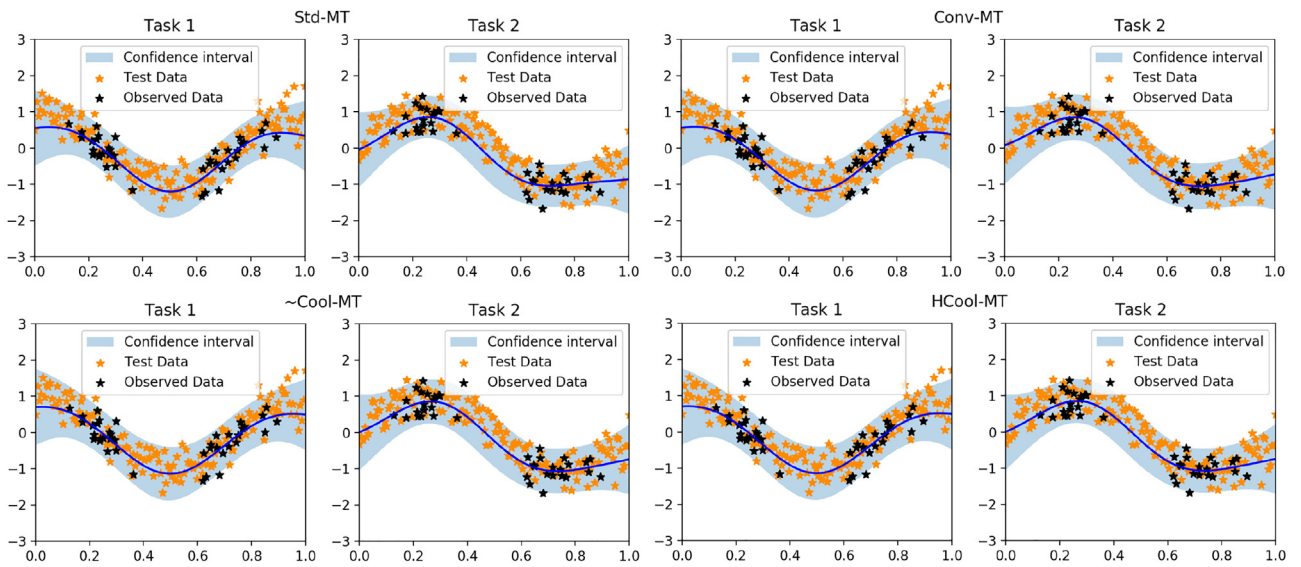

Fig. 5. Predictive mean and confidence interval (vertical axis) with respect to the different input values (horizontal axis) for both tasks and over the different models under study..

$$p(\mathbf{y}) = \mathcal{N}\left(\mathbf{y} \middle| \begin{pmatrix} \cos(2\pi x) \\ \sin(2\pi x) \end{pmatrix}, \begin{pmatrix} 0.1 & 0.05 \\ 0.05 & 0.1 \end{pmatrix}\right) \tag{46}$$

where $x \in (0, 1)$. Specifically, we have generated a training set in which the input data is clustered into two groups to force the presence of areas of lower training sample density.

For this study we have considered the Std-MTGP, the convolutional model (Conv-MT) proposed in [30] and the two proposed versions of the Cool-MT. The Σ-MT model was left out due to the lack of a non-linear kernel implementation. For the reference models (Std-MT and Conv-MT), we have found that their libraries only provide an estimation of the confidence interval task-by-task. Therefore, we start by analyzing these intervals. As can be seen in Fig. 5, we observe that all models arrive at similar task-wise confidence intervals.

Going deeper into this analysis, we have recovered the complete predictive distribution for the Cool-MT and for the Std-MT (this can be easily done using their learnt $C_{1:T,1:T}$ and $\Sigma_{1:T,1:T}$ matrices); however, we have not been able to obtain this distribution for the Conv-MT due to the complexity of this model and the black-box nature of its implementation, which has

made it difficult to recover the hyperparameters. To analyse the complete predictive posterior for these models we have selected three test samples: $x^* = 0, x^* = 0.25$ and $x^* = 0.5$. This way we cover regions with high and low predictive confidence. The results, depicted in Fig. 6 allow us to appreciate clear differences:

- While Std-MT does offer a full predictive distribution, it considers the noise to be independent among tasks. This prevents the method from modelling the relationships among the tasks in the predictive distribution. Therefore its predictive distribution covariance matrix tends to be diagonal.
- Both versions of Cool-MT produce predictive posteriors that adequately model the correlation among tasks with a full covariance matrix.
- In regions where the confidence is lower, the distribution for the Cool-MT model widens yet it adequately retains its shape. This isn't the case for Std-MT, which seems to be insensitive to the level of confidence for a particular region.
- There are no appreciable differences between the predictive distributions obtained by HCool-MT and ∼Cool-MT.
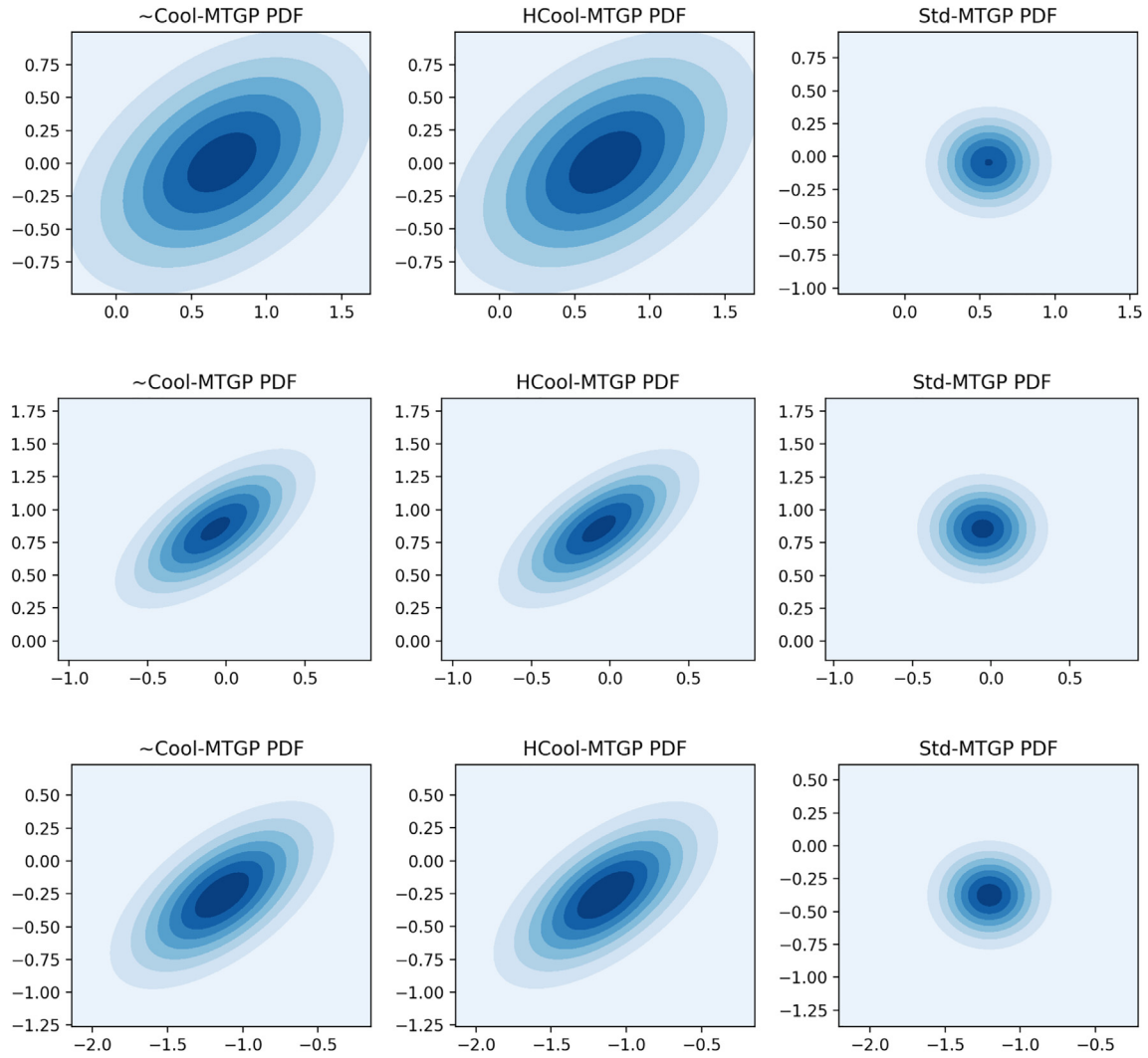
Fig. 6. Level curves of the joint predictive distribution in the output space, represented by $f_1(x^*)$ in the horizontal axis and $f_2(x^*)$ in the vertical axis for different values of x^* ($x^* = 0$ (top), $x^* = 0.25$ (center) and $x^* = 0.5$ (bottom)).

5.3. Real data benchmarks

In this experiment we use ten real world scenarios to compare the Cool-MT model's capabilities to those of the Std-MT and Conv-MT models[2].

To accomplish this, we have made use of the collection of datasets featured in [31], which can be found online at http://mulan.sourceforge.net/. These datasets offer a good variety in sample size, input dimensionality and number of tasks, as can be seen in Table 1. Due to the small performance differences seen in the previous section between the hierarchical and approximate versions of the Cool-MT, for the real world benchmarks we have only used the approximate model.

In all cases a standard normalization of the data was applied to both the input variables and output targets. The normalization parameters were obtained using the training partition only. Ten iterations were run with a random 80%/20% training/test partitioning of the data. In larger datasets (# samples > 1000) we dedicated 800 samples to the training set and 200 to the test set, limiting the number of total samples. All models were

Table 1
Real world datasets used in this work.

Dataset	Samples	Features	Tasks
andro	49	30	6
enb	768	8	2
edm	154	16	2
slump	103	7	3
oes10	403	298	16
oes97	334	263	16
atp1d	337	411	6
atp7d	296	411	6
scm1d	9803	280	16
scm20d	8966	61	16

trained with both a linear kernel and a squared exponential (SE) kernel.

Regarding to hyperparameter setting, Cool-MT doesn't need validation of any hyperparameters since they are all inferred by the algorithm. The kernel objects require the definition of an exploration region for their relevant parameters, so a sufficiently wide value range of $(10^{-10}, 10^3)$ was used to ensure good convergence of the optimizers. The Std-MT model requires the

---

[2] We have been unable to include Σ-MT in this part of our study due to convergence issues with the available implementation, as well as the lack of a non-linear kernel version.

Table 2

Real dataset benchmark results using a linear kernel: RMSE averaged over tasks. Best results in bold.

Dataset	Indep. GPs	Std-MT	Conv-MT	Cool-MT
andro	**0.74 ± 0.14**	0.81 ± 0.15	0.77 ± 0.10	0.75 ± 0.10
enb	**0.31 ± 0.02**	**0.31 ± 0.03**	**0.31 ± 0.02**	**0.31 ± 0.02**
edm	0.79 ± 0.05	**0.78 ± 0.05**	**0.78 ± 0.05**	**0.78 ± 0.05**
slump	0.68 ± 0.07	0.68 ± 0.07	0.68 ± 0.07	**0.67 ± 0.07**
oes10	0.47 ± 0.17	0.38 ± 0.14	**0.35 ± 0.13**	**0.35 ± 0.13**
oes97	0.56 ± 0.18	0.41 ± 0.12	**0.39 ± 0.11**	**0.39 ± 0.11**
atp1d	0.50 ± 0.05	0.49 ± 0.05	**0.42 ± 0.07**	**0.42 ± 0.07**
atp7d	0.70 ± 0.12	0.64 ± 0.11	**0.56 ± 0.07**	**0.56 ± 0.07**
scm1d	0.29 ± 0.02	0.27 ± 0.02	**0.24 ± 0.02**	**0.24 ± 0.02**
scm20d	0.36 ± 0.03	0.36 ± 0.02	**0.36 ± 0.02**	**0.36 ± 0.02**

Table 3

Real dataset benchmark results using a squared exponential kernel. RMSE averaged over tasks. Best results in bold.

Dataset	Indep. GPs	Std-MT	Conv-MT	Cool-MT
andro	0.62 ± 0.10	**0.42 ± 0.07**	0.46 ± 0.08	**0.42 ± 0.07**
enb	0.30 ± 0.02	0.15 ± 0.02	0.16 ± 0.02	**0.13 ± 0.02**
edm	0.70 ± 0.06	0.73 ± 0.06	**0.71 ± 0.06**	0.72 ± 0.05
slump	0.68 ± 0.11	0.67 ± 0.07	**0.61 ± 0.08**	0.63 ± 0.07
oes10	0.76 ± 0.51	0.85 ± 0.45	1.03 ± 0.43	**0.57 ± 0.41**
oes97	0.80 ± 0.56	0.81 ± 0.51	0.99 ± 0.49	**0.63 ± 0.46**
atp1d	0.49 ± 0.10	0.81 ± 0.12	0.90 ± 0.13	**0.41 ± 0.07**
atp7d	0.94 ± 0.15	0.88 ± 0.14	0.94 ± 0.15	**0.56 ± 0.10**
scm1d	0.26 ± 0.03	0.23 ± 0.02	0.99 ± 0.06	**0.22 ± 0.02**
scm20d	0.33 ± 0.03	**0.27 ± 0.03**	0.52 ± 0.28	0.28 ± 0.03

selection of hyperparameter P; for this setting, after trying values $1, T/2$ and T on a few datasets (where T is the number of tasks) using a small validation partition, it became apparent that the best value was consistently $P = 1$; and the kernel parameters were initialised using their default settings. The Conv-MT model requires the user to set the number of inducing points. After a brief exploration we settled on 50% of the size of the training partition, achieving a good performance while avoiding convergence issues.

Tables 2,3 show the benchmark results in terms of the root mean squared error (RMSE) for the linear and SE kernels respectively. In the linear case all models perform similarly, with a slight advantage in favour of both the Conv-MT and Cool-MT. A strong improvement in performance is obtained in all cases using a non-linear kernel, where the Cool-MT comes clearly on top in most datasets. After analysing the values for the SE kernel length-scale parameter learnt by all the models, it is clear that Conv-MT and, in some cases, Std-MT are unable to achieve a correct estimation.

We believe that this is due to our model's reduced number of parameters to be learnt, making its adjustment easier.

The linear algorithms show a low performance in the dataset **andro** (Table 2), but slightly better for the independent GP and the Cool-MTGP in spite of having to fit more parameters in the case of the MTGP model. However, Table 3 shows that the performance for this dataset is significantly increased if a nonlinear model is used, and we can therefore conclude that none of the linear models show a good performance. In this particular case, the Std-MTGP and the Cool MTGP show similar performances, suggesting that the problem is nonlinear and presents correlation between tasks, but the noise model of the standard multitask GP seems to be adequate.

5.4. Computational performance analysis

In this last section we evaluate the computational performance of some of the methods under study when executed on both a CPU
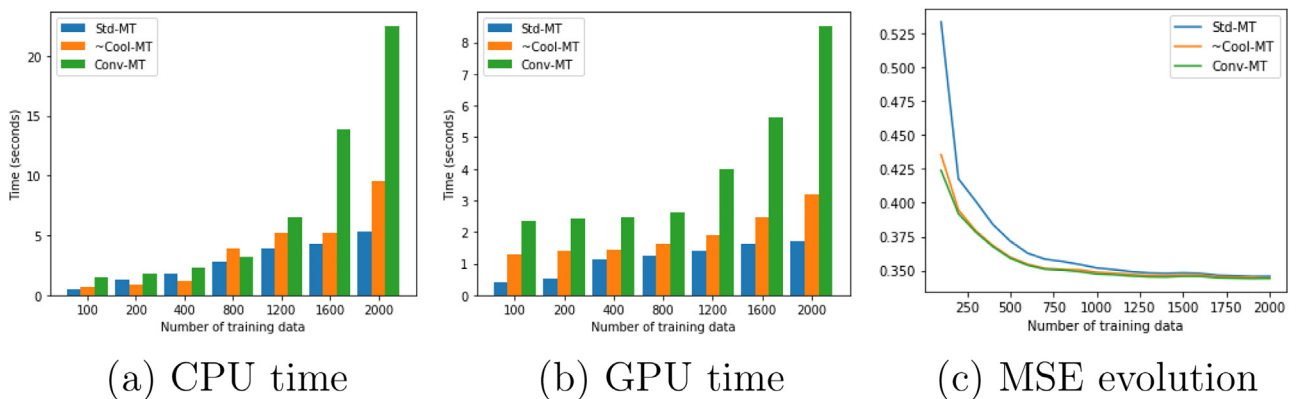

(a) CPU time   (b) GPU time   (c) MSE evolution

Fig. 7. Runtime and accuracy comparison on the scm20d dataset with 4 tasks for different numbers of training samples. Conv-MT and ~Cool-MT are competitive in MSE, but ~Cool-MT scales much better with the number of data. Std-MT offers the smallest computational burden, but has poor performance with a low number of data.

and a GPU. For this purpose, we have selected the Std-MT and Conv-MT, since they are efficiently implemented over Pytorch and TensorFlow, and we have designed a wrapper over the Pytorch GP implementation for the proposed ∼Cool-MT approach in order to run it on GPUs. We have measured the runtime and MSE performances of each algorithm with a linear kernel for different sized (*N*) training partitions of the scm20d dataset considering only 4 tasks. Computational times are averaged over 50 iterations. 50 optimization iterations were used for the Std-MT and ∼Cool-MT methods, whereas Conv-MT needed 200 to obtain accurate results. The experiment was carried out on an Intel Core i9 Processor using a single core (3.3 GHz, 98 GB RAM) and a GeForce RTX 2080Ti GPU (2944 Cuda Cores, 1.545 GHz, 10.76 GB VRAM).

Fig. 7 shows the evolution of the runtime and MSE with *N*. Conv-MT and ∼Cool-MT show similar MSE, but the computational time of Conv-MT grows much faster with the number of data. We conclude that ∼Cool-MT presents the best trade-off between accuracy and computational burden.

Notably, while Std-MT's implementation is specific to GPUs using the optimizers provided by Pytorch, for now ∼Cool-MT only uses a wrapper. Despite this, ∼Cool-MT achieves comparable performance. Additional improvements can be expected with an implementation tailored for parallelization.

6. Conclusions

In this paper we have proposed a novel solution for the MTGP problem that, compared to previous formulations, eliminates the need to validate any model hyperparameters and dramatically reduces the number of parameters to be learnt. Similarly to other existing models, this proposal assumes that an intertask and a noise covariances exist. The novelty lies in the parameter inference, which is solved through the factorization of the joint MT likelihood into a product of conditional one-output GPs. Once these parameters are learnt, with either a hierarchical or an approximate approach, a recursive algorithm can be used to recover the MT intertask and noise covariances. Experimental results show an accurate estimation of the MT intertask and noise matrices, which translates into an improved error performance. At the same time, we have integrated the model with standard GP toolboxes, showing that it is computationally competitive with the state of the art.

CRediT authorship contribution statement

Óscar García-Hinde: Data curation, Methodology, Software, Investigation, Validation, Visualization, Writing - original draft, Writing - review & editing. **Manel Martínez-Ramón:** Methodology, Software, Investigation, Writing - review & editing, Supervision, Funding acquisition. **Vanessa Gómez-Verdejo:** Methodology, Investigation, Writing - review & editing, Supervision, Funding acquisition.

Data availability

Data will be made available on request.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] C.E. Rasmussen, C.K. Williams, Gaussian process for machine learning, MIT press, 2006.

[2] J. Wiens, J. Guttag, E. Horvitz, Patient risk stratification with time-varying parameters: a multitask learning approach, The Journal of Machine Learning Research 17 (1) (2016) 2797–2819.

[3] M. Boubnovski, M. Chen, K. Linton-Reid, J. Posma, S. Copley, E. Aboagye, Development of a multi-task learning V-Net for pulmonary lobar segmentation on CT and application to diseased lungs, Clinical Radiology (2022).

[4] X. Sun, W. Xu, H. Jiang, Q. Wang, A deep multitask learning approach for air quality prediction, Annals of Operations Research 303 (1) (2021) 51–79.

[5] A. Turetskyy, J. Wessel, C. Herrmann, S. Thiede, Battery production design using multi-output machine learning models, Energy Storage Materials 38 (2021) 93–112.

[6] R. Dürichen, M.A.F. Pimentel, L. Clifton, A. Schweikard, D.A. Clifton, Multitask gaussian processes for multivariate physiological time-series analysis, IEEE Transactions on Biomedical Engineering 62 (1) (2014) 314–322.

[7] B. Shen, R. Gnanasambandam, R. Wang, and Z.J. Kong, "Multi-task Gaussian process upper confidence bound for hyperparameter tuning and its application for simulation studies of additive manufacturing," IISE Transactions, April 2022.

[8] Y. Li, T. Bao, Z. Chen, Z. Gao, X. Shu, K. Zhang, A missing sensor measurement data reconstruction framework powered by multi-task Gaussian process regression for dam structural health monitoring systems, Measurement 186 (2021) 110085.

[9] Y. Zhou, Y. Liu, D. Wang, G. De, Y. Li, X. Liu, Y. Wang, A novel combined multi-task learning and Gaussian process regression model for the prediction of multi-timescale and multi-component of solar radiation, Journal of Cleaner Production 284 (2021) 124710.

[10] S. Ketu, P.K. Mishra, Enhanced Gaussian process regression-based forecasting model for COVID-19 outbreak and significance of iot for its detection, Applied Intelligence 51 (3) (2021) 1492–1512.

[11] D. Xu, Y. Shi, I.W. Tsang, Y.-S. Ong, C. Gong, X. Shen, Survey on multi-output learning, IEEE Transactions on Neural Networks and Learning Systems 31 (7) (2020) 2409–2429.

[12] E. Adiyeke, M.G. Baydoğan, The benefits of target relations: A comparison of multitask extensions and classifier chains, Pattern Recognition 107 (2020) 107507.

[13] E.V. Bonilla, K.M. Chai, C. Williams, Multi-task Gaussian process prediction, in: J.C. Platt, D. Koller, Y. Singer, S.T. Roweis (Eds.), Advances in Neural Information Processing Systems 20, Curran Associates Inc, 2008, pp. 153–160.

[14] A.M. Schmidt and A.E. Gelfand, "A Bayesian coregionalization approach for multivariate pollutant data," Journal of Geophysical Research: Atmospheres, vol. 108, no. D24, 2003.

[15] T.R. Fanshawe and P.J. Diggle, "Bivariate geostatistical modelling: a review and an application to spatial variation in radon concentrations," Environmental and ecological statistics, vol. 19, no. 2, pp. 139–160, 2012.

[16] P. Goovaerts et al., Geostatistics for natural resources evaluation, Oxford University Press, 1997.

[17] O. Stegle, C. Lippert, J.M. Mooij, N.D. Lawrence, and K. Borgwardt, "Efficient inference in matrix-variate Gaussian models with iid observation noise," in Advances in neural information processing systems, pp. 630–638, 2011.

[18] B. Rakitsch, C. Lippert, K. Borgwardt, O. Stegle, It is all in the noise: Efficient multi-task Gaussian process inference with structured residuals, Advances in Neural Information Processing Systems 26 (2013) 1466–1474.

[19] H.K. Lee, C.H. Holloman, C.A. Calder, and D.M. Higdon, "Flexible Gaussian processes via convolution," Duke University, 2002.

[20] P. Boyle, M. Frean, Dependent Gaussian processes, Advances in Neural Information Processing Systems 17 (2005) 217–224.

[21] M. Alvarez, N.D. Lawrence, Sparse convolved Gaussian processes for multi-output regression, Advances in Neural Information Processing Systems 21 (2009) 57–64.

[22] M.A. Álvarez, N.D. Lawrence, Computationally efficient convolved multiple output Gaussian processes, Journal of Machine Learning Research 12 (May) (2011) 1459–1500.

[23] K. Chen, T. van Laarhoven, E. Marchiori, F. Yin, S. Cui, Multitask Gaussian process with hierarchical latent interactions, in: ICASSP 2022–2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2022, pp. 4148–4152.

[24] O. Hamelijnck, T. Damoulas, K. Wang, and M. Girolami, "Multi-resolution multi-task Gaussian processes," in Advances in Neural Information Processing Systems, vol. 32, Curran Associates Inc, 2019.

[25] M. Nabati, S.A. Ghorashi, R. Shahbazian, JGPR: a computationally efficient multi-target Gaussian process regression algorithm, in: Machine Learning, 2022, pp. 1–24.

[26] J. Gardner, G. Pleiss, K.Q. Weinberger, D. Bindel, and A.G. Wilson, "Gpytorch: Blackbox matrix-matrix Gaussian process inference with gpu acceleration," in Advances in Neural Information Processing Systems, pp. 7576–7586, 2018.

[27] T. de Wolff, A. Cuevas, F. Tobar, MOGPTK: The multi-output Gaussian process toolkit, Neurocomputing 424 (2021) 49–53.

[28] J. Shawe-Taylor, N. Cristianini, Kernel Methods for Pattern Analysis, Cambridge University Press, Cambridge, UK, 2004.

[29] C.M. Bishop, Pattern recognition and machine learning, ch. 2. Probability Distributions. Springer, 2006.

[30] A.G. d. G. Matthews, M. van der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, and J. Hensman, "GPflow: A Gaussian process library using TensorFlow," Journal of Machine Learning Research, vol. 18, pp. 1–6, Apr 2017.

[31] E. Spyromitros-Xioufis, G. Tsoumakas, W. Groves, I. Vlahavas, Multi-target regression via input space expansion: treating targets as inputs, Machine Learning 104 (1) (2016) 55–98.

Óscar **García-Hinde** was born in Madrid in 1983. He received his degree in Telecommunications Engineering in 2014 from the Carlos III University of Madrid, where he also obtained his masters degree in 2016. He finished his PhD with honors on March 2022 in the department of Signal Theory and Communications at Carlos III University.

Manel Martínez Ramón is a professor with the ECE department of University of New Mexico (UNM). He holds the King Felipe VI Endowed Chair (UNM). He is a Telecommunications Engineer (Universitat Politècnica de Catalunya, Spain, 1996) and PhD in Communications Technologies (Universidad Carlos III de Madrid, Spain, 1999).

Vanessa Gómez-Verdejo was born in Madrid in 1979. She received the telecommunication engineering degree from the Universidad Politécnica de Madrid (2002) and the Ph.D.degree from the Universidad Carlos III de Madrid (2007). She is currently Associate Professor in the Department of Signal Theory and Communications, Universidad Carlos III de Madrid.