

Gaussian processes for missing value imputation

Bahram Jafrasteh^{a,*}, Daniel Hernández-Lobato^b, Simón Pedro Lubián-López^{a,c},
Isabel Benavente-Fernández^{a,c,d}

^a Biomedical Research and Innovation Institute of Cádiz (INIBICA) Research Unit, Puerta del Mar University, Cádiz, Spain

^b Computer Science Department, Universidad Autónoma de Madrid, Madrid, Spain

^c Division of Neonatology, Department of Pediatrics, Puerta del Mar University Hospital, Cádiz, Spain

^d Area of Pediatrics, Department of Child and Mother Health and Radiology, Medical School, University of Cádiz, Cádiz, Spain

ARTICLE INFO

Article history:

Received 13 December 2022

Received in revised form 10 March 2023

Accepted 26 April 2023

Available online 29 April 2023

Keywords:

Missing values

Gaussian process

Deep learning

Deep Gaussian processes

Variational inference

ABSTRACT

A missing value indicates that a particular attribute of an instance of a learning problem is not recorded. They are very common in many real-life datasets. In spite of this, however, most machine learning methods cannot handle missing values. Thus, they should be imputed before training. Gaussian Processes (GPs) are non-parametric models with accurate uncertainty estimates that combined with sparse approximations and stochastic variational inference scale to large data sets. Sparse GPs (SGPs) can be used to get a predictive distribution for missing values. We present a hierarchical composition of sparse GPs that is used to predict the missing values at each dimension using the observed values from the other dimensions. Importantly, we consider that the input attributes to each sparse GP used for prediction may also have missing values. The missing values in those input attributes are replaced by the predictions of the previous sparse GPs in the hierarchy. We call our approach missing GP (MGP). MGP can impute all observed missing values. It outputs a predictive distribution for each missing value that is then used in the imputation of other missing values. We evaluate MGP on one private clinical data set and on four UCI datasets with a different percentage of missing values. Furthermore, we compare the performance of MGP with other state-of-the-art methods for imputing missing values, including variants based on sparse GPs and deep GPs. Our results show that the performance of MGP is significantly better.

© 2023 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

1. Introduction

Data in real-life sciences are often noisy, stored in databases, and contain missing values. This is particularly the case of clinical data. Specifically, our work is motivated by a clinical data set of newly born premature infants. This dataset has different variables that are related to various diagnoses at the current and previous states of life. It contains, as measurements, among other variables, the total brain volume estimated using ultrasound images. Often, for each infant, it is not possible to measure and store all variables considered in the dataset. Therefore, there are several missing values associated to some instances. Having missing values is also very common in datasets from other domains. Some examples include those related to laboratory measurements or shopping platforms [1].

Machine learning models can learn the underlying data structure. Moreover, they can also capture high dimensional and complex relationships between different variables, and hence, estimate the true distribution of the data to make predictions on previously unseen data [2]. They allow us to make a better decision, create high-quality clusters, detect outliers in the data, and make more accurate predictions of a parameter of interest. Nevertheless, machine learning methods for regression, classification and clustering most of the times do not account for missing values in the observed data. A simple solution to still be able to use the aforementioned methods is to systematically ignore data instances with missing values. Ignoring the impact of missing values can, however, lead to sub-optimal models without good enough generalization performance. Thus, taking into account missing values and not just ignoring data instances with missing values is a critical step in a machine learning method.

To be able to take into account missing values when fitting a machine learning method it is important to know what type of missing values one can find in practical applications. Specifically, there are three kinds of missing value mechanisms described in the literature:

* Corresponding author.

E-mail addresses: jafrasteh.bahram@inibica.es (B. Jafrasteh), daniel.hernandez@uam.es (D. Hernández-Lobato), simonp.lubian.sspa@juntadeandalucia.es (S.P. Lubián-López), isabel.benavente@uca.es (I. Benavente-Fernández).

- Missing completely at random (MCAR): The missingness mechanism is not related to any observed value or unobserved values from the dataset. Therefore missing values appear completely at random on the observed instances.
- Missing at random (MAR): Having missing values for one variable is related to the value of some other observed variable or variables from the dataset. For example, men are more likely to tell their weight and women are less likely. Therefore, the missing value mechanism for weight is explained by gender.
- Missing not at random (MNAR): The missing instances are related to some values of the same attribute. For example, if an attribute considers the level of education, people may be ashamed of answering that they have the lowest education level and they may not fill that information.

It is common to assume the missing mechanism as MAR and impute missing values using traditional methods or machine learning algorithms. The imputation of missing values plays a key role in the final model's performance since the chosen algorithm for this task directly impacts the resulting model. Removing instances with missing values from a dataset and training a model with all the remaining data is considered a minimal and simple approach that is expected to result in a suboptimal performance. Moreover such a model will not be able to consider new instances for prediction with missing values.

Most machine learning methods for regression, classification, and/or clustering inherently cannot deal with missing values. Hence, it is needed to provide a way to impute this missing data and/or change the machine learning method. The simplest approach is to impute the data with their mean/median values across data instances. However, several studies show that these approaches are not sufficient. The generated model can still be sub-optimal [3,4]. More sophisticated methods should be used to find the true distribution of missing values and impute them.

Recovering latent values associated to missing values can help the final decision makers to improve their predictions. Moreover, it can be useful to better understand the dependence of the target variable to predict on the explaining attributes. The data distribution of missing values can be extracted using a predictor for the missing value and the corresponding associated prediction uncertainty. This prediction uncertainty will be able to capture what are the potential values that a missing observation may have had. This uncertainty is expected to be important, since there is evidence that incorporating input noise in a final model can improve overall prediction performance [5].

A well-known non-parametric machine learning method with a probabilistic nature is a Gaussian process (GP) [6]. A GP can output a predictive distribution for the target variable that takes into account prediction uncertainty. This uncertainty arises from intrinsic noise in the data and also because of the fact that we have a finite amount of training data. A problem of GPs is, however, that given N observation points, the training process requires the inversion of an $N \times N$ covariance matrix, which is very expensive for a large N . Therefore, GPs become infeasible as the number of training instances increases. Approximate techniques must be used for the computation of the predictive distribution.

One of the most popular approximations methods to deal with the scalability of GPs is based on sparse inducing points representations [7,8]. In the sparse variational Gaussian process (SVGP) $M \ll N$ inducing points are optimized alongside with other hyper-parameters using variational inference [9,10]. Using this approximation, GPs can scale to very large datasets, with millions of instances, by a combination of sparse approximations and stochastic optimization techniques [11].

A concatenation of GPs corresponds to a deep GP (DGP). DGPs have been proposed to improve the performance of single-layer

GPs, similarly to what happened in the case of multi-layer neural networks [12–14]. DGPs overcome some limitations of the single layer sparse GPs, such as the reduced expressiveness of the kernel/covariance function. However, they still enjoy the nice properties of GPs such as the output of a predictive distribution and the scalability to large datasets [14]. DGPs and SVGPs can be used with multiple input and multiple outputs to learn the latent representation of the data and recover the data distribution. However, DGPs do not consider sequential relations between a set of variables in the dataset. Recurrent GPs have been introduced in [15] for sequential data sets.

In this work we are inspired by the DGP architecture [14] and the recurrent GP to develop a new method of imputing missing values. The method is a hierarchical composition of GPs, where there is a GP per dimension that predicts the missing values for that dimension using all the variables from the other dimensions. Of course, for this to work, an ordering on the dimensions has to be specified and also an initial value for the missing values.

Our method starts with the dimension that has the largest standard deviation (before standardizing the data). The missing values of that dimension are predicted using a GP that receives as an input all the other dimensions. Since these other dimensions will be used for prediction, their missing values are simply estimated initially using the mean value across the corresponding dimension. After this, a second GP is used to predict the missing values of the dimension with the second largest standard deviation (before standardizing the data). This second GP also receives all the other dimensions as an input. Importantly, however, this second GP receives as an input for the missing values corresponding to the dimension with the largest standard deviation (before standardizing the data) the predictions given by the first GP. This process is then repeated iteratively for the total number of dimensions with missing values, using the predictions of the previous GPs for the attributes with missing values.

Given the imputed missing values by the process just described and the observed data, we then have a last GP that predicts the target variable. That last GP receives as an input the observed data and the predictions given by the GPs in charge of imputing the missing values. Therefore, with the process described, all the missing values have an associated predictive distribution which is taken into account by the last GP for prediction. Critically, all the GPs are trained at the same time.

We have validated the method described, called missing GP (MGP), using one private clinical dataset and four datasets extracted from UCI repository [16]. The private dataset is provided by the “perinatal brain damage” group at the Biomedical Research and Innovation Institute of Cádiz (INiBICA) and the Puerta del Mar University Hospital, Cádiz, Spain.

Summing up, the contributions of this work are:

- A new method based on concatenated GPs is introduced for imputing missing values.
- The method outputs a predictive distribution for each missing value in the dataset. This can be used for missing value imputation.
- The final model can be trained simultaneously and can be scaled to large data sets.

The rest of the manuscript is organized as follows: In Section 3, we briefly describe the Gaussian processes, DGPs, and then we explain the proposed method. Section 2 introduces important related work from the literature. The configuration of the experiments and the datasets are explained in Section 4. In Section 5, we discuss the obtained results and, finally, Section 6 presents the conclusions.

2. Related work

Following a common practice in the literature about missing data imputation, we focus here on missing completely at random and describe related works.

A Gaussian mixture model (GMM) trained with the expectation-maximization algorithm has been proposed to impute missing values based on the acquired class [17,18]. Similarly, K-nearest neighbors (KNN) [19] has also been proposed to impute missing values. This method does not rely on a predictive model for the missing variable. However, its estimation accuracy is strongly affected by the number of neighbors. Self-organizing maps (SOM) [20] has also been used for data correction and imputation for continuous and categorical data. These techniques, i.e., GMM, KNN and SOM, do not require an iterative process to impute the missing values, unlike our method MGP. However, their performance is expected to be limited for the same reason. Specifically, the iterative process that MGP follows can use partial imputed information corresponding to some missing values to predict other missing values.

Multiple imputations using chained equations (MICE) [21] is another state-of-the-art algorithm that uses linear regression models to impute missing values. It considers variables with missing values as the dependent variable for each model, as in MGP. Initial missing values are also imputed using the mean value. We compare results in our experiments with this technique showing improved results for MGP. We believe this may be due to the extra flexibility of GPs for missing value imputation, compared to the linear regression models used by MICE.

Recently, more sophisticated methods such as Auto-encoders (AE) [22,23], variational AE (VAE) [24], and heterogeneous-incomplete VAE [25] have been proposed to impute missing values. In general, AE based methods use neural networks to impute missing values. Generative adversarial network (GAIN) for missing data imputation [26] is another method based on neural networks. In GAIN, a generator neural network is used to generate new values for missing values. Similarly, a discriminator neural network is used for training the discriminator efficiently. We compare our method, MGP, with GAIN and VAE [27] imputation showing improved results. We believe the reason for this is that GAIN is expected to perform well in big datasets. By contrast, a GP based approach is expected to perform better in a small data regime. The reason for this is the robustness of the GP predictive distribution that incorporates uncertainty estimation about the predictions made.

Deep Belief Networks (DBNs) have been used in [28] to impute missing values. DBNs fall in the deep learning methods category. They are composed of multiple layers where the connections are between layers but not between units within each layer. More precisely, the layers of the network contain restricted Boltzmann machines (RBMs). The output of each layer serves as the input for the next layer. The lowest layer is the training set. MGP and DBN can have different number of layers. However, the number of layers in MGP is fixed and specified by the number of the missing features, while in a DBN the optimal number of layers should be determined by methods such as K-fold cross-validation. Moreover, one can add a final regression layer to MGP and make predictions of the target variable. In MGP all the parameters can be learned at the same time. However, in a DBN one needs to follow a two steps approach for training. First, the DBN should be trained to reconstruct the inputs. Then, it can be used for further prediction tasks and missing value imputation. To conclude, a DBN is similar to the method GAIN, and is expected to have a better performance on bigger datasets. By contrast, our method, i.e., MGP, is expected to be more robust on smaller datasets.

There are a few studies on using GP based methods for imputing missing values. In particular, [29] proposes a combination of

GP and VAE for imputing missing values. According to our knowledge, however, there is no study on imputing missing values using deep GPs [14] nor SVGP [11]. The proposed model from [29] uses a GP evaluated on the latent space inferred by a VAE to model time series and impute missing values. The model is exclusively working on multi-variate time series data. Moreover, it has a fixed GP kernel which cannot benefit from joint optimization. In our work, we use a network of SVGPs that resembles a deep GP to impute missing values after mean pre-imputation of missing values. MGP learns from the observed value of each attribute and, similar to what happens in MICE, it uses previously imputed missing values for this task.

3. Gaussian processes for missing data

This section gives a briefly introduction to Gaussian process (GPs) and Deep GPs (DGPs). It provides the needed background to correctly explain the proposed method for learning under missing values using GPs. We call such a method missing GP (MGP).

3.1. Gaussian processes

A Gaussian process (GP) is a stochastic process whose values at any finite set of points follow a multi-variate Gaussian distribution [30]. From a machine learning perspective, a GP is used as a prior over a latent function f , where the posterior of that function, computed given the observed data, is another GP. This results in a non-parametric machine learning method whose level of expressiveness grows with the dataset size. Consider a set of points $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ and $y_i = f(\mathbf{x}_i) + \epsilon_i$, where ϵ_i is a Gaussian noise with variance σ^2 . A GP prior for f is typically specified by a mean function $m(\mathbf{x})$ and a covariance function $k_\theta(\mathbf{x}, \mathbf{x}')$ with trainable parameters θ . Assuming a zero mean function, given a dataset \mathcal{D} , the predictive distribution for the value of f, f^* , at a new test point \mathbf{x}^* is Gaussian. Namely,

$$p(f^*|\mathcal{D}) = \mathcal{N}(\mu(\mathbf{x}^*), \sigma^2(\mathbf{x}^*)), \quad (1)$$

with mean and variance given by

$$\mu(\mathbf{x}^*) = \mathbf{k}(\mathbf{x}^*)^T (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{y}, \quad (2)$$

$$\sigma^2(\mathbf{x}^*) = k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}(\mathbf{x}^*)^T (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{k}(\mathbf{x}^*), \quad (3)$$

where $\mu(\mathbf{x}^*)$ and $\sigma^2(\mathbf{x}^*)$ denotes the predictive mean and variance, respectively. $\mathbf{k}(\mathbf{x}^*)$ is a vector with the covariances between $f(\mathbf{x}^*)$ and each $f(\mathbf{x}_i)$, simply given by $k(\mathbf{x}^*, \mathbf{x}_i)$, with $k(\cdot, \cdot)$ the covariance function. \mathbf{K} is a $N \times N$ matrix with the covariances between each $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$ in the training set. That is, $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. \mathbf{I} stands for the identity matrix.

Learning the GP hyper-parameters θ can be done by maximizing the marginal likelihood of the model. Namely, $p(\mathbf{y}|\theta)$, which is Gaussian [30]. It is possible to show that the marginal likelihood penalizes models that are either too simple or too complicated to explain the observed data [2].

Importantly, GPs are unsuitable for large datasets as they need the inversion of matrix \mathbf{K} , with a computational complexity in $O(N^3)$. However, one can use sparse GPs to overcome this problem. Sparse GPs are explained in the next section.

3.2. Sparse Gaussian processes

One can reduce the computational cost of GPs by introducing a set of $M \ll N$ additional data instances called inducing points $\mathbf{Z} = (\mathbf{z}_1, \dots, \mathbf{z}_M)^T$ [8]. The inducing points are in the same space as each \mathbf{x}_i . Let $\mathbf{u} = (f(\mathbf{z}_1), \dots, f(\mathbf{z}_M))^T$ be the process values at the inducing points and $\mathbf{f} = (f(\mathbf{x}_1), \dots, f(\mathbf{x}_N))^T$ be the process values at the training data. As a consequence of using a GP with

mean function $m(\cdot)$, $p(\mathbf{u}) \sim \mathcal{N}(\mathbf{m}(\mathbf{Z}), \mathbf{K}(\mathbf{Z}, \mathbf{Z}))$, with $\mathbf{K}(\mathbf{Z}, \mathbf{Z})$ the covariance matrix that results from evaluating the covariance function $k(\cdot, \cdot)$ on the inducing points and $\mathbf{m}(\mathbf{Z})$ the vector of the prior GP means for each $f(\mathbf{z}_j)$.

Consider now the use of variational inference (VI) to find an approximate posterior for \mathbf{f} and \mathbf{u} given the observed data [7]. Specifically, the goal is to find an approximate posterior $q(\mathbf{f}, \mathbf{u})$ which resembles to true posterior $p(\mathbf{f}, \mathbf{u}|\mathbf{y})$. Following, [7,31] we can specify a constrained form for q . Namely,

$$q(\mathbf{f}, \mathbf{u}) = p(\mathbf{f}|\mathbf{u})q(\mathbf{u}), \quad (4)$$

where the first factor is fixed and given by the GP predictive distribution, and the second factor is a multi-variate Gaussian $q(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\mathbf{r}, \mathbf{S})$, where \mathbf{r} is the mean and \mathbf{S} is the covariance matrix. These are tunable parameters that specify the shape of the posterior approximation $q(\mathbf{f}, \mathbf{u})$.

One can marginalize out \mathbf{u} in order to compute the mean and variances of the predictive distribution at the inputs. That is, $q(f(\mathbf{x}_i))$ is Gaussian with parameters

$$\mu_{\mathbf{r}, \mathbf{Z}}(\mathbf{x}_i) = m(\mathbf{x}_i) + \alpha(\mathbf{x}_i)^T(\mathbf{r} - m(\mathbf{Z})), \quad (5)$$

$$\sigma_{\mathbf{S}, \mathbf{Z}}^2(\mathbf{x}_i, \mathbf{x}_i) = k(\mathbf{x}_i, \mathbf{x}_i) - \alpha(\mathbf{x}_i)^T(\mathbf{K}(\mathbf{Z}, \mathbf{Z}) - \mathbf{S})\alpha(\mathbf{x}_i), \quad (6)$$

where $\alpha(\mathbf{x}_i) = \mathbf{K}(\mathbf{Z}, \mathbf{Z})^{-1}\mathbf{k}(\mathbf{Z}, \mathbf{x}_i)$, with $\mathbf{k}(\mathbf{Z}, \mathbf{x}_i)$ the vector that results from evaluating $k(\mathbf{z}_j, \mathbf{x}_i)$ for $j = 1, \dots, M$.

The variational parameters \mathbf{Z} , \mathbf{r} , \mathbf{S} and hyper-parameters θ are optimized by maximizing the evidence lower bound \mathcal{L} (ELBO) on the log-marginal likelihood, as described in [7,31]. This results in the sparse-variational GP model (SVGP). Namely,

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{f}, \mathbf{u})}[\log \frac{p(\mathbf{y}, \mathbf{f}, \mathbf{u})}{q(\mathbf{f}, \mathbf{u})}], \quad (7)$$

where $p(\mathbf{y}, \mathbf{f}, \mathbf{u}) = \prod_{i=1}^N p(y_i|f_i)p(\mathbf{f}|\mathbf{u})p(\mathbf{u})$. In this last expression, the first factors correspond to the likelihood and the other two factors represent the GP prior on \mathbf{f} and \mathbf{u} . After some simplifications, and some factors cancellation, the lower bound is computed as follows

$$\mathcal{L} = \sum_{i=1}^N \mathbb{E}_{q(f_i)}[\log p(y_i|f_i)] - \text{KL}[q(\mathbf{u})|p(\mathbf{u})], \quad (8)$$

where KL stands for the Kullback–Leibler divergence between the distributions $q(\mathbf{u})$ and $p(\mathbf{u})$, and $f_i = f(\mathbf{x}_i)$. Since both distributions are Gaussian, we can analytically compute the KL value. In the case of regression, where a Gaussian likelihood is used, the expectation has a closed form and there is no need to use extra approximation methods. The evaluation of (8) can be done with a cost that is in $\mathcal{O}(NM^2)$. Critically, the objective function, i.e., the lower-bound \mathcal{L} , involves a sum over training instances and hence, can be combined with mini-batch sampling and stochastic optimization techniques for inference on large datasets [31], resulting in a cost in $\mathcal{O}(M^3)$.

Instead of a one dimensional output $y_i \in \mathbb{R}$, one can consider D -dimensional outputs. Namely, $\mathbf{y}_i \in \mathbb{R}^D$. These problems can be addressed by considering D independent GPs. The GP prior is changed to a factorizing prior of D GPs. Namely, $\prod_{d=1}^D p(\mathbf{f}^d|\mathbf{u}^d)p(\mathbf{u}^d) = p(\mathbf{F}|\mathbf{U})p(\mathbf{U})$, with \mathbf{f}^d and \mathbf{u}^d being the d sparse GP process values at the training points and the inducing points, respectively. Therefore, $\mathbf{F} = (\mathbf{f}^1, \dots, \mathbf{f}^D)$ and $\mathbf{U} = (\mathbf{u}^1, \dots, \mathbf{u}^D)$. Moreover, we can assume that the inducing points \mathbf{Z} are shared across each of the D different sparse GPs. The joint distribution of all the variables can be rewritten

$$p(\mathbf{Y}, \mathbf{F}, \mathbf{U}) = \prod_{i=1}^N p(\mathbf{y}_i|\mathbf{f}_i)p(\mathbf{F}|\mathbf{U})p(\mathbf{U}), \quad (9)$$

where \mathbf{f}_i is the i th row of \mathbf{F} , a D dimensional vector with the values of each of the D latent functions at \mathbf{x}_i . One can also consider a

similar approximate distribution q . Namely, $q(\mathbf{F}, \mathbf{U}) = p(\mathbf{F}|\mathbf{U})q(\mathbf{U})$. Then, the ELBO is

$$\begin{aligned} \mathcal{L} &= \sum_{i=1}^N \mathbb{E}_{q(\mathbf{f}_i)}[\log p(\mathbf{y}_i|\mathbf{f}_i)] - \text{KL}[q(\mathbf{U})|p(\mathbf{U})] \\ &= \sum_{i=1}^N \mathbb{E}_{q(\mathbf{f}_i)}[\log p(\mathbf{y}_i|\mathbf{f}_i)] - \sum_{d=1}^D \text{KL}[q(\mathbf{u}^d)|p(\mathbf{u}^d)]. \end{aligned} \quad (10)$$

Note that the method described can be used to map all input attributes to themselves for missing value imputation. In this case, $\mathbf{Y} = \mathbf{X}$. Of course, for this to work one needs to have an initial guess for the missing values so that they can be considered as input attributes of each latent function. Missing values can be initially estimated using a simple technique such as mean imputation. After learning the latent representation of the inputs, the missing values can be predicted using the predictive distribution of the method described.

3.3. Deep Gaussian processes

Deep Gaussian processes (DGPs) [12,13] are a concatenation of independent GPs. Namely, the GPs at layer l receive as an input the output of the GPs at layer $l-1$, in the same spirit as a deep neural network, but where each unit in a hidden layer is a GP. Consider a DGP of L layers with H units or GPs in each layer. Fig. 1 illustrates this architecture. Let $\mathbf{F}^{(l)}$ be the function values associated to the input points in layer l . That is, $\mathbf{F}^{(l)}$ is a matrix of size $N \times H$. For computational reasons, sparse GPs based on inducing points are used instead of standard GPs in each layer. Thus, each layer l has inducing points $\mathbf{Z}^{(l)}$, a noisy inputs $\mathbf{F}^{(l-1)}$ received from the previous layer. Note that here we assume shared inducing points for the GPs of each layer. The inducing points values of layer l are denoted by $\mathbf{U}^{(l)}$, a $M \times H$ matrix. Given a DGP, the joint distribution of all the variables in the model is

$$\begin{aligned} p(\mathbf{y}, \{\mathbf{F}^{(l)}, \mathbf{U}^{(l)}\}_{l=1}^L) &= \prod_{l=1}^L p(\mathbf{F}^{(l)}|\mathbf{U}^{(l)}, \mathbf{F}^{(l-1)}, \mathbf{Z}^{(l)})p(\mathbf{U}^{(l)}|\mathbf{Z}^{(l)}) \times \\ &\times \prod_{i=1}^N p(y_i|f_i^1) \end{aligned} \quad (11)$$

where the inputs to the first layer are the observed data instances \mathbf{X} and f_i^1 is the corresponding function value associated to the DGP at the last layer for instance \mathbf{x}_i . Moreover, $p(\mathbf{F}^{(l)}|\mathbf{U}^{(l)}, \mathbf{F}^{(l-1)}, \mathbf{Z}^{(l)})$ is given by each GP predictive distribution at layer l , as in the single-layer sparse GP described in the previous section. Since exact inference in the model is not tractable, approximate inference has to be used. The work in [14] introduces a method based on variational inference and the following form for the posterior approximation

$$q(\{\mathbf{F}^{(l)}, \mathbf{U}^{(l)}\}_{l=1}^L) = \prod_{l=1}^L p(\mathbf{F}^{(l)}|\mathbf{U}^{(l)}, \mathbf{F}^{(l-1)}, \mathbf{Z}^{(l)})q(\mathbf{U}^{(l)}), \quad (12)$$

where $p(\mathbf{F}^{(l)}|\mathbf{U}^{(l)}, \mathbf{F}^{(l-1)}, \mathbf{Z}^{(l)})$ and $q(\mathbf{U}^{(l)})$ factorize across units in a layer as in the single-layer sparse GP described in the previous section. Moreover, $p(\mathbf{F}^{(l)}|\mathbf{U}^{(l)}, \mathbf{F}^{(l-1)}, \mathbf{Z}^{(l)})$ is fixed and given by the product of each GP predictive distribution and $q(\mathbf{U}^{(l)})$ is a product of multi-variate Gaussian distributions whose means and covariance matrices can be adjusted. After marginalizing out \mathbf{U} from each layer, the posterior distribution is a product of Gaussian

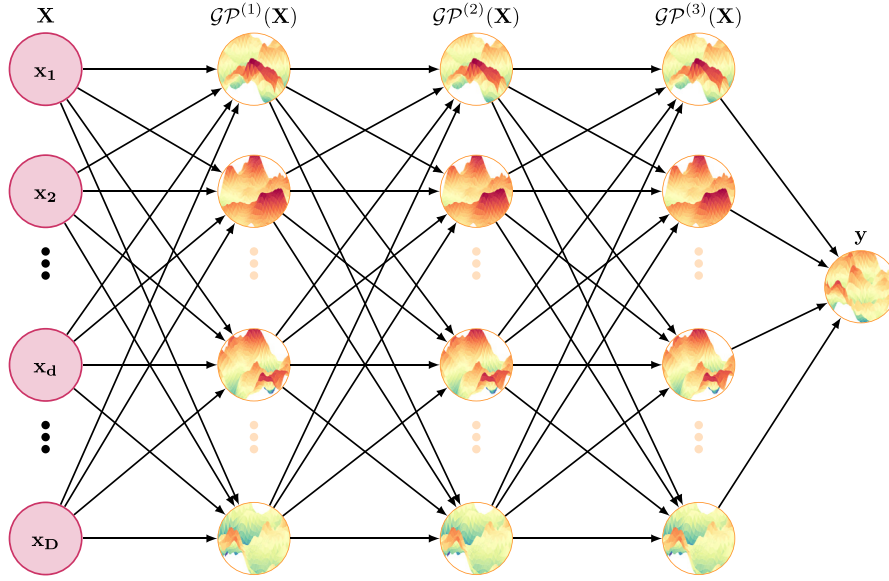


Fig. 1. Three layer Deep Gaussian process.

distributions

$$q(\{\mathbf{F}^{(l)}\}_{l=1}^L) = \prod_{l=1}^L q(\mathbf{F}^{(l)} | \mathbf{r}^{(l)}, \mathbf{S}^{(l)}, \mathbf{F}^{(l-1)}, \mathbf{Z}^{(l)})$$

$$= \prod_{l=1}^L \mathcal{N}(\mathbf{F}^{(l)} | \boldsymbol{\mu}^{(l)}, \boldsymbol{\Sigma}^{(l)}). \quad (13)$$

where the mean and variance of each marginal Gaussian distribution are computed as in (5) and (6). For each sample i and unit h at layer l the mean is $\mu_{i,h}^{(l)} = \mu_{\mathbf{r}_h^{(l)}, \mathbf{Z}^{(l)}}(\mathbf{f}_i^{(l-1)})$ and the variance is $(\Sigma_{i,h}^{(l)}) = \sigma_{\mathbf{r}_h^{(l)}, \mathbf{Z}^{(l)}}^2(\mathbf{f}_i^{(l-1)}, \mathbf{f}_i^{(l-1)})$, where $\mathbf{f}_i^{(l-1)}$ is the i th row of $\mathbf{F}^{(l-1)}$. Consider the joint distribution in (11) and the approximate posterior q in (12). Let them be put together into (7). The ELBO of a DGP is

$$\mathcal{L}_{DGP} = \sum_{i=1}^N \mathbb{E}_q[\log p(y_i | \mathbf{f}_i^L)]$$

$$- \sum_{l=1}^L \text{KL}[q(\mathbf{U}^{(l)}) | p(\mathbf{U}^{(l)} | \mathbf{Z}^{(l)})], \quad (14)$$

where \mathbf{f}_i^L are the latent functions of the last layer associated to the input \mathbf{x}_i . Critically, $\mathbb{E}_q[\log p(y_i | \mathbf{f}_i^L)]$ is intractable and requires a Monte Carlo approximation. Specifically, samples from q for \mathbf{f}_i^L can be obtained by propagating inputs through the DGP network and iteratively sampling from the predictive distribution of each layer [14]. The resulting approximation can be combined with stochastic optimization techniques that maximize (14) and train the model [14]. The training cost is in $\mathcal{O}(LHM^3)$, with L the number of layers, H the number of units in each layer and M the number of inducing points considered.

In a DGP, the predictive distributions of layer l for the output associated to \mathbf{x}_i , denoted with $\hat{\mathbf{f}}_i^l$, depends on the output of the previous layer $\hat{\mathbf{f}}_i^{(l-1)}$. Let $f_{i,h}^l$ be the output of unit h at layer l for the data instance \mathbf{x}_i . Using this property, one can use the reparameterization trick [32,33] to recursively sample $\hat{f}_{i,h}^{(l)} \sim q(f_{i,h}^{(l)} | \mathbf{r}_h^{(l)}, \mathbf{S}_h^{(l)}, \hat{\mathbf{f}}_i^{(l-1)}, \mathbf{Z}^{(l)})$ with

$$\hat{f}_{i,h}^{(l)} = \mu_{\mathbf{r}_h^{(l)}, \mathbf{Z}^{(l)}}(\hat{\mathbf{f}}_i^{(l-1)}) + \epsilon_{i,h}^{(l)} \sqrt{\sigma_{\mathbf{r}_h^{(l)}, \mathbf{Z}^{(l)}}^2(\hat{\mathbf{f}}_i^{(l-1)}, \hat{\mathbf{f}}_i^{(l-1)})} \quad (15)$$

where $\mathbf{f}_i^{(0)} = \mathbf{x}_i$ and $\epsilon_{i,h}^{(l)} \sim \mathcal{N}(0, 1)$.

The prediction for a test point is made by drawing K samples and propagating them across the DGP network until the L th layer using (15). We denote $f_*^{(L)}$ as the latent function value at a new test location \mathbf{x}_* in the last layer, L . The approximate predictive distribution for $f_*^{(L)}$ is

$$q(\mathbf{f}_*^{(L)}) \approx \frac{1}{K} \sum_{k=1}^K q(\mathbf{f}_*^{(L)} | \mathbf{r}^{(L)}, \mathbf{S}^{(L)}, \hat{\mathbf{f}}_*^{(L-1),k}, \mathbf{Z}^{(L)}) \quad (16)$$

where $\hat{\mathbf{f}}_*^{(L-1),k}$ denotes the k th sample from layer $L-1$.

Importantly, the formulation in (14) also allows for mini-batch sampling to train the model, which enables scaling to very large datasets. The predictive distribution for $y_* \in \mathbb{R}$ can be easily obtained in the case of a Gaussian likelihood. One only has to incorporate the variance of the additive Gaussian noise in (16).

3.4. Missing Gaussian process

In this subsection we introduce our method, missing Gaussian process (MGP), to impute missing values. In many practical problems, the imputed value from one dimension highly depends on the values from all other dimensions as it has been observed in [21]. Specifically, chained equation algorithms have been very successful in imputing missing values, as explained in the related work section. Here, we propose a new version of these algorithms based on chained Gaussian processes. The idea of MGP is inspired from DGPs [14] and recurrent Gaussian processes [15], where the output of each GP depends on a previous GP.

Let us denote the D -dimensional input matrix of observed data with $\hat{\mathbf{X}} = (\hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_N)^T$, where $\hat{\mathbf{x}}_i$ is the i th sample that randomly has some missing values, which have been initially imputed with the mean of the observed values at each dimension. The total attributes containing missing values are denoted by D_m . We sort these variables according to their standard deviations (before standardizing the data), from lowest to highest. For example, the attribute with the smallest standard deviation becomes the first attribute, followed by the attribute with the second smallest standard deviation, etc. The attributes without missing values are left as the last ones in the ordering. The ordering of these attributes is not important. Later, we show that the ordering of the missing attributes also does not impact the results of MGP and the ordering can be considered as a suggestion.

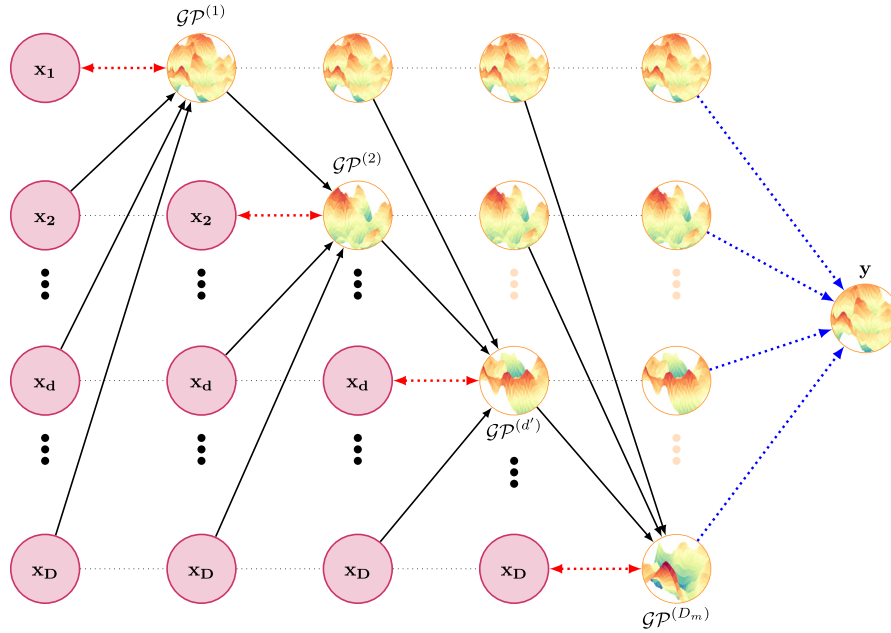


Fig. 2. Missing GP architecture. It consists of D_m GPs, where D_m is the number of missing dimensions in the data. Each GP is used to impute the missing value in the dimension d' . The input of $GP^{(d')}$ are all other dimensions excluding d' . The missing values of dimension $1, \dots, d-1$ are imputed using the predictions of the corresponding GPs. Black arrows show connections. Blue arrows indicate concatenation and red arrows demonstrate log-likelihood computation. Dashed black lines are copying the information.

Our method works as follows. First, it uses a GP to predict the missing values corresponding to the first attribute (after the ordering of the attributes) in terms of all the other attributes. After this step, all missing values corresponding to the first attribute are replaced with the corresponding GP predictions, which are random variables determined by the GP predictive distribution. The first attribute, alongside with the GP predictions for missing values and all the other attributes, are fed to a second GP to predict the missing values corresponding to the second attribute (after the ordering of the attributes). Therefore, some inputs to this second GP are random variables, i.e., those corresponding to the missing values of the first attribute. Next, the first and two attributes, where missing values are replaced by the corresponding GP predictive distribution, are fed, alongside with the remaining attributes, to a third GP to predict the missing values corresponding to the third attribute (after the ordering of the attributes). The process will be iterated until all dimensions with missing values have an associated GP that predicts their values. The observed input attributes alongside with the GP predictions for missing values are then feed to a final GP that predicts the target variable. Fig. 2 shows the architecture described. The resulting method can hence be understood as a particular case of a DGP in which the GPs in the inner layers predict the missing values in the observed data.

Let D_m be the total number of dimensions with associated missing values. Let $\mathbf{f}^{(l)} \in \mathbb{R}^N$ be the process values at layer l for the training data. Similarly, let $\mathbf{u}^{(l)} \in \mathbb{R}^M$ be the process values at layer l for the inducing points. We denote the input of the GP at layer l with $\tilde{\mathbf{X}}^{(l-1)}$. This is a $N \times D$ matrix equal to $\tilde{\mathbf{X}}$, but where missing values corresponding to dimensions 1 to $l-1$ are replaced by the corresponding GP predictions of the previous layer. Therefore, $\tilde{\mathbf{X}}^{(l-1)}$ can be computed in terms of $\tilde{\mathbf{X}}$ and $\{\mathbf{f}^{(l'-1)}\}_{l'=1}^{l-1}$, the predictions of the previous GPs in the sequence. For the first GP, we simply define $\tilde{\mathbf{X}}^{(0)} = \tilde{\mathbf{X}}$. In the last GP, the input is $\tilde{\mathbf{X}}^{(D_m)}$. The joint distribution of all the observed

variables in our model is

$$p(\mathbf{y}, \{\mathbf{f}^{(l)}, \mathbf{u}^{(l)}\}_{l=1}^{D_m+1}) = \prod_{l=1}^{D_m+1} p(\mathbf{f}^{(l)} | \mathbf{u}^{(l)}, \tilde{\mathbf{X}}^{(l-1)}, \mathbf{Z}^{(l)}) p(\mathbf{u}^{(l)} | \mathbf{Z}^{(l)}) \times \prod_{i=1}^N p(y_i | f_i^{(D_m+1)}, \tilde{\mathbf{x}}_i^{(D_m)}) \times \prod_{i=1}^N \prod_{l \notin \mathcal{M}_i} p(x_{i,l} | f_i^{(l)}), \quad (17)$$

where \mathcal{M}_i is the set of attributes with missing values associated to instance \mathbf{x}_i and $p(x_{i,l} | f_i^{(l)}) = \mathcal{N}(x_{i,l} | f_i^{(l)}(\tilde{\mathbf{x}}_i^{(l-1)}), \sigma_l^2)$. That is, we assume a Gaussian likelihood for predicting the corresponding observed values of an attribute with missing values. This is just a particular case of the DGP model described in the previous section, but with extra likelihood factors.

Similar to (12), the variational distribution q is defined as

$$q(\{\mathbf{f}^{(l)}, \mathbf{u}^{(l)}\}_{l=1}^{D_m+1}) = \prod_{l=1}^{D_m+1} p(\mathbf{f}^{(l)} | \mathbf{u}^{(l)}, \tilde{\mathbf{X}}^{(l-1)}, \mathbf{Z}^{(l)}) q(\mathbf{u}^{(l)}), \quad (18)$$

where we can again marginalize out all $\mathbf{u}^{(l)}$ in closed form to obtain

$$q(\{\mathbf{f}^{(l)}\}_{l=1}^{D_m+1}) = \prod_{l=1}^{D_m+1} q(\mathbf{f}^{(l)} | \mathbf{r}^{(l)}, \mathbf{S}^{(l)}; \tilde{\mathbf{X}}^{(l-1)}, \mathbf{Z}^{(l-1)}) \\ = \prod_{l=1}^{D_m+1} \mathcal{N}(\mathbf{f}^{(l)} | \boldsymbol{\mu}^{(l)}, \boldsymbol{\Sigma}^{(l)}). \quad (19)$$

Moreover, $\boldsymbol{\mu}^{(l)}$ and $\boldsymbol{\Sigma}^{(l)}$ are computed as in (5) and (6). Then, the variational ELBO of MGP is

$$\mathcal{L}_{MGP} = \sum_{i=1}^N \mathbb{E}_q[\log p(y_i | f_i^{(D_m+1)}, \tilde{\mathbf{x}}_i^{(D_m)})] + \sum_{i=1}^N \sum_{l \notin \mathcal{M}_i} \mathbb{E}_q[\log p(x_{i,l} | f_i^{(l)})] \\ - \sum_{l=1}^{D_m+1} \text{KL}[q(\mathbf{u}^{(l)}) | p(\mathbf{u}^{(l)} | \mathbf{Z}^{(l)})], \quad (20)$$

Table 1
Characteristics of the datasets.

| Dataset | N | d |
|------------------|--------|----|
| Protein | 45,730 | 10 |
| KeggD | 53,414 | 23 |
| KeggUD | 65,554 | 28 |
| Parkinson | 1,040 | 24 |
| TotalBrainVolume | 867 | 31 |

where the required expectations can be approximated via Monte Carlo simply by propagating samples through the GP network displayed in Fig. 2, as in the case of a DGP. Importantly, our formulation optimizes all hyper-parameters and variational parameters at the same time by maximizing \mathcal{L}_{MGP} .

Algorithm 1 shows the training details of MGP. This algorithm uses a mini-batch to obtain a noisy estimate of (20) and its gradient, which is then used to update the parameters of each $q(\mathbf{u}^{(l)})$ and the hyper-parameters. The data-dependent term of (20) is corrected to account for the fact that it is estimated using a single mini-batch.

When making a prediction for a new data instance \mathbf{x}_* , one can propagate K samples through the GP network. This results in a Gaussian mixture to predict the latent function at layer $D_m + 1$. That is,

$$q(f_{*}^{(D_m+1)}) \approx \frac{1}{K} \sum_{k=1}^K q(f_{*}^{(D_m+1)} | \mathbf{r}^{(D_m+1)}, \mathbf{S}^{(D_m+1)}, \tilde{\mathbf{x}}_i^{(D_m),k}, \mathbf{Z}^{(D_m+1)}), \quad (21)$$

where $\tilde{\mathbf{x}}_i^{(D_m),k}$ is the k th sample of $\tilde{\mathbf{x}}_i^{(D_m)}$, the input to the last GP in the network. When making predictions for the target variable, y_* , one simply has to add the variance of the additive Gaussian noise to each component of the previous Gaussian mixture.

The computational cost of the proposed method, MGP, is squared in the number of dimensions with missing values, since all other dimensions are used for prediction. The cost is, however, cubic in the number of inducing points considered. If mini-batch training is used, the training cost does not depend on the number of training instances.

Algorithm 1 Training algorithm for MGP

Require: Training data \mathcal{D} and D_m attributes with missing values, initial matrix of observed attributes $\hat{\mathbf{X}}$, and M inducing points for each layer.

Ensure: Optimal parameters of the model

Initialize model's hyper-parameters θ

while stopping criteria is False **do**

Gather mini-batch \mathbf{mb} of size n from \mathcal{D}

for (\mathbf{x}_i) in \mathbf{mb} **do**

Compute predictive distribution for $\hat{f}_i^{(l)}$, for each layer l , and propagate samples.

end for

Estimate ELBO using (20) and the propagated samples $\hat{f}_i^{(l)}$:

$\hat{\mathcal{L}}_{MGP} \leftarrow \frac{N}{n} \times \log_{\text{ik}} - \text{KL}$

Update parameters of the model using the noisy gradient of

$\hat{\mathcal{L}}_{MGP}$

end while

4. Experimental setup

We use 5 different data sets to evaluate the proposed method MGP. Table 1 describes the datasets. Four datasets are publicly available from the UCI repository [16]. The last dataset, called *TotalBrainVolume* (TBV), is a private dataset obtained from

“Perinatal brain damage” group at Biomedical Research and Innovation Institute of Cádiz (INiBICA) Research Unit, Puerta del Mar University Hospital University of Cádiz, Spain [34]. It is related to preterm infants and it contains different categorical and continuous attributes corresponding to the clinical information associated to those infants. It initially contains 3.2 percent missing values. All the datasets are standardized using a Z-score transformation method. All categorical variables are converted to continuous variables using a one-hot encoding strategy. For each dataset, we generate five different splits, where 70 percent of the data are used for training, and the rest, 30 percent, are used for testing. Then, we randomly removed 10, 20, 30, 40 percent of the observed data in each dataset split to randomly introduce missing values. We report results for each different level of missing values. The performance of the proposed method, MGP, is compared to:

- **Mean:** The mean value of each variable is computed and used to impute missing values.
- **Median:** The median value of each variable is used to impute missing values.
- **KNN:** A K-nearest neighbor is used to estimate and replace the missing values. The number of neighbors is fixed to be 2 in all the problems. We tried other numbers of neighbors and we found out that such a number works well in general in all the experiments considered.
- **GAIN:** Generative adversarial network for missing data imputation [26] is also used to compute the missing values. The number of iterations are fixed to 20,000. The α value is set to be 10, as recommended, and all the other specifications are similar to what is suggested in [26]. We observed that GAIN suffers from over-fitting and often does not perform well on previously unseen data.
- **MICE:** Multiple imputation using chained equations [21] is another state-of-the-art algorithm that has been used in this experiment. Linear regression is used to estimate each missing value and the number of repetitions used is 10, as proposed in [35].
- **SVGP:** Sparse variational Gaussian process [11], as described in Section 3.2. Required missing values are estimated using mean imputation. The number of inducing points and the number of training iterations are fixed to be 100 and 10,000, respectively.
- **DGP:** Five layered deep Gaussian process, as described in [14], and in Section 3.3. Again, we use mean imputation to estimate required missing values. The specifications are similar to SVGP.
- **DBN:** This is the method for missing value imputation described in [28].
- **VAE:** Variational AutoEncoder (VAE) is the method for missing value imputation described in [27]. There are 256 hidden units and the latent dimension is 512 in all experiments. The encoder and decoder have two equal layers.
- **MGP:** Our proposed method. It is also trained for a total of 10,000 iterations, except for the TotalBrainVolume and Parkinson datasets where 2000 iterations are used for training as they have fewer number of training samples.

The mini-batch size for all GP based algorithms is 100. All GP based methods and GAIN are optimized using Adam [36] and a learning rate equal to 0.01. We use 20 samples when training and testing in all GP based methods. We use Matern 5/2 kernel in all experiments with GP based methods

$$K(\mathbf{x}_i, \mathbf{x}^*) = \sigma_f^2 \left(\sqrt{5} \frac{\mathbf{r}}{\ell} + 1 + \frac{5\mathbf{r}^2}{3\ell^2} \right) \exp \left(-\sqrt{5} \frac{\mathbf{r}}{\ell} \right) \quad (22)$$

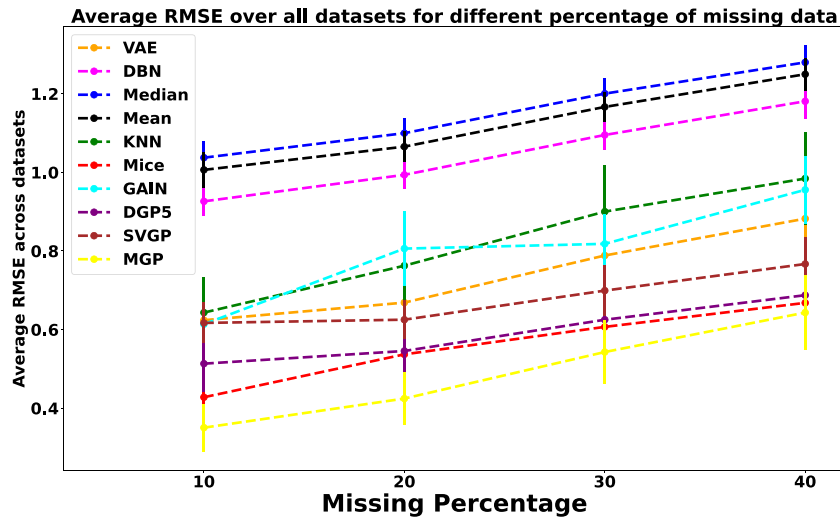


Fig. 3. Average RMSE values obtained by the used methods in this study at various missing rates using KeggUD, Parkinson, KeggD, Protein and Total brain volume dataset. The error bar shows the standard deviation of the average RMSE values.

where σ_f^2 is the scale factor, ℓ is the length scale parameter, and \mathbf{r} is

$$\mathbf{r} = \|\mathbf{x}_i - \mathbf{x}^*\|_2 \quad (23)$$

All the experiments have been executed using two RTX A5000 GPUs (24 Gb), available at INIBICA.

Although most of the methods described can be used to predict a target variable y associated to each dataset, in our experiments we focus exclusively on missing value imputation. That is, we try to predict all missing values present in the data and do not consider a target variable y to be predicted. That is straight-forward to do in our proposed method, MGP, and other approaches we compare with. In DGP and SVGP (SVGP is just a DGP with one layer) we simply have at the output layer D different GPs, one for each attribute with missing values. We then have a likelihood factor for each observed attribute.

We compare all methods in terms of the root mean squared error of missing value imputation in the test set. Namely,

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N \frac{1}{|\mathcal{M}_i|} \sum_{d \in \mathcal{M}_i} (X_i^d - X_i'^d)^2} \quad (24)$$

where X_i^d is the i th true missing value, $X_i'^d$ is i th estimated value at dimension d , and \mathcal{M}_i is the set of attributes with missing values associated to instance \mathbf{x}_i . In the GP based methods we use the mean of the predictive distribution as the model's prediction.

In all these experiments, we focus on regression inside each layer of MGP. However, one can use classification GPs, besides regression, whenever the output is binary, as in [14]. This also happens in the case of SVGP and DGP. Furthermore, we compare the performance of MGP with that of DBN [28] in the task of imputing missing values using the TBV dataset. The configuration of DBN is adapted from [28]: the number of hidden nodes in each layer are two times the number of features, the mini-batch sizes are 256, learning rate is 0.001 and the number of epochs set to be 1000. Finally, we also consider using the imputed values of MGP as an input to a support vector machine (SVM) classifier to distinguish between healthy and non-healthy patients on the TBV dataset. The idea is that the quality of missing value imputation should be reflected in a better performance of the SVM classifier.

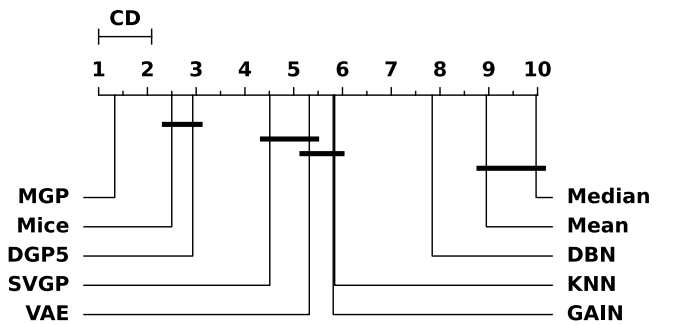


Fig. 4. Average rank of each method alongside with the corresponding critical distance on all datasets and splits when considering all levels of missing values 10%, 20%, 30% and 40%.

5. Results and discussion

Fig. 3 shows summarized results graphically for all the datasets and different level of missing values. The figure displays the average RMSE for each method across all five datasets, for each percentage of missing values considered. We observe that MGP obtains the best results.

Tables 2 to 5 show the RMSE for each method after randomly removing 10%, 20%, 30% and 40% of the values from the data, respectively. We observe that the proposed method, MGP, most of the times, has a better performance than the other methods on each dataset. In general, mean and median imputation based methods are the worst methods in all cases. DBN is following them. KNN is the fourth method while is 20 percentage of missing values is doing better than GAIN. GAIN at 10 percentage of missing value is very close to SVGP and VAE. After that it will get closer to KNN. The imputation accuracy of SVGP is generally better than VAE. The accuracy of MICE is better than DGP5 at 10 percent of missing values. After that their accuracy is getting close to each other. MGP is performing very well at lower percentage of missing values. At higher percentage it gets close to MICE and DGP5.

Moreover, we computed the average rank of each method across all datasets splits and levels of missing values. In particular, if a method obtains the best performance for a dataset split and level of noise, it gets rank 1. If it obtains the second best performance, it gets rank 2, etc. Fig. 4 shows the average rank of each

Table 2

Average RMSE values for 10% missing values. The numbers in parentheses are standard errors. Best mean values are highlighted.

| | Protein | KeggD | KeggUD | Parkinson | TBV |
|--------|-------------------|-------------------|-------------------|-------------------|-------------------|
| Median | 1.13(0.02) | 0.90(0.04) | 1.09(0.01) | 1.09(0.08) | 0.98(0.05) |
| Mean | 1.11(0.02) | 0.87(0.04) | 1.06(0.01) | 1.07(0.08) | 0.93(0.04) |
| KNN | 0.73(0.03) | 0.27(0.02) | 0.73(0.04) | 0.80(0.06) | 0.69(0.06) |
| MICE | 0.51(0.03) | 0.29(0.03) | 0.29(0.01) | 0.47(0.03) | 0.59(0.02) |
| GAIN | 0.65(0.05) | 0.48(0.11) | 0.51(0.02) | 0.68(0.04) | 0.75(0.04) |
| DGP5 | 0.57(0.03) | 0.34(0.04) | 0.33(0.01) | 0.65(0.03) | 0.68(0.02) |
| SVGP | 0.72(0.03) | 0.47(0.04) | 0.51(0.01) | 0.68(0.04) | 0.70(0.01) |
| DBN | 0.96(0.02) | 0.80(0.04) | 0.97(0.02) | 1.02(0.08) | 0.88(0.03) |
| VAE | 0.76(0.03) | 0.48(0.04) | 0.43(0.01) | 0.80(0.06) | 0.65(0.05) |
| MGP | 0.47(0.04) | 0.23(0.04) | 0.17(0.02) | 0.43(0.05) | 0.46(0.05) |

Table 3

Average RMSE values for 20% missing values. The numbers in parentheses are standard errors. Best mean values are highlighted.

| | Protein | KeggD | KeggUD | Parkinson | TBV |
|--------|-------------------|-------------------|-------------------|-------------------|-------------------|
| Median | 1.15(0.03) | 0.95(0.04) | 1.16(0.03) | 1.14(0.06) | 1.10(0.04) |
| Mean | 1.13(0.03) | 0.93(0.04) | 1.13(0.03) | 1.12(0.05) | 1.03(0.03) |
| KNN | 0.97(0.04) | 0.32(0.02) | 0.81(0.04) | 0.88(0.03) | 0.83(0.03) |
| MICE | 0.49(0.03) | 0.37(0.01) | 0.37(0.02) | 0.59(0.03) | 0.86(0.03) |
| GAIN | 0.68(0.08) | 1.16(1.37) | 0.57(0.01) | 0.72(0.04) | 0.90(0.04) |
| DGP5 | 0.54(0.03) | 0.38(0.07) | 0.37(0.03) | 0.68(0.03) | 0.76(0.03) |
| SVGP | 0.64(0.02) | 0.51(0.07) | 0.52(0.03) | 0.68(0.04) | 0.78(0.03) |
| DBN | 0.99(0.02) | 0.87(0.04) | 1.04(0.03) | 1.08(0.05) | 0.99(0.03) |
| VAE | 0.79(0.04) | 0.48(0.04) | 0.49(0.03) | 0.84(0.05) | 0.74(0.05) |
| MGP | 0.50(0.03) | 0.28(0.04) | 0.24(0.03) | 0.48(0.03) | 0.61(0.02) |

Table 4

Average RMSE values for 30% missing values. The numbers in parentheses are standard errors. Best mean values are highlighted.

| | Protein | KeggD | KeggUD | Parkinson | TBV |
|--------|-------------------|-------------------|-------------------|-------------------|-------------------|
| Median | 1.23(0.06) | 1.04(0.02) | 1.23(0.01) | 1.25(0.06) | 1.25(0.02) |
| Mean | 1.21(0.06) | 1.02(0.02) | 1.20(0.01) | 1.23(0.06) | 1.17(0.02) |
| KNN | 1.20(0.03) | 0.43(0.03) | 0.87(0.05) | 1.02(0.04) | 0.98(0.02) |
| MICE | 0.56(0.06) | 0.43(0.01) | 0.44(0.03) | 0.67(0.06) | 0.93(0.08) |
| GAIN | 0.77(0.18) | 0.64(0.07) | 0.69(0.01) | 0.92(0.05) | 1.07(0.04) |
| DGP5 | 0.59(0.06) | 0.43(0.02) | 0.42(0.02) | 0.80(0.05) | 0.89(0.02) |
| SVGP | 0.68(0.06) | 0.56(0.02) | 0.55(0.02) | 0.80(0.05) | 0.90(0.02) |
| DBN | 1.07(0.06) | 0.96(0.02) | 1.12(0.01) | 1.19(0.06) | 1.13(0.02) |
| VAE | 0.85(0.09) | 0.61(0.07) | 0.59(0.01) | 0.96(0.06) | 0.94(0.03) |
| MGP | 0.58(0.06) | 0.39(0.02) | 0.31(0.02) | 0.64(0.06) | 0.79(0.03) |

Table 5

Average RMSE values for 40% missing values. The numbers in parentheses are standard errors. Best mean values are highlighted.

| | Protein | KeggD | KeggUD | Parkinson | TBV |
|--------|-------------------|-------------------|-------------------|-------------------|-------------------|
| Median | 1.31(0.05) | 1.11(0.05) | 1.32(0.01) | 1.34(0.06) | 1.33(0.02) |
| Mean | 1.29(0.05) | 1.08(0.05) | 1.30(0.01) | 1.31(0.05) | 1.27(0.02) |
| KNN | 1.26(0.04) | 0.52(0.03) | 0.92(0.03) | 1.11(0.05) | 1.11(0.01) |
| MICE | 0.61(0.03) | 0.48(0.03) | 0.47(0.01) | 0.72(0.03) | 1.06(0.12) |
| GAIN | 0.90(0.13) | 0.71(0.06) | 0.85(0.04) | 1.09(0.10) | 1.23(0.09) |
| DGP5 | 0.63(0.05) | 0.50(0.07) | 0.47(0.01) | 0.86(0.04) | 0.98(0.02) |
| SVGP | 0.72(0.05) | 0.63(0.09) | 0.61(0.02) | 0.87(0.05) | 1.00(0.02) |
| DBN | 1.15(0.05) | 1.02(0.05) | 1.22(0.01) | 1.28(0.05) | 1.23(0.02) |
| VAE | 0.94(0.06) | 0.70(0.19) | 0.65(0.02) | 1.05(0.06) | 1.07(0.03) |
| MGP | 0.65(0.05) | 0.47(0.04) | 0.39(0.02) | 0.73(0.06) | 0.97(0.01) |

method. Following [37], we carried out a Bonferroni–Dunn post-hoc test to look for statistical differences among average ranks. If the average ranks of two methods are far apart one from another by a distance bigger than the critical distance (CD) shown in Fig. 4, the differences are statistically significant. The critical distance is computed in terms of the number of methods compared, 8, and the number of datasets and splits considered. Namely, $5 \times 5 = 25$. The reason for this is that the missing values are different in each dataset split. We observe that mean and median imputations are the worst methods, overall. According to Fig. 4, MGP is the best performing method overall, followed by DGP and MICE, which

have similar overall performance. SVGP, VAE, GAIN, and KNN perform similarly, DBN is doing better than Mean and Median, and mean and median imputation are performing close together. Appendix shows similar results for each individual dataset.

Table 6 presents the results of an extra experiment where we compare the accuracy of the proposed MGP with the suggested ordering (i.e., ordering the attributes according to their standard deviation before standardization) against the random ordering of the attributes (MGP-Random). We consider different level of missing values and the Protein data-set. The results show that the

Table 6

Average RMSE value for imputing missing values of Protein dataset using MGP when random ordering of variables and the proposed variable ordering is used. The numbers in parentheses are standard errors.

| | 10% | 20% | 30% | 40% |
|------------|------------|------------|------------|------------|
| MGP | 0.47(0.04) | 0.50(0.03) | 0.58(0.06) | 0.65(0.05) |
| MGP-Random | 0.48(0.04) | 0.50(0.03) | 0.57(0.06) | 0.65(0.05) |

Table 7

Average training time per epoch and prediction time per instance on the Protein data set.

| Method | Training | Prediction |
|---------------------|--------------|---------------|
| MGP | 51(0.92) | 0.11(0.01) |
| DGP5 | 37.4(0.75) | 0.06(0.00) |
| SVGP | 10.35(0.07) | 0.03(0.01) |
| DBN | 9.86(0.70) | 0.001(0.00) |
| VAE | 3.92(0.85) | 0.002(0.00) |
| GAIN | 6.29(0.08) | ^b |
| Mean ^a | 0.007(0.00) | 0.001(0.00) |
| Median ^a | 0.028(0.00) | 0.001(0.00) |
| KNN ^a | 37.44(0.298) | 16.163(0.207) |
| MICE ^a | 1.32(0.075) | 0.096(0.017) |

^aTotal training time has been mentioned.

^bThere is no test time.

differences between the accuracies of the two methods is negligible. This indicates that the particular ordering of the attributes considered by MGP is not relevant.

Table 7 shows the training time per epoch and prediction time per instance, for each method, on the Protein dataset. As expected, MGP needs more training time and needs more prediction time. However, it still in the range of time needed by DGP5. The training time of SVGP and DBN are very close to each other. MICE, KNN, Median and Mean imputation have the smallest training times.

Table 8 shows, for each method considered to impute missing values, the average accuracy of a SVM classifier trained on the imputed data when distinguishing healthy and non-healthy patients. We considered the TBV dataset, when 10% and 20% of the data are missing. The results show that the pre-imputation with MGP leads to better accuracy of the SVM classifier than with the other methods. This indicates a better imputation of missing values by our method, MGP.

MGP is performing also better than SVGP and DGP5. The reason is that MGP uses the predictions for the missing values of each dimension to predict the missing values of the other dimensions. By contrast, DGP5 and SVGP cannot consider directly such missing value predictions across dimensions for missing value imputation. Moreover, MGP has a fixed number of layers which are determined by the number of dimensions containing missing values. Actually, MGP is more related to MICE, which is one of the best know algorithms for missing value imputation. MICE considers the chained relations between each missing attribute. MGP acts better than MICE as it is powered by the flexibility and the probabilistic nature of GPs. MICE simply relies on linear models.

Finally, we would like to point out that our MGP implementation is coded using PyTorch and is publicly available at <https://github.com/BahramJafrasteh/MissingGPs>.

6. Conclusions

We have presented a novel hierarchical composition of sparse variational GPs to impute missing values, inspired by deep GPs and recurrent GPs. The proposed method, called missing GP (MGP), has been evaluated on four UCI benchmark data sets and

on one real-life private medical dataset, where 10, 20, 30, and 40 percent of the data attributes contain missing values.

In our experiments, we observed a statistically significant better performance of MGP than other state-of-the-art methods for missing value imputation. Namely, KNN, MICE, GAIN, DBN, VAE, Mean and Median imputation. We also observed that MGP provides better results than other methods, *i.e.*, deep GPs and sparse variational GPs (SVGP) in terms of the RMSE of missing value imputation. This is particularly the case when the fraction of missing values is not very high. By contrast, when this fraction is high, we believe that there is not enough data to train the sparse GPs inside MGP and the gains obtained are better, but not as significant. The proposed method, MGP, works also better in terms of the accuracy of a SVM classifier trained on the imputed missing values to solve a classification task.

Regarding the computational cost, MGP is more expensive than simple alternatives for missing value imputation. MGP has a cost that is similar to that of a standard DGP that directly aims at predicting missing values. Nevertheless, in spite of its higher computational cost, MGP performs better than the other methods according to our experiments, which means that the extra computational time of MGP is worth it.

In our work, we used only regression GPs inside MGP. However, it is also possible to use a combination of classification or regression layers to impute missing values with binary attributes. This will make approximate inference more challenging since the Bernoulli distribution is not re-parameterizable. However, it may be possible to leave the binary attributes as the last ones in the hierarchical structure of MGP so that their output is not used for the imputation of other variables.

Selecting the best method for missing value imputation for a given task is a challenging problem. However, we have shown that MGP gives over-all good predictive performance for missing value imputation. Therefore, it is a good candidate for this task. More principled techniques to choose the best strategy for a particular dataset may consider the nature of the observed data and the use of cross-validation. Investigating these techniques is left for future work.

CRedit authorship contribution statement

Bahram Jafrasteh: Conceptualization, Methodology, Software, Writing – original draft, Data curation. **Daniel Hernández-Lobato:** Conceptualization, Writing – original draft, Supervision. **Simón Pedro Lubián-López:** Writing – review & editing. **Isabel Benavente-Fernández:** Writing – review & editing.

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.

Data availability

Data will be made available on request.

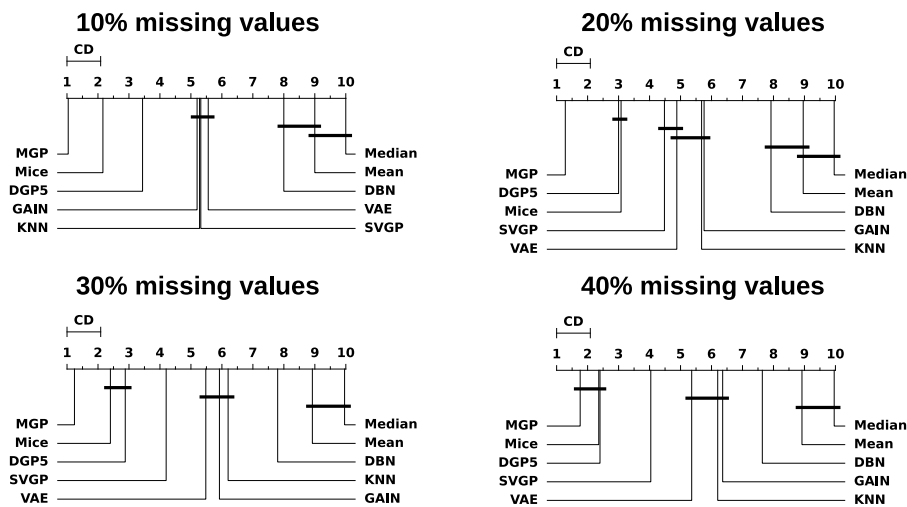
Acknowledgments

This study was funded by the Cádiz integrated territorial initiative for biomedical research, European Regional Development Fund (ERDF) 2014–2020 (<http://dx.doi.org/10.13039/501100008530>). Andalusian Ministry of Health and Families, Spain. Registration number: ITI-0019-2019. DHL acknowledges financial support from Spanish Plan Nacional I+D+i, grant PID2019-106827GB-I00/AEI/10.13039/501100011033.

Table 8

Average accuracy of SVM for TBV dataset with different methods. The numbers in parentheses are standard errors. Best mean values are highlighted.

| Missing | Mean | KNN | DBN | MICE | MGP | SVGP | GAIN | VAE | Median | DGP5 |
|---------|------------|------------|------------|------------|-------------------|------------|-------------|-------------|-------------|--------------|
| 10% | 0.89(0.04) | 0.90(0.04) | 0.89(0.04) | 0.91(0.03) | 0.97(0.02) | 90(0.03) | 0.89 (0.04) | 0.86(0.03) | 0.87 (0.04) | 0.91 (0.04) |
| 20% | 0.80(0.03) | 0.81(0.02) | 0.81(0.02) | 0.85(0.05) | 0.90(0.04) | 0.87(0.03) | 0.80 (0.09) | 0.78 (0.10) | 0.77 (0.02) | 0.087 (0.04) |

**Fig. A.5.** Average rank of each method alongside with the corresponding critical distance for different level of missing values, 10%, 20%, 30% and 40%.

Appendix. Average rank over different percentages of missing values

Fig. A.5 CD diagram for each different level of missing values. Namely, 10%, 20%, 30% and 40%, respectively. We observe that in general the results are similar to those of Fig. 4 and MGP is the best method overall. However, when the level of missing values increases to 40% the differences between MGP, MICE and DGP become smaller.

References

- [1] R.J. Little, D.B. Rubin, Statistical analysis with missing data, John Wiley & Sons, 2019.
- [2] C.M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer, 2006.
- [3] B.K. Beaulieu-Jones, J.H. Moore, Missing data imputation in the electronic health record using deeply learned autoencoders, in: Pacific Symposium on Biocomputing, 2017, pp. 207–218.
- [4] S. Ryu, M. Kim, H. Kim, Denoising autoencoder-based missing value imputation for smart meters, IEEE Access 8 (2020) 40656–40666.
- [5] C. Villacampa-Calvo, B. Zaldívar, E.C. Garrido-Merchán, D. Hernández-Lobato, Multi-class Gaussian process classification with noisy inputs, J. Mach. Learn. Res. 22 (1) (2021) 1696–1747.
- [6] C.K. Williams, C.E. Rasmussen, Gaussian Processes for Machine Learning, MIT Press, 2006.
- [7] M. Titsias, Variational learning of inducing variables in sparse Gaussian processes, in: International Conference on Artificial Intelligence and Statistics, 2009, pp. 567–574.
- [8] E. Snelson, Z. Ghahramani, Sparse Gaussian processes using pseudo-inputs, Adv. Neural Inf. Process. Syst. (2006) 1257–1264.
- [9] J. Hensman, A.G. Matthews, M. Filippone, Z. Ghahramani, MCMC for variationally sparse Gaussian processes, Adv. Neural Inf. Process. Syst. (2015) 1648–1656.
- [10] C. Villacampa-Calvo, D. Hernández-Lobato, Scalable multi-class Gaussian process classification using expectation propagation, in: International Conference on Machine Learning, 2017, pp. 3550–3559.
- [11] J. Hensman, A. Matthews, Z. Ghahramani, Scalable variational Gaussian process classification, in: Artificial Intelligence and Statistics, 2015, pp. 351–360.
- [12] A. Damianou, N.D. Lawrence, Deep Gaussian processes, in: International Conference on Artificial Intelligence and Statistics, 2013, pp. 207–215.
- [13] T. Bui, D. Hernández-Lobato, J. Hernandez-Lobato, Y. Li, R. Turner, Deep Gaussian processes for regression using approximate expectation propagation, in: International Conference on Machine Learning, 2016, pp. 1472–1481.
- [14] H. Salimbeni, M. Deisenroth, Doubly stochastic variational inference for deep Gaussian processes, Adv. Neural Inf. Process. Syst. (2017) 4588–4599.
- [15] C.L.C. Mattos, Z. Dai, A. Damianou, J. Forth, G.A. Barreto, N.D. Lawrence, Recurrent Gaussian processes, in: International Conference on Learning Representations, 2016.
- [16] D. Dua, C. Graff, UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences, 2017, URL <http://archive.ics.uci.edu/ml>.
- [17] J.L. Schafer, Analysis of Incomplete Multivariate Data, CRC Press, 1997.
- [18] P. Melchior, A.D. Goulding, Filling the gaps: Gaussian mixture models from noisy, truncated or incomplete samples, Astron. Comput. 25 (2018) 183–194.
- [19] G.E. Batista, M.C. Monard, et al., A study of K-nearest neighbour as an imputation method, Hybrid Intell. Syst. 87 (2002) 251–260.
- [20] L. Folguera, J. Zupan, D. Cicerone, J.F. Magallanes, Self-organizing maps for imputation of missing data in incomplete data matrices, Chemometr. Intell. Lab. Syst. 143 (2015) 146–151.
- [21] P. Royston, I.R. White, Multiple imputation by chained equations (MICE): implementation in stata, J. Stat. Softw. 45 (2011) 1–20.
- [22] W.-C. Lin, C.-F. Tsai, Missing value imputation: a review and analysis of the literature (2006–2017), Artif. Intell. Rev. 53 (2) (2020) 1487–1509.
- [23] X. Ning, Y. Xu, X. Gao, Y. Li, Missing data of quality inspection imputation algorithm base on stacked denoising auto-encoder, in: International Conference on Big Data Analysis, 2017, pp. 84–88.
- [24] R.C. Pereira, P.H. Abreu, P.P. Rodrigues, Vae-bridge: Variational autoencoder filter for Bayesian ridge imputation of missing data, in: International Joint Conference on Neural Networks, 2020, pp. 1–7.
- [25] A. Nazabal, P.M. Olmos, Z. Ghahramani, I. Valera, Handling incomplete heterogeneous data using vae, Pattern Recognit. 107 (2020) 107501.
- [26] J. Yoon, J. Jordon, M. Schaar, Gain: Missing data imputation using generative adversarial nets, in: International Conference on Machine Learning, 2018, pp. 5689–5698.
- [27] D.P. Kingma, M. Welling, Auto-encoding variational Bayes, in: International Conference on Learning Representations, 2014.
- [28] W.-C. Lin, C.-F. Tsai, J.R. Zhong, Deep learning for missing value imputation of continuous data and the effect of data discretization, Knowl.-Based Syst. 239 (2022) 108079.

- [29] V. Fortuin, D. Baranchuk, G. Rätsch, S. Mandt, Gp-vae: Deep probabilistic time series imputation, in: *International Conference on Artificial Intelligence and Statistics*, 2020, pp. 1651–1661.
- [30] C.E. Rasmussen, C.K.I. Williams, *Gaussian Processes for Machine Learning*, The MIT Press, 2006.
- [31] J. Hensman, N. Fusi, N.D. Lawrence, Gaussian processes for big data, in: *Uncertainty in Artificial Intelligence*, 2013.
- [32] D.J. Rezende, S. Mohamed, D. Wierstra, Stochastic backpropagation and approximate inference in deep generative models, in: *International Conference on Machine Learning*, 2014, pp. 1278–1286.
- [33] D.P. Kingma, T. Salimans, M. Welling, Variational dropout and the local reparameterization trick, *Adv. Neural Inf. Process. Syst.* 28 (2015) 2575–2583.
- [34] I. Benavente-Fernández, E. Ruiz-González, M. Lubian-Gutiérrez, S.P. Lubián-Fernández, Y. Cabrales Fontela, C. Roca-Cornejo, P. Olmo-Duran, S.P. Lubián-López, Ultrasonographic estimation of total brain volume: 3D reliability and 2D estimation. Enabling routine estimation during NICU admission in the preterm infant, *Front. Pediatr.* (2021) 740.
- [35] T.E. Raghunathan, P.W. Solenberger, J. Van Hoewyk, Iweware: Imputation and variance estimation software, Tech. rep. Survey Methodology Program, Survey Research Center, Institute for Social Research, University of Michigan, 2002.
- [36] D.P. Kingma, J. Ba, ADAM: A method for stochastic optimization, in: *International Conference on Learning Representations*, 2015.
- [37] J. Demšar, Statistical comparisons of classifiers over multiple data sets, *J. Mach. Learn. Res.* 7 (2006) 1–30.