IEICE TRANSACTIONS

on Fundamentals of Electronics, Communications and Computer Sciences

DOI:10.1587/transfun.2024EAL2092

Publicized:2025/01/10

A PUBLICATION OF THE ENGINEERING SCIENCES SOCIETY

The Institute of Electronics, Information and Communication Engineers

Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3 chome, Minato-ku, TOKYO, 105-0011 JAPAN

PAPER

Distributed and Secured Gaussian Process Learning over Networks*

Ling ZHU†, *Nonmember*, **Takayuki NAKACHI**††, *Senior Member*, **Bai ZHANG**†, *and* **Yitu WANG**†, *Nonmembers*

SUMMARY Gaussian Process (GP) has been acknowledged as a powerful kernel-based machine learning technique with broad application areas, such as time series prediction and system state estimation. However, in the era of big data, new challenges are raised for GP. For example, in the presence of huge amount of data distributed at different locations, how to perform GP without being faced with significant privacy concerns? In this paper, we are aiming at constructing a distributed and secured GP learning framework over networks. Specifically, we first propose the idea of secured GP by incorporating random unitary transform, such that locally, the processing of data is guaranteed to be secure. Then, noticing that gathering data to a central node for GP learning is neither efficient nor secure, we extend secured GP into distributed learning over networks through invoking Alternating Direction Method of Multipliers (ADMM) technique, such that global optimality can be asymptotically reached only with local computations and parameter exchange. Finally, we demonstrate the performance improvements through simulation.

key words: *Gaussian process, Random unitary transform, ADMM, Secured learning, Distributed learning.*

1. Introduction

As smart mobile devices are equipped with advanced technology and design, it becomes possible to realize many novel applications, such as digital twin, smart factory and virtual reality, wherein artificial intelligence is a key enabler [2], [3]. However, to render machine learning feasible for addressing more complicated applications and problems with larger scales, a substantial amount of training data is a prerequisite [4]. Conventionally, the cloud computing paradigm is promising, i.e., a powerful computing server is deployed near a data center to provide large enough storage and computation capability [5]. Nevertheless, devices with data are globally and remotely distributed, which makes it challenging to collect data over networks, especially in the existence of numerous unreliable service providers, unauthorized users and even eavesdroppers [6].

In the literature, to protect the data privacy, some researches adopt Advanced Encryption Standard (AES) and Secure Hash Algorithm (SHA) for data encryption [7], which is computationally infeasible to decrypt the data through brute force approach. However, key distribution and inability to compute on cipher-texts diminish the efficiency, especially when large amount of devices and data are involved. To address this issue, Homomorphic Encryption (HE) and secure Multi-Party Computation (MPC) become popular for machine learning over networks, as computation directly on cipher-texts is allowed [8]. However, they are faced with the curse of dimensionality, which makes it difficult to scale to big data scenarios. Some researchers study cancelable biometrics to lower the computational complexity for computation in the encrypted domain [9]. For instance, Random projection (RP) could project an input signal into a low dimensional sub-space using a random matrix generated from random numbers [11]. Bio-Hashing [10] performs encryption based on RP that transforms the input data into a binary string called *hash-code*. Nevertheless, these encryption schemes are irreversible. Such irreversibility is preferable for preserving privacy, while it left a problem to deterministically guarantee whether the processing of the cipher-texts degrades the performance or not.

To address the above problems, we consider to utilize random unitary transform for data encryption [12]. The reasons are three-fold,

1. It not only preserves the privacy of the system, but also enables computing directly on cipher-texts.
2. It is with a desired low computational complexity, which facilitates extending the proposed algorithm to circumstances with a large cipher-text size.
3. It guarantees the reversibility of the transform, which analytically incurs no performance loss.

Based on random unitary transform, the authors have proposed a secure sparse coding method for image compression, pattern recognition, and data analysis [6], [13]–[15] Moving one step ahead, we would like to expend the application scenarios of random unitary transform into GP learning. GP is a light-weight and non-parametric learning technique, which proves to be empirically effective in various fields, e.g., non-linear regression and classification. In a nutshell, GPs encode domain and expert knowledge into kernel functions to handle both linear and non-linear data. The rea-

†Ling Zhu, Bai Zhang and Yitu Wang are with the School of Electrical and Information Engineering, in addition, Yitu Wang is also with Microelectronics and Solid-state Electronics Device Research Center (Director: Juinjei Liou) and Intelligent Equipment and Precision Measurement Technology R&D Group (2022BSB03104), North Minzu University, Yinchuan 750021, China.

††Takayuki Nakachi is with the Information Technology Center, University of the Ryukyus, Nishihara-cho, Okinawa 9030213, Japan.

*This work is in part supported by Fundamental Research Funds for Central Universities, North Minzu University (No. 2022QNPY07), National Natural Science Foundation of China (No. 62301007), NingXia Natural Science Foundation for Young Elite Scientists Sponsorship Program, and JSPS Grant-in-Aid for Scientific Research (22K04089). Part of this work has been presented at EURASIP EUSIPCO 2021 [1].

sons for considering GP are two-fold,

1. Not only a single-layer fully-connected Neural Network (NN) is equivalent to a GP, but also the exact equivalence between infinitely wide Deep Neural Networks (DNNs) and GPs is derived in [16]. Especially, the trained NN accuracy approaches that of the corresponding GP with increasing layer width, and thus, GP could achieve comparable performance as DNN with appropriate kernel design.
2. The parameters in GP can be explicitly optimized based on the Bayes theorems without stochastic gradient-based training, which allows manipulating the structure of GP for achieving excellent performance in networks.

In this paper, we propose an analytical framework for distributed and secured GP over networks. The contributions of this paper are two-fold,

1. Through invoking random unitary transform, we extend GP into secured GP, which enables directly learning on cipher-texts. In addition, it is proved both theoretically and through simulation that such encryption will not affect the performance.
2. With ADMM, parallel parameter optimization in the training phase of GP is allowed, such that a secured and scalable GP framework is established in a principled way for solving large-scale problems.

The rest of this article is organized as follows. Section II introduces GP and random unitary transform. In Section III, we propose the distributed and secured GP based on ADMM and random unitary transform. Section IV compares the performance with several baselines. Finally, Section V concludes this article.

2. Preliminary of GP and Random Unitary Transform

2.1 Preliminary of GP

In a nutshell, GP is a powerful statistical modeling tool, whose performance is highly controlled by the kernel [17], which determines the correlation between any two data points.

First of all, we briefly introduce GP. Given input $\mathbf{X}$ and noisy observations $\mathbf{Y}$

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

where

$$\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_N \end{bmatrix}, \mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_N \end{bmatrix}, \tag{2}$$

ϵ is an i.i.d. Gaussian noise with zero mean and σ^2 variance, which is caused by system errors, such as measurement and modeling inaccuracy. Under the framework of GP, the mapping function $f(\cdot)$ is approximated according to a probability distribution as

$$f(\mathbf{X}) \sim \mathcal{GP}(m(\mathbf{X}), K(\mathbf{X}, \mathbf{X})). \tag{3}$$

It is seen that the approximation accuracy is entirely controlled by the mean function $m(\mathbf{X})$, w.l.o.g. set to zero, and the covariance function $K(\mathbf{X}, \mathbf{X})$, which is called kernel of GP.

When applying GP to perform data regression, i.e., inferring the distribution of $\mathbf{Y}^*$ given a new input $\mathbf{X}^*$, where

$$\mathbf{X}^* = \begin{bmatrix} x_1^* \\ x_2^* \\ \cdot \\ \cdot \\ \cdot \\ x_M^* \end{bmatrix}, \mathbf{Y}^* = \begin{bmatrix} y_1^* \\ y_2^* \\ \cdot \\ \cdot \\ \cdot \\ y_M^* \end{bmatrix}, \tag{4}$$

we first derive the joint prior distribution of $\mathbf{Y}$ together with $\mathbf{Y}^*$ as $\begin{bmatrix} \mathbf{Y} \\ f(\mathbf{X}^*) \end{bmatrix}$, which follows

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

where the kernel $\mathbf{K}(\mathbf{X}, \mathbf{X})$ is generally chosen as a Positive and Symmetric semi-Definite (PSD) function, and the entries of $\mathbf{K}(\cdot)$ are given as $\mathbf{K}_{ij} = K(x_i, x_j)$. Then, by conditioning the joint Gaussian prior distribution on $\mathbf{X}$, the posterior distribution of $f(\mathbf{X}^*)$ can be analytically derived as

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

where the regression mean and variance are

$$\begin{aligned} \hat{f}(\mathbf{X}^*) &= \mathbf{K}_*^T (\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I})^{-1} \mathbf{Y} \\ \sigma^2(\mathbf{X}^*) &= \mathbf{K}(\mathbf{X}^*, \mathbf{X}^*) - \mathbf{K}_*^T (\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I})^{-1} \mathbf{K}(\mathbf{X}, \mathbf{X}^*). \end{aligned} \tag{7}$$

The parameters in $\mathbf{K}$ and σ are trained according to

$$\min_{\mathbf{K}, \sigma} \mathbf{Y}^T (\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I})^{-1} \mathbf{X} + \log |\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I}|, \tag{8}$$

which can be solved by gradient algorithms, such as Adaptive Moment Estimation (Adam) [17].

2.2 Preliminary of Random Unitary Transform

In order to not only preserve the privacy of the system, but also enable computing on cipher-texts, the random unitary transform is one promising method, which proves to be effective for biometric template protection [12]. Moreover, random unitary transform provides us with a desired low computational complexity, which makes it possible to apply the proposed algorithm to the scenarios with a large cipher-text size. Therefore, the encrypted training and testing samples are generated using random unitary transform.

Any vector $\boldsymbol{v} \in \mathbb{R}^{m \times 1}$ encrypted by random unitary

matrix $\boldsymbol{Q}_p \in \mathbb{C}^{m\times m}$ with private key p can be expressed as follows,

$$\overline{\boldsymbol{v}} = f(p, \boldsymbol{v}) = \boldsymbol{Q}_p \boldsymbol{v}, \tag{9}$$

where $\overline{\boldsymbol{v}}$ is the encrypted vector, and the unitary matrix $\boldsymbol{Q}_p$ satisfies

$$\boldsymbol{Q}_p^* \boldsymbol{Q}_p = \boldsymbol{I}, \tag{10}$$

where $[\cdot]^*$ and $\boldsymbol{I}$ represents the Hermitian transpose and identity matrix, respectively. Gram-Schmidt orthogonalization can be adopted for generating $\boldsymbol{Q}_p$[†]. The encrypted vector has three properties as follows,

- Conservation of the Euclidean distances

$$||\boldsymbol{v}_i - \boldsymbol{v}_j||_2^2 = ||\overline{\boldsymbol{v}}_i - \overline{\boldsymbol{v}}_j||_2^2, \tag{11}$$

- Norm isometry

$$||\boldsymbol{v}||_2^2 = ||\overline{\boldsymbol{v}}||_2^2, \tag{12}$$

- Conservation of inner products

$$\boldsymbol{v}_i \times \boldsymbol{v}_j^T = \overline{\boldsymbol{v}}_i \times \overline{\boldsymbol{v}}_j^T, \tag{13}$$

where $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ are two distinct vectors with the same dimension.

3. Distributed and Secured GP over Networks

In this section, we first propose the idea of secured GP by utilizing random unitary transform. However, when applying secured GP to solve large-scale problems, where data are distributed at different locations, we are faced with two significant challenges,

1. The same key must be used for all devices involved to encrypt the data, which significantly diminishes the utility of secured GP.
2. Data should be gathered at the central node for GP learning, which consumes a large quantity of communication resource.

To address the above problems, we invoke ADMM to propose a distributed and secured GP learning framework, where devices could use different keys for encryption, in addition, only local computation and parameter exchange are needed to achieve global optimality.

3.1 System Configuration

In this paper, we consider a general system consisting of one central node with communication capability, i.e., the cloud server, and K edge servers, who is in charge of data collection from local user devices, and with certain communication and computation capability. To perform GP learning, denote the global training dataset as $\mathcal{D} = \{\mathbf{X}, \mathbf{Y}\}$, which is formulated by aggregating K training subsets at the edge servers, i.e., $\mathcal{D} = \{\mathcal{D}_1 \cup \mathcal{D}_2, \cdots, \cup\mathcal{D}_K\}$, where subset at the i-th edge server is denoted as $\mathcal{D}_i = \{\mathbf{X}_i, \mathbf{Y}_i\}$.

[†]Such encrypting technique has been proved to be robust in terms of brute-face attack, diversity and irreversibility [12].

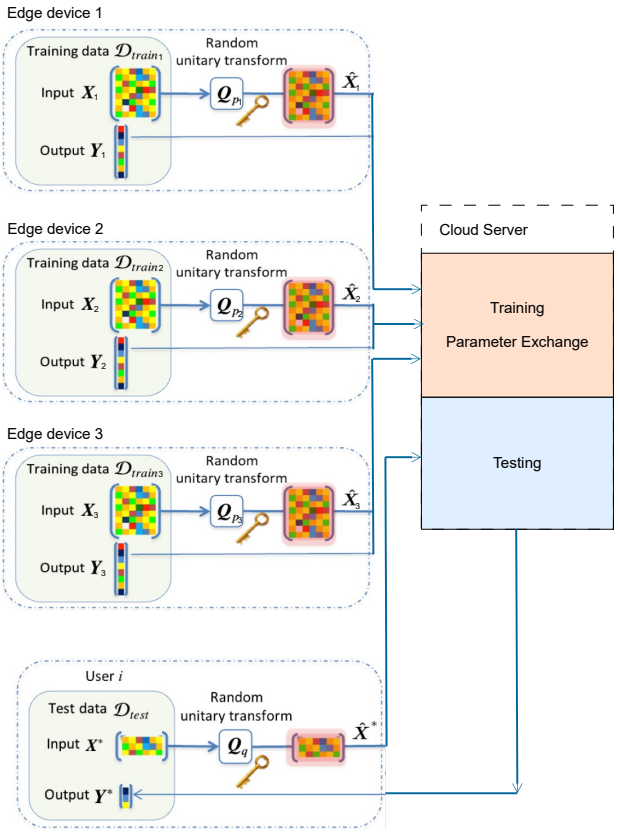

Fig. 1: System configuration

The objective of the design is as follows,

- To protect data privacy when the i-th edge server collects data from local user devices, the cloud allocates a designated key p_i to the i-th edge server so as to encrypt the data based on random unitary transform.
- To achieve the best GP learning performance without actually aggregating data from all edge servers, i.e., without the knowledge of $\mathcal{D}$, we adopt ADMM, where the cloud server is mainly responsible for updating the global ADMM parameters based on the local ADMM parameters from edge servers at the training phase, and each local GP model is trained based on its own subset $\mathcal{D}_i$.

Fig. 1 shows the system configuration the proposed distributed and secured GP learning framework.

3.2 Secured GP

Since the data is encrypted when the i-th edge server collects data from local user devices, in this subsection, we consider the problem at the i-th edge server-side.

As for the encryption of the training data, the GP input $\mathbf{X}_i$ and GP output $\mathbf{Y}_i$ are transformed into the encrypted GP input $\hat{\mathbf{X}}_i$ and GP output $\hat{\mathbf{Y}}_i$ by using the random unitary matrix $\mathbf{Q}_{p_i}$ generated with the private key p_i, which can be

calculated as follows,

$$\hat{\mathbf{X}}_i = \mathbf{X}_i\mathbf{Q}_{p_i} = \begin{bmatrix} \hat{x}_{i,1} \\ \hat{x}_{i,2} \\ \cdot \\ \cdot \\ \cdot \\ \hat{x}_{i,N} \end{bmatrix}, \hat{\mathbf{Y}}_i = \mathbf{Y}_i\mathbf{Q}_{p_i} = \begin{bmatrix} \hat{y}_{i,1} \\ \hat{y}_{i,2} \\ \cdot \\ \cdot \\ \cdot \\ \hat{y}_{i,M} \end{bmatrix}. \tag{14}$$

As for the encryption of the testing data, the encrypted GP input $\hat{\mathbf{X}}_i^*$ and GP output $\hat{\mathbf{Y}}_i^*$ for testing are generated using the random unitary matrix $\mathbf{Q}_{q_i}$ with a private key q_i as follows,

$$\hat{\mathbf{X}}_i^* = \mathbf{X}_i^*\mathbf{Q}_{q_i} = \begin{bmatrix} \hat{x}_{i,1}^* \\ \hat{x}_{i,2}^* \\ \cdot \\ \cdot \\ \cdot \\ \hat{x}_{i,N}^* \end{bmatrix}, \hat{\mathbf{Y}}_i^* = \mathbf{Y}_i^*\mathbf{Q}_{q_i} = \begin{bmatrix} \hat{y}_{i,1}^* \\ \hat{y}_{i,2}^* \\ \cdot \\ \cdot \\ \cdot \\ \hat{y}_{i,M}^* \end{bmatrix}. \tag{15}$$

Given encrypted input $\hat{\mathbf{X}}_i$ and noisy output observation

$$\hat{\mathbf{Y}}_i = f(\hat{\mathbf{X}}_i) + \epsilon, \tag{16}$$

Secured GP seeks to infer the latent function $f(\cdot)$. Similarly, the joint prior distribution of $\hat{\mathbf{Y}}_i$ together with $f(\hat{\mathbf{X}}_i^*)$ is given by the following equation

$$\begin{bmatrix} \hat{\mathbf{Y}}_i \\ f(\hat{\mathbf{X}}_i^*) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{K}(\hat{\mathbf{X}}_i, \hat{\mathbf{X}}_i) + \sigma^2\mathbf{I} & \mathbf{K}(\hat{\mathbf{X}}_i, \hat{\mathbf{X}}_i^*) \\ \mathbf{K}(\hat{\mathbf{X}}_i^*, \hat{\mathbf{X}}_i) & \mathbf{K}(\hat{\mathbf{X}}_i^*, \hat{\mathbf{X}}_i^*) \end{bmatrix} \right), \tag{17}$$

by conditioning the joint Gaussian prior distribution on $\hat{\mathbf{X}}_i$, the posterior distribution of $f(\hat{\mathbf{X}}_i^*)$ can be analytically derived as

$$p(f(\hat{\mathbf{X}}_i^*)|(\hat{\mathbf{X}}_i, \hat{\mathbf{X}}_i^*)) \sim \mathcal{N}(\hat{f}(\hat{\mathbf{X}}_i^*), \sigma^2(\hat{\mathbf{X}}_i^*)), \tag{18}$$

where the regression mean and variance are

$$\begin{aligned} \hat{f}(\hat{\mathbf{X}}_i^*) &= \mathbf{K}_*^T(\mathbf{K}(\hat{\mathbf{X}}_i, \hat{\mathbf{X}}_i) + \sigma^2\mathbf{I})^{-1}\hat{\mathbf{Y}}_i \\ \sigma^2(\hat{\mathbf{X}}_i^*) &= \mathbf{K}(\hat{\mathbf{X}}_i^*, \hat{\mathbf{X}}_i^*) - \mathbf{K}_*^T(\mathbf{K}(\hat{\mathbf{X}}_i, \hat{\mathbf{X}}_i) + \sigma^2\mathbf{I})^{-1}\mathbf{K}(\hat{\mathbf{X}}_i, \hat{\mathbf{X}}_i^*). \end{aligned} \tag{19}$$

To determine the relationship between Eq. (7) before encryption and Eq. (19) after encryption, it is necessary to consider the kernel function in closed form. In this paper, we consider two commonly adopted candidates,
1). The Radial Basis Function (RBF) kernel

$$k_{RBF}(x_k, x_j) = \exp\left(-\frac{|x_k - x_j|^2}{2\gamma^2} \right). \tag{20}$$

2). The Rational Quadratic (RQ) kernel

$$k_{RQ}(x_k, x_j) = \left(1 + \frac{|x_k - x_j|^2}{2\nu\gamma^2} \right). \tag{21}$$

When using $k_{RBF}(\cdot)$ and $k_{RQ}(\cdot)$ to process encrypted data, we have

$$\begin{aligned} k_{RBF}(\hat{x}_k, \hat{x}_j) &= \exp\left(-\frac{|\hat{x}_k - \hat{x}_j|^2}{2\gamma^2} \right) \\ &= \exp\left(-\frac{|(x_k - x_j)\mathbf{Q}_{p_i}|^2}{2\gamma^2} \right) \\ &= \exp\left(-\frac{|x_k - x_j|^2}{2\gamma^2} \right) \\ &= k_{RBF}(x_k, x_j), \end{aligned} \tag{22}$$

and

$$\begin{aligned} k_{RQ}(\hat{x}_k, \hat{x}_j) &= \left(1 + \frac{|\hat{x}_k - \hat{x}_j|^2}{2\nu\gamma^2} \right) \\ &= \left(1 + \frac{|(x_k - x_j)\mathbf{Q}_{p_i}^2}{2\nu\gamma^2} \right) \\ &= k_{RQ}(x_k, x_j). \end{aligned} \tag{23}$$

However, the covariance between the encrypted input $\hat{\mathbf{X}}_i$ for training and the encrypted input $\hat{\mathbf{X}}_i^*$ for testing does not match the covariance for the non-encrypted data. This is because the encrypted training data $\hat{\mathbf{X}}_i$ and the encrypted test data $\hat{\mathbf{X}}_i^*$ are generated using different random unitary transforms $\mathbf{Q}_{p_i}$ and $\mathbf{Q}_{q_i}$. Take RBF kernel for example,

$$\begin{aligned} k_{RBF}(\hat{x}_k, \hat{x}_j^*) &= \exp\left(-\frac{|\hat{x}_k - \hat{x}_j^*|^2}{2\gamma^2} \right) \\ &= \exp\left(-\frac{|x_k\mathbf{Q}_{p_i} - x_j\mathbf{Q}_{q_i})|^2}{2\gamma^2} \right) \\ &\neq k_{RBF}(x_k, x_j^*), \end{aligned} \tag{24}$$

When the random unitary transform is generated using the same private key ($p_i = q_i$) for training and testing, $k_{RBF}(\hat{x}_k, \hat{x}_j^*) = k_{RBF}(x_k, x_j^*)$ is satisfied. Therefore, the prediction mean and variance estimated by secured GP are equal to those estimated by GP for non-encrypted data, i.e.,

$$\begin{aligned} \hat{f}(\hat{\mathbf{X}}_i^*) &= \hat{f}(\mathbf{X}_i^*)\mathbf{Q}_{p_i}, \\ \sigma^2(\hat{\mathbf{X}}_i^*) &= \sigma^2(\mathbf{X}_i^*). \end{aligned} \tag{25}$$

We can use the keys p_i and q_i to control privacy. For legitimate users, we distribute the same key ($p_i = q_i$).

3.3 Distributed and Secured GP

To perform secured GP learning based on $\mathcal{D}$ without actually aggregating $\{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_K\}$ to the cloud server, we propose a distributed and secured GP learning framework based on ADMM in this sub-section.

For this purpose, we first approximate the probability distribution of $\mathcal{D}$ by a product of the probability distributions of $\mathcal{D}_i, \forall i \in \{1, 2, \cdots, K\}$ as follows [18],

$$p(\hat{\mathbf{Y}}|\hat{\mathbf{X}}; \theta) \approx \prod_{i=1}^{K} p_i(\hat{\mathbf{Y}}_i|\hat{\mathbf{X}}_i; \theta), \tag{26}$$

$$\log p(\hat{\mathbf{Y}}|\hat{\mathbf{X}}; \theta) \approx \sum_{i=1}^{K} \log p_i(\hat{\mathbf{Y}}_i|\hat{\mathbf{X}}_i; \theta), \tag{27}$$

where θ denotes the parameters in $\mathbf{K}$ and σ that need to be trained. The philosophy behind such approximation is to approximate the covariance matrix of full dataset with a block-diagonal matrix of the same size. In this stand, the standard centralized training Eq. (8) can be partitioned into parallel parameter training, which is expressed as

$$\begin{aligned}\min_{\theta} \sum_{i=1}^{K} \hat{\mathbf{Y}}_i^T (\mathbf{K}(\hat{\mathbf{X}}_i, \hat{\mathbf{X}}_i) + \sigma^2 \mathbf{I})^{-1} \hat{\mathbf{X}}_i \\ + \log |\mathbf{K}(\hat{\mathbf{X}}_i, \hat{\mathbf{X}}_i) + \sigma^2 \mathbf{I}|.\end{aligned} \tag{28}$$

Therefore, each local GP model at edge servers merely needs to optimize its own cost function as in Eq. (28) w.r.t θ, where the calculation only involves operations on the small covariance matrix $\mathbf{K}(\hat{\mathbf{X}}_i, \hat{\mathbf{X}}_i)$ instead of the full covariance matrix, whose size is $n_i \times n_i$, where n_i denotes the number of data points in the subset $\mathcal{D}_i$. In this sense, the computational complexity for each local secured GP model can be reduced from $O((\sum_i n_i)^3)$ to $O(n_i^3)$.

However, the training in Eq. (28) is non-trivial. Specifically, for each iteration step, the local cost and local derivative information must be collected at the cloud server and coordinated as the global cost and derivative among the edge servers. Since the number of iterations required for the convergence is generally not small, the communication overhead should be large. In the meantime, the rigorous synchronous requirement also significantly restricts its practical application in real systems. To address this problem, we invoke the ADMM technique, where the original problem is decomposed into small local subproblems that can be solved in a coordinated way [19]. Based on ADMM, problem in Eq. (28) an be recast into the following optimization problem

$$\begin{aligned}\min_{\theta_i} \sum_{i=1}^{K} \hat{\mathbf{Y}}_i^T (\mathbf{K}(\hat{\mathbf{X}}_i, \hat{\mathbf{X}}_i) + \sigma^2 \mathbf{I})^{-1} \hat{\mathbf{X}}_i \\ + \log |\mathbf{K}(\hat{\mathbf{X}}_i, \hat{\mathbf{X}}_i) + \sigma^2 \mathbf{I}|, \\ s.t. \theta_i - \theta = 0, \forall i \in \{1, 2, \cdots, K\},\end{aligned} \tag{29}$$

where θ_i represents the local parameter. With the new optimization problem in Eq. (29), each local secured GP model is free to train its own local parameter θ_i based on the local subset $\mathcal{D}_i$. By applying ADMM, the local parameters will eventually converge to the global parameter θ after certain iterations.

To solve problem in Eq. (29) with ADMM, we formulate the augmented Lagrangian as

$$\begin{aligned}L = \sum_{i=1}^{K} \hat{\mathbf{Y}}_i^T (\mathbf{K}(\hat{\mathbf{X}}_i, \hat{\mathbf{X}}_i) + \sigma^2 \mathbf{I})^{-1} \hat{\mathbf{X}}_i \\ + \log |\mathbf{K}(\hat{\mathbf{X}}_i, \hat{\mathbf{X}}_i) + \sigma^2 \mathbf{I}| \\ + \sum_{i=1}^{K} \lambda_i (\theta_i - \theta) + \sum_{i=1}^{K} \frac{\alpha}{2} ||\theta_i - \theta||_2^2,\end{aligned} \tag{30}$$

where λ_i is the dual variable and $\alpha > 0$ is a fixed augmented Lagrangian parameter. The sequential update of ADMM parameters in the $(t+1)$-th iteration can be derived as

$$\begin{aligned}\theta_i^{t+1} &= \arg\min_{\theta_i^t} L^t(\theta_i^t, \theta^t, \lambda_i^t), \\ \theta^{t+1} &= \frac{1}{K} \sum_{i=1}^{K} (\theta_i^{t+1} + \frac{1}{\alpha} \lambda_i^t), \\ \lambda_i^{t+1} &= \lambda_i^t + \alpha(\theta_i^{t+1} - \theta^{t+1}),\end{aligned} \tag{31}$$

where the first equation optimizes the local cost function; the second equation reduces the distance between θ_i and θ. Since the marginal likelihood function for hyper-parameter optimization is a well established non-convex optimization problem, the convergence of ADMM cannot be guaranteed. But fortunately, a few iterations is often sufficient for ADMM to converge to an acceptable accuracy level in practical applications. Therefore, ADMM has been applied to a variety of distributed settings in machine learning such as model fitting, regression, and classification, where the objective function is non-convex, such as [20]–[22]. We verify through simulation that the centralized GP and the distributed GP will converge to the same loss level, which proves that there will be little performance loss by adopting ADMM. In addition, we evaluated the performance loss induced by non-convergence or convergence to another local optima for such non-convex problem by evaluating on two large-scale datasets, namely the Protein Dataset and the Chem Dataset. As shown in Fig. 7and Fig. 8, it is discovered that such performance loss becomes smaller when the number of iterations is chosen larger. With sufficient number of iterations, the performance gap could be small enough.

3.4 Performance Analysis

In this subsection, we analyze both the computational complexity as well as the communicational overhead of the proposed distributed computing framework, which are further compared with those of the corresponding centralized algorithm.

1. Computational complexity.

Regarding the conventional centralized algorithm, since the central node should collect the full dataset from each node as $\mathcal{D} = \cup_{i \in \{1,2,\cdots,K\}} \mathcal{D}_i$, the associated computational complexity of parameter training by solving Eq. (28) is $O((\sum_{i=1}^{K} |\mathcal{D}_i|)^3) = O((\sum_{i=1}^{K} n_i)^3)$.

Regarding the proposed distributed algorithm, as the i-th node locally optimizes the optimization problem Eq. (30), the associated computational complexity is $O((|\mathcal{D}_i|)^3) = O((n_i)^3)$, which is greatly reduced compared with the centralized algorithm. However, the computing time of the proposed distributed framework follows the order of $O(r(\max_i\{n_i\})^3)$, where r denotes the number of iterations, which is not guaranteed to be smaller than the computing time of the centralized algorithm in case $r > K^3$. But fortunately, a few iterations is often sufficient for ADMM to converge to an acceptable accuracy level in practical applications.

2. Communicational overhead.

Regarding the conventional centralized algorithm, since the central node should collect the full dataset from each node as $\mathcal{D} = \cup_{i\in\{1,2,\cdots,K\}}\mathcal{D}_i$, the associated communication overhead is $O(\sum_{i=1}^{K} n_i)$.

Regarding the proposed distributed algorithm, as iterative approach is adopted for parameter training, which is Eq. (31). During this process, $\theta_i, \forall i$ and $\lambda_i, \forall i$ are gathered at the central node, then θ is distributed to all the nodes, where the associated communication overhead follows the order of $O(rK)$, where r denotes the number of iterations and K is the number of nodes. Note that the amount of data exchange is significantly reduced compared with the centralized algorithm.

Remark 1. *To obtain the best global model parameters θ without aggregating all the training data at the center node. We propose a distributed computing framework, such that the i-th node merely needs to optimize its parameter θ_i locally. Only with parameter exchange, $\theta_i, \forall i$ could converge to θ. In this sense, huge amount of data exchange can be totally avoided, at the cost of slight performance degradation*[†].

4. Future Work

Since the kernel function $K(\mathbf{X}, \mathbf{X})$ controls the performance of GP, there exists a bunch of viable kernel functions. Some commonly used kernel functions are as follows:
1). RBF kernel in Eq. (20).
2). RQ kernel in Eq. (21).
It is proven in this research that they are compatible with the encryption method random unitary transform, i.e., the prediction mean and variance estimated by secured GP are equal to those estimated by GP for non-encrypted data. Even though with RBF kernel and RQ kernel, GP could achieve adequate performance in a wide variety of scenarios, there exists several circumstances that GP cannot perform well. For instance, in Fig. 2, we provide the analysis of two different time series from a network traffic dataset, namely GEANT, in spectrum domain in our previous research [23]. Specifically, these time series are collected from different starting points (t = 1 and t = 100) with the same length (400 data points), i.e., $\{y(1), y(2), \cdots, y(400)\}$ and $\{y(100), y(101), \cdots, y(499)\}$. Besides the dominant pattern (peaks with the highest amplitude in y-axis), nondominant patterns (peaks with lower amplitude in y-axis) also exist. Moreover, the positions of these peaks (in x-axis) vary according to time. RBF kernel and RQ kernel can only capture the peak located at the origin, and thus leading to large modeling error and low prediction accuracy. To address such problem, there exists other useful kernels, such as

[†]The performance comparison between the centralized algorithm and the proposed distributed one is evaluated through simulation

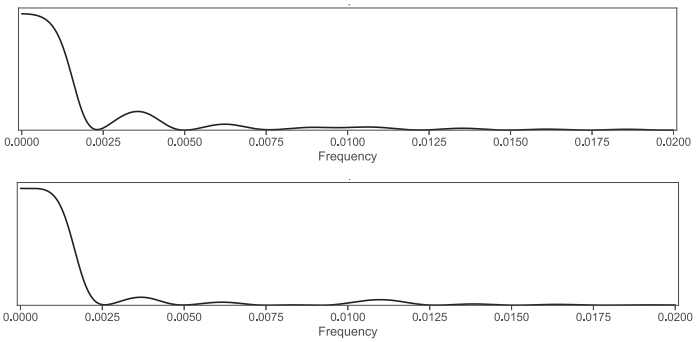

Fig. 2: Spectrum graph of two different network traffic time series

3). The Periodic kernel

$$k_{Periodic}(x_k, x_j) = \exp\left(-\frac{2\sin^2(\pi|x_k - x_j|/p)}{l^2}\right), \tag{32}$$

where p denotes the periodicity, and l is the length scale.
4) The Matern kernel

$$k_{Martern}(x_k, x_j) = \frac{2^{l-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu|x_k - x_j|}}{l}\right)^{\nu} K_\nu\left(\frac{\sqrt{2\nu|x_k - x_j|}}{l}\right), \tag{33}$$

where ν denotes the smoothness, l is the length scale, $K_.$ represents modified Bessel function of the second kind, and $\Gamma(\cdot)$ is the Gamma function.
However, they are not compatible with the encryption method random unitary transform, which limits the implementation scenarios of the proposed distributed and secured computing framework. In the future, we are willing to investigate on discovering more compatible kernel functions, and theoretically consider how to orchestrate random unitary transform with kernels, such as Periodic kernel, Matern kernel and etc., so as to enhance the significance of the proposed framework.

5. Simulation Results

In this section, we perform the following experiments using diabetes data from the medical analysis field and synthetic data to investigate the effectiveness of the proposed distributed and secured GP learning framework.

5.1 Dataset Description

We first evaluate the privacy preserving property of secured GP. The diabetes data includes quantitative measures for 442 diabetes patients on 10 baseline variables such as age, gender, BMI, and disease progression one year after baseline [24], [25]. Our proposed framework predicts a measure of disease progression from the 10 baseline variables. The GP input $\mathbf{X}$ is set as the test data of 10 baseline variables

Table 1: Performance comparison in terms of MSE and PPMCC

RBF kernel:

Private key	$p = q$	$p \neq q$
MSE	$2.16 \times 10^{-}24$	8044
PPMCC	1.0	-0.3

RQ kernel:

Private key	$p = q$	$p \neq q$
MSE	$2.10 \times 10^{-}18$	8040
PPMCC	1.0	-0.3

(i.e., $D = 10$), and the GP output $\mathbf{Y}$ is set as the disease progression. Data from 353 patients are used for training ($N = 397$) and data from 89 patients are used for testing ($M = 45$).

The input $\mathbf{X}$ is encrypted using the random unitary transform $\mathbf{Q}_p$ generated by the following equation,

$$\mathbf{Q}_p = \mathbf{H}_{PR}\mathbf{G}_p, \tag{34}$$

where $\mathbf{Q}_p$ is generated by the Gram-Schmidt orthogonalization. $\mathbf{H}_{PR}$ is a permutation matrix of $D \times D$ dimension that randomly replaces each element of the input signal $\mathbf{x} \in R^D$. An example of $\mathbf{H}_{PR}$ when the number of dimensions of the input data is $D = 4$ is given below,

$$\mathbf{H}_{PR} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}. \tag{35}$$

Since both $\mathbf{H}_{PR}$ and $\mathbf{G}_p$ have a unitary matrix, $\mathbf{Q}_p$ satisfies the condition of the random unitary matrix:

$$\mathbf{Q}_p^H \mathbf{Q}_p = (\mathbf{H}_{PR}\mathbf{G}_p)^H(\mathbf{H}_{PR}\mathbf{G}_p) = \mathbf{I}. \tag{36}$$

5.2 Performance Metrics

The estimation accuracy of secured GP is compared with that of GP for non-encrypted data based on the similarity between disease progression $\mathbf{Y}^*$ estimated by GP and $\mathbf{S}^*$ estimated by secured GP. The Mean Square Error (MSE) and Pearson Product-Moment Correlation Coefficient (PPMCC) are used to measure similarity indexes:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i^* - s_i^*)^2, \tag{37}$$

$$\text{PPMCC} = \frac{\sum_{i=1}^{N}(y_i^* - \overline{y}^*)(s_i^* - \overline{s}^*)}{\sqrt{\sum_{i=1}^{N}(y_i^* - \overline{y}^*)^2}\sqrt{\sum_{i=1}^{N}(s_i^* - \overline{s}^*)^2}}, \tag{38}$$

where $y_i^* \in \mathbf{Y}^*$ is disease progression estimated by GP and $s_i^* \in \mathbf{S}^*$ is disease progression estimated by secured GP, and $\overline{y}^*$ and $\overline{s}^*$ are the respective average values. When the PPMCC is close to 1, there is a strong correlation, and secured GP can estimate the same values as GP. We evaluated the performance of two different kernels (RBF, RQ). In addition, two patterns are verified for the random unitary transform for encrypting the training and testing data: one when the same private key ($p = q$) is used and the other when different private keys ($p \neq q$) are used.

Table I shows the MSE and PPMCC of $\mathbf{Y}^*$ and $\mathbf{S}^*$. When the private keys are the same ($p = q$), secured GP has a small MSE and the associated PPMCC $= 1$, indicating that the estimation performance of secured GP does not deteriorate compared with conventional GP. However, when the private keys are different, the MSE is large and the PPMCC is small, demonstrating that we can control the privacy through key distribution. Fig. 3 shows the relationship between the disease progression $\mathbf{Y}^*$ and $\mathbf{S}^*$ when using the RBF kernel. It illustrates that the output $\mathbf{Y}^*$ estimated by secured GP is almost the same as that of GP when $p = q$. Similar results can be obtained when using RQ kernel.

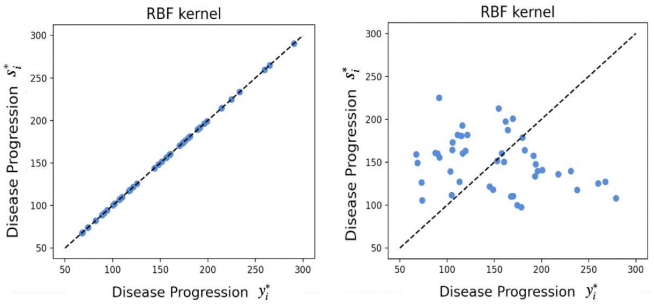

Fig. 3: Relationship between disease progression $\mathbf{Y}^*$ estimated by GP and $\mathbf{S}^*$ estimated by secured GP when using RBF kernel

5.3 Security Strength

i). Key Space:

We evaluate the safety of the encrypted input $\hat{\mathbf{X}} = \mathbf{X}\mathbf{Q}_p$ in terms of the key space of $\mathbf{Q}_p$. The key space is calculated assuming a case of restoration by brute force attack. We consider a case of the random unitary transform being generated by Eq. (34).

First, elements of the unitary transform are limited to real numbers (orthogonal matrix) for $\mathbf{G}_p$. The degree of freedom is D^2, which is equal to the number of matrix elements. However, the unitary matrix is subject to the following conditions:

1. The column vectors of the unitary matrix are orthogonal to each other. The number of conditional expressions imposed is $D^{C2} = D(D-1)/2$ (number of combinations that select two from D column vectors) from the condition.
2. The norm of each column vector $= 1$. The number of conditional expressions imposed is D from the condition.

Table 2: Absolute value of PPMCC for diabetes input $\mathbf{X}$ ($D = 10$)

Ave	Max	Min
0.122	0.526	8.51×10^{-6}

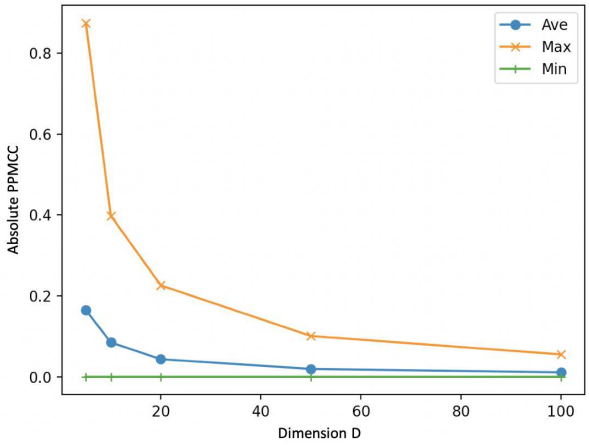

Fig. 4: Absolute value of PPMCC for synthetic input $\mathbf{X}$

Therefore, the degree of freedom is $D^2 - [D(D-1)/2 + D] = D(D-1)/2$ for the random unitary transform $\mathbf{G}_p$. Assuming each element is represented by an 8-bit fixed point number, the size of the key space is $8^{D(D-1)/2}$.

Next, the combination pattern is $D!$ for the permutation matrix $\mathbf{H}_{PR}$. Therefore, the size of the key space of the random unitary transform $\mathbf{Q}_p$ is $8^{D(D-1)/2} \times D!$. Compared with the key space used in the Advanced Encryption Standard (AES), the key space is wider than the 128-bit case and narrower than the 256-bit space when the number of elements used in this simulation is $D = 10$. If $D > 13$, it will be wider than the 256-bit key space.

ii). Irreversibility:

We investigate the security strength of the encrypted input $\hat{\mathbf{X}} = \mathbf{X}\mathbf{Q}_p$ via simulations. The security strength is evaluated based on the absolute value of the PPMCC between the original input $\mathbf{X}$ and the decrypted input $\hat{\mathbf{X}}\mathbf{Q}_q^H$ that is attacked by the illegitimate users (i.e., $p \neq q$). Generally, the two samples can be regarded as uncorrelated when the absolute value of the PPMCC between two data samples is less than 0.2. We assume 100 legitimate users (i.e., generate 100 kinds of random unitary matrices $\mathbf{Q}_p$). Then the illegitimate users try to decrypt each piece of encrypted input using 100 kinds of random unitary matrices $\mathbf{Q}_q$ that differed from the ones used in encryption. We test 10000 matrix combination patterns.

Table II shows the average, maximum, and minimum values of the absolute PPMCC for diabetes data. The absolute PPMCC is small on average, but the maximum value is relatively large. The key space is not considered to be large enough when the dimension $D = 10$. Fig. 4 shows the absolute value of the PPMCC for synthetic input $\mathbf{X}$. The input data $\mathbf{X}$ (in which each element follows a normal Gaussian distribution) is generated with different dimensions ($D = 5, 10, 20, 50, 100$). The absolute value of the PPMCC clearly decreases as the dimension D increases. If D is greater than around 30, both the average value and the maximum value are less than 0.2. From the perspective of irreversibility, security is stronger when the dimension D is larger.

5.4 Convergence of the Distributed and Secured GP

We evaluate the convergence property of the proposed distributed and secured GP framework, where the system consists of one cloud server and 10 edge servers. The number of training data at the edge servers satisfies $n_k = n_j, \forall i, j \in \{1, 2, \cdots, K\}$, and the subsets satisfies $\mathcal{D}_k \cap \mathcal{D}_j = \emptyset, \forall k \neq j$. It is observed in Fig. 5 that the convergence speed of the GP with the knowledge of the full dataset $\mathcal{D}$ is faster, i.e., with approximately 40 iterations, while the distributed GP converges slower, i.e., requires about 120 iterations. However, the centralized GP and the distributed GP will converge to the same loss level, which proves that there will be little performance loss by adopting ADMM at the cost of small communicational overhead.

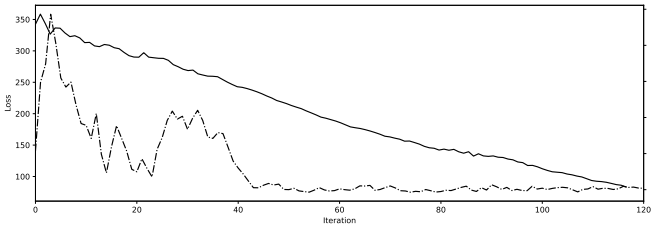

Fig. 5: Loss function of ADMM

5.5 Performance comparison

We compare the performance of the proposed distributed computing framework and the corresponding centralized computing framework using several large-scale datasets as in Fig. 6 following [26], where the Protein dataset describes the physicochemical properties of the protein tertiary structure, and the Chem dataset concerns the physical simulations relating to electron energies in molecules. 10 nodes are considered, each of which possesses non-overlapping 10% of the full training dataset. The performance of regression is quantified using the standardized mean square error (SMSE), which is defined as

$$\text{SMSE} = \frac{\sum_{i=1}^{N}(y_i^* - s_i^*)^2}{N\text{var}(y^*)}. \tag{39}$$

Fig. 7 demonstrates the performance comparison on the Protein dataset, and Fig. 8 presents the performance comparison on the Chem dataset. The number of iterations

for the centralized algorithm denotes the number of iterations of the gradient algorithm for minimizing the marginal likelihood, while the number of iterations for the proposed distributed algorithm represents that of ADMM. It is discovered that with larger number of iterations for the proposed distributed framework, the performance in terms of SMSE becomes closer to the centralized algorithm. However, there exists a small performance gap, which is caused by the non-convergence (or convergence to another local optima) of ADMM induced by the non-convexity of the marginal likelihood function.

Dataset	Dimension of the data	Training samples	Testing samples
Protein	9	35000	10730
Chem	15	60000	11969

Fig. 6: Brief Introduction of the Datasets

	100 iterations	300 iterations	500 iterations
Centralized algorithm	0.324		
Proposed framework	0.446	0.371	0.329

Fig. 7: Results on the Protein Dataset in terms of SMSE

	100 iterations	500 iterations	1000 iterations
Centralized algorithm	0.022		
Proposed framework	0.053	0.031	0.024

Fig. 8: Results on the Chem Dataset in terms of SMSE

6. Conclusions

In this paper, we proposed a distributed and secured G-P learning framework. Especially, we have two findings: 1). The proposed secured GP is deterministically reversible, and thus, we will not suffer from performance degradation when directly computing on cipher-texts. 2). By incorporating ADMM, not only each edge server could use a distinct key for encryption, but also the communication overhead is largely reduced for reaching the global optimality, which significantly extends the implementation scenarios of the proposed framework. In addition, we verified the effectiveness of the proposed framework using diabetes data from the medical analysis field and synthetic data. In the future, we are willing to theoretically consider how to orchestrate random unitary transform with more kernel functions, such as Periodic kernel, Matern kernel and etc., so as to further extend the implementation scenarios of the proposed framework.

References

[1] T. Nakachi and Y. Wang, "Secure computation of Gaussian process regression for data analysis," *EURASIP EUSIPCO 2021*, pp. 1441-1445, Aug. 2021.

[2] Y. Wang, W. Wang, V. K. Lau, T. Nakachi and Z. Zhang, "Stochastic resource allocation and delay analysis for mobile edge computing systems," *IEEE Trans. Commun.*, vol. 71, no. 7, pp. 4018-4033, Jul. 2023.

[3] H. Jiang, X. Dai, Z. Xiao and A. K. Iyengar, "Joint task offloading and resource allocation for energy-constrained mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 7, pp. 4000-4015, Jul. 2023.

[4] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen and J. S. Rellermeyer, "A survey on distributed machine learning," *ACM Comput. Surveys*, vol. 53, no. 2, pp. 1-33, Mar. 2020.

[5] M. M. Sadeeq, N. M. Abdulkareem, S. R. Zeebaree, D. M. Ahmed, A. S. Sami and R. R. Zebari, "IoT and cloud computing issues, challenges and opportunities: A review," *Qubahan Academic J.*, vol. 1, no. 2, pp. 1-7, Mar. 2021.

[6] Y. Wang and T. Nakachi, "A privacy-preserving learning framework for face recognition in edge and cloud networks," *IEEE Access*, vol. 8, pp. 136056-136070, Jul. 2020.

[7] A. Khalid, A. Aziz, C. Wang, M. ONeill and W. Liu, "Resource-shared crypto-coprocessor of AES Enc/Dec with SHA-3," *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 67, no. 12, pp. 4869-4882, Dec. 2020.

[8] C. V. Mouchet, "Multiparty homomorphic encryption: From theory to practice," *EPFL Press*, Aug. 2023.

[9] V. Patel, N. Ratha, and R. Chellappa, "Cancelable biometrics: A review," *IEEE Signal Process. Mag.*, vol. 32, no. 5, pp. 54-65, 2015.

[10] 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 Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 12, pp. 1892-1901, 2006.

[11] M. Binjubeir, A. A. Ahmed, M. A. B. Ismail, A. S. Sadiq and M. K. Khan, "Comprehensive survey on big data privacy protection," *IEEE Access*, vol. 8, pp. 20067-20079, Dec. 2019.

[12] I. Nakamura, Y. Tonomura, and H. Kiya, "Unitary transform-based template protection and its application to l2-norm minimization problems," *IEICE Trans. Inf. Syst.*, vol. E99-D, no. 1, pp. 60-68, 2016.

[13] T. Nakachi, Y. Bandoh and H. Kiya, "Secure overcomplete dictionary learning for sparse representation," *IEICE Trans. Inf. Syst.*, vol. E103.D, no. 1, pp. 50-58, Jan. 2020.

[14] T. Nakachi and H. Kiya, "Secure OMP computation maintaining sparse representations and its application to EtC systems," *IEICE Trans. Inf. Syst.*, vol. E103.D, no. 9, pp. 1988-1997, 2020.

[15] Y. Bandoh, T. Nakachi and H. Kiya, "Distributed secure sparse modeling based on random unitary transform," *IEEE Access*, vol. 8, pp. 211762-211772, Aug. 2020.

[16] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein, "Deep neural networks as gaussian processes," arXiv preprint arXiv:1711.00165, 2017.

[17] C. E. Rasmussen and C. I. K. Williams, "Gaussian processes for machine learning," *MIT Press*, 2006.

[18] J. W. Ng and M. P. Deisenroth, "Hierarchical mixture-of-experts model for large-scale Gaussian process regression," arXiv preprint arXiv:1412.3078, 2014.

[19] Y. Xu, F. Yin, W. Xu, J. Lin, and S. Cui, "Wireless traffic prediction with scalable Gaussian process: Framework, algorithms, and verification," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1291-1306, 2019.

[20] H. Zhang, J. Gao, J. Qian, J. Yang, C. Xu, and B. Zhang, "Linear regression problem relaxations solved by nonconvex ADMM with convergence analysis," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 34, no. 2, Feb. 2024.

[21] R. F. Barber, and E. Y. Sidky, "Convergence for nonconvex ADMM,

with applications to CT imaging," *J. Machine Learning Research*, vol. 25, no. 38, pp. 1-46, 2024.

[22] Z. Zha, X. Zhang, Y. Wu, Q. Wang, X. Liu, L. Tang, and X. Yuan, "Non-convex weighted l_p nuclear norm based ADMM framework for image restoration," *Neurocomputing*, vol. 311, pp. 209-224, Oct. 2018.

[23] Y. Wang, T. Nakachi, and W. Wang, "Pattern discovery and multi-slot-ahead forecast of network traffic: A revisiting to Gaussian process," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 2, pp. 1691-1706, Jun. 2023.

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

[25] scikit-learn, https://scikit-learn.org/stable/index.html.

[26] H. Liu, J. Cai, Y. S. Ong, and Y. Wang, "Understanding and comparing scalable Gaussian process regression for big data," *Knowledge-Based Syst.*, vol. 164, pp. 324-335, Jan. 2019.

Ling Zhu received the master's degree from Tongji University, Shanghai, China, in 2011. She is currently a Lecturer with the School of Electrical and Information Engineering, North Minzu University, China.

Takayuki Nakachi received the Ph.D degree in electrical engineering from Keio University, Tokyo, Japan, in 1997. From 1997 to 2021, he was a senior researcher with Nippon Telegraph and Telephone Corporation (NTT), Japan. From 2006 to 2007, he was a visiting scientist at Stanford University. He is currently a Professor with the Information Technology Center, University of the Ryukyus, Japan.

Bai Zhang received the Ph.D degree from Beijing University of Technology, Beijing, China, in 2014. He is currently an associated professor with the School of Electrical and Information Engineering, North Minzu University, China.

Yitu Wang received the Ph.D. degree from Zhejiang University, Hangzhou, China, in 2018. From August to November 2014, he was a visiting scholar with the University of Paris-Sud, Orsay, France. From January 2019 to March 2022, he was a researcher with NTT Innovation Laboratories, NTT Corporation, Japan. He is currently a Lecturer with the School of Electrical and Information Engineering, North Minzu University, China.