Machine Learning (2022) 111:1987–2010
https://doi.org/10.1007/s10994-022-06170-3

JGPR: a computationally efficient multi-target Gaussian process regression algorithm

Mohammad Nabati[1] · **Seyed Ali Ghorashi**[1,2] · **Reza Shahbazian**[3]

Received: 7 June 2021 / Revised: 10 February 2022 / Accepted: 18 March 2022 /
Published online: 11 May 2022

Abstract
Multi-target regression algorithms are designed to predict multiple outputs at the same time, and allow us to take all output variables into account during the training phase. Despite the recent advances, this context of machine learning is still an open challenge for developing a low-cost and high accurate algorithm. The main challenge in multi-target regression algorithms is how to use different targets' information in the training and/or test phases. In this paper, we introduce a low-cost multi-target Gaussian process regression (GPR) algorithm, called joint GPR (JGPR) that employs a shared covariance matrix among the targets during the training phase and solves a sub-optimal cost function for optimization of hyperparameters. The proposed strategy reduces the computational complexity considerably during the training and test phases and simultaneously avoids overfitting of the multi-target regression algorithm upon the targets. We have performed extensive experiments on both simulated data and 18 benchmark datasets to assess the proposed method compared with other multi-target regression algorithms. Experimental results show that the proposed JGPR outperforms the state-of-the-art approaches on most of the given benchmark datasets.

Keywords Machine learning · Gaussian process regression · Multi-task learning · Multi-target regression

Editor: Dragi Kocev.

✉ Seyed Ali Ghorashi
s.a.ghorashi@uel.ac.uk

Mohammad Nabati
mo.nabati@mail.sbu.ac.ir

Reza Shahbazian
r.shahbazian@standard.ac.ir

[1] Cognitive Telecommunication Research Group, Department of Telecommunications, Faculty of Electrical Engineering, Shahid Beheshti University, Tehran 19839-69411, Iran

[2] School of Architecture, Computing and Engineering, University of East London, London E16 2RD, UK

[3] Department of Electrical Engineering, Faculty of Technology and Engineering, Standard Research Institute, Alborz 31745-139, Iran

1 Introduction

Conventional machine learning (ML) algorithms are often designed for the prediction of a single target (output variable). However, there are other types of data that contain multiple targets. For example, in Andromeda dataset (Hatzikos et al., 2008) the goal is to predict 6 targets which determines the quality of seawater, or in See Click Predict Fix dataset (Spyromitros-Xioufis et al., 2016) the goal is to find the number of clicks, views and comments based on input features. For these types of data, the conventional ML algorithms can independently be implemented for each target (Li et al., 2014). Although establishing this strategy is a straightforward solution, it does not use the other outputs' information in the training and/or test phases. Muti-target regression algorithms, which are the subset of multi-task learning (MTL) (Zhang & Yang, 2021), are used to predict multiple and continuous outputs at the same time (Zhen et al., 2018).

MTL algorithms are built for the simultaneous implementation of relevant tasks. In the MTL algorithms, a current task uses the other related tasks' information for achieving a better performance (Zhang & Yang, 2017). The concept of MTL has been derived from the human-learning paradigm in which people gain knowledge from previous learning tasks (Xu et al., 2018). Multi-target regression has a similar definition in which the goal is to predict multiple outputs simultaneously (Spyromitros-Xioufis et al., 2016; Petković et al., 2020). The MTL is a general form in which the input samples and the number of those samples can be different for each task, while, in multi-target regression problems, input samples are the same for all targets. Generally, the MTL-based algorithms can also be employed for multi-target regression problems but not vice versa; however, in most cases, the MTL-based algorithms force enormous calculation costs due to their general form.

Various multi-target regression algorithms have been introduced, which are the extended versions of the baseline ML algorithms. Spyromitros-Xioufis et al. (2016) proposed input space expansion such that the other outputs are fed into the input space for each target. Tsoumakas et al. (2014) employed a random linear target combination scenario, in which new output variables are created via random multiplications of available outputs. Melki et al. (2017) introduced a multi-target support vector regression-based correlation chain to take the outputs' correlation into account. Struyf and Džeroski (2005) proposed a constraint-based system to build a multi-objective regression tree.

On the other side, some researchers have tried to design MTL-based Gaussian process regression (GPR) algorithms. Bonilla et al. (2008) proposed an MTL-based GPR algorithm that uses the Kronecker product to extract the correlation between the targets. Nguyen et al. (2014) proposed a collaborative GPR-based algorithm to draw the tasks' correlations by sharing multiple sets of inducing samples. Also, there are lots of attempts to elicit the outputs' correlations through the convolutional process (Álvarez & Lawrence, 2011; Álvarez et al., 2010). The earlier MTL-based GPR algorithms force huge computational costs for multi-target regression problems.

Despite all the recent advances in this topic, lots of efforts are underway to design a low-cost and high accurate algorithm. The accuracy of an implemented algorithm on one hand, and the complexity, on the other hand, are two main challenges of the MTL or multi-target regression algorithms. Although the MTL-based algorithms can handle the multi-target regression problems, they might implicate huge calculation costs due to the structure of implemented techniques in the general form. A more complex structure of an algorithm does not give always a better performance; because, when an ML algorithm becomes more

and more complex, only the training error may reduce, but the test error increases due to the overfitting effect (Hastie et al., 2009).

The overfitting problem occurs when an ML algorithm is fit to the training data too well in the presence of noise (Liu et al., 2008). Various techniques have been introduced to avoid this problem. The regularization (Hastie et al., 2009; Tibshirani, 1996) is used in a wide range of ML algorithms to prevent overfitting of a regression model. Srivastava et al. (2014) proposed a dropout connection strategy for deep neural networks to hinder the model from co-adapting too much in which the units are randomly removed during the training process. The basic idea behind most techniques for solving the overfitting problem is to prevent the hyperparameters of a model to reach a globally optimum point (Hastie et al. 2009). The overfitting also can be prevented by stopping the iteration when the validation data error increases. When the instances of data are not enough, using a portion of data as validation part may decrease the performance, and in general, using an algorithm without needing a part of dataset for validation would be more desired.

In multi-target regression problems, the tasks (targets) have a common set of features, and thereby, it helps to reduce the complexity calculation of the multi-target regression algorithms. On the other hand, the concept of overfitting reduction by finding a sub-optimal solution can either help to increase the accuracy or to reduce the complexity. Hence, there are two facts

1. The input features are common for output variables in multi-target regression problems. This fact can help to reduce the complexity.
2. The overfitting problem can be solved by finding a sub-optimal solution. By using this fact, we can reduce the complexity and at the same time enhance the accuracy.

The key idea behind multi-target regression algorithms is the transfer of knowledge between the targets. In this paper, we introduce a GPR-based multi-target regression algorithm that combines the transfer of knowledge between the targets and overfitting reduction concepts by solving a sub-optimal solution. The proposed method, named joint Gaussian process regression (JGPR), maximizes the log-likelihood of the joint probabilities with a shared covariance matrix over the targets. The proposed JGPR algorithm not only benefits from a low-complexity structure in the training and test phases, but also achieves better accuracy in the test phase compared with conventional GPR (CGPR) in both toy problem and real-world datasets. Unlike the existing works, the JGPR uses a shared covariance matrix in a completely joint optimization process for all targets. This subtle assumption significantly reduces the complexity in the optimization process, and at the same time, avoids overfitting of the model upon the targets. Experimental results on 18 diverse benchmark datasets show that the proposed JGPR has better performance compared with state-of-the-art multi-target regression algorithms. It is because the JGPR is not overfitted when all targets take part during the training process. This paper is the extended version of our previous work (Nabati et al., 2021) in which we only used a 2D-based algorithm for a fingerprint-based positioning problem, and also, we did not perform sufficient mathematical analyses such as convergence and complexity. In summary, the main contributions of this paper can be summarized as follows

– We propose a novel multi-target Gaussian process regression algorithm, named joint Gaussian process regression (JGPR), which not only benefits a low-complexity struc-

ture but also prevents the regression algorithm from overfitting problem and enhances the accuracy.
– We perform a mathematical convergence analysis for the proposed JGPR algorithm and conclude that the proposed method is converged in the optimization process.
– We provide a complexity analysis for the proposed JGPR algorithm and conclude that it has a lower calculation cost even compared with the conventional GPR algorithm in multi-target regression problems.

The rest of this paper is organized as follows: In Sect. 2, we present the conventional Gaussian process regression framework. Our proposed algorithm is explained in Sect. 3. Experimental results and conclusions are presented in Sect. 4 and 5, respectively.

Notations: We use lower-case letters to show scalars (e.g., u), bold-face lower-case letters to indicate vectors (e.g., $\mathbf{u}$) and bold-face capital letters to denote matrices (e.g., $\mathbf{U}$). Also, u_i demonstrates the i^{th} element of $\mathbf{u}$, $\mathbf{U}_i$ and $\mathbf{U}^i$ show the i^{th} row and i^{th} column of $\mathbf{U}$, respectively. The $(\hat{.})$ symbol is used to show the test dataset.

2 Conventional GPR (CGPR)

In this section, we present a background for conventional GPR in the training and test phases. This section helps the reader to better understand the proposed JGPR in the next section.

2.1 Training phase of CGPR

Gaussian process regression is one of the most popular non-linear ML algorithms and has a wide variety of applications (Williams & Rasmussen, 2006; Nguyen et al., 2018). This algorithm can predict the variance besides the outputs, which is derived from the Bayes rule. The goal of CGPR in the training phase is to obtain a non-linear function that maps the input features to a single output variable. We commence via the assumption that a non-linear relationship between input features and a single output variable is as follows

$$f_i = \mu(\mathbf{x}_i) + \varepsilon \quad \forall \mathbf{x}_i, \tag{1}$$

where $\mathbf{x}$ is the input vector, μ is a function that converts the input features to the desired target, and $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$ models the noise. In the training phase, the function values $\mu(\mathbf{x}_i)$ are unknown; however, we have access to the noisy observations f_i and the goal is to obtain the function μ with a training dataset. We shall first consider the training dataset as follows

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1M} \\ x_{21} & x_{22} & \cdots & x_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{NM} \end{pmatrix}, \mathbf{f} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{bmatrix}, \tag{2}$$

where $\mathbf{X} \in \Re^{N \times M}$ is the input matrix with M features and N observations, and $\mathbf{f}$ is the vector of outputs. We can assume a zero-mean Gaussian process, and hence, the joint distribution of input observations can be written in the following form

$$\mathbf{f} \sim \mathcal{GP}(\mathbf{0}, \mathbf{C}), \tag{3}$$

where $\mathbf{C} \in \Re^{N \times N}$ is the covariance matrix, which must be positive definite (Ambikasaran et al., 2016). Each element of this matrix c_{ij} is calculated by two pairs of input observations via kernel functions $\mathcal{K}(\mathbf{X}_i, \mathbf{X}_j)$, where $\mathbf{X}_i$ and $\mathbf{X}_j$ are the i^{th} and j^{th} row of the matrix $\mathbf{X}$ in (2). There are several kernel functions to calculate the elements of covariance matrix. In this paper we use squared exponential kernel $\mathcal{K}_{SE}(\mathbf{X}_i,\mathbf{X}_j) = \gamma^2 \exp\left(-d^2(\mathbf{X}_i, \mathbf{X}_j)\big/ l^2\right)$ to capture non-linear dependencies of input features and Noise kernel $\mathcal{K}_n(\mathbf{X}_i,\mathbf{X}_j) = \sigma_n^2 \delta_{ij}$ to model the noise ϵ in (1), where $\delta_{ij} = \{1 \text{ if } i = j, \ 0 \text{ o.w}\}$. Therefore, the combined kernel can be written as follows

$$\mathcal{K}(\mathbf{X}_i,\mathbf{X}_j) = \gamma^2 \exp\left(-\frac{d^2(\mathbf{X}_i, \mathbf{X}_j)}{l^2}\right) + \sigma_n^2 \delta_{ij}, \tag{4}$$

where $\boldsymbol{\theta} = [\gamma, \ l, \ \sigma_n]^T$ is the vector of hyperparameters that should be optimized in the training phase, and d is the Euclidean distance between two vectors $\mathbf{X}_i$ and $\mathbf{X}_j$. In order to optimize the hyperparameters, the multivariate probability density function (PDF) of observations can be maximized as follows

$$\tilde{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \log(p(\mathbf{f})) = \arg\min_{\boldsymbol{\theta}}(-\log(p(\mathbf{f}))), \tag{5}$$

where $p(\mathbf{f})$ is the multivariate PDF as follows

$$p(\mathbf{f}) = \frac{1}{(2\pi)^{N/2}|\mathbf{C}|^{1/2}} \exp(-\frac{1}{2}\mathbf{f}^T\mathbf{C}^{-1}\mathbf{f}), \tag{6}$$

Thus, the objective function in (5) can be written as follows

$$\mathcal{L}(\boldsymbol{\theta}) = -\log p(\mathbf{f}) = \frac{1}{2}\log|\mathbf{C}| + \frac{N}{2}\log(2\pi) + \frac{1}{2}\mathbf{f}^T\mathbf{C}^{-1}\mathbf{f}. \tag{7}$$

The Eq. (5) is a non-convex optimization problem; however, a gradient-based optimizer can be used to solve this problem for a locally optimum point such as conjugate gradient algorithm (Nocedal and Wright 2006). The conjugate gradient requires the first order gradient, which for the j^{th} hyperparameter can be derived as follows

$$\nabla\mathcal{L}(\theta_j) = -\frac{1}{2}\text{tr}((\mathbf{s}\mathbf{s}^T - \mathbf{C}^{-1})\frac{\partial \mathbf{C}}{\partial \theta_j}) \ \text{ where } \ \mathbf{s} = \mathbf{C}^{-1}\mathbf{f}, \tag{8}$$

where $\frac{\partial \mathbf{C}}{\partial \theta_j} \in \Re^{N \times N}$ is a squared matrix, and its elements are calculated by gradient of the j^{th} hyperparameter. The $\nabla\mathcal{L}(\theta_j)$ is calculated for all hyperparameters, and then, they can be fed to the j^{th} element of gradient vector $\boldsymbol{\vartheta} = [\nabla\mathcal{L}(\theta_1), \nabla\mathcal{L}(\theta_2), \ldots, \nabla\mathcal{L}(\theta_G)]^T$, which here $G = 3$, since there are three hyperparameters for definition of kernel function in (4). The hyperparameters in $\boldsymbol{\theta}$ are updated till reaching a convergence point.

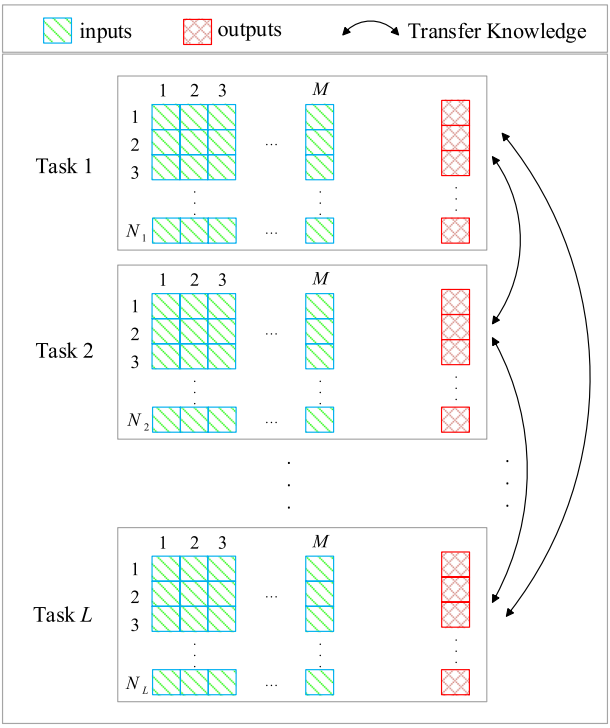

Fig. 1 Multi-Task Learning structure

2.2 Test phase of CGPR

The optimized function for prediction of outputs in the test phase is derived from the Bayes theorem. Initially, the joint distribution of the test and train observations is written as follows

$$\begin{pmatrix} \mathbf{f} \\ \hat{\mathbf{f}} \end{pmatrix} \sim \mathcal{N}\left[\begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \begin{pmatrix} \mathbf{C} & \mathbf{C}(\mathbf{x}, \hat{\mathbf{x}}) \\ \mathbf{C}(\hat{\mathbf{x}}, \mathbf{x}) & \mathbf{C}(\hat{\mathbf{x}}, \hat{\mathbf{x}}) \end{pmatrix}\right] \tag{9}$$

where $\hat{\mathbf{f}}$ is a vector contains the outputs of test data, $\mathbf{C} = \mathbf{C}(\mathbf{x}, \mathbf{x}) \in \Re^{N \times N}$ is the covariance matrix between train observations, $\mathbf{C}(\hat{\mathbf{x}}, \mathbf{x}) = \mathbf{C}^T(\mathbf{x}, \hat{\mathbf{x}}) \in \Re^{\hat{N} \times N}$ is the covariance matrix between the test and train observations, and $\mathbf{C}(\hat{\mathbf{x}}, \hat{\mathbf{x}}) \in \Re^{\hat{N} \times \hat{N}}$ is the covariance matrix between the test observations. The posterior distribution can be derived by conditioning over the training observations $\hat{\mathbf{f}}|\mathbf{f}$ as follows

$$\begin{aligned} &\hat{\mathbf{f}}|\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Phi}) \\ &\boldsymbol{\mu} = \mathbf{C}(\hat{\mathbf{x}}, \mathbf{x})\, \mathbf{C}^{-1}\mathbf{f} \\ &\boldsymbol{\Phi} = \mathbf{C}(\hat{\mathbf{x}}, \hat{\mathbf{x}}) - \mathbf{C}(\hat{\mathbf{x}}, \mathbf{x})\mathbf{C}^{-1}\mathbf{C}(\mathbf{x}, \hat{\mathbf{x}}) \end{aligned} \tag{10}$$

where $\boldsymbol{\mu}$ is prediction of the output values, and $\boldsymbol{\Phi}$ is the posterior covariance matrix whose diagonal elements are predicted variances for the elements of $\boldsymbol{\mu}$.

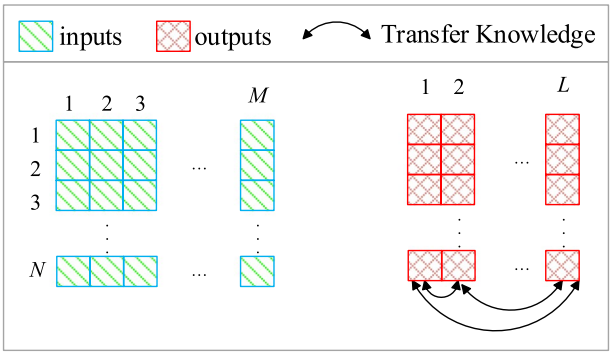

Fig. 2 Multi-Target Regression structure

3 Proposed joint GPR (JGPR)

This section describes the proposed multi-target regression algorithm, named joint Gaussian process regression (JGPR). The structures of the MTL and multi-target regression algorithms are illustrated in Figs. 1 and 2, respectively. As can be seen in the multi-target regression structure, the input features are the same for all targets, and this structure helps to reduce the complexity. Initially, the train and test phases of the JGPR are explained in detail. Then, a pre/post-processing step is proposed that can be performed before and after the training and test phases of JGPR, which is a crucial step to get equal information from all targets in the optimization process. Finally, we discuss the convergence and complexity analyses of the proposed JGPR.

3.1 Training phase of JGPR

The CGPR is independently implemented for each output in multi-target regression problems. Nevertheless, all outputs can contribute in the training phase. Although there are other MTL-based GPR algorithms, most of them have high complexity, because they have been designed for multi-task problems. The proposed algorithm has lower complexity even compared with CGPR for L dimensional outputs, which will be discussed in Sect. 3.5. The basic idea behind this algorithm is to use a shared covariance matrix with the same hyperparameters across the targets. This strategy prevents the model from overfitting problem since it is not fit to a single target as well. Assuming that there are L targets, and $\mathbf{F}$ is a matrix that consists of all targets on its columns, the distribution of input observations over these outputs can be written as follows

$$\mathbf{F}^i \sim \mathcal{GP}(\mathbf{0}, \mathbf{C}), \quad \text{where} \quad i = 1, 2, ...L \tag{11}$$

where $\mathbf{C}$ is the shared covariance matrix over all targets and is filled with a user-defined kernel function as discussed in the previous section. The joint distribution of these targets is as follows

$$
\begin{aligned}
&p(\mathbf{F}) = p(\mathbf{F}^1, \mathbf{F}^2, \cdots, \mathbf{F}^L) = p(\mathbf{F}^L|\mathbf{F}^{L-1}, \mathbf{F}^{L-2}, \cdots, \mathbf{F}^1) \\
&p(\mathbf{F}^{L-1}|\mathbf{F}^{L-2}, \cdots, \mathbf{F}^1) \cdots p(\mathbf{F}^3|\mathbf{F}^2, \mathbf{F}^1)\ \ p(\mathbf{F}^2|\mathbf{F}^1)\ \ p(\mathbf{F}^1)
\end{aligned} \tag{12}
$$

where $p(\mathbf{F}) = p(\mathbf{F}^1, \mathbf{F}^2, \cdots, \mathbf{F}^L)$ is the joint distribution of all targets. In (12), the posterior distributions cannot be derived, since they have a shared covariance matrix and hyperparameters. Even if the restriction does not exist, posterior derivation increases the complexity. Here, it can be assumed that outputs are independent, since we already modeled the dependency of output observations by a shared covariance matrix. Therefore, Eq. (12) is simplified as follows

$$
p(\mathbf{F}) = p(\mathbf{F}^1, \mathbf{F}^2, \cdots, \mathbf{F}^L) = \prod_{i=1}^{L} p(\mathbf{F}^i) \tag{13}
$$

We propose the following optimization problem to obtain the hyperparameters

$$
\begin{aligned}
\tilde{\boldsymbol{\theta}} &= \arg\max_{\boldsymbol{\theta}} \log(p(\mathbf{F}^1, \mathbf{F}^2, \cdots, \mathbf{F}^L)) \\
&= \arg\min_{\boldsymbol{\theta}} (-\log(p(\mathbf{F}^1, \mathbf{F}^2, \cdots, \mathbf{F}^L)))
\end{aligned} \tag{14}
$$

where the objective function is calculated as follows

$$
\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}) =& -\log(p(\mathbf{F}^1, \mathbf{F}^2, \cdots, \mathbf{F}^L)) \\
=& \frac{1}{2}\log|\mathbf{C}| + \frac{N}{2}\log(2\pi) + \frac{1}{2}[\mathbf{F}^1]^T\mathbf{C}^{-1}\mathbf{F}^1 + \\
& \frac{1}{2}\log|\mathbf{C}| + \frac{N}{2}\log(2\pi) + \frac{1}{2}[\mathbf{F}^2]^T\mathbf{C}^{-1}\mathbf{F}^2 + \\
& \qquad \vdots \\
& \frac{1}{2}\log|\mathbf{C}| + \frac{N}{2}\log(2\pi) + \frac{1}{2}[\mathbf{F}^L]^T\mathbf{C}^{-1}\mathbf{F}^L \\
=& \frac{1}{2}\sum_{i=1}^{L}\log|\mathbf{C}| + N\log(2\pi) + [\mathbf{F}^i]^T\mathbf{C}^{-1}\mathbf{F}^i
\end{aligned} \tag{15}
$$

The first-order gradient $\nabla\mathcal{L}(\theta_j)$ is needed for optimization process, where θ_j indicates the j^{th} hyperparameter. Therefore, we derive the gradient of objective function as presented below

$$
\begin{aligned}
&\nabla\mathcal{L}(\theta_j) = \frac{\partial(-\log(p(\mathbf{F}^1, \mathbf{F}^2, \cdots, \mathbf{F}^L)))}{\partial\theta_j} \\
&= -\sum_{i=1}^{L}(\frac{1}{2}[\mathbf{F}^i]^T\mathbf{C}^{-1}\frac{\partial\mathbf{C}}{\partial\theta_j}\mathbf{C}^{-1}[\mathbf{F}^i]^T - \frac{1}{2}\mathrm{tr}(\mathbf{C}^{-1}\frac{\partial\mathbf{C}}{\partial\theta_j})) \\
&= -\frac{1}{2}\sum_{i=1}^{L}\mathrm{tr}((\mathbf{S}^i[\mathbf{S}^i]^T - \mathbf{C}^{-1})\frac{\partial\mathbf{C}}{\partial\theta_j}) \\
&= -\frac{1}{2}\mathrm{tr}((\sum_{i=1}^{L}\mathbf{S}^i[\mathbf{S}^i]^T - L\mathbf{C}^{-1})\frac{\partial\mathbf{C}}{\partial\theta_j}), \text{ where } \mathbf{S}^i = \mathbf{C}^{-1}\mathbf{F}^i
\end{aligned} \tag{16}
$$

where $\mathbf{S}^i$ is calculated for each target independently. The term $\sum_{i=1}^{L}\mathbf{S}^i[\mathbf{S}^i]^T - L\mathbf{C}^{-1}$ plays a transfer knowledge role for optimization of the hyperparameters, and $\nabla\mathcal{L}(\theta_j)$ shows a

sub-optimal direction. Here, the term $\sum_{i=1}^{L} \mathbf{S}^i[\mathbf{S}^i]^T - L\mathbf{C}^{-1}$ also is interpreted as the overfitting reduction part; because it hinders the hyperparameters from adapting to a single target as well. The $\mathbf{S}^{i^{th}}$ can be calculated in matrix form for calculation of $\nabla\mathcal{L}(\theta_j)$ as follows

$$\begin{aligned} &\nabla\mathcal{L}(\theta_j) = -\frac{1}{2}\mathrm{tr}((\mathbf{S}\mathbf{S}^T - L\mathbf{C}^{-1})\frac{\partial\mathbf{C}}{\partial\theta_j}) \\ &\text{where, } \mathbf{S} = \mathbf{C}^{-1}\mathbf{F} \end{aligned} \tag{17}$$

Then, a gradient-based algorithm can be utilized to optimize the hyperparameters (e.g., conjugate gradient algorithm).

3.2 Test phase of JGPR

All targets have the same kernel function with the same hyperparameters during the training and test phases, and due to having a shared covariance matrix, the joint distribution of targets can be expressed as follows

$$\begin{aligned} &\begin{pmatrix} \mathbf{F}^i \\ \hat{\mathbf{F}}^i \end{pmatrix} \sim \mathcal{N}\left[\begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \begin{pmatrix} \mathbf{C} & \mathbf{C}(\mathbf{x}, \hat{\mathbf{x}}) \\ \mathbf{C}(\hat{\mathbf{x}}, \mathbf{x}) & \mathbf{C}(\hat{\mathbf{x}}, \hat{\mathbf{x}}) \end{pmatrix}\right] \\ &i = 1, 2, ..., L \end{aligned} \tag{18}$$

The same scenario of (10) can be used to obtain the posterior distribution for each target. Therefore, we can write the following distribution for prediction of the i^{th} target

$$\begin{aligned} &\boldsymbol{\mu}^i = \mathbf{C}(\hat{\mathbf{x}}, \mathbf{x})\,\mathbf{C}^{-1}\mathbf{F}^i \\ &\boldsymbol{\Phi}^i = \mathbf{C}(\hat{\mathbf{x}}, \hat{\mathbf{x}}) - \mathbf{C}(\hat{\mathbf{x}}, \mathbf{x})\mathbf{C}^{-1}\mathbf{C}(\mathbf{x}, \hat{\mathbf{x}}) \\ &i = 1, 2, ..., L \end{aligned} \tag{19}$$

As can be seen, each target has its own prediction function $\boldsymbol{\mu}^i$, because the real train outputs $\mathbf{F}^i$ are different for each target. However, all targets have the same posterior covariance matrix $\boldsymbol{\Phi}^i$. The posterior covariance matrix is not used for prediction of outputs; however, we will discuss this issue in Sect. 3.3 in more detail.

3.3 Target equalization

The proposed JGPR algorithm is more robust when all targets' values are in the same range. This condition helps the $\nabla\mathcal{L}(\theta_j)$ to extract equal information from each output variable, since the matrix $\mathbf{S}$ is directly related to the gradient direction and computed by $\mathbf{S} = \mathbf{C}^{-1}\mathbf{F}$. However, sometimes the outputs are not in the same range and have diverse effects on the gradient direction $\nabla\mathcal{L}(\theta_j)$. To tackle this problem, the output variables can be normalized or standardized before the training phase. Generally, the scaling function can be written in the following form

$$\mathbf{F}^i_{sc} = \frac{\mathbf{F}^i - \eta_i}{\Omega_i}, \quad i = 1, 2, ..., L \tag{20}$$

where $\mathbf{F}^i$ is the real values of the i^{th} train target, $\mathbf{F}^i_{sc}$ is the scaled values of the i^{th} target, η_i and Ω_i are defined based on standardization or normalization process. In the normalization process, these two parameters are described below

$$\begin{aligned} &\eta_i = \min(\mathbf{F}^i) \\ &\Omega_i = \max(\mathbf{F}^i) - \min(\mathbf{F}^i) \\ &i = 1, 2, ..., L \end{aligned} \tag{21}$$

and in the standardization process, the mentioned parameters are defined as follows

$$\begin{aligned} &\eta_i = \text{mean}(\mathbf{F}^i) \\ &\Omega_i = \text{variance}(\mathbf{F}^i) \\ &i = 1, 2, ..., L \end{aligned} \tag{22}$$

After the normalization process, all outputs fall into the [0, 1] range, and by using the standardization process, the output variables have a zero-mean normal distribution with unit variance. In the training phase, the model is trained via the $\mathbf{F}_{sc}$ instead of $\mathbf{F}$. Therefore in the test phase, the predicted outputs are in the same range as $\mathbf{F}_{sc}$. We can re-scale the distribution of the estimated outputs to real ranges via the following equations

$$\begin{aligned} &\boldsymbol{\mu}^i_{re} = \boldsymbol{\mu}^i_{re} \Omega_i + \boldsymbol{\eta}_i \\ &\boldsymbol{\Phi}^i_{re} = \boldsymbol{\Phi}^i_{re}\ \Omega_i^2 \\ &i = 1, 2, ..., L \end{aligned} \tag{23}$$

where $\boldsymbol{\mu}^i_{re}$ and $\boldsymbol{\Phi}^i_{re}$ are the i^{th} rescaled estimation target and corresponding covariance matrix, respectively. This pre/post-processing also can solve the problem presented in Sect. 3.2, which concerns the equal estimated of posterior covariance matrix for all targets in (24), since the covariance matrices for each target are different for all targets due to the Ω_i^2 term in (23).

3.4 Convergence analysis of JGPR

In this section, we provide a theoretical analysis for the convergence of proposed JGPR algorithm. Initially, we need the below Lemma to provide a rigorous proof for the JGPR.

Lemma *If a sequence is bounded from below and monotonically decreased, it will converge.*

It should be proved that the objective function in (15) is bounded from below. We know that the inequality $0 < \int_{\mathbf{F}^{i\prime}}^{\mathbf{F}^{i}} \int_{\theta'}^{\theta''} p(\mathbf{F}^i, \boldsymbol{\theta})\ d\boldsymbol{\theta} d\mathbf{F}^i < 1$ holds for the i^{th} target, where $\int_{\mathbf{F}^{i\prime}}^{\mathbf{F}^{i}} .\, d\mathbf{F}^i$ and $\int_{\theta'}^{\theta''} .\, d\boldsymbol{\theta}$ are translated to $\int_{\mathbf{F}_N^{i\prime}}^{\mathbf{F}_N^{i''}} \cdots \int_{\mathbf{F}_1^{i\prime}}^{\mathbf{F}_1^{i''}} .d\mathbf{F}^i_N \cdots d\mathbf{F}^i_1$ and $\int_{\gamma'}^{\gamma''} \int_{l'}^{l''} \int_{\sigma_n'}^{\sigma_n''} .\, d\sigma_n dl\, d\gamma$, respectively. The same concept holds for joint distribution of all targets $p(\mathbf{F};\boldsymbol{\theta})$ as follows

$$\begin{aligned} &0 < \int_{\mathbf{F}'}^{\mathbf{F}''} \int_{\theta'}^{\theta''} p(\mathbf{F};\boldsymbol{\theta})\ \ d\boldsymbol{\theta}\ d\mathbf{F} < 1 \\ &0 < \int_{\mathbf{F}'}^{\mathbf{F}''} \int_{\theta'}^{\theta''} \prod_{i=1}^{L} p(\mathbf{F}^i;\boldsymbol{\theta})\ \ d\boldsymbol{\theta}\ d\mathbf{F} < 1 \end{aligned} \tag{24}$$

where the multiple integral $\int_{\mathbf{F}'}^{\mathbf{F}''} .d\mathbf{F}$ is translated to $\int_{\mathbf{F}^{L\prime}}^{\mathbf{F}^{L''}} \int_{\mathbf{F}^{(L-1)\prime}}^{\mathbf{F}^{(L-1)''}} \cdots \int_{\mathbf{F}^{1\prime}}^{\mathbf{F}^{1''}} .d\mathbf{F}^L d\mathbf{F}^{L-1} \cdots d\mathbf{F}^1$.

The above inequality can be multiplied by a constant $\frac{1}{\varsigma}$

$$
\begin{aligned}
0 < \frac{1}{\varsigma}\int_{\mathbf{F}'}^{\mathbf{F}''}\int_{\theta'}^{\theta''}\prod_{i=1}^{L} p(\mathbf{F}^i;\theta)\ d\theta\, d\mathbf{F} < \frac{1}{\varsigma} \\
\varsigma = (\mathbf{F}^{L''} - \mathbf{F}^{L'})\cdots(\mathbf{F}^{1''} - \mathbf{F}^{1'})(\theta'' - \theta')
\end{aligned} \tag{25}
$$

where $(\mathbf{F}^{i''} - \mathbf{F}^{i'})$ and $(\theta'' - \theta')$ are translated to $(\mathbf{F}_N^{i''} - \mathbf{F}_N^{i'})(\mathbf{F}_{N-1}^{i''} - \mathbf{F}_{N-1}^{i'})\cdots(\mathbf{F}_1^{i''} - \mathbf{F}_1^{i'})$ and $(\gamma'' - \gamma')(l'' - l')(\sigma_n'' - \sigma_n')$, respectively. Now, we can get the $-\log$ from the above inequality

$$
\xi < -\log\left(\frac{1}{\varsigma}\int_{\mathbf{F}'}^{\mathbf{F}''}\int_{\theta'}^{\theta''}\prod_{i=1}^{L} p(\mathbf{F}^i;\theta) d\theta d\mathbf{F}\right) \tag{26}
$$

where $\xi = \log(\varsigma)$ and from the Jensen's inequality (Kuczma, 2009) we can write

$$
\begin{aligned}
\xi < \frac{1}{\varsigma}\int_{\mathbf{F}'}^{\mathbf{F}''}\int_{\theta'}^{\theta''} -\log\left(\prod_{i=1}^{L} p(\mathbf{F}^i;\theta)\right) d\theta\, d\mathbf{F} \\
\xi < \frac{1}{\varsigma}\int_{\mathbf{F}'}^{\mathbf{F}''}\int_{\theta'}^{\theta''} \mathcal{L}(\mathbf{F};\theta)\, d\theta\, d\mathbf{F}
\end{aligned} \tag{27}
$$

The middle term of the above inequality is the average of $\mathcal{L}(\mathbf{F},\theta)$ over the ranges of $\mathbf{F}^i$ and θ, where $1 \le i \le L$. For simplicity we use $\mathbf{F}$ to indicate all targets ($\mathbf{F}^{i^{th}}$). Here, there are two possible presumptions for $\mathcal{L}(\mathbf{F},\theta)$

1. $\mathcal{L}(\mathbf{F},\theta)$ is higher than that of ξ for all ranges of $\mathbf{F}$ and θ. In this case, the $\mathcal{L}(\mathbf{F},\theta)$ is bounded from below because of the primary assumption $\mathcal{L}(\mathbf{F},\theta) > \xi$.
2. $\mathcal{L}(\mathbf{F},\theta)$ is higher than that of ξ for some ranges of $\mathbf{F}$ and θ, and smaller than that of ξ for the other ranges. We shall first consider the symbol $\bar{\Delta}$ for sum of the areas where $\mathcal{L}(\mathbf{F},\theta) > \xi$, and the symbol Δ for sum of the areas where $\mathcal{L}(\mathbf{F},\theta) < \xi$. Then with respect to (27), we can conclude that $\bar{\Delta} > \Delta$, and Δ must finite. In order to prove the finiteness of Δ, the inequality (27) can be decomposed into several parts such that $\mathcal{L}(\mathbf{F},\theta)$ does not cross over the ξ on those regions. Therefore, the inequality (27) can be written as follows

$$
\xi < \frac{1}{\mathbf{F}'' - \mathbf{F}'}\frac{1}{\theta'' - \theta'}(\bar{\Delta} + \Delta) \tag{28}
$$

   where $\bar{\Delta} + \Delta/(\mathbf{F}'' - \mathbf{F}')(\theta'' - \theta')$ is the average of $\mathcal{L}(\mathbf{F},\theta)$ over all regions. The inequality (28) holds when $\bar{\Delta} > \Delta$, and Δ must be finite. This means that $\mathcal{L}(\mathbf{F},\theta)$ is bounded from below. Since sum of the regions under the ξ must be finite, it implies the limitation of $\mathcal{L}(\mathbf{F},\theta)$ under the constant ξ.

It can be concluded that the $\mathcal{L}(\mathbf{F},\theta)$ is converged; because, it is bounded from below and monotonically decreases by a gradient based algorithm.

3.5 Complexity analysis

Here, we compare the multiplication complexity of the proposed JGPR and CGPR in multi-target regression problems. The $\mathcal{O}$ notation is used to indicate the order of complexity.

Complexity of CGPR in the training phase: The conjugate gradient algorithm for optimization process only needs $\nabla\mathcal{L}(\boldsymbol{\theta})$ to optimize the hyperparameters, where the complexity of the first-order gradient is $\mathcal{O}(N^3)$ due to the need for inversion of covariance matrix in (8). The other operations such as matrix elements' calculations of $\frac{\partial \mathbf{C}}{\partial \theta_j}$ and $\mathbf{C}$ lead to $\mathcal{O}(MN^2)$ computation cost. Assuming that $M \ll N$, the complexity of CGPR is $\mathcal{O}(N^3)$ in the training phase. Therefore, the implementation of CGPR in multi-target regression algorithm forces $\mathcal{O}(LN^3)$ calculation cost.

Complexity of CGPR in the Test Phase: Calculation of matrix elements in (10), including $\mathbf{C}(\hat{\mathbf{x}}, \hat{\mathbf{x}})$ and $\mathbf{C}(\hat{\mathbf{x}}, \mathbf{x})$ have $\mathcal{O}(M\hat{N}^2)$ and $\mathcal{O}(MN\hat{N})$ costs, respectively. The other conclusions are based on the assumption that $M \ll N$ and $\hat{N} \ll N$ are satisfied. To calculate the $\boldsymbol{\mu}$ we must pay $\mathcal{O}(N^3)$, $\mathcal{O}(N^2)$ and $\mathcal{O}(\hat{N}N)$ computation costs for $\mathbf{A}_1 = \mathbf{C}^{-1}$, $\mathbf{A}_2 = \mathbf{A}_1\mathbf{f}$ and $\mathbf{C}(\hat{\mathbf{x}}, \mathbf{x})\mathbf{A}_2$, respectively. Therefore, the complexity of calculating $\boldsymbol{\mu}$ is $\mathcal{O}(N^3)$. Also, to calculate the $\boldsymbol{\Phi}$ we must pay $\mathcal{O}(N^3)$, $\mathcal{O}(N^2\hat{N})$ and $\mathcal{O}(\hat{N}^2N)$ computation costs for $\mathbf{A}_1 = \mathbf{C}^{-1}$, $\mathbf{A}_2 = \mathbf{A}_1\mathbf{C}^T(\hat{\mathbf{x}}, \mathbf{x})$ and $\mathbf{C}(\hat{\mathbf{x}}, \mathbf{x})\mathbf{A}_2$, respectively. Therefore, the complexity of calculating $\boldsymbol{\Phi}$ is $\mathcal{O}(N^3)$. When this process implemented for L targets, the complexity will be $\mathcal{O}(LN^3)$ for both $\boldsymbol{\mu}$ and $\boldsymbol{\Phi}$.

Complexity of JGPR in the training phase: The proposed JGPR in the training phase is not implemented for each target independently. We use a shared covariance matrix for all targets and need the gradient of the introduced objective function, which has been stated in (17). To calculate the gradient of objective function in (17), we must pay $\mathcal{O}(LN^2 + N^3)$ cost for $\mathbf{S} = \mathbf{C}^{-1}\mathbf{F}$. Therefore, if $L \leq N$, the complexity of calculating $\mathbf{S}$ is $\mathcal{O}(N^3)$. Also, the complexity calculation of $\mathbf{S}\mathbf{S}^T$ is $\mathcal{O}(LN^2)$. By doing so, the complexity of calculating $\nabla\mathcal{L}(\boldsymbol{\theta})$ equals to $\mathcal{O}(N^3)$. While this process is not repeated for each target independently. Thus, the complexity of JGPR in the training phase equals to $\mathcal{O}(N^3)$.

Complexity of JGPR in the test phase: To calculate the $\boldsymbol{\mu}^i$ in (19), the $\mathbf{C}(\hat{\mathbf{x}}, \mathbf{x})\,\mathbf{C}^{-1}$ is the same for all targets, which forces $\mathcal{O}(N^3)$ complexity due to inversion of the training covariance matrix. Then, it is multiplied by $\mathbf{F}^i$ with complexity of $\mathcal{O}(\hat{N}N)$, and for the L targets the complexity equals to $\mathcal{O}(L\hat{N}N)$. If $\hat{N} \ll N$ and $\hat{L} \ll N$, the complexity calculation of targets equals to $\mathcal{O}(N^3)$. Since the posterior covariance matrix $\boldsymbol{\Phi}^i$ is equal for all targets in (19), the complexity of this process is $\mathcal{O}(N^3)$.

4 Experiments

In this section, we perform extensive experiments in both the simulated data and real-world datasets to show the effectiveness of the proposed JGPR algorithm. Accuracy is measured in terms of relative root mean squared error (RRMSE). The simulated data figuratively helps to understand the effect of simultaneous optimization in the training phase of JGPR to reduce the overfitting problem, and the real-world datasets show the robustness of the proposed algorithm compared with that of others.

4.1 Evaluation metric

In the first experiment, two toy problems are investigated to compare our proposed JGPR algorithm with the CGPR. In the second experiment, the JGPR is scrutinized on 18

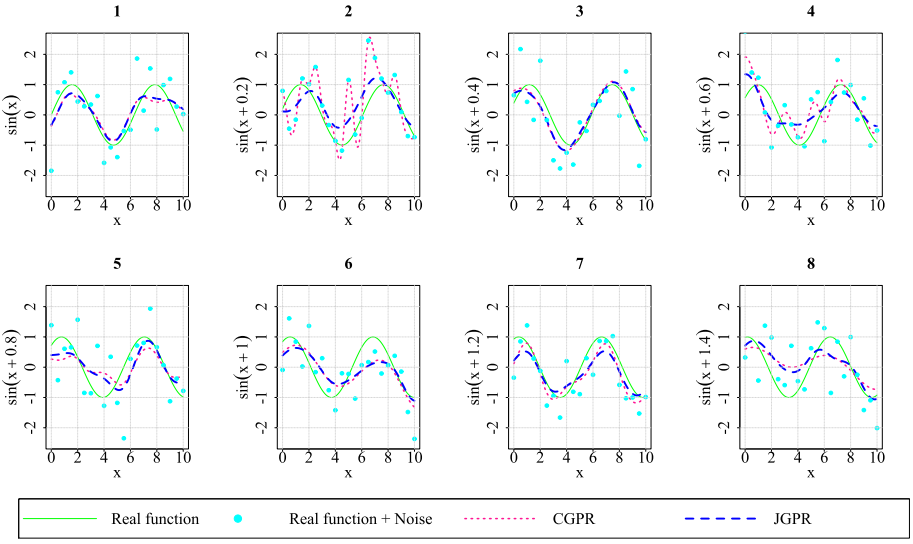

Fig. 3 Simulation results for the toy problem in experiment 1, consists of 8 shifted sinusoidal functions

Table 1 Comparison of CGPR and proposed JGPR in terms of RRMSE, where Ti is the i^{th} target

	T1	T2	T3	T4	T5	T6	T7	T8	aRRMSE
JGPR	**0.428**	**0.619**	**0.339**	**0.583**	**0.579**	0.639	**0.441**	**0.637**	**0.533**
CGPR	0.479	1.162	0.361	0.762	0.684	**0.624**	0.442	0.718	0.654

Bold numbers are the best results

real-world benchmark datasets and compared with other algorithms. The RRMSE metric is used for the experiments, which is defined as follows

$$\text{RRMSE} = \sqrt{\frac{\sum_{i=1}^{\hat{N}} \left(y_i - \hat{y}_i\right)^2}{\sum_{i=1}^{\hat{N}} \left(y_i - \bar{y}\right)^2}} \tag{29}$$

where $\hat{N}$ is the number of test observations, $\hat{y}_i$ is the i^{th} estimated test value, y_i is the i^{th} real test value, and $\bar{y}$ is the average of outputs within the training set. In this paper, $\mathbf{y} = \hat{\mathbf{f}}$ and $\hat{\mathbf{y}} = \boldsymbol{\mu}$. For multi-target regression problems, the RRMSE is independently computed for each output variable, and the averages of RRMSEs (aRRMSE) are reported.

The JGPR has been written in the R programming language[1] using Microsoft Open R (MOR version 3.5.3) due to its parallel computation ability and high-speed performance. We run the implemented code on 64-bit operating system PC with the processor Intel(R) Core(TM) i7-6700 CPU @ 3.4 GHz, and 16 GB RAM.

[1] The code is available in: https://github.com/m-nabati/JGPR/tree/main

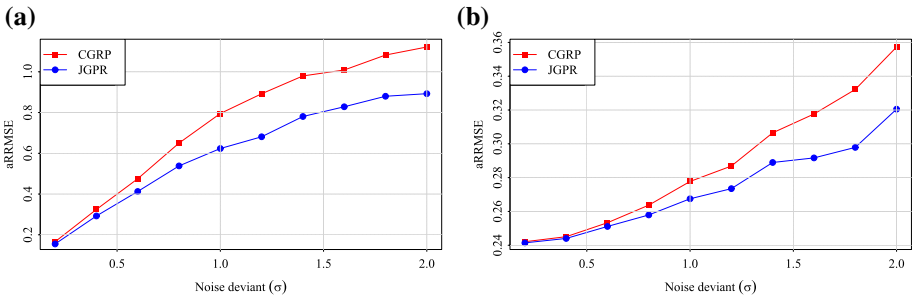

Fig. 4 Comparing CGPR and JGPR algorithms with different noise deviants for **a** toy problem 1 in Eq. 30 and **b** toy problem 2 in Eq. 31

4.2 Experiment 1

In this experiment, we use 8 shifted sinusoidal functions for generating multi-target samples. Then, Gaussian noise with zero mean and 0.8 deviant is added to the generated samples of these 8 functions independently as follows

$$\mathbf{F}^i = \sin(x + 0.2(i-1)) + \varepsilon_i, \quad i = 1, 2, ..., 8 \tag{30}$$

where $\varepsilon_i \sim \mathcal{N}(0,\ 0.8)$. Here, we compare the JGPR with CGPR in terms of RRMSE, which is depicted in Table 1. The last column in this table shows the average RRMSE (aRRMSE) over targets. As can be seen in Table 1, the proposed JGPR on most targets has better performance compared with CGPR. The reason is that sometimes the CGPR algorithm is overfitted in the training phase, as depicted in Fig. 3. In this figure, the green solid lines show real sine functions, the cyan points are noisy observations which are used for the training phase, deep pink dotted lines are the outputs predicted by CGPR, and dashed blue lines are outputs predicted by the JGPR algorithm.

To more precisely compare the CGPR and JGPR, we run the algorithms with different noise levels (noise deviants). At each noise deviant, the algorithms are run with different seeds, and the average of aRRMSE is plotted in Fig. 4a. As can be seen, the proposed JGPR has better performance, especially when the noise deviant is large. In the next step, we run the algorithms in another toy problem. We test the algorithms over the following targets

$$\mathbf{F}^i = \sin(6x + 0.2(i-1)) + x + \varepsilon_i, \quad i = 1, 2, ..., 8 \tag{31}$$

where ε_i is the noise added to the real function. The aRRMSE of algorithms are plotted in Fig. 4b for this problem. As can be seen, the JGPR again achieves better results compared with CGPR in the new toy problem, and this shows the robustness of the proposed method.

4.3 Experiment 2

In this experiment, we use 18 real-world datasets[2] to compare the proposed JGPR with the other algorithms. We directly report the results from Spyromitros-Xioufis et al. (2016) and Zhen et al. (2018), and follow the same simulation methodologies (such as using

[2] The datasets can be downloaded from: https://github.com/lefman/mulan-extended/tree/master/datasets

cross-validation, choosing the number of folds, choosing targets of datasets, etc.) that have been performed by Spyromitros-Xioufis et al. (2016). In the following, we describe the datasets and the strategies which are conducted for simulations.

Solar Flare (SF): The solar flare dataset (Lichman, 2013) consists of two versions, and the goal is to predict 3 output variables from 10 input features. The outputs are not continuous variables. However, the regression algorithms can be used to predict these categorical variables.

JURA: In this dataset (Goovaerts et al., 1997), seven heavy metals concentrations, in addition to the land and rock types, have been collected at 359 locations. Three out of seven metals that are more expensive than the others are used for output variables, and the outputs are continuous.

Water Quality (WQ): The water quality dataset (Džeroski et al., 2000) consists of 14 output variables that have been gathered for six years from most of the Slovenian rivers, and the outputs are categorical variables. Also, there are 16 physical and chemical input features to predict the outputs.

Energy Building (ENB): The energy building dataset (Tsanas and Xifara, 2012) has been generated by Ecotect software on 12 buildings. The goal is to find a relation between building features (including surface area, wall area, etc.) and two continuous output variables (heating cool and loading cool). These two variables are needed to maintain comfortable indoor air conditions.

Electrical Discharge Machining (EDM): The EDM dataset (Karalič and Bratko, 1997) has two categorical output variables and 16 continuous input features. The goal is to decrease the machining time via reproducing human operator behavior that adjusts the values of output variables.

SLUMP: In this dataset (Yeh, 2007), the slump flow is used as an output variable to create high-performance concrete. However, Spyromitros-Xioufis et al. (2016) use 3 output variables. Both of the input features and output variables are continuous.

ANDRO: The Andromeda dataset (Hatzikos et al., 2008) has been collected by the under-water system based on a set of sensors. The goal is to predict the quality of seawater, which is determined by 6 continuous variables (the outputs) based on 30 continuous input features.

Occupational Employment Survey (OES): There are two versions for the OES dataset (Spyromitros-Xioufis et al., 2016), which have been collected in 1997 (OES97) and 2010 (OES10). In both versions, each sample includes full-time equivalent employees across many employment types. Among the employment types, 10 variables have randomly been selected as outputs.

OSALES: The goal of this dataset (Spyromitros-Xioufis et al., 2016) is to find a function for the prediction of online sales of consumer products. There are 12 output variables and 401 input features. The missing values are imputed by samples' mean.

See Click Predict Fix (SCPF): In this dataset (Spyromitros-Xioufis et al., 2016), the goal is to find the number of clicks, views, and comments (The outputs), which show the reaction of people on 311 topics. There are 23 input features (including city, source, etc.) and the missing values are imputed by samples' mean.

Airline Ticket Price (ATP): There are two versions for this dataset (Spyromitros-Xioufis et al., 2016). One of them (ATP1D) concerns the prediction of the ticket price for the next day, and the goal of the other one (ATP7D) is to predict the minimum ticket price after 7 days with 411 input features.

River Flow (RF): In the river flow dataset (Spyromitros-Xioufis et al., 2016), the goal is to find the river network flows for 48 hours in the future based on previous states in the

Table 2 A summary description of 18 benchmark datasets used in experiments. Cat O shows the type of outputs whether they are categorical or continuous. AAC indicates the average of absolute correlation, which is defined in (33)

Dataset	Observation (N)	Inputs (M)	Targets (L)	k-fold cv	Cat O	AAC
SF1	323	10	3	10	✓	0.231
SF2	1066	10	3	10	✓	0.199
JURA	359	15	3	10	✗	0.199
WQ	1060	16	14	10	✓	0.095
ENB	768	8	2	10	✗	0.975
EDM	154	16	2	10	✓	0.005
SLUMP	103	7	3	10	✗	0.417
ANDRO	49	30	6	10	✗	0.396
OES97	334	263	16	10	✗	0.786
OES10	403	298	16	10	✗	0.822
OSALES	639	401	12	10	✓	0.621
SCPF	1137	23	3	10	✓	0.734
ATP1D	337	411	6	10	✗	0.795
ATP7D	296	411	6	10	✗	0.677
RF1	9125	64	8	5	✗	0.391
RF2	9125	576	8	5	✗	0.391
SCM1D	9803	280	16	2	✗	0.638
SCM20D	8966	61	16	2	✗	0.596

past (6, 12, 18, 24, 30, 36, 48 past hours). There are two versions for this dataset (RF1 and RF2), and the missing values are imputed by samples' mean.

Supply Chain Management (SCM): The SCM dataset (Spyromitros-Xioufis et al., 2016) has two versions with the goal of predicting the mean price for the next day (SCM1D) and 20 days (SCM20D) in the future. There are 280 and 61 input features for SCM1D and SCM20D, respectively.

A summary of these datasets is stated in Table 2. Due to the high computations, two-fold and five-fold cross-validation is used for large datasets[3]. Some of the outputs are not continuous, and the ranges of these values are limited; therefore, we tick them as categorical variables. We also define a criterion, named the average of absolute correlation (AAC), which shows the average of targets' correlation for a dataset. The Pearson correlation coefficient (Benesty et al., 2009) between two vector variables is calculated as follows

[3] Since large datasets have enough samples, it is reliable to perform the experiments with 2-fold and 5-fold cross-validation, and this reliability previously has been reported by Spyromitros-Xioufis et al. (2016).

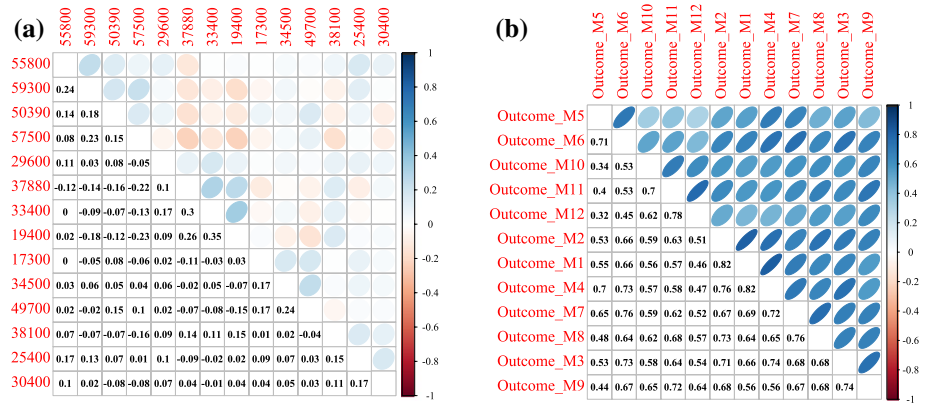

Fig. 5 pairwise correlation between targets of **a** WQ and **b** OSALES datasets. The right-skewed shows the positive correlation and the left-skewed means negative correlation

$$\rho(\mathbf{u},\mathbf{w}) = \frac{E\left[(\mathbf{u}-\bar{u})^T(\mathbf{w}-\bar{w})\right]}{\sqrt{E\left[(\mathbf{u}-\bar{u})^T(\mathbf{u}-\bar{u})\right]E\left[(\mathbf{w}-\bar{w})^T(\mathbf{w}-\bar{w})\right]}} = \frac{\sum_{i=1}^{N}(u_i-\bar{u})(w_i-\bar{w})}{\sqrt{\sum_{i=1}^{N}(u_i-\bar{u})^2\sum_{i=1}^{N}(w_i-\bar{w})^2}} \tag{32}$$

where N is the number of all samples, $\mathbf{u} = [u_1, u_2, ..., u_N]$ and $\mathbf{w} = [w_1, w_2, ..., w_N]$ are two output vectors, $\bar{u}$ and $\bar{w}$ are averages of $\mathbf{u}$ and $\mathbf{w}$, respectively. The AAC is defined as follows

$$\text{AAC} = \frac{1}{\omega}\sum_{i=1}^{L}\sum_{j=2}^{i}\left|\rho(\mathbf{F}^i,\mathbf{F}^j)\right| \tag{33}$$

where $\omega = \frac{L^2-L}{2}$ is the number of elements in lower triangular of the correlation matrix. L^2 is the number of all elements in the correlation matrix, L the number of diagonal elements must be subtracted from the number of all elements. Considering only the lower triangular elements, the result $L^2 - L$ must be divided by 2. The summations $\sum_{i=1}^{L}\sum_{j=2}^{i}$ probe the elements in the lower triangular matrix, and $|.|$ returns the absolute value.

Table 3 aRRMSE performance of proposed JGPR algorithm compared with others

	MMR	ST	SST	ERC	RLC	MORF	MSVR	AKRF	MTFL	MROTS	OKL	CGPR	JGPR
SF1	0.958	1.135	1.068	1.089	1.163	1.282	1.021	1.114	1.112	1.155	1.059	0.871	**0.826**
SF2	0.984	1.149	1.055	1.088	1.228	1.425	1.043	1.135	1.127	1.201	1.004	0.860	**0.820**
JURA	0.582	0.589	0.591	0.590	0.596	0.597	0.611	0.618	0.608	0.625	0.599	0.657	**0.577**
WQ	**0.889**	0.908	0.909	0.906	0.902	0.899	0.899	0.918	0.962	0.913	0.891	0.921	0.918
ENB	0.111	0.117	0.121	0.114	0.120	0.121	0.220	0.234	0.316	0.257	0.138	0.294	**0.063**
EDM	0.716	0.742	0.740	0.741	0.735	0.734	0.737	0.740	0.851	0.812	0.741	0.846	**0.697**
SLUMP	0.587	0.688	0.695	0.689	0.690	0.694	0.711	0.729	0.681	0.778	0.699	0.620	**0.568**
ANDRO	0.527	0.602	0.579	0.567	0.570	0.510	0.627	0.623	0.803	0.635	0.553	0.707	**0.501**
OES97	0.497	0.525	0.524	0.524	0.523	0.549	0.557	0.581	0.818	0.605	0.535	0.555	**0.465**
OES10	0.403	0.420	0.421	0.420	0.419	0.452	0.447	0.446	0.532	0.558	0.432	0.436	**0.390**
OSALES	**0.709**	0.748	0.726	0.713	0.741	0.753	0.778	0.775	1.682	0.800	0.718	0.852	0.798
SCPF	0.812	0.837	0.831	0.830	0.835	0.833	0.828	0.831	0.899	0.901	0.820	0.844	**0.808**
ATP1D	0.332	0.374	0.372	0.372	0.384	0.422	0.381	0.412	0.415	0.404	0.364	0.294	**0.226**
ATP7D	0.443	0.525	0.507	0.512	0.461	0.551	0.477	0.531	0.553	0.549	0.475	0.266	**0.170**
RF1	**0.089**	0.097	0.094	0.091	0.121	0.123	0.109	0.114	0.983	0.154	0.112	0.140	**0.089**
RF2	0.095	0.102	0.097	0.095	0.130	0.148	0.144	0.157	1.103	0.198	0.118	0.275	**0.090**
SCM1D	0.318	0.348	0.336	0.330	0.345	0.352	0.367	0.368	0.437	0.449	0.342	0.387	**0.309**
SCM20D	0.389	0.475	0.413	0.394	0.443	0.482	0.493	0.655	0.643	0.456	0.443	0.584	**0.377**

Bold numbers are the best results

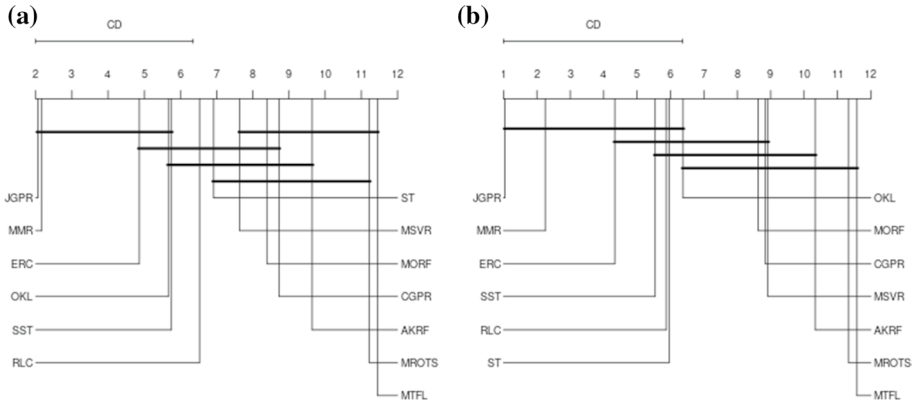

Fig. 6 Friedman-nemanyi test for **a** all datasets and **b** excluding the datasets with categorical targets

We compare the JGPR with other algorithms including multi-layer multi-target regression (MMR) (Zhen et al., 2018), single-target regression (ST) (Spyromitros-Xioufis et al., 2016), stacked single target (SST) (Spyromitros-Xioufis et al., 2016), ensemble of regressor chains (ERC) (Spyromitros-Xioufis et al., 2016), random linear target combinations (RLC) (Tsoumakas et al., 2014), multi-object random forests (MORF) (Kocev et al., 2007), multi-dimensional support vector regression (M-SVR) (Sánchez-Fernández et al., 2004), multi-task feature learning (MTFL) (Argyriou et al., 2007), multi-output regression with output and task structures (MROTS) (Rai et al., 2012), output kernel learning (OKL) (Dinuzzo et al., 2011) and CGPR.

The aRRMSE of algorithms for 18 benchmark datasets have been reported in Table 3. As can be seen, the JGPR has better accuracy on most of the given benchmark datasets unless in WQ and OSALES datasets. The experiments show that the JGPR does not work properly for the categorical output variables as well as continuous ones. The WQ and OSALES datasets contain the categorical variables on their targets. Also, the experimental results show that the JGPR does not need any correlation between the targets. Fig. 5a shows the pairwise correlation between the output variables of the WQ dataset. As shown, the correlations between the outputs are small. On the other side, the OSALES dataset has sufficient correlation on its outputs, while JGPR does not increase the accuracy of this dataset as well as the other methods. In Fig 5b, we plot the pairwise correlation between outputs for the OSALES dataset. We conclude that the categorical variable is a poisonous factor for the proposed JGPR, while it outperforms the accuracy over the datasets with continuous output variables.

In order to investigate the effect of output type, we have performed the friedman-nemenyi test (Demšar, 2006) on the results of Table 3. Two different tests have been performed: one test with all datasets and the other by excluding the datasets with categorical variables. In both experiments, the null hypothesis is rejected ($p < 0.05$), and we can further proceed with the nemanyi post-hoc test. Instead of pair-wise comparison, we draw the visualization of result proposed by Demšar (2006), which is known to critical difference (CD) plot. The results have been plotted in Fig. 6 for these two tests. A CD plot shows the average rank and group of different algorithms, which are connected with distinct links. In Fig. 6a the proposed JGPR and MMR are in the same group, and their average rank is similar. The reason is that the proposed method does not have a good performance on the WQ and OSALES datasets, which have

categorical targets. In Fig. 6b, we excluded the datasets with categorical targets. Although the proposed JGPR is again in the same group as MMR, it stands in rank 1. It is a clear evidence that a statistical test may not be able to find the significance of an algorithm. In other words, the statistical test can say a group of algorithms are different from the other groups, and not finding a difference between algorithms does not mean that their performance is equivalent (Carrasco et al., 2020).

Besides the performance improvement, the proposed method does not implicate higher complexity for performing the optimization and prediction. Here, we compare the complexity of the proposed method with those algorithms that have provided the complexity analysis for their models. The ST refers to the single target regression for each output independently and could be any machine learning algorithm. According to the reports of Spyromitros-Xioufis et al. (2016), which bagged regression trees has a better performance compared with other methods, we report the complexity of ST_{BAG} which equals to $\mathcal{O}(NM^2)$ and $\mathcal{O}(N \log_2 M)$ for the training and test phases of a single target regression problem, respectively. For multi-target regression problems with L targets, the ST has $\mathcal{O}(LNM^2)$ and $\mathcal{O}(LN \log_2 M)$ complexity for training and test phases. SST and ERC have the same complexity level for multi-target regression problems. According to Zhen et al. (2018), the MMR implies the complexity of $\mathcal{O}(N^3)$ in the training phase. The proposed method has a better performance compared with MMR in most of the given benchmark datasets, and it has the same complexity level as MMR.

5 Conclusion

In this paper, we studied the multi-task learning and multi-target regression problems and presented the main difference between them. We emphasized that the input features are different for the targets in multi-task learning problems, while for multi-target regression is not the case. We proposed a novel multi-target regression-based GPR algorithm, named joint GPR (JGPR), which solves a sub-optimal cost function to optimize the hyperparameters of a shared covariance matrix between the targets. The proposed method improved the accuracy of conventional GPR in toy problem and the other state-of-the-art approaches on 16 out of 18 benchmark datasets. Experiments show that the JGPR is not overfitted over each target during the training phase. Solving the sub-optimal solution also helped to reduce the complexity. In the test phase, we did not use other outputs' information, while it can be considered as future work to design an algorithm for capturing the other targets' information in the test phase.

Appendix: detailed results for proposed JGPR method

	c-class	m-class	x-class
SF1	1.014	0.990	0.474

	c-class	m-class	x-class
SF2	1.049	1.004	0.408

	Cd	Co	Cu
JURA	0.648	0.588	0.495

	25400	29600	30400	33400	17300	19400	34500	38100	49700	50390	55800	57500	59300	37880
WQ	0.937	0.971	0.946	0.883	0.916	0.871	0.970	0.922	0.819	0.906	0.935	0.918	0.954	0.906

	Y1	Y2
ENB	0.058	0.069

	DFlow	DGap
EDM	0.599	0.795

	SLUMP_cm	FLOW_cm	Compressive_Strength_Mpa
SLUMP	0.814	0.730	0.162

	Target	Target_2	Target_3	Target_4	Target_5	Target_6
ANDRO	0.362	0.355	0.490	0.499	0.639	0.668

	58028	15014	32511	15017	98502	92965	32314	13008	21114	85110	27311	98902	65032	92998	27108	53905
OES97	0.180	0.263	0.868	0.312	0.622	0.621	0.697	0.232	0.182	0.466	0.563	0.449	0.477	0.511	0.518	0.489

	513021	292071	392021	151131	151141	291069	119032	432011	419022	292037	519061	291051	172141	431011	291127	412021
OES10	0.362	0.321	0.420	0.440	0.403	0.596	0.310	0.337	0.574	0.297	0.346	0.206	0.558	0.189	0.481	0.409

	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12
OSALES	0.747	0.711	0.805	0.789	0.829	0.809	0.781	0.827	0.838	0.812	0.804	0.828

	num_views	num_votes	num_comments
SCPF	0.753	0.730	0.942

	'ALLminpA'	'ALLminp0'	'aCOminpA'	'aDLminpA'	'aFLminpA'	'aUAminpA'
ATP1D	0.338	0.148	0.160	0.231	0.340	0.141

	'ALLminpA'	'ALLminp0'	'aCOminpA'	'aDLminpA'	'aFLminpA'	'aUAminpA'
ATP7D	0.234	0.025	0.176	0.183	0.245	0.158

	CHSI2__0	NASI2__0	EADM7__0	SCLM7__0	CLKM7__0	VALI2__0	NAPM7__0	DLDI4__0
RF1	0.019	0.497	0.028	0.022	0.041	0.042	0.036	0.030

	CHSI2__0	NASI2__0	EADM7__0	SCLM7__0	CLKM7__0	VALI2__0	NAPM7__0	DLDI4__0
RF2	0.016	0.470	0.029	0.025	0.049	0.057	0.043	0.034

	LBL	MTLp2	MTLp3	MTLp4	MTLp5	MTLp6	MTLp7	MTLp8	MTLp9	MTLp10	MTLp11	MTLp12	MTLp13	MTLp14	MTLp15	MTLp16
SCM1D	0.268	0.292	0.289	0.302	0.310	0.331	0.321	0.339	0.292	0.313	0.305	0.337	0.306	0.328	0.301	0.312

	LBL	MTLp2A	MTLp3A	MTLp4A	MTLp5A	MTLp6A	MTLp7A	MTLp8A	MTLp9A	MTLp10A	MTLp11A	MTLp12A	MTLp13A	MTLp14A	MTLp15A	MTLp16A
SCM20D	0.338	0.353	0.363	0.371	0.378	0.394	0.379	0.380	0.371	0.394	0.384	0.407	0.392	0.395	0.368	0.379

Author Contributions MN proposed and implemented the idea and wrote the first draft of the paper. SAG managed the project and revised the paper, and RS has contributed to the revision of the paper.

Funding Not Applicable.

Data availability The datasets can be downloaded from: https://github.com/lefman/mulan-extended/tree/master/datasets

Declaration

Conflict of interest The authors declare that they have no conflict of interest.

Code availability The implementation of the proposed JGPR is available at: https://github.com/m-nabati/JGPR/tree/main (R programming language).

References

Álvarez, M., Luengo, D., Titsias, M., Lawrence, N. D. (2010). Efficient multioutput gaussian processes through variational inducing kernels. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, pp. 25–32.

Álvarez, M. A., & Lawrence, N. D. (2011). Computationally efficient convolved multiple output gaussian processes. *The Journal of Machine Learning Research, 12*, 1459–1500.

Ambikasaran, S., Foreman-Mackey, D., Greengard, L., Hogg, D. W., & O'Neil, M. (2016). Fast direct methods for gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 38*(2), 252–265.

Argyriou, A., Evgeniou, T., Pontil, M. (2007) Multi-task feature learning. In Advances in neural information processing systems, pp. 41–48.

Benesty, J., Chen, J., Huang, Y., Cohen, I. (2009) Pearson correlation coefficient. In Noise reduction in speech processing, Springer, pp. 1–4.

Bonilla, E. V., Chai, K. M., Williams, C. (2008). Multi-task gaussian process prediction. In Advances in neural information processing systems, pp. 153–160.

Carrasco, J., García, S., Rueda, M., Das, S., & Herrera, F. (2020). Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review. *Swarm and Evolutionary Computation, 54*, 100665.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research, 7*, 1–30.

Dinuzzo, F., Ong, C. S., Pillonetto, G., Gehler, P. V. (2011). Learning output kernels with block coordinate descent. In Proceedings of the 28th International Conference on Machine Learning (ICML-11), pp. 49–56.

Džeroski, S., Demšar, D., & Grbović, J. (2000). Predicting chemical parameters of river water quality from bioindicator data. *Applied Intelligence, 13*(1), 7–17.

Goovaerts, P., et al. (1997). *Geostatistics for natural resources evaluation*. Oxford University Press on Demand.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction*. Springer.

Hatzikos, E. V., Tsoumakas, G., Tzanis, G., Bassiliades, N., & Vlahavas, I. (2008). An empirical study on sea water quality prediction. *Knowledge-Based Systems, 21*(6), 471–478.

Karalič, A., & Bratko, I. (1997). First order regression. *Machine learning, 26*(2–3), 147–176.

Kocev, D., Vens, C., Struyf, J., Džeroski, S. (2007) Ensembles of multi-objective decision trees. In European conference on machine learning, Springer, pp. 624–631.

Kuczma, M. (2009). *An introduction to the theory of functional equations and inequalities: Cauchy's equation and Jensen's inequality*. Springer Science & Business Media.

Li, G., Hoi, S. C. H., Chang, K., Liu, W., & Jain, R. (2014). Collaborative online multitask learning. *IEEE Transactions on Knowledge and Data Engineering, 26*(8), 1866–1876.

Lichman M (2013) Uci machine learning repository. http://archive.ics.uci.edu/ml

Liu, Y., Starzyk, J. A., & Zhu, Z. (2008). Optimized approximation algorithm in neural networks without overfitting. *IEEE Transactions on Neural Networks, 19*(6), 983–995.

Melki, G., Cano, A., Kecman, V., & Ventura, S. (2017). Multi-target support vector regression via correlation regressor chains. *Information Sciences, 415*, 53–69.
Nabati, M., Ghorashi, S. A., & Shahbazian, R. (2021). Joint coordinate optimization in fingerprint-based indoor positioning. *IEEE Communications Letters, 25*(4), 1192–1195. https://doi.org/10.1109/LCOMM.2020.3047352
Nguyen, T. N. A., Bouzerdoum, A., & Phung, S. L. (2018). Stochastic variational hierarchical mixture of sparse gaussian processes for regression. *Machine Learning, 107*(12), 1947–1986.
Nguyen, T. V., Bonilla, E. V., et al. (2014). Collaborative multi-output gaussian processes. In UAI, pp. 643–652.
Nocedal, J., & Wright, S. (2006). *Numerical optimization*. Springer Science & Business Media.
Petković, M., Kocev, D., & Džeroski, S. (2020). Feature ranking for multi-target regression. *Machine Learning, 109*(6), 1179–1204.
Rai. P., Kumar, A., Daume, H. (2012) Simultaneously leveraging output and task structures for multiple-output regression. In Advances in Neural Information Processing Systems, pp. 3185–3193.
Sánchez-Fernández, M., de Prado-Cumplido, M., Arenas-García, J., & Pérez-Cruz, F. (2004). Svm multiregression for nonlinear channel estimation in multiple-input multiple-output systems. *IEEE transactions on signal processing, 52*(8), 2298–2307.
Spyromitros-Xioufis, E., Tsoumakas, G., Groves, W., & Vlahavas, I. (2016). Multi-target regression via input space expansion: treating targets as inputs. *Machine Learning, 104*(1), 55–98.
Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research, 15*(1), 1929–1958.
Struyf, J., & Džeroski, S. (2005). Constraint based induction of multi-objective regression trees. In International Workshop on Knowledge Discovery in Inductive Databases, Springer, pp. 222–233.
Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological), 58*(1), 267–288.
Tsanas, A., & Xifara, A. (2012). Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. *Energy and Buildings, 49*, 560–567.
Tsoumakas, G., Spyromitros-Xioufis, E., Vrekou, A., Vlahavas, I. (2014) Multi-target regression via random linear target combinations. In Joint european conference on machine learning and knowledge discovery in databases, Springer, pp. 225–240.
Williams, C. K., & Rasmussen, C. E. (2006). *Gaussian processes for machine learning*. MIT press Cambridge.
Xu, Y., Li, X., Chen, D., & Li, H. (2018). Learning rates of regularized regression with multiple gaussian kernels for multi-task learning. *IEEE Transactions on Neural Networks and Learning Systems, 29*(11), 5408–5418.
Yeh, I. C. (2007). Modeling slump flow of concrete using second-order regressions and artificial neural networks. *Cement and concrete composites, 29*(6), 474–480.
Zhang, Y., & Yang, Q. (2017) A survey on multi-task learning. arXiv preprint arXiv:1707.08114
Zhang, Y., & Yang, Q. (2021). A survey on multi-task learning. IEEE Transactions on Knowledge and Data Engineering pp. 1.
Zhen, X., Yu, M., He, X., & Li, S. (2018). Multi-target regression via robust low-rank learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 40*(2), 497–504.