ACCESS CONTROL FOR PRIVACY-PRESERVING GAUSSIAN PROCESS REGRESSION

Takayuki Nakachi

Information Technology Center
University of the Ryukyus
Senbaru, Nishihara, Okinawa 903-0213, Japan

Yitu Wang

NTT Network Innovation Labs
Nippon Telegraph and Telephone Corp.
Yokosuka, Kanagawa 239-0847, Japan

ABSTRACT

In this paper, we propose access control for privacy-preserving Gaussian process regression (GPR), in which the encrypted data are generated through a random unitary transform (RUT). The proposed secure GPR enables computation in both encrypted input and output domains, and the access to inputs and prediction results can be controlled. We prove that our GPR for encrypted data has the same prediction accuracy as GPR for non-encrypted data. Furthermore, we demonstrate the effectiveness of our method by experimenting with diabetes data from the medical analysis field.

Index Terms— Gaussian process regression, random unitary transform, machine learning, secure computation

1. INTRODUCTION

Edge/cloud computing has rapidly become widespread in machine learning fields including deep neural networks (DNNs). Because edge/cloud computing depends on the reliability of service providers, privacy concerns have arisen in response to unreliability and the unauthorized use of data or loss of data due to accidents [1]. Secure computation has arisen as a possible solution to this problem [2–4]. Considerable efforts have been made in the fields of fully homomorphic encryption (HE) and multi-party computation (MPC), which enable processing in the encrypted domain [5, 6]. However, those approaches incur high computational complexity, large ciphertext sizes, and other disadvantages. Cancelable biometrics [7] and random projection (RP) [8] have thus been studied as methods to achieve low complexity. These encryption schemes are irreversible, which is preferable for security. However, it is difficult to deterministically guarantee that analysis of the encrypted data does not degrade performance.

Accordingly, we have focused on secure computation based on the random unitary transform (RUT) [9], which has much lower computational complexity and a smaller ciphertext size than either HE or MPC. Moreover, an RUT has reversibility, which deterministically guarantees that analysis of the encrypted data does not degrade performance. By applying RUTs, we proposed a secure sparse coding method for image compression and pattern recognition [10–13] and privacy-preserving Gaussian process regression (GPR) [14].

The Gaussian process (GP) based on Bayesian nonparametric analysis has attracted attention in the machine learning field [15]. GPs can encode expert knowledge into kernel functions, which enables accurate estimation based on Bayes' theorem. In addition, a GP can output the uncertainty of prediction based on the amount of data used for learning, whereas a DNN cannot provide the uncertainty. Moreover, a neural network with an infinite number of units in a single hidden layer can be represented by a GP [16]. The neural network GP (NNGP) has been reported to be equivalent to deep learning that corresponds to a multi-layer neural network [17].

In this paper, we propose an access control scheme for the privacy-preserving GPR. The paper's main contributions can be summarized as follows: (1) secure GPR computation can be performed on not only the input domain but also the output (prediction results) domain; and (2) the access to inputs and prediction results can be controlled.

The organization of the paper is as follows. Section 2 overviews GPR. Section 3 then explains the privacy-preserving GPR with access control capability. Section 4 describes the results of a simulation, and finally, Section 5 describes our conclusions and future work.

2. BACKGROUND

In this section, we overview GPR.

2.1. Gaussian Process

We consider a regression problem in which the input is $\boldsymbol{x}_i \in \mathbb{R}^D$ (a row vector) and the output is $y_i \in \mathbb{R}$. Let $\mathcal{D}_{train} = \{\boldsymbol{X}, \boldsymbol{Y}\}$ be a pair of N training data, and define $\boldsymbol{X} = [\boldsymbol{x}_1^T, \boldsymbol{x}_2^T, \cdots, \boldsymbol{x}_N^T]^T$ and $\boldsymbol{Y} = [y_1, y_2, \cdots, y_N]^T$. Suppose we are given $\boldsymbol{X}$ and a noisy output observation

$$\boldsymbol{Y} = f(\boldsymbol{X}) + \epsilon, \tag{1}$$

where ϵ represents i.i.d. Gaussian noise with a mean of zero and a variance of σ^2. Then, GP seeks to infer the latent function $f(\boldsymbol{X})$. It is expressed by

$$f(\boldsymbol{X}) \sim \mathcal{GP}(\boldsymbol{0}, \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})), \tag{2}$$

which is completely defined by the mean function (usually set to zero without loss of generality) and the so-called kernel function $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})$. The kernel function is a symmetric, positive, semidefinite covariance matrix with $K_{i,j} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The radial basis function (RBF) kernel is often used in the GP and is defined by the following equation:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \theta_1 \, exp\left(-\frac{|\boldsymbol{x}_i - \boldsymbol{x}_j|^2}{\theta_2}\right). \tag{3}$$

The hyperparameters θ_1, θ_2 can be trained well by optimizing the negative log marginal likelihood

$$\min_{\mathbf{K},\sigma} \boldsymbol{Y}^T(\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2\mathbf{I})^{-1}\boldsymbol{Y} + \log_2 |\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2\mathbf{I}|, \tag{4}$$

which can be solved by the efficient gradient descent algorithm via the partial derivatives of the marginal likelihood with respect to the hyperparameters [18].

2.2. Gaussian Process Regression

Next, we consider solving the regression problem based on the GP to obtain a prediction of the target value $y_i^* \in \mathbb{R}$ for a new input $\boldsymbol{x}_i^* \in \mathbb{R}^D$ (a row vector). We define a sample data set $\boldsymbol{X}^* = [\boldsymbol{x}_1^{*T}, \boldsymbol{x}_2^{*T}, \cdots, \boldsymbol{x}_M^{*T}]^T$, where M is the number of new input samples. Thus, the joint prior distribution of $\boldsymbol{Y}$ with $f(\boldsymbol{X}^*)$ is obtained from

$$\begin{bmatrix} \boldsymbol{Y} \\ f(\boldsymbol{X}^*) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \boldsymbol{K} + \sigma^2\mathbf{I} & (\boldsymbol{K}^*)^T \\ \boldsymbol{K}^* & \boldsymbol{K}^{**} \end{bmatrix} \right). \tag{5}$$

Here, we simplify the notation by denoting $\boldsymbol{K} = \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})$, $\boldsymbol{K}^* = \boldsymbol{K}(\boldsymbol{X}^*, \boldsymbol{X})$, and $\boldsymbol{K}^{**} = \boldsymbol{K}(\boldsymbol{X}^*, \boldsymbol{X}^*)$. By conditioning the joint Gaussian prior distribution on $\boldsymbol{Y}$, the posterior distribution of $f(\boldsymbol{X}^*)$ can be analytically derived as

$$p(f(\boldsymbol{X}^*)|(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{X}^*)) \sim \mathcal{N}(f(\boldsymbol{X}^*), \sigma^2(\boldsymbol{X}^*)), \tag{6}$$

where the prediction mean and variance are respectively given as

$$f(\boldsymbol{X}^*) = \boldsymbol{K}^*[\boldsymbol{K} + \sigma^2\boldsymbol{I}]^{-1}\boldsymbol{Y}, \tag{7}$$

$$\sigma^2(\boldsymbol{X}^*) = \boldsymbol{K}^{**} - \boldsymbol{K}^*[\boldsymbol{K} + \sigma^2\boldsymbol{I}]^{-1}\boldsymbol{K}^{**}. \tag{8}$$

3. PRIVACY-PRESERVING GAUSSIAN PROCESS REGRESSION

In this section, we propose secure GPR that allows computation in both encrypted input and output domains.

3.1. System Configuration

Figure 1 shows the system configuration of secure GP regression. At a local site, a pair of N training data $\mathcal{D}_{train} = \{\boldsymbol{X}, \boldsymbol{Y}\}$ is prepared. First, the input $\boldsymbol{X}$ is transformed into the encrypted input $\hat{\boldsymbol{X}}_{int}$ by using an RUT $\boldsymbol{Q}_p$ generated with a private key p. Next, sample elements of the pair $\{\hat{\boldsymbol{X}}_{int}, \boldsymbol{Y}\}$

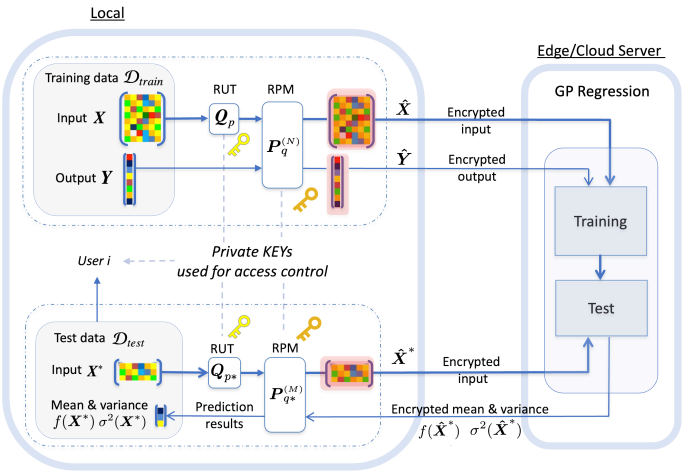


Fig. 1. System configuration of secure GP regression.

are permuted to $\hat{\mathcal{D}}_{train} = \{\hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}}\}$ by using a random permutation matrix (RPM) $\boldsymbol{P}_q^{(N)}$ generated with a private key q. The RPM is a specific type of RUT. Then, $\hat{\mathcal{D}}_{train}$ is transferred to the edge/cloud, where the kernel function and its hyperparameters θ are estimated using $\hat{\mathcal{D}}_{train}$.

At another local site, the input $\boldsymbol{X}^*$ for testing is transformed into the encrypted input $\hat{\boldsymbol{X}}_{int}^*$ by using an RUT $\boldsymbol{Q}_{p^*}$ with a private key p^*. Then, sample elements of $\hat{\boldsymbol{X}}_{int}^*$ are permuted to $\hat{\boldsymbol{X}}^*$ by using an RPM $\boldsymbol{P}_{q^*}^{(M)}$ generated with a private key q^*. The encrypted input $\hat{\boldsymbol{X}}^*$ is transferred to the edge/cloud. There, by using the predelivered $\hat{\mathcal{D}}_{train} = \{\hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}}\}$ and $\hat{\boldsymbol{X}}^*$, the corresponding encrypted mean value $f(\hat{\boldsymbol{X}}^*)$ and the encrypted variance $\sigma^2(\hat{\boldsymbol{X}}^*)$ are estimated.

As listed in Table 1, we define three access levels for decryption of the encrypted training data $\hat{\mathcal{D}}_{train} = \{\hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}}\}$ and the test data $\{\hat{\boldsymbol{X}}^*, f(\hat{\boldsymbol{X}}^*), \sigma^2(\hat{\boldsymbol{X}}^*)\}$. Access level 0 (deny) does not permit decryption of any data. Access level 1 (restricted) permits decryption only of the output $\{\hat{\boldsymbol{Y}}, f(\hat{\boldsymbol{X}}^*), \sigma^2(\hat{\boldsymbol{X}}^*)\}$. Finally, access level 2 (allow) permits decryption of all the encrypted data. These access levels are controlled by the private keys (p, q) and (p^*, q^*), which are shared with legitimate users via the public-key scheme.

3.2. Secure Computation

We design the RUTs $\{\boldsymbol{Q}_p, \boldsymbol{Q}_{p^*}\}$ and RPMs $\{\boldsymbol{P}_q^{(N)}, \boldsymbol{P}_{q^*}^{(M)}\}$ according to the following criteria:

1. Encrypt input/output as preprocessing of GPR.
2. Use the GPR algorithm without any changes.
3. Do not reduce the prediction performance of GPR.

To satisfy these criteria, the inputs $\{\boldsymbol{x}_i \in \boldsymbol{X}, \boldsymbol{x}_i^* \in \boldsymbol{X}^*\}$ are encrypted by $\{\boldsymbol{Q}_p, \boldsymbol{Q}_{p^*}\}$, and each sample element of $\mathcal{D}_{train} = \{\hat{\boldsymbol{X}}_{int}, \boldsymbol{Y}\}$ and $\hat{\boldsymbol{X}}_{int}^*$ is randomly replaced by $\{\boldsymbol{P}_q^{(N)}, \boldsymbol{P}_{q^*}^{(M)}\}$. These elements are formulated as follows:

Table 1. Access control levels in privacy-preserving GPR for encrypted training data $\hat{\mathcal{D}}_{train} = \{\hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}}\}$ and test data $\{\hat{\boldsymbol{X}}^*, f(\hat{\boldsymbol{X}}^*), \sigma^2(\hat{\boldsymbol{X}}^*)\}$. *OK* indicates that the encrypted data can be decrypted, while *NG* indicates that it cannot.

(a) Training data

Access level	0 (deny)	1 (restricted)	2 (allow)
Shared private keys	*not shared*	q	p, q
$\hat{\boldsymbol{Y}}$	*NG*	*OK*	*OK*
$\hat{\boldsymbol{X}}$	*NG*	*NG*	*OK*

(b) Test data

Access level	0 (deny)	1 (restricted)	2 (allow)
Shared private keys	*not shared*	q^*	p^*, q^*
$f(\hat{\boldsymbol{X}}^*), \sigma^2(\hat{\boldsymbol{X}}^*)$	*NG*	*OK*	*OK*
$\hat{\boldsymbol{X}}^*$	*NG*	*NG*	*OK*

$$\hat{\mathcal{D}}_{train} = \{\hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}}\} = \{\boldsymbol{P}_q^{(N)} \boldsymbol{X} \boldsymbol{Q}_p, \boldsymbol{P}_q^{(N)} \boldsymbol{Y}\}, \quad (9)$$

$$\hat{\boldsymbol{X}}^* = \boldsymbol{P}_{q^*}^{(M)} \boldsymbol{X}^* \boldsymbol{Q}_{p^*}. \quad (10)$$

Given an encrypted input $\hat{\boldsymbol{X}}$ and a noisy output observation

$$\hat{\boldsymbol{Y}} = f(\hat{\boldsymbol{X}}) + \epsilon, \quad (11)$$

GP seeks to infer the latent function $f(\hat{\boldsymbol{X}})$. As in the case when the input data is not encrypted, the joint prior distribution of $\hat{\boldsymbol{Y}}$ and $f(\hat{\boldsymbol{X}}^*)$ is given by the following equation:

$$\begin{bmatrix} \hat{\boldsymbol{Y}} \\ f(\hat{\boldsymbol{X}}^*) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}, \begin{bmatrix} \hat{\boldsymbol{K}} + \sigma^2 \mathbf{I} & (\hat{\boldsymbol{K}}^*)^T \\ \hat{\boldsymbol{K}}^* & \hat{\boldsymbol{K}}^{**} \end{bmatrix} \right). \quad (12)$$

Here, we again simplify the notation as follows: $\hat{\boldsymbol{K}} = \boldsymbol{K}(\hat{\boldsymbol{X}}, \hat{\boldsymbol{X}})$, $\hat{\boldsymbol{K}}^* = \boldsymbol{K}(\hat{\boldsymbol{X}}^*, \hat{\boldsymbol{X}})$, and $\hat{\boldsymbol{K}}^{**} = \boldsymbol{K}(\hat{\boldsymbol{X}}^*, \hat{\boldsymbol{X}}^*)$. By conditioning the joint Gaussian prior distribution on $\hat{\boldsymbol{Y}}$, the posterior distribution of $f(\hat{\boldsymbol{X}}^*)$ can be analytically derived as

$$p(f(\hat{\boldsymbol{X}}^*)|(\hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}}, \hat{\boldsymbol{X}}^*)) \sim \mathcal{N}(f(\hat{\boldsymbol{X}}^*), \sigma^2(\hat{\boldsymbol{X}}^*)). \quad (13)$$

The predicted mean and variance for encrypted data are then given by

$$f(\hat{\boldsymbol{X}}^*) = \hat{\boldsymbol{K}}^* [\hat{\boldsymbol{K}} + \sigma^2 \boldsymbol{I}]^{-1} \hat{\boldsymbol{Y}}, \quad (14)$$

$$\sigma^2(\hat{\boldsymbol{X}}^*) = \hat{\boldsymbol{K}}^{**} - \hat{\boldsymbol{K}}^* [\hat{\boldsymbol{K}} + \sigma^2 \boldsymbol{I}]^{-1} \hat{\boldsymbol{K}}^{**}, \quad (15)$$

where $\hat{\boldsymbol{K}} = \boldsymbol{K}(\hat{\boldsymbol{X}}, \hat{\boldsymbol{X}})$, $\hat{\boldsymbol{K}}^* = \boldsymbol{K}(\hat{\boldsymbol{X}}^*, \hat{\boldsymbol{X}})$, and $\hat{\boldsymbol{K}}^{**} = \boldsymbol{K}(\hat{\boldsymbol{X}}^*, \hat{\boldsymbol{X}}^*)$. Substituting Eqs. (9) and (10) into Eqs. (14) and (15), we obtain the following:

$$f(\hat{\boldsymbol{X}}^*) = \boldsymbol{P}_M \boldsymbol{K}^* [\boldsymbol{K} + \sigma^2 \boldsymbol{I}]^{-1} \boldsymbol{Y} = \boldsymbol{P}_M f(\boldsymbol{X}^*), \quad (16)$$

$$\sigma^2(\hat{\boldsymbol{X}}^*) = \boldsymbol{P}_M \{\boldsymbol{K}^{**} - \boldsymbol{K}^* [\boldsymbol{K} + \sigma^2 \boldsymbol{I}]^{-1} \boldsymbol{K}^{**}\} \boldsymbol{P}_M^T = \boldsymbol{P}_M \{\sigma^2(\boldsymbol{X}^*)\} \boldsymbol{P}_M^T, \quad (17)$$

where we simplify the notation by denoting $\boldsymbol{P}_M = \boldsymbol{P}_{q^*}^{(M)}$. The detailed derivation is omitted here to conserve space, but it can be obtained from the properties of the RUT and RPM described in Sec. 3.3. From Eqs. (16) and (17), the encrypted prediction mean and variance can be decrypted by $f(\boldsymbol{X}^*) = \boldsymbol{P}_M^T f(\hat{\boldsymbol{X}}^*)$ and $\sigma^2(\boldsymbol{X}^*) = \boldsymbol{P}_M^T \{\sigma^2(\hat{\boldsymbol{X}}^*)\} \boldsymbol{P}_M$, respectively.

3.3. Encryption Based on RUT and RPM

(1) Encryption based on an RUT: $\boldsymbol{Q}_p \in \mathbb{R}^{D \times D}$ satisfies $\boldsymbol{Q}_p^H \boldsymbol{Q}_p = \boldsymbol{I}$, where $[\cdot]^H$ denotes the Hermitian transpose operation and $\boldsymbol{I}$ is the identity matrix. In addition to unitarity, $\boldsymbol{Q}_p$ must fulfill randomness. Gram-Schmidt orthogonalization is a typical method of generating $\boldsymbol{Q}_p$. The encrypted input $\hat{\boldsymbol{x}}_i = \boldsymbol{x}_i \boldsymbol{Q}_p$ has the following properties:

· Property 1: L2 norm isometry,
$$||\boldsymbol{x}_i||_2^2 = ||\hat{\boldsymbol{x}}_i||_2^2.$$
· Property 2: Conservation of Euclidean distances,
$$||\boldsymbol{x}_i - \boldsymbol{x}_j||_2^2 = ||\hat{\boldsymbol{x}}_i - \hat{\boldsymbol{x}}_j||_2^2.$$
· Property 3: Conservation of inner products,
$$\boldsymbol{x}_i^H \boldsymbol{x}_j = \hat{\boldsymbol{x}}_i^H \hat{\boldsymbol{x}}_j.$$

The same properties hold for $\boldsymbol{Q}_{p^*}$. Next, we consider the application of an RUT to a kernel function. By using property 2 above, we can prove that the following relation is satisfied: $\hat{\boldsymbol{K}} = \boldsymbol{K}$ and $\hat{\boldsymbol{K}}^{**} = \boldsymbol{K}^{**}$ for an RBF kernel. As for the RBF kernel between the training and test data, $\hat{\boldsymbol{K}}^* = \boldsymbol{K}^*$ when $\boldsymbol{Q}_p = \boldsymbol{Q}_{p^*}$, and $\hat{\boldsymbol{K}}^* \neq \boldsymbol{K}^*$ when $\boldsymbol{Q}_p \neq \boldsymbol{Q}_{p^*}$.

(2) Encryption based on an RPM: $\boldsymbol{P}_q^{(N)} \in \{1, 0\}^{N \times N}$ is an RPM with a private key q. The permutation matrix is a square binary matrix that has exactly one entry of 1 in each row and each column and 0s elsewhere. It randomly replaces each element of the training data $\{\boldsymbol{X}, \boldsymbol{Y}\}$. $\boldsymbol{P}_q^{(N)}$ is a type of RUT, i.e., it satisfies the condition of an RUT that $\{\boldsymbol{P}_q^{(N)}\}^H \boldsymbol{P}_q^{(N)} = \boldsymbol{I}$ and the condition of fulfilling randomness. The same properties hold for $\boldsymbol{P}_{q^*}^{(M)}$.

4. NUMERICAL SIMULATION

To investigate the effectiveness of our scheme, we performed numerical experiments using diabetes data from the medical analysis field.

4.1. Simulation Conditions

The diabetes data included quantitative measures for 442 diabetes patients on 10 baseline variables such as age, gender, BMI, and disease progression one year after the baseline [19] [20]. Our secure GPR (secGPR) predicted a measure of disease progression from the 10 baseline variables. The input $\boldsymbol{X}$ was the test data on the 10 baseline variables, and the output $\boldsymbol{Y}$ was the disease progression. Data from 353 patients was used for learning ($N = 353$), and the remaining data from 89 patients was used for testing ($M = 89$). We evaluated an RBF kernel. We used RUTs $\{\boldsymbol{Q}_p, \boldsymbol{Q}_{p^*}\} \in \mathbb{R}^{10 \times 10}$ generated by Gram-Schmidt orthogonalization to encrypt the inputs $\boldsymbol{X}$ and $\hat{\boldsymbol{X}}$. As for RPMs, $\boldsymbol{P}_q^{(353)}$ and $\boldsymbol{P}_{q^*}^{(89)}$ were used.

Table 2. Pearson product-moment correlation coefficient (PPMC) and mean square error (MSE) between the decrypted data of a prediction pair $\{f(\hat{\boldsymbol{X}}^*), \sigma^2(\hat{\boldsymbol{X}}^*)\}$ with input $\hat{\boldsymbol{X}}^*$ and the corresponding non-encrypted GPR data.

(a) PPMC

Access level	0 (deny)	1 (restricted)	2 (allow)
Shared private keys	*none*	q^*	p^*, q^*
$f(\hat{\boldsymbol{X}}^*)$	0.064	1.0	1.0
$\sigma^2(\hat{\boldsymbol{X}}^*)$	0.108	1.0	1.0
$\hat{\boldsymbol{X}}^*$	0.018	0.23	1.0

(b) MSE

Access level	0 (deny)	1 (restricted)	2 (allow)
Shared private keys	*none*	q^*	p^*, q^*
$f(\hat{\boldsymbol{X}}^*)$	4947.5	$\simeq 0$	$\simeq 0$
$\sigma^2(\hat{\boldsymbol{X}}^*)$	1.039	$\simeq 0$	$\simeq 0$
$\hat{\boldsymbol{X}}^*$	2.056	2.498	$\simeq 0$

4.2. Estimation Accuracy with Access Control

The estimation accuracy of the disease progression prediction results was compared between secGPR and GPR for non-encrypted data, in terms of the Pearson product-moment correlation coefficient (PPMC) and the mean square error (MSE). Generally, two samples could be regarded as uncorrelated when the absolute value of the PMCC between them was less than 0.2. When the PMCC was close to 1 and the MSE was small, there was a strong correlation, and secGPR could estimate the same values as GPR.

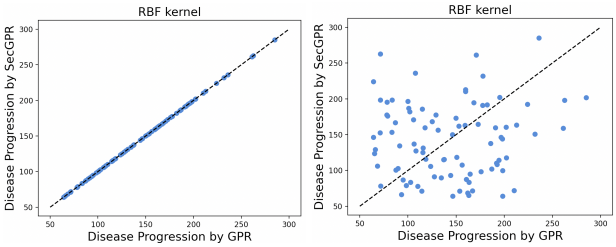
Table 2 summarizes the PPMC and MSE results. In addition, Fig. 2 shows the relationship between the disease progressions estimated by secGPR and GPR for access levels (1, 2) and for access level 0. These results show that the estimation by secGPR was almost the same as that by GPR when the access level was 1 or 2. The results also demonstrate that the access for $f(\hat{\boldsymbol{X}}^*)$ and $\sigma^2(\hat{\boldsymbol{X}}^*)$ could be controlled.

4.3. Security Strength

We also evaluated the security of the encrypted inputs $\hat{\boldsymbol{X}}$, $\hat{\boldsymbol{Y}}$, $\hat{\boldsymbol{X}}^*$, and $\{f(\hat{\boldsymbol{X}}^*), \sigma^2(\hat{\boldsymbol{X}}^*)\}$ in terms of the RUT and RPM key spaces. The key space was calculated by assuming a case of restoration by a brute-force attack. First, we consider $\hat{\boldsymbol{X}} = \boldsymbol{P}_q^{(N)} \boldsymbol{X} \boldsymbol{Q}_{\boldsymbol{p}}$. Regarding $\boldsymbol{Q}_p \in \mathbb{R}^{D \times D}$, the elements are limited to real numbers (i.e., an orthogonal matrix). The number of degrees of freedom is initially D^2, which is equal to the number of matrix elements. However, the unitarity is subject to the following conditions:

1. The column vectors of the unitary matrix must be orthogonal to each other. Therefore, the number of imposed conditional expressions is ${}_D\mathrm{C}_2 = D(D-1)/2$.

2. The norm of each column vector must be 1. Therefore, the number of imposed conditional expressions is D.



(a) Access levels 1, 2 (b) Access level 0

Fig. 2. Relationship between the disease progressions estimated by secGPR and GPR.

Table 3. Key spaces of the RUT and RPM for encrypted targets $\hat{\boldsymbol{X}}$, $\hat{\boldsymbol{Y}}$, $\hat{\boldsymbol{X}}^*$, and $\{f(\hat{\boldsymbol{X}}^*), \sigma^2(\hat{\boldsymbol{X}}^*)\}$.

Access level	0 (deny)	1 (restricted)
$\hat{\boldsymbol{Y}}$	$N!$	1
$\hat{\boldsymbol{X}}$	$8^{D(D-1)/2} \times N!$	$8^{D(D-1)/2}$
$f(\hat{\boldsymbol{X}}^*), \sigma^2(\hat{\boldsymbol{X}}^*)$	$M!$	1
$\hat{\boldsymbol{X}}^*$	$8^{D(D-1)/2} \times M!$	$8^{D(D-1)/2}$

Consequently, the number of degrees of freedom is $D^2 - [D(D-1)/2 + D] = D(D-1)/2$. Assuming that each element is represented by an 8-bit fixed-point number, the size of the key space is $8^{D(D-1)/2}$. As for $\boldsymbol{P}_q^{(N)}$, the permutation pattern is $N!$; therefore, the size of the key space (combinations of $\boldsymbol{Q}_p$ and $\boldsymbol{P}_q^{(N)}$) is $8^{D(D-1)/2} \times N!$. Next, we consider $\hat{\boldsymbol{Y}} = \boldsymbol{P}_q^{(N)} \boldsymbol{Y}$. The size of the key space depends only on $\boldsymbol{P}_q^{(N)}$, i.e., on $N!$. The key spaces for $\hat{\boldsymbol{X}}^*$ and $\{f(\hat{\boldsymbol{X}}^*), \sigma^2(\hat{\boldsymbol{X}}^*)\}$ can be calculated in the same way.

Table 3 summarizes the sizes of the key spaces. When the access level was 0, as compared with the key space used in the Advanced Encryption Standard (AES), the key space was larger than the 256-bit space in the simulation (with $N = 353$, $M = 89$, and $D = 10$). The PPMC and MSE results listed in Table 2 confirmed this. When the access level was 1, however, the key space was not large enough in comparison with the 256-bit space in the simulation ($D = 10$). Our future research will need to consider the security when the D value is small.

5. CONCLUSIONS AND FUTURE WORK

We have proposed GPR for encrypted data that is generated from an RUT. Our GPR scheme enables computation in both encrypted input and output domains. Furthermore, the access to inputs and prediction results can be controlled. The proposed secure GPR has exactly the same regression performance as non-encrypted variants of the GPR scheme. We demonstrated the effectiveness of our secure GPR on diabetes data from the medical analysis field. In the future, we will further research the security strength when the dimension of the input data is low.

6. REFERENCES

[1] C. T. Huang, L. Huang, Z. Qin, H. Yuan, L. Zhou, V. Varadharajan, and C.-C. Jay Kuo, "Survey on securing data storage in the cloud," *APSIPA Transactions on Signal and Information Processing*, vol. 3, pp. e7, 2014.

[2] J. Zhang, B. Chen, Y. Zhao, X. Cheng, and F. Hu, "Data security and privacy-preserving in edge computing paradigm: Survey and open issues," *IEEE Access*, vol. 6, pp. 18209–18237, 2018.

[3] M. Ghasemi E. Hesamifard, H. Takabi and C. Jones, "Privacy-preserving machine learning in cloud," in *Cloud Comput. Secur. Workshop (CCSW)*, 2017, p. 39–43.

[4] H. C. Tanuwidjaja, R. Choi, S. Baek, and K. Kim, "Privacy-preserving deep learning on machine learning as a service—a comprehensive survey," *IEEE Access*, vol. 8, pp. 167425–167447, 2020.

[5] R. L. Lagendijk, Z. Erkin, and M. Barni, "Encrypted signal processing for privacy protection: Conveying the utility of homomorphic encryption and multiparty computation," *IEEE Signal Processing Magazine*, vol. 30, no. 1, pp. 82–105, Jan 2013.

[6] A. Chatterjee and I. Sengupta, "Sorting of fully homomorphic encrypted cloud data: Can partitioning be effective?," *IEEE Transactions on Services Computing*, vol. 13, no. 3, pp. 545–558, 2020.

[7] Vishal M. Patel, Nalini K. Ratha, and Rama Chellappa, "Cancelable biometrics: A review," *IEEE Signal Processing Magazine*, vol. 32, no. 5, pp. 54–65, 2015.

[8] A.B.J. Teoh, A. Goh, and D.C.L. Ngo, "Random multispace quantization as an analytic mechanism for biohashing of biometric and random identity inputs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 12, pp. 1892–1901, 2006.

[9] I. Nakamura, Y. Tonomura, and H. Kiya, "Unitary transform-based template protection and its application to l2-norm minimization problems," *IEICE Transactions on Information and Systems*, vol. E99.D, pp. 60–68, 01 2016.

[10] T. Nakachi, Y. Bandoh, and H. Kiya, "Secure overcomplete dictionary learning for sparse representation," *IEICE Transactions on Information and Systems*, vol. E103.D, no. 1, pp. 50–58, 2020.

[11] T. Nakachi and H. Kiya, "Secure omp computation maintaining sparse representations and its application to etc systems," *IEICE Transactions on Information and Systems*, vol. E103.D, no. 9, pp. 1988–1997, 2020.

[12] Y. Wang and T. Nakachi, "A privacy-preserving learning framework for face recognition in edge and cloud networks," *IEEE Access*, vol. E103.D, no. 8, pp. 136056–136070, 2020.

[13] T. Nakachi Y. Bandoh and H. Kiya, "Distributed secure sparse modeling based on random unitary transform," *IEEE Access*, vol. E103.D, no. 8, pp. 211762–211772, 2020.

[14] T. Nakachi and Y. Wang, "Secure computation of gaussian process regression for data analysis," in *European Signal Processing Conference (EUSIPCO2021)*, Aug. 2021.

[15] H. Liu, Y. S. Ong, X. Shen, and J. Cai, "When Gaussian process meets big data: A review of scalable GPs," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 11, pp. 4405–4423, 2020.

[16] R. M. Neal, "Bayesian learning for neural networks," *Lecture Notes in Statistics*, 1996.

[17] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein, "Deep neural networks as gaussian processes," *arXiv*, 2018.

[18] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*, The MIT Press, 2005.

[19] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *The Annals of Statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[20] "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.