Abstract—Gaussian process regression (GPR) is a fundamental model used in machine learning (ML). Due to its accurate prediction with uncertainty and versatility in handling various data structures via kernels, GPR has been successfully used in various applications. However, in GPR, how the features of an input contribute to its prediction cannot be interpreted. Here, we propose GPR with local explanation, which reveals the feature contributions to the prediction of each sample while maintaining the predictive performance of GPR. In the proposed model, both the prediction and explanation for each sample are performed using an easy-to-interpret locally linear model. The weight vector of the locally linear model is assumed to be generated from multivariate Gaussian process priors. The hyperparameters of the proposed models are estimated by maximizing the marginal likelihood. For a new test sample, the proposed model can predict the values of its target variable and weight vector, as well as their uncertainties, in a closed form. Experimental results on various benchmark datasets verify that the proposed model can achieve predictive performance comparable to those of GPR and superior to that of existing interpretable models and can achieve higher interpretability than them, both quantitatively and qualitatively.

Index Terms—Explainability, feature relevance, Gaussian processes (GPs), interpretable machine learning (ML), locally linear models.

I. INTRODUCTION

GAUSSIAN processes (GPs) have been well studied for constructing probabilistic models as priors of nonlinear functions in the machine learning (ML) community. They have demonstrated great success in various problem settings, such as regression [1], [2], classification [1], [3], time-series forecasting [4], and black-box optimization [5]. A fundamental model on GPs is Gaussian process regression (GPR) [1]; due to its high predictive performances and versatility in using various data structures via kernels, it has been used in not only the ML community but also in various other research areas, such as finance [6], geostatistics [7], material science [8], and medical science [9], [10].

GPR is defined on an infinite-dimensional feature space via kernel functions. Therefore, it requires the values of the kernels defined on pairs of samples, i.e., a covariance matrix of the samples as an input, rather than the samples themselves. Due to the nonlinearity of the kernel, GPR enables nonlinear predictions. In terms of interpretability, the covariance is useful for understanding the relationship between the samples; however, since the kernels make the features invisible, GPR cannot explain which features contribute to the predictions, such as linear regression models. Therefore, it prevents us from judging whether the predictions by GPR are performed to use the features that are likely to be correct from users’ point of view and performed by fair decision, where “fair” means that the features used in the predictions do not include discrimination.

The objective of this article is to “explain” the feature contributions to the predictions made by GPR while maintaining its own predictive power. The explanations, i.e., the feature contributions to the predictions, should be “faithful” and “stable.” Faithful explanation means that explanations are indicative of true importance in predictions, as defined in [11]. This indicates that the explanation is required to change for each sample since complex real-world behaviors that appear in data cannot be represented by a single explanation. Stable explanation means that explanations are similar for similar samples, as defined in [12]. In other words, it means that the explanation is robust to slight variation in the features appearing in the sample. High stability is desirable for users to trust the model and its prediction.

Several methodologies that explain the features that contribute to the outputs of flexible prediction models, including GPR, have been proposed; in this case, the prediction models are often regarded as black boxes [12], [13]. Their representative methods are local interpretable model-agnostic explanations (LIME) [14] and Shapley additive explanations (SHAP) [15], which approximates the prediction for each test sample by a locally linear explanation model. Since the weights of the learned explanation model represent feature contributions to the prediction, they can assist ML practitioners and scientists to understand the behavior of the prediction models and the functionality of the features in the prediction. However, some limitations regarding faithfulness and stability exist in these methods. First, because the forms of the prediction and explanation models differ, it is unsure whether the estimated feature contributions reflect those of the prediction model. Second, because the explanation model is learned on each test sample, it may not obtain consistent explanations on similar samples.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
To overcome the aforementioned limitations, we propose a novel framework for GP-based regression models, \textit{Gaussian process regression with local explanation}, called \textit{GPX}, which reveals the feature contributions to the prediction for each sample while maintaining the predictive performance of GPR. In GPX, both the prediction and explanation for each sample are performed using an easy-to-interpret locally linear model. Therefore, no gap exists between the prediction and the explanation. The weight vector of the locally linear model is assumed to be generated from multivariate GP priors [17].

As the multivariate GP priors have a covariance function defined as kernels on the samples, GPX ensures that similar samples have similar weights. The hyperparameters of GPX are estimated by maximizing the marginal likelihood, in which the weight vectors for all the training samples are integrated out. For a test sample, the predictions with their uncertainties of the target variable and weight vector are obtained by computing their predictive distributions. The explanation for the predicted target variable is provided using the estimated weight vector with uncertainty, as shown in Fig. 1. Depicting the explanation with uncertainty helps users of GPX judge the reasonability of the predicted weights.

In experiments, we evaluated GPX both qualitatively and quantitatively in terms of predictive performance and interpretability on various benchmark datasets. The experimental results show that: 1) GPX can achieve predictive errors comparable to GPR and lower errors compared with existing interpretable methods; 2) it can outperform model-agnostic interpretable methods and locally linear methods in terms of three interpretability measurements; and 3) the feature contributions produced by GPX are appropriate.

The predictions and explanations by GPX bring various benefits to users. For engineers who deploy GPX in ML systems, the explanations can be used for judging whether the trained GPX works as they expect. In particular, by investigating the explanations for wrong predictions, they can identify the cause of the errors. For users of ML services that display the predictions and the explanations by GPX, the explanations are useful for judging whether the predictions are reliable. For example, when GPX is used in a recommender system, users can see why an item is recommended from its corresponding explanation. As a result, the users may rely on the recommendation result, and if the recommendation and the explanation are wrong, developers may have a chance to improve the system by asking the users to point out the errors. For scientists, the explanations by GPX are useful to extract knowledge from data. Since GPX provides different explanations over samples, it may find the knowledge that feature selection methods that provide a single explanation over samples cannot find. Some of the above benefits can be said of other interpretable ML methods. We believe that high predictive performances and explainability of GPX make the benefits more confident.

\section{A. Notation}

Unless otherwise noted, we basically use the following notation of variables and special operators throughout this article. A plain lowercase symbol, such as $x$, represents a scalar. A bold lowercase symbol, such as $\mathbf{x}$, represents a vector. A bold uppercase symbol, such as $\mathbf{X}$, represents a matrix. A calligraphic symbol, such as $\mathcal{X}$, represents a set. $\mathbf{I}_n$ denotes an identity matrix of order $n$. $\mathbf{0}_d$ denotes an $d$-dimensional all-zero vector. $\mathbf{X}_i$ and $\mathbf{X}_{ij}$ denote the $i$th row and $ij$th column vectors of matrix $\mathbf{X}$, respectively. $\text{vec}(\mathbf{X})$ is a function that flattens input matrix $\mathbf{X}$ in a column-major order. $\text{diag}(\mathbf{x})$ is a diagonal matrix whose diagonal elements possess the values of input vector $\mathbf{x}$. $\odot$ denotes the element-wise product operator. $\mathbf{T}^\top$ denotes the transpose of vector and matrix. $|\cdot|$ represents the absolute value for a scalar or the number of elements for a set.
in each sample. Unlike the filter methods, the predictive accuracy of GPX is consistently comparable with that of GPR. In addition, unlike the wrapper methods, GPX can be used by training it with all the features only once.

Linear regression models are simple types of interpretable models, as their weight for each feature directly represents the contribution to the output of the models if the feature has a binary value, where the contribution helps us to interpret the effectiveness of the feature on a task. For real-valued features, the contribution for each feature is calculated as the product of the feature value and its corresponding weight. A number of studies introducing various regularizations have been conducted to produce methods, such as the ridge regression [21] and lasso [22] methods. These models have a global weight vector that is shared across all samples. In kernel methods, automatic relevance determination (ARD), which considers the global relevance of each feature contained in kernel functions, is widely used [23], [24]. The above approaches are beneficial in understanding global effectiveness of features. However, since linear regression models often do not fit real complicated problems, resulting in low predictive accuracy, the estimated weights may also be unreliable. In addition, these approaches cannot estimate weights/relevances appropriate for individual samples, which means that they cannot cope with the need of the significant changes of the weights/relevances among the samples. For example, as shown in the weight visualization result on Digits dataset in Section VI-C, this is crucial in image classification tasks that important pixels (i.e., features) and their weights change depending on individual images (i.e., samples).

On the other hand, some locally linear models for regression have been proposed, such as the network lasso [25] and localized lasso [26], which have a weight vector for each sample. Therefore, these methods can avoid the drawbacks of globally linear models. To receive the benefit, we focus on generating predictions with explanations using locally linear models. In the network and localized lasso, the weights of locally linear models are estimated via optimization with network-based regularization, where the network must be defined on samples in advance. If the network is not provided, as assumed in standard regression problems, we can construct a \( k \)-nearest-neighbor graph of samples to create a network, where \( k \) is a hyperparameter that must be optimized via cross validation. Meanwhile, GPX can estimate weights and their uncertainties without constructing graphs by assuming that weights are determined by functions generated from GPs.

For GP models, Paananen et al. [27] proposed feature selection methods that quantify the feature relevances by measuring the difference between the predictive posterior for each sample and that for its vicinity. The feature relevance can be used for evaluating the importance of the feature in prediction. Meanwhile, the feature contribution produced by GPX indicates whether the feature makes a positive or negative contribution to an individual prediction and how much it contributes.

With regard to research on deep neural networks (DNNs), a number of studies have been conducted on making predictions generated by DNNs interpretable [28]–[30]. Some of these studies have developed methods that make interpretable predictions by generating locally linear models for each sample using DNNs [11], [31], [32]. These concepts inspired our study, but we formalize our model without DNNs. To the best of our knowledge, our study is the first to develop a GP-based regression model with local explanations. Compared to the DNN-based locally linear models, GPX has mainly two benefits: 1) GPX can produce the feature contributions with uncertainty and 2) GPX can be straightforwardly used for tasks in which GPR can be used advantageously, such as Bayesian optimization [5], [33].

Wilson et al. [2] proposed GPR networks, which have a similar structure with multilayer perceptron. Here, the weights of the networks are generated from GPs. Although their idea is related to GPX, their work did not focus on improving the explainability of the predictions by GPR.

Conceptually, back constrained Gaussian process latent variable models (BC-GPLVMs) [34] are related to GPX because both assume that the latent variables (the weight vectors for GPX) are generated by the functions of input data, and their outputs are determined based on the latent variables. A key difference is that BC-GPLVMs enable us to interpret input data by visualizing the estimated latent variables, while GPX enables us to interpret the feature contributions to a prediction via the weights generated from the functions on a GP. Therefore, the proposed model is appropriate to know how the input of each sample affects its corresponding prediction.

In the context of Bayesian neural networks, Karalekos et al. [35] proposed neural networks with input-dependent weights, where a GP prior with covariance function depending on inputs, called as auxiliary kernel, are placed for the weights for improving generalization performance, uncertainty quantification, and active learning. Although the idea of generating input-dependent weights from the GP is common between the proposed model and their one, our study focuses on exploiting the weights with uncertainty for the explanation of each prediction and then investigating the explanation power of the weights in experiments.

In econometrics, regression models with time-varying coefficients have been well studied for time-series analysis [36], [37]. Those models have different coefficients of a linear regression model for each time, where the coefficients are determined by a nonparametric function over time. By replacing the functions over time with those over input feature space, we can find the connection between those models and GPX. However, in the existing studies, the explanation power of the time-varying coefficients has not been investigated. Besides, those models need to have the simplified representation as input to obtain explanations from the time-varying coefficients. This may cause a low predictive performance. On the other hand, GPX is formalized to receive two inputs, the original representation and the simplified representation, to both achieve a high predictive performance and generate easy-to-interpret explanations.

III. PROPOSED MODEL

In this section, we describe the proposed model, i.e., GPR with local explanation, called GPX.
We consider a scalar-valued regression problem. Suppose that training data $D = \{(x_i, y_i, z_i)\}_{i=1}^n$ containing $n$ samples are provided. $x_i \in \mathcal{X}$ is an original input representing the $i$th sample, where $\mathcal{X}$ is an original input space. A typical representation for $x_i$ is a vector. For example, on the task of predicting some scores from text data, we can use embedding vectors calculated from the text data by using a neural language model, such as sentence transformers [38]. In addition to the vector representation, we can use any data representation on which some scores from text data, we can use embedding vectors from a multivariate GP. More specifically, weight vector $w$ results in avoiding overfitting, the dimension should be easily understood by humans, such as a value in a column in tabular data and the occurrence of a word in text data. In another case that the original input is a graph, its simplified representation can be bag-of-edges or a vector of the properties representing the structure of the graph, such as diameter and clustering coefficient. Since the original input contains rich information about the corresponding sample to achieve high predictive performance, but the content of the original input is complex or difficult to understand by humans, we interpret the predictions made by GPX via the simplified representation. $z_i$ is an optional input; therefore, if $x_i$ can be used as a simplified representation, one can define $z_i = x_i$. Let us denote $X = \{x_i\}_{i=1}^n$, $y = \{y_i\}_{i=1}^n \in \mathbb{R}^n$ and $Z = \{z_i\}_{i=1}^n \in \mathbb{R}^{n \times d}$.

In GPX, both the prediction of target variables $y$ and their explanations are performed via easy-to-interpret locally linear models, i.e., target variable $y_i$ for the $i$th sample is assumed to be obtained using locally linear function $f_i : \mathcal{X}^d \rightarrow \mathbb{R}$, which is defined as follows:

$$ f_i(z_i) = w_i^\top z_i + \epsilon_y \quad (1) $$

where $w_i \in \mathbb{R}^d$ is a $d$-dimensional weight vector for the $i$th sample and $\epsilon_y \sim \mathcal{N}(0, \sigma_y^2)$ is a Gaussian noise with variance $\sigma_y^2 > 0$. Here, the explanation for the $i$th sample is obtained using either weight vector $w_i$ or feature contributions $\phi_i = \{(w_i^\top z_i)\}_{i=1}^n$.

In principle, the locally linear functions can express all the training samples completely. However, estimating $W = \{w_i\}_{i=1}^n \in \mathbb{R}^{n \times d}$ without any constraints is an ill-posed problem because the number of free parameters in $W$, $nd$, is larger than that of target variable $n$. To avoid this problem in GPX, we assume that functions determining $W$ are generated from a multivariate GP. More specifically, weight vector $w_i$ for the $i$th sample is obtained as follows:

$$ w_i = g(x_i) + \epsilon_w \quad (2) $$

where $\epsilon_w \sim \mathcal{N}(0, \sigma_w^2 I_d)$ is a $d$-dimensional Gaussian noise with variance $\sigma_w^2 > 0$, and $I_d$ is an identity matrix of order $d$.

Here, vector-valued function $g : \mathcal{X} \rightarrow \mathbb{R}^d$ is a function that determines the weight vector for each sample, and each element of $g$ is generated from a univariate GP independently.

1Since the existence of noise term $\epsilon_w$ results in avoiding overfitting, the posterior of $w$ would be more smooth than without the noise term. The smoothness can contribute to improving the stability (34).
maintaining its mathematical meanings as follows:

\[
p(y, W, G|X, Z) = \mathcal{N}(\text{vec}(G)|0_n, \tilde{K})\mathcal{N}(\text{vec}(W)|\text{vec}(G), \sigma_\omega^2 I_{nd}) \times \mathcal{N}(y|\bar{Z}\text{vec}(W), \sigma_\gamma^2 I_{nd})
\]

where \(\tilde{K}\) is a block diagonal matrix of order \(nd\) whose block is \(K\) and \(\text{vec}(\cdot)\) is a function that flattens the input matrix in a column-major order. Here

\[\bar{Z} = (\text{diag}(Z, 1), \text{diag}(Z, 2), \ldots, \text{diag}(Z, d)) \in \mathbb{R}^{n \times nd}\]

where \(\text{diag}(\cdot)\) is a diagonal matrix whose diagonal elements possess the values of the input vector. In (6), \(d\) functions that output \(n\)-dimensional column vectors in \(W\) are generated from GPs; however, in (43), it is rewritten such that a single function that outputs an \(nd\)-dimensional flatten vector \(\text{vec}(W)\) is generated from a single GP. Consequently, the likelihood of target variables \(y\) can be rewritten as a single multivariate normal distribution.

Subsequently, we derived the marginal likelihood by integrating out \(G\) and \(W\) in (43). Due to the property of normal distributions, it can be obtained analytically as follows:

\[
p(y|X, Z) = \int p(y, W, G|X, Z)dWdG = \mathcal{N}(y|0_n, C)
\]

where

\[
C = \sigma_\gamma^2 I_n + Z(K + \sigma_\omega^2 I_{nd})Z^T
\]

\[
= \sigma_\gamma^2 I_n + (K + \sigma_\omega^2 I_n) \odot ZZ^T.
\]

B. Hyperparameter Estimation

If \(k_y(x, x')\) is differentiable with respect to \(\theta\), all the hyperparameters, i.e., \(\theta, \sigma_\omega,\) and \(\sigma_\gamma\), can be estimated by maximizing the logarithm of the marginal likelihood with respect to them for the training data using gradient-based optimization methods, e.g., L-BFGS [42].

One of the explainability measures used in our experiments, stability, measures the consistency of explanations for similar samples. Since GPX is modeled such that two samples with a large kernel value have similar weights, and otherwise, they have different weights, an easiest way to achieve the highest stability is to set lengthscale parameter \(\theta_2\) in kernel (5) to a large value or \(\infty\). However, in such a case, the predictive power of GPX becomes worse, because when all pairs of samples have similarly large kernel values, their own weights are much the same, that is, the lengthscale parameter can be seen as a tradeoff parameter of predictive power and stability. By adjusting the lengthscale parameter by hand or by tuning on a validation set, one can also balance between predictive power and stability.

C. Predictive Distributions

For a new test sample \((x_*, z_*)\), our goal is to infer the predictive distributions of target variable \(y_\bullet\) and weight vector \(w_\bullet\).

First, the predictive distribution of \(y_\bullet\) is obtained similarly as in the standard GPR as follows:

\[
p(y_\bullet|x_\bullet, z_\bullet, D) = \mathcal{N}(y_\bullet|c_y^\top C^{-1} y, c_y - c_y^\top C^{-1} c_y)
\]

where \(c_y = (k_\phi(x_\bullet, x_i)z_i^\top z_i)_{i=1}^n \in \mathbb{R}^n\) and \(c_{y_\bullet} = \sigma_y^2 + (k_\phi(x_\bullet, x_i) + \sigma_\omega^2)z_i^\top z_i \in \mathbb{R}^d\).

Second, the predictive distribution of \(w_\bullet\) is obtained by solving the following integral:

\[
p(w_\bullet|x_\bullet, z_\bullet, D) = \int p(w_\bullet|W, x_\bullet) p(W|D)dW
\]

\[
p(w_\bullet|W, x_\bullet) = \mathcal{N}(w_\bullet|\text{AVec}(W), \bar{c}_{w_\bullet} - \bar{A}\bar{k}_\bullet)
\]

\[
p(W|D) = \mathcal{N}(\text{vec}(W)|\sigma_\gamma^2 S\bar{Z}^\top y, S)
\]

where we define \(A = \bar{k}_\bullet (\bar{K} + \sigma_\omega^2 I_{nd})^{-1}, S = L - L\bar{Z}^\top C^{-1} \bar{Z}L, \bar{K} = \bar{K} + \sigma_\omega^2 I_{nd},\) and \(\bar{c}_{w_\bullet} = (k_\phi(x_\bullet, x_i) + \sigma_\omega^2)\bar{z}_i\) for \(l = 1, 2, \ldots, d,\) and the other blocks are zero matrices. Solving the integral analytically according to the property of the normal distributions, we obtain

\[
p(w_\bullet|x_\bullet, z_\bullet, D) = \mathcal{N}(w_\bullet|\sigma_y^{-2} AS\bar{Z}^\top y, \bar{c}_{w_\bullet} - \bar{A}\bar{k}_\bullet + ASA^\top).
\]

We provide the detailed derivation of predictive distributions (14) and (18) in the Appendix.

The marginal likelihood (12) and the predictive distribution for \(y_\bullet\) (14) are similar to those of GPR, except that GPX can obtain the predictive distribution for \(w_\bullet\) (18). Since GPX can be used with the same input as GPR if \(Z = X\), it can be employed in existing ML models, instead of GPR.

D. Computational Efficiency

As with ordinary GPR, the computational cost of GPX is dominated by the inverse computation. The computation of \(A\) requires inverting a square matrix of order \(nd, \bar{K} + \sigma_\omega^2 I_{nd}\). However, because the matrix is block diagonal and every diagonal block comprises \(K + \sigma_\omega^2 I_n\), a square matrix of order \(n\) and \(A\) can be obtained by inverting \(K + \sigma_\omega^2 I_n\) only once. The remaining inverse matrix \(C^{-1}\) is of order \(n\). Therefore, all the inverse matrices appearing in GPX can be obtained using a naive implementation with a computational time complexity of \(O(n^3)\), which is the same as that in GPR. To significantly reduce the computational cost, efficient computation methods for GPR, such as the inducing variable method [43] and KISS-GP [44], can be used for GPX. When the inducing variable method can be applied, the space complexity is reduced from \(O(n^2)\) to \(O(nM + M^2)\) and the time complexity is reduced from \(O(n^3)\) to \(O(nM^2 + M^3)\), where \(M\) is the number of induced points; in general, \(M\) is set to \(M \ll N\). In addition, because \(\bar{k}_\bullet, \bar{K}\) and \(\bar{Z}\) are sparse matrices, one can obtain the predictive distributions efficiently using libraries for sparse matrix computation.
V. ANALYSIS AND DISCUSSION

In this section, we analyze the behaviors of GPX in special cases to understand the properties of GPX. In addition, we discuss the relationship and the difference between GPX and related methods.

A. Behaviors in Special Cases

Considering some special cases, we discuss the behavior of GPX and the relation between GPX and the standard GPR.

1) Case of Z With All-One Vectors: First, let us consider a case that simplified inputs $Z = (z_i^T)^n_{i=1}$, including $z_i$ for a test sample, are composed only of all-one vectors, although original inputs $X = [x_i]^n_{i=1}$, including $x_i$ for the test sample, have no constraints. This is the easiest way to construct $Z$ for users; therefore, it is meaningful to analyze the behavior of GPX in this case.

For ease of explanation, we let $\sigma_w = 0$ and $z_i = (1/(d^{1/2}))1_d$ for all $i = 1, 2, \ldots, n$, instead of $z_i = 1_d$. By substituting $Z$ into (14), the predictive distribution of target variable become equivalent to that of the standard GPR, because of $ZZ^T$ being an all-one matrix, and $z_i^Tz_j = 1$ for any $i$ and $j$. Similarly, we substitute $Z$ into predictive distribution of weights (18). Since $A$ is determined only by $X$ and $x_*$, it is constant for feature index $l$. Also, since $Z$ is now constant, $S$ is also constant for feature index $l$. Consequently, although the mean and covariance change depending only on $k_s = (k(x_*, x_i))^n_{i=1}$ and $y$, they have the same values for all feature indices $l = 1, 2, \ldots, d$. If the kernel used is squared exponential kernel (5), the above discussion is also valid even when $Z$ is an all-b matrix for some constant $b$ other than zero since parameter $\theta_l$ in the kernel can buffer the effect of constant $b$. Let $Z$ be an all-zero matrix that is meaningless since it makes both the posterior means of $y_s$ and $w_s$ be zero. Thus, using $Z$ being a constant value leads to lose the explainability of the feature contributions.

2) Case of X With Constant Vectors: Second, we consider the opposite case, i.e., original inputs $X$ and $x_*$ are composed only of constant vectors, although simplified inputs $Z$ and $z_*$ have no constraints. Since the original inputs have the same value, kernel $k_0(x, x')$ for any two original inputs is constant. For ease of explanation, assuming that the value of $k_0(x, x')$ has a range of $[0, 1]$, we let $k_0(x, x') = 1$. In this case, the predictive distribution of target variable $y_s$ is equivalent to that of GPR with the linear kernel of the simplified inputs. In addition, predictive distribution of weights $w_s$ is no longer dependent on the test sample since it originally depends on $x_*$, rather than $z_*$. As a result, this model behaves like a globally linear model with the simplified inputs as input.

B. Comparison With the Derivative of GPR Posterior

To obtain the feature contributions in GPR, a possible approach is to calculate the derivative of the posterior in GPR. For test sample $z_*$ which is a simplified input, the GPR posterior mean of target variable $y_s$ is given by

$$ m(z_*) = k_s^T K^{-1} y $$

where $k_s = (k(z_i, z_*))^n_{i=1}$ is the vector of kernel values between training samples $[z_i]^n_{i=1}$ and test sample $z_*$ and $K^{-1}$ is the inverse of the kernel matrix of the training samples. Then, the derivative of the posterior mean is calculated as follows:

$$ \frac{\partial m(z_*)}{\partial z_i} = (\frac{\partial k_s}{\partial z_i})^T K^{-1} y. \quad (20) $$

Since the derivative shows how posterior mean $m(z_*)$ changes for small changes in input $z_*$, we can interpret the feature contributions to the prediction by investigating the values of the derivative. Note that, the feature contributions in this approach are different from those in GPX because they do not satisfy the local linear assumption for target variables, that is, $y_s = (\partial m(z_*)^T / \partial z_i) z_*$.

This approach is reasonable, but there are the following limitations compared to GPX. First, the approach cannot be applied for GPR with a nondifferentiable kernel. Second, even when using a differentiable kernel, the inputs of GPR are limited to be only the simplified inputs, i.e., $Z$, for interpretability. When information valuable for prediction can be represented only in the original inputs, i.e., $X$, this could be a disadvantage.

C. Relationship to Bayesian Warped Gaussian Processes

Bayesian warped GPs (BWGPs) are the composition model of two GPs [45]. In BWGPs, target variable $y_i$ for the $i$th sample is modeled as

$$ y_i = h(f(x_i)) + \epsilon_i \quad (21) $$

$$ f(x) \sim GP(\mu_0, k^{(f)}(x, x')) \quad (22) $$

$$ h(f) \sim GP(f, k^{(h)}(f, f')) \quad (23) $$

$$ \epsilon_i \sim N(0, \sigma^2) \quad (24) $$

where $f(x)$ is a latent function with vector $x$, $h(f)$ is an arbitrary warping function with scalar inputs, $\epsilon$ is a Gaussian noise term, and $k^{(f)}$ and $k^{(h)}$ are arbitrary kernel functions. Then, variational inference is performed to learn BWGPs because the exact posterior of BWGPs is analytically intractable. Since the formulation of BWGPs is quite general, some of the existing GP-based models, such as GP regression, GP classification, ordinal regression, and maximum-likelihood warped GPs, can be regarded as a special case of BWGPs by modifying the definition of $f$, $h$, and $\epsilon$. Therefore, an interest is to know whether GPX can also be seen as a special case of BWGPs.

Mimicking the formulation of BWGPs, GPX can be written as follows:

$$ y_i = h_1(f(x_i)) + \epsilon_i \quad (25) $$

$$ f(x) = (f_i(x))^d_{i=1} \quad (26) $$

$$ f_i(x) \sim GP(0, k^{(f)}(x, x')) \quad (27) $$

$$ h_1(f) = f^T z_i \quad (28) $$

$$ \epsilon_i \sim N(0, \sigma^2_i) \quad (29) $$

Unlike BWGPs, $f(x)$ in GPX is a vector, each of which, $f_i(x)$, is generated from a GP. Also, we denote $h$ in BWGPs by $h_1$ instead because its value depends on sample index $i$, and $h_1$ is in fact a linear function of $z_i$. Due to the simple
formulation of \( h_i \), the posteriors of target variables and weights in GPX are analytically obtained.

VI. EXPERIMENTS

In this section, we demonstrate the effectiveness of the proposed model, GPX, quantitatively and qualitatively, by comparing various interpretable models. Through a quantitative evaluation, we evaluated the models based on the following perspectives.

1) **Accuracy:** How accurate is the prediction of the interpretable model?
2) **Faithfulness:** Are feature contributions indicative of “true” importance?
3) **Sufficiency:** Do \( k \)-most important features reflect the prediction?
4) **Stability:** How consistent are the explanations for similar or neighboring examples?

In addition, we qualitatively evaluated whether the feature contributions produced by the models were appropriate by visualizing them. Subsequently, we experimentally compared the computational efficiency of the models.

All the experiments were done with a computer with Intel Xeon Gold 6132 2.6 GHz CPU with 16 cores and 120 GB of main memory.

A. Preparation

1) **Datasets:** To evaluate weights estimated by GPX and comparing methods with ground truth ones, we used four simulated datasets, namely, “Corr \( d = 20 \),” “Corr \( d = 100 \),” “Weak corr,” and “No corr.” All the datasets have \( n = 100 \) samples, where the original inputs for the first 50 samples \( \{ x_i \}_{i=1}^{50} \) are drawn from an isotropic normal distribution with mean 0.5 and variance 1 and those of the remaining samples \( \{ x_i \}_{i=51}^{100} \) are drawn from an isotropic normal distribution with mean \(-0.5\) and variance 1. Here, the original and simplified inputs \( x_i, z_i \) are identical. The dimensionality of \( x_i \) is commonly \( d = 20 \) except that Corr \( d = 100 \) is \( d = 100 \). The datasets have ground truth weight vectors for the samples generated in different ways each other, represented as matrix \( \hat{W} = (\hat{w}_i)^T \in \mathbb{R}^{n \times d} \). In Corr \( d = 20 \) and Corr \( d = 100 \), the \( l \)th column vector of \( \hat{W} \) is drawn from a multivariate normal distribution with mean vector \( \mu \in \mathbb{R}^d \) and covariance \( \mathbf{K} \in \mathbb{R}^{d \times d} \), where the first 50 elements of \( \mu \) are 0.5, while the remaining elements are \(-0.5\), and the \((i,j)\)-element of \( \mathbf{K} \) is calculated by squared exponential kernel (5) of two inputs \( x_i \) and \( x_j \). We set \( \theta_2 \) to 0.05 and 0.005 for Corr \( d = 20 \) and Corr \( d = 100 \), respectively, while we fixed \( \theta_1 \) at 1.0. As a result, the ground truth weights are correlated with the inputs. In Weak corr, we generated each of the weight vectors from the same distribution as its corresponding input so that the weight vectors are weakly correlated with the inputs. In No corr, all the weight vectors are drawn independently of an isotropic normal distribution with zero mean and variance 1. Finally, in all the datasets, target variable \( y_i \) is calculated as \( y_i = \hat{w}_i^T z_i \).

We also used eight real datasets in the UCI ML repository [46], referred to as Digits, Abalone, Diabetes, Boston, Fish, Wine, Paper, and Drug in our experiments. Digits dataset was originally developed as a classification dataset for recognizing handwritten digits from 0 to 9. Since each sample in the dataset is an \( 8 \times 8 \) grayscale image, we flattened the image into a 64-dimensional vector and used it as an input feature vector. To use it as a regression problem, we transformed the digit labels of the images into target variables \( y \) of scalar values, i.e., the target variables for the labels from 0 to 4 were \(-1\) and those for the remaining labels were 1. Here, we used only the testing set from the original Digits dataset, as scikit-learn [47] does. Abalone dataset is a dataset for predicting the age of abalone based on physical measurements. Diabetes dataset is a dataset for predicting the onset of diabetes based on diagnostic measures. Boston dataset is a dataset for predicting house prices, as described in Section VI-C7. Fish dataset is a dataset for predicting acute aquatic toxicity toward the fish Pimephales promelas for a set of chemicals. Wine dataset is a dataset for predicting the quality of white and red wines based on physicochemical tests. With Abalone, Diabetes, Boston, Fish, and Wine, the input of each sample is a feature vector with integer or continuous values and its output is an integer or continuous value to be predicted. The remaining two datasets are text datasets. Paper dataset is a dataset for predicting evaluation scores for papers based on review texts written mainly in Spanish. Drug dataset is a drug review dataset for predicting 10-star ratings for drugs based on patient review texts. With Paper and Drug datasets whose samples were represented as sentences, the original input \( X \) and the simplified input \( Z \) differed, i.e., we used the 512-dimensional sentence vectors obtained using sentence transformers [38] as \( X \), while we used bag-of-words binary vectors for the sentences as \( Z \). Each of the remaining datasets had the same \( X \) and \( Z \). In all the real datasets, the values of \( X \) and \( y \) were standardized to have mean 0 and variance 1 before training and prediction. Table I lists the number of samples \( n \) and number of features \( d \) in each dataset.

For a quantitative evaluation of each dataset, we evaluated the average scores over five experiments performed on different training/test splittings, where the training set was 80% of the entire dataset, whereas the remaining was the test set.

2) **GPX Setup:** In GPX, we consistently used a squared exponential kernel defined as (5). The hyperparameters of GPX were estimated based on the method described in Section IV, where they were initialized with \( \theta_1 = 1.0, \sigma_y = 0.1, \) and \( \sigma_w = 0.1 \). In addition, we initialized bandwidth \( 2^\theta \) from (12), the mean of \( y \) is assumed to be zero; therefore, one should use standardized \( y \). In addition, since the covariance function in (12) is a nonstationary kernel, different predictive distributions are obtained by transforming \( Z \) via \( Z + c \) for some constant \( c \). One can make it robust against such a transformation by using standardized \( Z \).
parameter $\theta_2$ using median heuristics [48]. We implemented GPX and GPR explained below using PyTorch v1.5.0\textsuperscript{3} and GPytorch.\textsuperscript{4}

3) Comparing Methods: We compared GPX with several methods with globally or locally linear weights that can be used as interpretable feature contributions or relevances for predictions. Lasso [22] and Ridge [21] are standard linear regression models with $\ell_1$ and $\ell_2$ regularizers, respectively, where their weights are globally shared across all samples. For Lasso and Ridge, we used the implementations provided by scikit-learn [47]. The hyperparameters that regularize the strengths of the $\ell_1$ and $\ell_2$ regularizers in Lasso and Ridge, respectively, were optimized through a grid search using functions provided by scikit-learn (i.e., sklearn. linear_model.LassoCV and sklearn.linear_model.RidgeCV) with the default options. The search range for the hyperparameters for Lasso was limited to within 100 grid points such that the ratio of its minimum value to its maximum value was capped at 0.001, while that for Ridge was limited to be within a range of $[0.1, 1, 10]$. GPR/ARD is a GPR model with ARD kernel that identifies the relevance of each feature. Here, the ARD kernel is defined as

$$k_\theta(x, x') = \alpha \exp\left(-\frac{1}{2} \sum_{l=1}^{d} (x_l - x'_l)^2 / \theta_l\right), \quad (a, \theta_l > 0)$$

(30)

where $\alpha$ is a scale parameter, $\theta_l$ is the relevance of the $l$th feature, and $\theta = \{a, \theta_1, \theta_2, \ldots, \theta_d\}$ is a set of parameters to be estimated. All hyperparameters for GPR/ARD were estimated by maximizing marginal likelihood [1], where we initialized the hyperparameters to the same values as those for GPX. The network lasso ("network" for short) is a locally linear model that regularizes the weights of nodes such that neighboring nodes in a network have similar weights [25]. In our case, each node represents a sample, and the network is a $k$-nearest neighbor graph on the samples based on the cosine similarity on $X$. The localized lasso ("localized" for short) is an extension of the network lasso; it can estimate the sparse and exclusive weights of each sample by further incorporating an $\ell_{1,2}$ regularizer into the network lasso [26].

For the localized lasso, we used the original implementation written in Python.\textsuperscript{5} The hyperparameters and their search ranges for the localized lasso are the strength of network regularization $\lambda_1 \in \{1, 3, 5, 7\}$, strength of the $\ell_{1,2}$ regularizer $\lambda_2 \in \{0.01, 0.1, 1, 10\}$, and $k \in \{5, 10, 15\}$ for the $k$-nearest-neighbor graph. The hyperparameters were optimized through a grid search. The network lasso is a special case of the localized lasso. If $\lambda_2$ for the localized lasso is zero, then the localized lasso is identical to the network lasso. Therefore, we used the implementation of the localized lasso and set $\lambda_2 = 0$ for the network lasso. The hyperparameter search for the network lasso was the same as that for the localized lasso, except for the setting of $\lambda_2$. The locally weighted scatter plot smoothing (LOESS) is a kernel-weighted locally linear model [49] that computes the values of RBF kernel between training samples and each test sample on $X$ followed by solving a kernel-weighted least squared problem in the training samples on $Z$. Here, RBF kernel with bandwidth parameter $\gamma > 0$ is defined as

$$k_\gamma(x, x') = \exp\left(-\frac{1}{\gamma} \|x - x'\|_2^2\right).$$

(31)

To compare model-agnostic interpretable methods with GPX in terms of explainability for prediction, we used LIME [14] and Kernel SHAP [15], which produce a locally linear model for each test sample to explain the prediction by a black-box prediction model. For LIME and Kernel SHAP, we used the original implementations.\textsuperscript{6} For a fair comparison, we used GPR with squared exponential kernel ("GPR/SE" for short) as the prediction model, which is optimized as with GPR/ARD. In addition, we used a Kullback–Leibler (KL) divergence-based feature selection method for GPR/ARD ("KL" for short) [27]. We implemented KL by mimicking its original implementation.\textsuperscript{7} Here, we set to amount of perturbation $\Delta = 0.001$ throughout our experiments. The hyperparameters of GPR/SE and GPR/ARD were estimated by maximizing marginal likelihood, similarly as for GPX. Meanwhile, those of the remaining comparing methods were optimized by grid search.

B. Results on Simulated Datasets

First, we demonstrate the predictive performances of GPX and the comparing methods on the simulated datasets in Table II. For Corr ($d = 20$), Corr ($d = 100$), and Weak corr, GPX consistently achieved the lowest predictive errors. As shown in the results on Corr ($d = 20$) and Corr ($d = 100$), the performances of some comparing methods became much worse as the dimensionality $d$ increases. Even in such a case, GPX could achieve one of the lowest predictive errors. For No corr, there was no method that was particularly good. In fact, these scores are no different from that of “random guess” that predicts target variables as samples from the distribution learned on training target variables. Therefore, none of the methods can predict well in no correlation setting.

Then, to demonstrate the faithfulness of the methods, we show the errors of the estimated weights in Table III. For Corr ($d = 20$) and Corr ($d = 100$), GPX clearly outperformed the comparing methods. For Weak corr and No corr, although GPX achieved one of the lowest errors, the errors are much the same as those of some comparing methods. In summary, the results on Tables II and III indicate that when the inputs and their corresponding weights are correlated at least weakly, GPX can achieve better predictive performances and output the contributions to the predictions more accurately.

C. Results on Real Datasets

1) Accuracy: First, we demonstrate the predictive performances of GPX and the comparing methods in Table IV.

\textsuperscript{1}https://pytorch.org/
\textsuperscript{2}https://gpytorch.ai/
\textsuperscript{3}https://riken-yamada.github.io/localizedlasso.html
\textsuperscript{4}https://github.com/marcotcr/lime, Kernel SHAP: https://github.com/slundberg/shap
\textsuperscript{5}https://github.com/otopia/gp-varsel-kl-var
GPX achieved one of the lowest predictive errors on all the datasets, compared to the other globally or locally linear models. In addition, their predictive errors were comparable to that of GPR on all the datasets. This result indicates that GPR can be replaced by GPX to achieve similar predictive performances.

2) Faithfulness: Assessing the correctness of the estimated contribution of each feature to a prediction requires a reference “true” contribution for comparison. As this is rarely available, a typical approach for measuring the faithfulness of the contributions produced by interpretable models is to rely on the proxy notion of the contributions: observing the effect of removing features on the model’s prediction. Following previous studies [11], [50], we computed the faithfulness score by removing features one-by-one, measuring the differences between the original predictions and the predictions from the inputs without the removed features, and calculating the correlation between the differences and the contributions of the removed features. Formally, the faithfulness score for set of test samples \( D_{te} = \{(x_s, z_s)\} \) is calculated as follows:

\[
\text{Fa}(D_{te}) = \text{corr}_{x_s, z_s \in D_{te}}(\{|w_{s,l}|, |y_s - y_{s,-l}|\})
\]

where \( y_s \) is the output of the predictor for input \( x_s, z_s \), \( y_{s,-l} \) is the output of the predictor when removing the \( l \)th feature from \( x_s, w_{s,l} \) is the \( l \)th dimension of the estimated weights for test sample \( (x_s, z_s) \), and \( \text{corr}(\cdot, \cdot) \) is a function that returns the Pearson correlation coefficient between two arguments for all combinations of variables on the bottom of corr.

Table V shows the faithfulness scores of GPX and the comparing methods. Here, we denote the results of LIME and Kernel SHAP using GPR/SE as the black-box prediction model by GPR/SE + LIME and GPR/SE + SHAP, respectively. We found that GPX achieved one of the best faithfulness scores on all the datasets except for Boston and Wine. As GPX predicts and explains using a single locally linear model for each test sample, when removing a feature from the input, the contribution of the feature is subtracted from the prediction directly. Meanwhile, because GPR/SE + LIME, GPR/SE + SHAP, and KL have different prediction and explanation models, a gap may exist between the estimated contribution in the explanation model and the latent contribution in the prediction. Because the predictions by GPX and GPR were performed using similar calculations, their faithfulness differences were likely due to the gap. With GPR/ARD, it cannot estimate feature relevances appropriate for each sample, as the feature relevances are shared over samples; therefore, it produced
relatively low faithfulness scores. As with GPX, LOESS achieved one of the best faithfulness scores on four datasets. However, as LOESS showed the worst score on Abalone, it was not stable compared to GPX.

3) Sufficiency: In general, the inputs contain many irrelevant features that do not contribute to the predictions, and discovering important features in all the features is difficult for users of the models. Therefore, a desirable property of the interpretable models is that it can assign high contributions only for important features that affect the predictions well. To quantify how each method satisfies the property, we define the sufficiency score at \( k \), where \( k \) is the number of important features. In particular, the sufficiency score at \( k \) was computed by identifying \( k \) important features in the descending order of the absolute values of their estimated contributions, predicting from the inputs having only \( k \) important features, and comparing them against the original predictions. More formally, the sufficient score at \( k \) for set of test samples \( D_{te} \) is calculated as follows:

\[
\text{Su@}k(D_{te}) = \frac{1}{|D_{te}|} \sum_{x_i, z_i \in D_{te}} \left( y_{i} - y_{i, \Omega(k; w)} \right)^2
\]

where \( y_{i} \) and \( w_{i} \) denote the output of the predictor and its corresponding weights, respectively, and \( \Omega(k; w) \) denotes the indices of the dimensions associated with the top-\( k \) largest absolute values in \( w_{i} \); therefore, \( y_{i, \Omega(k; w)} \) denotes the output of the predictor when keeping the values in \( z_{i} \) associated with the dimensions of \( \Omega(k; w) \) and setting the other values to zero. Because the number of important features varied according to the sample and dataset, we evaluated them fixed at \( k = 1, 2, \ldots, 10 \).

Fig. 3 shows the sufficiency scores of GPX and the comparing methods. Independent of the \( k \) values, GPX and LOESS outperformed the others on the Digits dataset. GPX, GPX/SE + LIME, GPX/SE + SHAP, and LOESS produced the best sufficiency scores on Diabetes, Fish, and Wine datasets, except that the scores of LOESS at \( k \leq 2 \) were slightly worse on the Diabetes dataset. These results indicate that GPX was appropriately assigned high contributions for the important features. On Abalone and Boston datasets, GPX was slightly inferior to the localized lasso at \( k = 1, 2 \), although GPX outperformed it at \( k \geq 3 \). This is because the localized lasso has a regularizer that induces sparse weights. This result suggests that GPX can be further improved by employing the mechanism for generating sparse weights.

4) Stability: To generate meaningful explanations, interpretable methods must be robust against local perturbations from the input, as explanations that are sensitive to slight changes in the input may be regarded as inconsistent by users. In particular, flexible models such as locally linear models might be sensitive to such changes for achieving better predictions. As with the work in [11], we used the following quantity for measuring the stability of the estimated weights for test sample \( (x_{*}, z_{*}) \) as follows:

\[
L(x_{*}, z_{*}) = \max_{x_{*} \in B(x_{*})} \frac{||w_{*} - w_{*}'||_2}{||z_{*} - z_{*}'||_2} \quad (34)
\]

where \( B(x_{*}) = \{(x_{*}', z_{*}') \in D_{te} | \frac{1}{m} ||x_{*}' - x_{*}||_2 < \epsilon \} \), \( D_{te} \) is a set of test samples, \( w_{*} \) and \( w_{*}' \) are the standardized estimated weights associated with test samples \( (x_{*}, z_{*}) \) and \( (x_{*}', z_{*}') \), respectively, \( \epsilon > 0 \) is a parameter that determines neighboring samples, and \( m \) is the dimensionality of \( x_{*} \). We set \( \epsilon = 0.05 \) in our experiments. Intuitively, the stability score will be high when the estimated weights for the sample and its neighboring samples are similar. Subsequently, we computed the stability score on a dataset by averaging the quantity (34) on all the test samples in the dataset.

Table VI shows the stability scores on each dataset. GPX achieved the best stability scores on four datasets. For Paper and Drug datasets in which simplified inputs \( Z \) are different to original ones \( X \), GPX had by far the lowest stability scores. This is because the weight vectors produced by GPX become similar among neighbor samples by using the embedding vectors for texts, which are robust for small changes in the texts, as the original inputs \( X \). With GPX/SE + LIME and GPX/SE + SHAP, their stability scores were higher than that of GPX, although the prediction powers of GPX and GPX/SE were comparable. This would be because LIME and Kernel SHAP estimated the weights independently over the test samples. The stability score of KL was as good as that of GPX only on the Paper dataset; however, on the other datasets, the stability scores of KL were worse than those of GPX. LOESS achieved the best stability scores on four different datasets from those of GPX. This would be because LOESS is prone to estimating similar weights at neighbor samples by definition of its objective function.

5) Qualitative Comparison on Digits Dataset: We qualitatively compared the estimated weights using GPX and the comparing methods on the Digits dataset, in which the appropriate contributions for predictions were apparent. For this comparison, we rescaled the inputs \( X \) and \( Z \) to be within \([0, 1]\).
Fig. 3. Average sufficiency scores on (a) Digits, (b) Abalone, (c) Diabetes, (d) Boston, (e) Fish, and (f) Wine datasets (lower scores are better). The filled area on each line indicates its standard deviation. For Paper and Drug datasets, we did not evaluate the scores as changes in $Z$ cannot reflect $X$.

**TABLE VI**

AVERAGE AND STANDARD DEVIATION OF STABILITY SCORES ON EACH DATASET (LOWER SCORES ARE BETTER). THIS TABLE CAN BE INTERPRETED SIMILARLY AS IN TABLE IV. WE DID NOT MEASURE THE SCORES FOR MODELS THAT HAVE GLOBAL FEATURE RELEVANCES/WEIGHTS, SUCH AS GPR/ARD, AS THEIR SCORES ARE OBVIOUSLY ZERO ACCORDING TO (34)

| Dataset  | GPX (ours) ± 0.015 | GPR/SE+LIME ± 0.079 | GPR/SE+SHAP ± 0.110 | KL ± 0.094 | Localized ± 0.453 | Network ± 0.119 | LOESS ± 0.085 |
|----------|-------------------|---------------------|---------------------|------------|-------------------|----------------|--------------|
| Digits   | 2.410 ± 0.079     | 2.274 ± 0.110       | 2.058 ± 0.104       | 1.996 ± 0.098 | 1.989 ± 0.222     | 2.292 ± 0.644 |
| Abalone  | 3.094 ± 0.249     | 11.809 ± 0.453      | 11.625 ± 0.362      | 4.485 ± 0.911 | 15.439 ± 1.193    | 15.772 ± 0.948 | 12.085 ± 0.959 |
| Diabetes | 1.164 ± 0.065     | 1.870 ± 0.119       | 1.401 ± 0.147       | 2.833 ± 0.085 | 2.693 ± 0.123     | 2.969 ± 0.077  |
| Boston   | 1.452 ± 0.081     | 4.180 ± 0.426       | 2.176 ± 0.304       | 2.956 ± 0.221 | 3.794 ± 1.215     | 3.467 ± 0.799  | 0.727 ± 0.023 |
| Fish     | 1.497 ± 0.075     | > 10^5              | 1.634 ± 0.057       | 3.289 ± 0.536 | 5.746 ± 0.563     | 5.625 ± 0.591  | 1.131 ± 0.106 |
| Wine     | 1.531 ± 0.113     | > 10^5              | 5.255 ± 0.906       | 3.989 ± 0.306 | 4.033 ± 0.282     | 0.725 ± 0.024  |
| Paper    | 0.004 ± 0.002     | 5.535 ± 0.161       | 4.548 ± 0.098       | 0.026 ± 0.052 | 6.482 ± 0.161     | 6.986 ± 0.442  | 7.327 ± 0.280 |
| Drug     | 0.067 ± 0.014     | 16.976 ± 0.399      | 12.037 ± 0.513      | 11.547 ± 0.265 | 16.908 ± 0.532    | 17.341 ± 0.589 | 12.608 ± 1.031 |

Fig. 4 shows the estimated weights on samples with some digits. On this dataset, the appropriate weights can be obtained by assigning weights having the same sign with the target variable to black pixels. We found that the methods except for GPX and the localized lasso could not estimate reasonable weights. Meanwhile, the weights estimated by GPX and the localized lasso were appropriate, although they exhibited different characteristics, i.e., dense weights from GPX, whereas sparse ones from the localized lasso. The task determines the better explanation; however, as showing important regions rather than pixels is meaningful for images, the estimated weights using GPX would be easier to interpret on the Digits dataset. Furthermore, the degree of sparsity in the localized lasso can be changed as a hyperparameter; if the value of the hyperparameter is zero, the localized lasso is identical to the network lasso. However, because the estimated weights using the network lasso were inappropriate, those using GPX cannot be mimicked by the localized lasso. Although LOESS achieved one of the highest accuracies on the Digits dataset, the weights estimated by LOESS tended to attend white pixels and were different from what we want.

In terms of the stability of explanations, estimated weights for the same digit should be similar. Fig. 5 presents three examples of estimated weights for digit two. We found that GPX estimated similar weights for all three examples.

6) Computational Efficiency: Table VII shows the computational times of each of the methods on the Digits dataset. First of all, LOESS was the fastest method in total time. Although it needs to determine a bandwidth parameter of RBF kernel (31) by hyperparameter search, even taking that time into account, LOESS was the fastest method. LOESS does nothing during training and solves a least-squares problem for each test sample during prediction. Since it is made only of simpler operations than the other methods, it can be executed quickly.

The training time of GPX was much the same as those of GPR/SE and GPR/ARD and significantly faster than those of the localized and network lasso. Since the localized and network lasso requires the hyperparameter search, their actual training times were about 48 and 12 times longer than the times shown in the table, respectively. The prediction time of GPX was significantly faster than those of GPR/SE + LIME.
GPR/SE + SHAP, and KL. This is because GPX does not require learning model parameters at the prediction phase. On the other hand, since GPR + LIME/SHAP learns an explanation model for each sample at that time and KL requires producing a lot of predictions with slight changes in the value of each dimension of the input, their prediction times lead to increase.

7) Examples of GPX on Boston Dataset: The Boston housing dataset, referred to as “Boston” in our experiments, contains information collected by the U.S. Census Service regarding housing in the area of Boston, Massachusetts [16], and is used for predicting house prices based on the information. Table VIII lists the names of the features and their descriptions for the Boston housing dataset.
Fig. 6 presents three examples of feature contributions estimated by GPX for the Boston housing dataset.

Table VIII
FEATURE NAMES AND DESCRIPTIONS FOR THE BOSTON HOUSING DATASET

| Feature name | Description |
|--------------|-------------|
| CRIM         | Per capita crime rate by town |
| ZN           | Proportion of residential land zoned for lots over 25,000 sq.ft. |
| INDUS        | Proportion of non-retail business acres per town |
| CHAS         | Charles River dummy variable (1 if tract borders river; 0 otherwise) |
| NOX          | Nitric oxides concentration (parts per 10 million) |
| RM           | Average number of rooms per dwelling |
| AGE          | Proportion of owner-occupied units built prior to 1940 |
| DIS          | Weighted distances to five Boston employment centers |
| RAD          | Index of accessibility to radial highways |
| TAX          | Full-value property-tax rate per $10,000 |
| PTRATIO      | Pupil-teacher ratio by town |
| B            | 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town |
| LSTAT        | % lower status of the population |
| MEDV         | Median value of owner-occupied homes in $1000's |

VII. CONCLUSION

We proposed a GP-based regression model with sample-wise explanations. The proposed model assumes that each sample has a locally linear model, which is used for both prediction and explanation, and the weight vector of the locally linear model is generated from multivariate GP priors. The hyperparameters of the proposed models were estimated by maximizing the marginal likelihood, in which all the weight vectors were integrated out. Subsequently, for a test sample, the proposed model predicted its target variable and weight vector with uncertainty. In the experiments, we confirmed that the proposed model outperformed the existing globally and locally linear models and achieved comparable performances with the standard GPR in terms of predictive performance, and the proposed model was superior to the existing methods, including model-agnostic interpretable methods, in terms of three interpretability measurements. Then, we showed that the feature weights estimated by the proposed model were appropriate as the explanation.

In future studies, we will explore the extensions to the proposed model in the following directions. First, we will confirm the effectiveness of the proposed model by applying its concept to various problems in which GPs have been successfully used, such as classification, black-box optimization, and time-series forecasting. Second, we will extend the proposed model for further improvements in interpretability, e.g., by employing the mechanism of inducing sparsity for the weight vectors. Also, we will extend the proposed model to estimate how much the weight of each feature for a sample is influenced by other samples. Since this can, in essence, determine whether the weight behaves like a locally linear model or an ordinary linear model, this will improve both predictive power and interpretability.

APPENDIX

DETAILED DERIVATION OF PREDICTIVE DISTRIBUTIONS

In this appendix, we describe the derivation of predictive distributions in detail. For a new test sample \((x_s, z_s)\), our goal is to infer the predictive distributions of the target variable \(y_s\) and weight vector \(w_s\).

**Predictive Distribution of \(y_s\):** The predictive distribution of \(y_s\) is obtained similar to the standard GPR [1]. In Section IV, we demonstrated that the marginal distribution of training target variables \(y\) for GPX is defined as

\[
p(y|X, Z) = \int \int p(y, W, G|X, Z) dW dG = \mathcal{N}(y|\mathbf{0}, C)
\]  

(36)

where \(C = \sigma^2 I_n + (K + \sigma^2 I_n) \otimes ZZ^T\). According to (36), the joint marginal distribution of \(y\) and \(y_s\) is defined as

\[
p(y, y_s|X, Z, x_s, z_s) = \mathcal{N}(\begin{bmatrix} y \\ y_s \end{bmatrix}|\begin{bmatrix} \mathbf{0} \\ c_s \end{bmatrix}, \begin{bmatrix} C & c_s \\ c_s^T & c_{ss} \end{bmatrix})
\]  

(37)

where \(c_s = (k_0(x_s, x_i)z_i^T z_i)_{i=1}^n \in \mathbb{R}^n\) and \(c_{ss} = \sigma^2 + (k_0(x_s, x_s) + \sigma^2)z_s^T z_s \in \mathbb{R}\). The predictive distribution of \(y_s\) is the conditional distribution of \(y_s\) given \(y\) with training and testing inputs. Therefore, it can be obtained by applying the formula of conditional distributions for normal distributions [51, eq. (354)] to (37) as follows:

\[
p(y_s|x_s, z_s, D) = \mathcal{N}(y_s|c_s^T C^{-1} y, c_{ss} - c_s^T C^{-1} c_s)
\]  

(38)
Predictive Distribution of $\mathbf{w}_*$: The predictive distribution of $\mathbf{w}_*$ can be obtained by solving the following equation:

$$p(\mathbf{w}_*|x_*, z_*, D) = \int p(\mathbf{w}_*|X, x_*) p(W|D) dW$$

(39)

where the first integrand $p(\mathbf{w}_*|X, x_*)$ is the conditional distribution of $\mathbf{w}_*$ and the second integrand $p(W|D)$ is the posterior distribution of $W$. The conditional distribution of $\mathbf{w}_*$ is derived similar to the conditional distribution of $\mathbf{y}_*$ (38). The distribution of $W$ in which the functions $G$ are integrated out is given by

$$p(W|X) = \int p(W|G)p(G|X)dG$$

$$= \prod_{l=1}^{d} \mathcal{N}(W_{-,l}|0, K + \sigma^2_{\mathbf{w}} I_n)$$

(40)

where $W_{-,l}$ is the $l$th column vector of $W$. According to this, the joint distribution of $W$ and $\mathbf{w}_*$ is defined as

$$p(W, \mathbf{w}_*|X, x_*) = \prod_{l=1}^{d} \mathcal{N}(\mathbf{w}_{-,l}|0, \left[K + \sigma^2_{\mathbf{w}} I_n, k_s, k_{**}\right])$$

where we let $k_s = (k_0(x_*, x_0))_{n=1}^n$ and $k_{**} = k_0(x_*, x_0) + \sigma^2_{\mathbf{w}}$. Consequently, we obtain the conditional distribution of $\mathbf{w}_*$ by applying the formula of conditional distributions for normal distributions [51, eq. (354)] to (41) as follows:

$$p(\mathbf{w}_*|W, X, x_*) = \prod_{l=1}^{d} \mathcal{N}(\mathbf{w}_{-,l}|k_s^{-1}(K + \sigma^2_{\mathbf{w}} I_n)^{-1} W_{-,l} - k_{**})$$

(42)

Here, we can rewrite (42) as a single $d$-dimensional multivariate normal distribution as follows:

$$p(y, W, G|X, Z) = \mathcal{N}(\text{vec}(G)|0_n, \bar{K}) \mathcal{N}(\text{vec}(W)|\text{vec}(G), \sigma^2_{\mathbf{w}} I_{nd})$$

$$\times \mathcal{N}(y|\bar{Z}\text{vec}(W), \sigma^2_{\mathbf{w}} I_{nd})$$

(43)

where $\bar{K}$ is a block diagonal matrix of order $nd$ whose block is $K$, vec() is a function that flattens the input matrix in column-major order, and $\bar{y}_* = (k_0(x_*, x_0) + \sigma^2_{\mathbf{w}} I_d) \mathbf{k}_s$ is an $nd \times d$ block matrix, where each block is an $n \times 1$ matrix and the $(l, l)$-block of the block matrix is $(k_0(x_*, x_0))_{n=1}^n$ for $l = 1, 2, \ldots, d$, while the other blocks are zero matrices. By letting $A = \mathbf{k}_s^T(K + \sigma^2_{\mathbf{w}} I_n)^{-1}$, we obtain

$$p(\mathbf{w}_*|W, X, x_*) = \mathcal{N}(\mathbf{w}_*|\text{Avec}(W), \bar{\mathbf{c}}_* - \mathbf{A}\bar{k}_*)$$

(44)

To derive the posterior distribution of $W$, $p(W|D)$, we first find the posterior distribution of $W$ and $D$. This distribution is straightforwardly obtained as

$$p(W, D) = \prod_{l=1}^{d} \mathcal{N}(W_{-,l}|0, K) \prod_{i=1}^{n} \mathcal{N}(y_i|w_i^T z_i, \sigma^2_{\mathbf{y}})$$

(45)

which can be rewritten as

$$p(W, D) = \mathcal{N}(\text{vec}(W)|0, \bar{K}) \mathcal{N}(y|\bar{Z}\text{vec}(W), \sigma^2_{\mathbf{y}} I_n)$$

(46)

where $\bar{Z} = (\text{diag}(Z_1), \text{diag}(Z_2), \ldots, \text{diag}(Z_d)) \in \mathbb{R}^{n \times nd}$. By applying the formula of conditional distributions of normal distributions [52, eqs. (2.113)-(2.117)] to (46), we can obtain

$$p(W|D) = \mathcal{N}(\text{vec}(W)|\sigma^2_{\mathbf{y}} \Sigma \bar{Z}^T y, \Sigma)$$

(47)

$$\Sigma = (\bar{K} - \bar{Z}^T (\sigma^2_{\mathbf{y}} I_n)^{-1} \bar{Z})^{-1}$$

(48)

Here, the computation of $\Sigma$ requires inverting a square matrix of order $nd$ with a computational complexity of $O(n^3 d^3)$. By using the Woodbury identity [51, eq. (156)] to compute this inversion efficiently, we can transform $\Sigma$ into $S = \bar{K} - \bar{K} \bar{Z}^T \bar{Z} \bar{K}$, which requires inverting a matrix of order $n$, $D = \sigma^2_{\mathbf{y}} I_n + \bar{K} \otimes \bar{Z} \bar{Z}^T$. Consequently, we obtain

$$p(W|D) = \mathcal{N}(\text{vec}(W)|\sigma^2_{\mathbf{y}} S \bar{Z}^T y, S)$$

(49)

From (44) and (49), one can see that (39) can be represented by the following equation:

$$p(\mathbf{w}_*|x_*, z_*, D) = \int p(\mathbf{w}_*|\text{Avec}(W), \bar{\mathbf{c}}_* - \mathbf{A}\bar{k}_*)$$

$$\times \mathcal{N}(\text{vec}(W)|\sigma^2_{\mathbf{y}} S \bar{Z}^T y, S) dW$$

(50)

This integral can be obtained in a closed form, as shown in [52, eqs. (2.113)-(2.117)]. Therefore, we can obtain the predictive distribution of $\mathbf{w}_*$ as follows:

$$p(\mathbf{w}_*|x_*, z_*, D) = \mathcal{N}(\mathbf{w}_*|\sigma^2_{\mathbf{y}} A S \bar{Z}^T y, \bar{\mathbf{c}}_* - \mathbf{A}\bar{k}_* + A S \mathbf{A}^T)$$

(51)

REFERENCES

[1] C. E. Rasmussen, Gaussian Processes in Machine Learning. Berlin, Germany: Springer, 2004, pp. 63–71, doi: 10.1007/978-3-540-28650-9_4.

[2] A. G. Wilson, D. A. Knowles, and Z. Ghahramani, “Gaussian process regression networks,” in Proc. 29th Int. Conf. Mach. Learn., 2012, pp. 1139–1146.

[3] L. Csató, E. Fokoué, M. Opper, B. Schottky, and O. Winther, “Efficient approaches to Gaussian process classification,” in Proc. Adv. Neural Inf. Process. Syst., 2000, pp. 251–257.

[4] S. Roberts, M. Osborne, M. Ebden, S. Reece, N. Gibson, and S. Aigrain, “Gaussian processes for time-series modelling,” Philos. Trans. Roy. Soc. London A, Math. Phys. Sci., vol. 371, no. 1984, 2013, Art. no. 20110550.

[5] J. Snoek, H. Larochelle, and R. P. Adams, “Practical Bayesian optimization of machine learning algorithms,” in Proc. Adv. Neural Inf. Process. Syst. 2012, pp. 2951–2959.

[6] J. Gonzalez, E. Lezmi, T. Roncalli, and J. Xu, “Financial applications of Gaussian processes and Bayesian optimization,” 2019, arXiv:1903.04841.

[7] G. Camps-Valls, J. Verrelst, J. Muñoz-Marí, V. Laparra, F. Mateo-Jiménez, and J. Gomez-Diaz, “A survey on Gaussian processes for earth-observation data analysis: A comprehensive investigation,” IEEE Geosci. Remote Sens. Mag., vol. 4, no. 2, pp. 58–78, Jun. 2016.

[8] Y. Zhang, D. W. Apley, and W. Chen, “Bayesian optimization for materials design with mixed quantitative and qualitative variables,” Sci. Rep., vol. 10, no. 1, Dec. 2020, Art. no. 4924.

[9] L.-F. Cheng et al., “Sparse multi-output Gaussian processes for medical time series prediction,” 2017, arXiv:1703.09112.

[10] J. Futoma, “Gaussian process-based models for clinical time series in healthcare,” Ph.D. dissertation, Dept. Stat. Sci., Duke Univ., Durham, NC, USA, 2018.

[11] D. A. Melis and T. Jaakkola, “Towards robust interpretability with self-explaining neural networks,” in Proc. Adv. Neural Inf. Process. Syst., 2018, pp. 7775–7784.
