Future Generation Computer Systems 82 (2018) 679–688

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Efficient multiple incremental computation for Kernel Ridge Regression with Bayesian uncertainty modeling

Bo-Wei Chen [a,*], Nik Nailah Binti Abdullah [a], Sangoh Park [b], Y. Gu [a,c]

[a] School of Information Technology, Monash University, Australia
[b] School of Computer Engineering, Chungang University, South Korea
[c] Beijing University of Science and Technology, China

HIGHLIGHTS

- Proposed method supports both incremental/decremental analyses for multiple samples.
- Large dataset can be divided into subsets and fed into the system batch by batch.
- Proper batch size for online learning in intrinsic and empirical space is derived.
- Multiple incremental and decremental computation are integrated.
- Earlier version of incremental Gaussian processes is accelerated by batch computation.

ARTICLE INFO

Article history:
Received 29 April 2017
Received in revised form 5 August 2017
Accepted 28 August 2017
Available online 25 September 2017

Keywords:
Multiple incremental analysis
Multiple decremental analysis
Incremental learning
Kernel Ridge Regression (KRR)
Recursive KRR
Uncertainty analysis
Kernelized Bayesian regression
Gaussian process
Batch learning
Online learning
Edge computing
Fog computing
Regression
Classification

ABSTRACT

This study presents an efficient incremental/decremental approach for big streams based on Kernel Ridge Regression (KRR), a frequently used data analysis in cloud centers. To avoid reanalyzing the whole dataset whenever sensors receive new training data, typical incremental KRR used a single-instance mechanism for updating an existing system. However, this inevitably increased redundant computational time, not to mention applicability to big streams. To this end, the proposed mechanism supports incremental/decremental processing for both single and multiple samples (i.e., batch processing). A large scale of data can be divided into batches, processed by a machine, without sacrificing the accuracy. Moreover, incremental/decremental analyses in empirical and intrinsic space are also proposed in this study to handle different types of data either with a large number of samples or high feature dimensions, whereas typical methods focused only on one type. At the end of this study, we further the proposed mechanism to statistical Kernelized Bayesian Regression, so that uncertainty modeling with incremental/decremental computation becomes applicable. Experimental results showed that computational time was significantly reduced, better than the original nonincremental design and the typical single incremental method. Furthermore, the accuracy of the proposed method remained the same as the baselines. This implied that the system enhanced efficiency without sacrificing the accuracy. These findings proved that the proposed method was appropriate for variable streaming data analysis, thereby demonstrating the effectiveness of the proposed method.

1. Introduction

Ridge regression extends linear regression techniques, where a ridge parameter is imposed on the objective function to regularize and prevent a model [1] from overfitting. Such regularization uses $\mathcal{L}_2$ norm, or Euclidean distance, as the criterion for constraining the searching path of objective functions. Kernel Ridge Regression (KRR) further advances ridge regression by mapping feature space into hyperspace with the use of kernel functions, for example, polynomial functions and Radial Basis Functions (RBFs). In machine learning, KRR and Support Vector Machines (SVMs) have been widely used in pattern classification, especially in recent decentralized wireless sensor networks and computing platforms for the Internet of Things (IoTs).

Although KRR has a closed-form solution, which involves the inverse of matrices, calculating these inverse matrices degrades computational speeds [2]. Literature reviews [1] showed that the complexity of KRR [3] was as high as $O(N^3)$, whereas that of SVMs was $O(N^2)$, in which N stands for the number of instances in data.

* Corresponding author.
E-mail address: bo-wei.chen@monash.edu (B.-W. Chen).

https://doi.org/10.1016/j.future.2017.08.053
0167-739X/

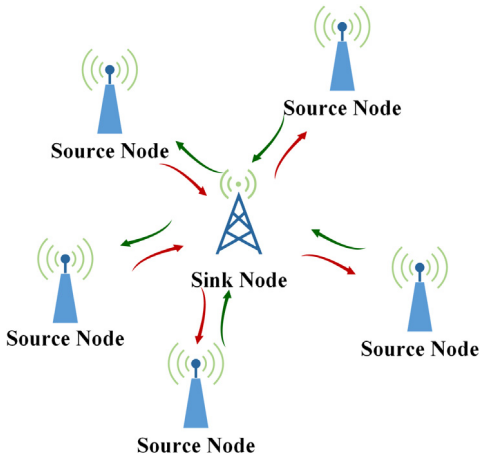

Fig. 1. Data pooling in sink nodes for wireless sensor networks.

Such a characteristic is a burden on cloud servers, which consume too much power for computation, not to mention online streaming data analysis for the IoTs [4]. The source nodes can rapidly collect information and transmit it to a fusion center, or a sink node [5,6] (see Fig. 1), which is designed for data pooling [7]. The massive amount of streams may deplete computational resources. This requires either distributed processing [8,9] or incremental analysis [10] to deal with big streams.

Unlike distributed processing, incremental analysis allows the system to add new training samples and to update itself without rescanning and reanalyzing existing datasets [11]. This is because incremental algorithms can reserve earlier calculation results for updating the system when new training samples arrive in the future. To enable incremental updates, the entire mechanism involves mathematical equations with differential forms. Furthermore, based on the equations, incremental mechanisms can be classified into two types. One is single incremental (i.e., single-instance incremental), and the other is multiple incremental (i.e., multiple-instance incremental, or equivalently batch incremental). They are both conducive to relief of computational loads.

When the size of data is too large and far beyond the capability of one machine, especially when the memory space cannot accommodate the entire data at once, incremental analysis is a feasible solution. As cloud computing for the IoTs receives significant attention in recent years, more and more incremental analyses [12–21] have been devoted to this research area. Cauwenberghs and Poggio [12] established a milestone for kernelized learning as they discovered the equilibrium between existing Lagrangian multipliers and newly added ones. A differential form was derived from the cost function of SVMs and the Karush–Kuhn–Tucker (KKT) [13] conditions. Such a differential form supported single incremental and decremental learning. The derivation was shown in a subsequent study [14]. A recursive procedure was introduced to update the matrix formed by the original support vectors and the kernel matrix when a single instance was changed. The authors also devised a strategy called "bookkeeping", or the accounting strategy mentioned in [15], to determine the largest increment/decremental amount of existing Lagrangian multipliers while maintaining the equilibrium. The model by Cauwenberghs and Poggio has inspired subsequent studies, for example [14,15], and [16]. Laskov *et al.* [15] summarized the methodology developed in [12] by presenting a systematic analytical solution. Such a solution explicitly and clearly elaborated the changes in Lagrangian multipliers with respect to three cases: Unbounded support vectors, bounded support vectors, and nonsupport vectors. Each vector was associated with one Lagrangian multiplier. Furthermore, they also presented recursive matrix updates and matrix decomposition that were conducive to incremental/decremental matrix computation. Karasuyama and Takeuchi [16] advanced the approach proposed by [12] and developed a strategy for multiple incremental/decremental learning. Multiple incremental and decremental processing were combined together during the update of the system, without being separately executed. Karasuyama and Takeuchi simplified the bookkeeping strategy mentioned in [12] by searching the shortest and easiest path when existing Lagrangian multipliers were changed. The definition of the path in their work represented a series of increment/decremental changes in the values of existing Lagrangian multipliers.

For KRR, incremental and decremental solutions become easier when compared with those of SVMs, because KRR has a closed-form solution. Recent works, such as [17,18] and [1], were examples for single-instance incremental regression. Based on kernel concepts, Engel *et al.* [17] developed a kernel recursive least squares algorithm, or incremental kernel regression. Their fundamental idea was equivalent to ordinary least squares (OLS) or linear least squares in statistics, but performed in hyperspace. The same algorithm was employed by Vaerenbergh *et al.* [18]. They furthered incremental kernel regression and integrated it into uncertainty analyses. However, no discussion on empirical-space or intrinsic-space computation was mentioned in [17] or [18]. In [1], a recursive version of KRR was introduced. It used a single incremental mechanism to update the weight vectors of the cost function. Moreover, a forgetting factor was integrated into the recursive form, where old and new training samples had different weights.

In frequentist methodologies, linear regression assumes there are sufficient observations. The weighting factor is calculated based on a deterministic process. Nonetheless, unlike frequentist concepts, Bayesian Regression concentrates on probabilistic modeling and Bayesian inference [19]. Given stochastic observations (i.e., predictor variables and dependent responses), Bayesian Regression examines uncertainty of a linear system by converting predictors and responses into statistical distributions. Posterior distributions derived from combination of likelihood and prior distributions are used for modeling a linear system in Bayesian Regression. As various statistical distributions can be used for modeling likelihood and prior distributions, resultant posterior distributions are different. When likelihood and prior distributions focus on Gaussian distributions, Kernelized Bayesian Regression is equivalent to Gaussian processes [20,22]. In contrast to KRR, which is a special case of OLS, both Kernelized Bayesian Ridge Regression and Bayesian Ridge Regression are special cases of Bayesian Regression. Incremental Kernelized Bayesian Regression is computationally intensive because it has to deal with the product of a series of inverse matrices in the exponential form along with conditional means and conditional covariance matrices. Quinonero-Candela and Winther [21] proposed an incremental solution for updating the hyperparameters (i.e., means and covariance matrices) of Gaussian distributions by devising an Expectation–Maximization (EM) algorithm when marginal likelihood distributions were computed. In this study, a multiple incremental mechanism is plugged into the updating process of Incremental Kernelized Bayesian Regression.

The contributions of this study are listed as follows.

- The proposed method supports both incremental and decremental analyses for multiple samples. A large dataset can be divided into subsets and fed into the system batch by batch. This enhances performance.
- The proper size of a batch for incremental and decremental learning in intrinsic and empirical space is derived in this article.

- Multiple incremental and decremental analyses are integrated together to update the system at the same time. Decremental learning becomes necessary when removal of unnecessary outliers is performed.
- The proposed mechanism furthers the earlier version of incremental Gaussian processes by introducing incremental/decremental mechanisms and batch learning. This speeds up the uncertainty computation.

The rest of this paper is organized as follows. Section 2 introduces the multiple incremental/decremental computation in intrinsic space for KRR, whereas Section 3 then describes the details of computation in empirical space. Next, Section 4 extends the proposed mechanism to Kernelized Bayesian Regression. Section 5 shows experimental results. Conclusions are finally drawn in Section 6.

2. Incremental/decremental kernel ridge regression in intrinsic space

KRR has two types of operation modes. One is in intrinsic space, and the other is in empirical space. Intrinsic space is used to describe dispersion matrices, also called intrinsic covariance matrices, computed based on the intrinsic dimensions of samples [1,23]. In contrast, empirical space refers to dispersion matrices, or empirical covariance matrices, computed based on the number of samples [1,24].

In KRR, after feature mapping by using kernel functions ϕ, intrinsic-space computation yields favorable complexity if the number of data N is far larger than the feature dimension M. Otherwise, empirical-space operations should be used.

Let $\{(\mathbf{x}_i, y_i)|i = 1, \ldots, N\}$ denote a pair of an M-dimensional feature vector $\mathbf{x}_i$ and its corresponding label y_i, where i specifies the indices of N training samples. The objective of KRR is to minimize the following cost function of least squares errors (LSEs).

$$\min_{\mathbf{u},b} E_{\mathrm{KRR}}(\mathbf{u}, b) = \min_{\mathbf{u},b} \left\{ \sum_{i=1}^{N} \left(\mathbf{u}^{\mathrm{T}} \phi(\mathbf{x}_i) + b - y_i \right)^2 + \rho \|\mathbf{u}\|^2 \right\} \tag{1}$$

where E_{KRR} is the cost function, $\mathbf{u}$ represents a J-by-1 weight vector, $\phi(\mathbf{x}_i)$ denotes the intrinsic-space feature vector of $\mathbf{x}_i$, b is a bias term, and ρ specifies the ridge parameter. Besides, T means the conjugate operator, and $\|\cdot\|^2$ calculates $\mathcal{L}_2$ norm. Notably, J is the degree of intrinsic space when feature vectors are transformed by a kernel function.

Eq. (1) can be rewritten as a matrix form, i.e.,

$$E_{\mathrm{KRR}}(\mathbf{u}, b) = \left\| \Phi^{\mathrm{T}} \mathbf{u} + b\mathbf{e}^{\mathrm{T}} - \mathbf{y}^{\mathrm{T}} \right\|^2 + \rho \|\mathbf{u}\|^2 \tag{2}$$

where $\mathbf{e}$ is a row vector of all ones. Individually differentiating (16) with respect to $\mathbf{u}$ and b followed by zeroing both equations gives

$$\mathbf{u} = \left(\Phi\Phi^{\mathrm{T}} + \rho\mathbf{I} \right)^{-1} \Phi \left(\mathbf{y}^{\mathrm{T}} - b\mathbf{e}^{\mathrm{T}} \right) \tag{3}$$

and

$$b = \frac{1}{N} \left(\mathbf{e}\mathbf{y}^{\mathrm{T}} - \mathbf{e}\Phi^{\mathrm{T}}\mathbf{u} \right). \tag{4}$$

Notice that $\mathbf{K} = \Phi^{\mathrm{T}}\Phi$ instead of $\Phi\Phi^{\mathrm{T}}$ mentioned in (3). Unlike the solution to kernel regression, i.e., $\mathbf{u} = (\Phi\Phi^{\mathrm{T}})^{-1}\Phi(\mathbf{y}^{\mathrm{T}} - b\mathbf{e}^{\mathrm{T}})$ where $\Phi\Phi^{\mathrm{T}}$ could be singular, KRR avoids such a problem by adding a ridge term inside. The solution to (3) and (4) can be obtained by solving a system of linear equations.

$$\begin{bmatrix} \mathbf{u} \\ b \end{bmatrix} = \begin{bmatrix} \mathbf{S} & \Phi\mathbf{e}^{\mathrm{T}} \\ \mathbf{e}\Phi^{\mathrm{T}} & N \end{bmatrix}^{-1} \begin{bmatrix} \Phi\mathbf{y}^{\mathrm{T}} \\ \mathbf{e}\mathbf{y}^{\mathrm{T}} \end{bmatrix} \tag{5}$$

where $\mathbf{S}$ denotes $\Phi\Phi^{\mathrm{T}} + \rho\mathbf{I}$.

Based on the Schur complement theory,

$$\begin{aligned} \begin{bmatrix} \mathbf{S} & \Phi\mathbf{e}^{\mathrm{T}} \\ \mathbf{e}\Phi^{\mathrm{T}} & N \end{bmatrix}^{-1} &= \begin{bmatrix} \mathbb{A} & \mathbb{U} \\ \mathbb{V} & \mathbb{N} \end{bmatrix}^{-1} \\ &= \begin{bmatrix} \mathbb{M} & -\mathbb{M}\mathbb{U}\mathbb{N}^{-1} \\ -\mathbb{N}^{-1}\mathbb{V}\mathbb{M} & \mathbb{N}^{-1}\mathbb{V}\mathbb{M}\mathbb{U}\mathbb{N}^{-1} + \mathbb{N}^{-1} \end{bmatrix} \end{aligned} \tag{6}$$

where

$$\begin{aligned} \mathbb{M} &= \left(\mathbb{A} - \mathbb{U}\mathbb{N}^{-1}\mathbb{V} \right)^{-1} \\ &= \mathbb{A}^{-1} + \mathbb{A}^{-1}\mathbb{U}\left(\mathbb{N} - \mathbb{V}\mathbb{A}^{-1}\mathbb{U} \right)^{-1}\mathbb{V}\mathbb{A}^{-1} \\ &= \mathbf{S}^{-1} + \mathbf{S}^{-1}\Phi\mathbf{e}^{\mathrm{T}}\left(\mathbf{e}\mathbf{e}^{\mathrm{T}} - \mathbf{e}\Phi^{\mathrm{T}}\mathbf{S}^{-1}\Phi\mathbf{e}^{\mathrm{T}} \right)^{-1}\mathbf{e}\Phi^{\mathrm{T}}\mathbf{S}^{-1}. \end{aligned} \tag{7}$$

This form becomes useful in the following incremental and decremental processes as $\mathbf{S}^{-1}$ is repeatedly used in the process rather than $\mathbf{S}$.

2.1. Single incremental and decremental processes

For intrinsic space, single incremental and decremental processes are straightforward. During the incremental phase, given a new training sample $(\mathbf{x}_c, y_c)$, the update of (3) becomes

$$\begin{aligned} &\mathbf{u}[\ell+1] = \\ &\quad \mathbf{S}[\ell+1]^{-1}\Phi[\ell+1]\left(\mathbf{y}^{\mathrm{T}}[\ell+1] - b[\ell+1]\,\mathbf{e}^{\mathrm{T}}[\ell+1] \right) \end{aligned} \tag{8}$$

and

$$\begin{aligned} &b[\ell+1] = \\ &\quad \frac{1}{N+1}\left(\mathbf{e}[\ell+1]\mathbf{y}^{\mathrm{T}}[\ell+1] - \mathbf{e}[\ell+1]\Phi^{\mathrm{T}}[\ell+1]\mathbf{u}[\ell+1] \right) \end{aligned} \tag{9}$$

where

$$\begin{cases} \mathbf{S}[\ell+1]^{-1} = \left(\mathbf{S}[\ell] + \phi(\mathbf{x}_c)\,\phi(\mathbf{x}_c)^{\mathrm{T}} \right)^{-1} \\ \Phi[\ell+1] = \left[\Phi[\ell] \quad \phi(\mathbf{x}_c) \right] \\ \mathbf{y}[\ell+1] = \left[\mathbf{y}[\ell] \quad y_c \right]. \end{cases} \tag{10}$$

In (8)–(10), ℓ denotes the current state of the system, and $\ell+1$ is the next state. To save computation of $\mathbf{S}^{-1}$, the Sherman–Morrison formula and Woodbury matrix identity [25] indicate that

$$\begin{aligned} \mathbf{S}[\ell+1]^{-1} &= \left(\mathbf{S}[\ell] + \phi(\mathbf{x}_c)\,\phi(\mathbf{x}_c)^{\mathrm{T}} \right)^{-1} \\ &= \mathbf{S}[\ell]^{-1} - \frac{\mathbf{S}[\ell]^{-1}\phi(\mathbf{x}_c)\,\phi(\mathbf{x}_c)^{\mathrm{T}}\mathbf{S}[\ell]^{-1}}{1 + \phi(\mathbf{x}_c)^{\mathrm{T}}\mathbf{S}[\ell]^{-1}\phi(\mathbf{x}_c)}. \end{aligned} \tag{11}$$

Regarding the decremental phase, given an index r of a sample, where $r \in \{1, \ldots, N\}$, a recursive form is created by considering the rth sample:

$$\begin{aligned} \mathbf{S}[\ell-1]^{-1} &= \left(\mathbf{S}[\ell] - \phi(\mathbf{x}_r)\,\phi(\mathbf{x}_r)^{\mathrm{T}} \right)^{-1} \\ &= \mathbf{S}[\ell]^{-1} + \frac{\mathbf{S}[\ell]^{-1}\phi(\mathbf{x}_r)\,\phi(\mathbf{x}_r)^{\mathrm{T}}\mathbf{S}[\ell]^{-1}}{1 - \phi(\mathbf{x}_r)^{\mathrm{T}}\mathbf{S}[\ell]^{-1}\phi(\mathbf{x}_r)}. \end{aligned} \tag{12}$$

For $\Phi[\ell-1]$ and $\mathbf{y}[\ell-1]$, we simply remove the corresponding column and row from $\Phi[\ell]$ and $\mathbf{y}[\ell]$, respectively.

2.2. Multiple incremental and decremental processes

For multiple incremental and decremental processes, assume the system is about to add $|C|$ new samples and remove $|R|$ existing data. The operator $|\cdot|$ denotes the size. Additionally, C and R are the sets that contain sample indices. Then, (11) and (12) respectively become

$$\begin{aligned} &\mathbf{S}[\ell+1]^{-1} \\ &\quad = \left(\mathbf{S}[\ell] + \Phi_C\Phi_C^{\mathrm{T}} \right)^{-1} \\ &\quad = \mathbf{S}[\ell]^{-1} - \mathbf{S}[\ell]^{-1}\Phi_C\left(\mathbf{I} + \Phi_C^{\mathrm{T}}\mathbf{S}[\ell]^{-1}\Phi_C \right)^{-1}\Phi_C^{\mathrm{T}}\mathbf{S}[\ell]^{-1} \end{aligned} \tag{13}$$

and

$$\begin{aligned}&\mathbf{S}[\ell-1]^{-1}\\&=\left(\mathbf{S}[\ell]-\Phi_R\Phi_R^{\mathrm{T}}\right)^{-1}\\&=\mathbf{S}[\ell]^{-1}+\mathbf{S}[\ell]^{-1}\Phi_R\left(\mathbf{I}-\Phi_R^{\mathrm{T}}\mathbf{S}[\ell]^{-1}\Phi_R\right)^{-1}\Phi_R^{\mathrm{T}}\mathbf{S}[\ell]^{-1}.\end{aligned}\tag{14}$$

To facilitate multiple incremental and decremental processes at once, combination of (13) and (14) is necessary. Let $\Phi_{\mathcal{H}}=[\Phi_C|\Phi_R]$ represent the concatenation of all the column vectors in Φ_C and Φ_R. Also denote $\Phi'_{\mathcal{H}}=[\Phi_C|-\Phi_R]^{\mathrm{T}}$ as the concatenation of all the column vectors in Φ_C and $-\Phi_R$. Therefore, combination of (13) and (14) becomes

$$\begin{aligned}&\mathbf{S}[\ell+1]^{-1}\\&=\left(\mathbf{S}[\ell]+\Phi_C\Phi_C^{\mathrm{T}}-\Phi_R\Phi_R^{\mathrm{T}}\right)^{-1}\\&=\left(\mathbf{S}[\ell]+\Phi_{\mathcal{H}}\Phi'_{\mathcal{H}}\right)^{-1}\\&=\mathbf{S}[\ell]^{-1}-\mathbf{S}[\ell]^{-1}\Phi_{\mathcal{H}}\left(\mathbf{I}+\Phi'{}_{\mathcal{H}}\mathbf{S}[\ell]^{-1}\Phi_{\mathcal{H}}\right)^{-1}\Phi'{}_{\mathcal{H}}\mathbf{S}[\ell]^{-1}.\end{aligned}\tag{15}$$

For $\Phi[\ell+1]$ and $\mathbf{y}[\ell+1]$, the system can simply remove the corresponding column(s) and row(s) from $\Phi[\ell]$ and $\mathbf{y}[\ell]$, respectively. Subsequently, new samples are appended to the end of $\Phi[\ell]$ and $\mathbf{y}[\ell]$ to generate $\Phi[\ell+1]$ and $\mathbf{y}[\ell+1]$.

The batch sizes of Φ_C and Φ_R, i.e., $|C|$ and $|R|$, can be different. Notably, the left-hand side of the two equations in (13) and (14) needs $O(J^3)$, whereas the inverse on the right-hand side requires $O(|C|^3)$ for (13) and $O(|R|^3)$ for (14), respectively [26]. To ensure performance, when the number of samples in a batch is smaller than the size of intrinsic-space features (i.e., $|C|<J$ and $|R|<J$), the system should perform an update if incremental and decremental computation is separate. For (15), $|\mathcal{H}|$ should be smaller than J. This implies a suitable batch size for time-series data, where new samples are rapidly generated and accumulated.

3. Incremental/decremental kernel ridge regression in empirical space

According to the Learning Subspace Property in [1], the weight vector $\mathbf{u}$ has the following relation between Φ and an unknown N-dimensional vector $\mathbf{a}$.

$$\mathbf{u}=\Phi\mathbf{a}.\tag{16}$$

Combining (2) and (16) yields

$$E'_{\mathrm{KRR}}(\mathbf{a},b)=\left\|\mathbf{K}\mathbf{a}+b\mathbf{e}^{\mathrm{T}}-\mathbf{y}^{\mathrm{T}}\right\|^2+\rho\mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{a}.\tag{17}$$

Rearranging the equations after differentiating (17) with respect to $\mathbf{a}$ and b yields

$$\mathbf{a}=(\mathbf{K}+\rho\mathbf{I})^{-1}\left(\mathbf{y}^{\mathrm{T}}-b\mathbf{e}^{\mathrm{T}}\right)\tag{18}$$

and

$$b=\frac{\mathbf{y}(\mathbf{K}+\rho\mathbf{I})^{-1}\mathbf{e}^{\mathrm{T}}}{\mathbf{e}(\mathbf{K}+\rho\mathbf{I})^{-1}\mathbf{e}^{\mathrm{T}}}.\tag{19}$$

3.1. Single incremental and decremental processes

Given a new training sample $(\mathbf{x}_c,y_c)$, the incremental phase is listed as follows.

$$\left(\mathbf{K}[\ell+1]+\rho\mathbf{I}[\ell+1]\right)^{-1}=\begin{bmatrix}\mathbf{K}[\ell]+\rho\mathbf{I}[\ell] & \boldsymbol{\eta}_{:,c}\\ \boldsymbol{\eta}_{:,c}^{\mathrm{T}} & K_{c,c}+\rho\end{bmatrix}^{-1}\tag{20}$$

where ":" signifies all the training samples except the new one, and $\boldsymbol{\eta}_{:,c}$ is part of the kernel matrix only based on the new sample. For simplicity, let $\mathbf{Q}$ denote $\mathbf{K}+\rho\mathbf{I}$ and $Q_{c,c}$ represent $K_{c,c}+\rho$. Then, (20) becomes

$$\mathbf{Q}^{-1}[\ell+1]=\begin{bmatrix}\mathbf{Q}[\ell] & \boldsymbol{\eta}_{:,c}\\ \boldsymbol{\eta}_{:,c}^{\mathrm{T}} & Q_{c,c}\end{bmatrix}^{-1}.\tag{21}$$

However, (20) does not save computational loads as the system calculates the inverse again. According to the Sherman–Morrison formula and Woodbury matrix identity [15,25], the inverse in (21) can be decomposed to two states. One is the current state $\mathbf{Q}^{-1}[\ell]$, and the other is $\mathbf{Q}^{-1}[\ell+1]$, shown as follows.

$$\begin{aligned}&\mathbf{Q}^{-1}[\ell+1]=\begin{bmatrix}\mathbf{Q}[\ell] & \boldsymbol{\eta}_{:,c}\\ \boldsymbol{\eta}_{:,c}^{\mathrm{T}} & Q_{c,c}\end{bmatrix}^{-1}\\&=\begin{bmatrix}\mathbf{Q}^{-1}[\ell]+z^{-1}\mathbf{G}_{:,c}\mathbf{G}_{:,c}^{\mathrm{T}} & z^{-1}\mathbf{G}_{:,c}\\ z^{-1}\mathbf{G}_{:,c}^{\mathrm{T}} & z^{-1}\end{bmatrix}\\&=\begin{bmatrix}\mathbf{Q}^{-1}[\ell] & \mathbf{0}\\ \mathbf{0} & 0\end{bmatrix}+\frac{1}{z}\begin{bmatrix}\mathbf{G}_{:,c}\\ \mathbf{1}\end{bmatrix}\begin{bmatrix}\mathbf{G}_{:,c}^{\mathrm{T}} & \mathbf{1}\end{bmatrix}\end{aligned}\tag{22}$$

where

$$\begin{cases}\mathbf{G}_{:,c}=-\mathbf{Q}^{-1}[\ell]\,\boldsymbol{\eta}_{:,c}\\ z=Q_{c,c}-\boldsymbol{\eta}_{:,c}^{\mathrm{T}}\mathbf{Q}^{-1}[\ell]\,\boldsymbol{\eta}_{:,c}.\end{cases}\tag{23}$$

Therefore, computation of the inverse in the previous state can be reserved for the next state. The incremental forms of (18) and (19) respectively become

$$\begin{aligned}&\mathbf{a}[\ell+1]\\&=\left(\mathbf{K}[\ell+1]+\rho\mathbf{I}[\ell+1]\right)^{-1}\\&\quad\times\left(\mathbf{y}^{\mathrm{T}}[\ell+1]-b[\ell+1]\,\mathbf{e}^{\mathrm{T}}[\ell+1]\right)\\&=\mathbf{Q}^{-1}[\ell+1]\left(\mathbf{y}^{\mathrm{T}}[\ell+1]-b[\ell+1]\,\mathbf{e}^{\mathrm{T}}[\ell+1]\right)\end{aligned}\tag{24}$$

and

$$b[\ell+1]=\frac{\mathbf{y}[\ell+1]\,\mathbf{Q}^{-1}[\ell+1]\,\mathbf{e}^{\mathrm{T}}[\ell+1]}{\mathbf{e}[\ell+1]\,\mathbf{Q}^{-1}[\ell+1]\,\mathbf{e}^{\mathrm{T}}[\ell+1]}.\tag{25}$$

For the decremental phase, given an index r of a sample that is about to be removed, where $r\in\{1,\ldots,N\}$, we can rearrange the elements in $\mathbf{Q}^{-1}$, so that r lies at the bottom-right corner of $\mathbf{Q}^{-1}$. Let Θ, ξ_r, and θ_r respectively signify the three blocks of $\mathbf{Q}^{-1}$, shown in (26). Besides, Θ is a matrix, ξ_r denotes a vector, and θ_r represents a scalar. Then,

$$\begin{aligned}&\mathbf{Q}^{-1}[\ell]=\begin{bmatrix}\Theta & \xi_r\\ \xi_r^{\mathrm{T}} & \theta_r\end{bmatrix}\\&=\begin{bmatrix}\mathbf{Q}^{-1}[\ell-1]+z^{-1}\mathbf{G}_{:,r}\mathbf{G}_{:,r}^{\mathrm{T}} & z^{-1}\mathbf{G}_{:,r}\\ z^{-1}\mathbf{G}_{:,r}^{\mathrm{T}} & z^{-1}\end{bmatrix}.\end{aligned}\tag{26}$$

The lower part of (26) comes from (22). Comparing the four blocks in the upper and lower parts of (22) [15] yields the following result.

$$\begin{aligned}\mathbf{Q}^{-1}[\ell-1]&=\Theta-\frac{\xi_r\xi_r^{\mathrm{T}}}{\theta_r}\\&=\mathbf{Q}^{-1}(1,1)[\ell]-\frac{\mathbf{Q}^{-1}(1,2)[\ell]\times\mathbf{Q}^{-1}(2,1)[\ell]}{\mathbf{Q}^{-1}(2,2)[\ell]}\end{aligned}\tag{27}$$

where $\mathbf{Q}^{-1}(\cdot,\cdot)$ indicates the blocks of $\mathbf{Q}^{-1}$. Substitution of (27) into (24) and (25) generates decremental forms.

3.2. Multiple incremental and decremental processes

Like Section 2.2, also assume that the system adds $|C|$ new samples and removes $|R|$ existing data. For batch incremental learning, (22) becomes

$$\mathbf{Q}^{-1}[\ell+1]=\begin{bmatrix}\mathbf{Q}^{-1}[\ell]+\mathbf{G}_{:,C}\mathbf{Z}^{-1}\mathbf{G}_{:,C}^{\mathrm{T}} & \mathbf{G}_{:,C}\mathbf{Z}^{-\mathrm{T}}\\ \mathbf{Z}^{-1}\mathbf{G}_{:,C}^{\mathrm{T}} & \mathbf{Z}^{-1}\end{bmatrix}\tag{28}$$

where $\mathbf{Z}$ and $\mathbf{G}$ are matrices computed based on $|C|$ new samples. Notably, $\mathbf{Z}$ is a matrix version of z in (23).

For batch decremental learning, (27) is replaced with (29).

$$\mathbf{Q}^{-1}[\ell-1]=\Theta-\xi_R\boldsymbol{\theta}_R^{-1}\xi_R^{\mathrm{T}}\tag{29}$$

where ξ_R and $\boldsymbol{\theta}_R$ are computed based on $|R|$ decremental samples and the residual data. This step requires the inverse of $\boldsymbol{\theta}_R$. If the

number of samples for $\mathbf{Q}^{-1}[\ell-1]$ is smaller than $|R|$, direct computation of $\mathbf{Q}^{-1}[\ell-1]$ saves more time.

To integrate multiple incremental/decremental processes together, the system should remove existing data first prior to adding new samples. Accordingly,

$$\mathbf{Q}^{-1}[\ell+1] = \begin{bmatrix} \boldsymbol{\Theta} - \boldsymbol{\xi}_R \boldsymbol{\theta}_R^{-1} \boldsymbol{\xi}_R^{\mathrm{T}} + \mathbf{G}_{:,C} \mathbf{Z}^{-1} \mathbf{G}_{:,C}^{\mathrm{T}} & \mathbf{G}_{:,C} \mathbf{Z}^{-\mathrm{T}} \\ \mathbf{Z}^{-1} \mathbf{G}_{:,C}^{\mathrm{T}} & \mathbf{Z}^{-1} \end{bmatrix}. \tag{30}$$

4. Incremental/decremental kernelized Bayesian regression

Unlike KRR that focuses on frequentist methodologies, where sufficient occurrences are observed, Bayesian Regression concentrates on uncertainty modeling. Thus, statistical distributions are introduced in Kernelized Bayesian Regression (KBR). The observed instances are sampled from a stochastic process that fits a statistical distribution. As Bayesian theory works for various distributions, this work uses Gaussian distributions as a case study for modeling incremental and decremental analysis.

To avoid confusion, this study uses the following notations to describe the relation between independent variables (i.e., predictors) and their conditional parameters in the subsequent functions.

- $P(\cdot|\cdot)$: When the transposition operator appears in the independent variable (i.e., the first operant) of a probabilistic function, the conditional parameters (i.e., the subsequent operants) still remain their original notations without adding the transposition operator.
- $\mathcal{N}(\cdot|\cdot)$: When the transpose operator is used in the independent variable of a normal distribution function, the subsequent hyperparameters reflect such a change and use the transpose operator.
- $\Sigma_{\cdot|\cdot}$ or $\boldsymbol{\Sigma}_{\cdot|\cdot}$: The first operant in the subscript is viewed as the independent argument of a covariance matrix function. When there is a transposition operation, the operant displays such an operator in the notation, e.g., $\boldsymbol{\Sigma}_{\mathbf{y}^{\mathrm{T}}|\mathbf{u},\boldsymbol{\Phi}}$. The conditional operant in the subscript remains the same form unless it is a dependent response of regression, e.g., $\boldsymbol{\Sigma}_{\mathbf{u}|\mathbf{y}^{\mathrm{T}},\boldsymbol{\Phi}}$
- $\mu_{\cdot|\cdot}$ or $\boldsymbol{\mu}_{\cdot|\cdot}$: They are based on the above-mentioned notations.

Moreover, for uncertainty modeling, b in (1) should be changed to a random variable b_i corresponding to its observed sample $(\mathbf{x}_i, y_i)$. Consider a regression model,

$$y_i = \mathbf{u}^{\mathrm{T}} \phi(\mathbf{x}_i) + b_i, \tag{31}$$

or in a matrix form,

$$\mathbf{y} = \mathbf{u}^{\mathrm{T}} \boldsymbol{\Phi} + \mathbf{b}$$

where $P(b) \sim \mathcal{N}(\mu_{\mathbf{b}}, \Sigma_{\mathbf{b}})$ and $P(\phi(\mathbf{x}_i) \in \boldsymbol{\Phi}) \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\Phi}}, \boldsymbol{\Sigma}_{\boldsymbol{\Phi}})$. Besides, $\mu_{\mathbf{b}}$ and $\Sigma_{\mathbf{b}}$ are scalars, and the dimensions of $\boldsymbol{\mu}_{\boldsymbol{\Phi}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\Phi}}$ are J-by-1 and J-by-J, respectively. Furthermore, for simplicity, assume $\mu_{\mathbf{b}} = 0$. Also, assume $\mathbf{x}_i$ and b_i are independent. Thus,

$$\begin{aligned} \mu_{\mathbf{y}} &= \mathbf{u}^{\mathrm{T}} \boldsymbol{\mu}_{\boldsymbol{\Phi}} + \mu_{\mathbf{b}} \\ &= \mathbf{u}^{\mathrm{T}} \boldsymbol{\mu}_{\boldsymbol{\Phi}} \end{aligned} \tag{32}$$

and

$$\begin{aligned} \Sigma_{\mathbf{y}} &= \mathbf{u}^{\mathrm{T}} \boldsymbol{\Sigma}_{\boldsymbol{\Phi}} \mathbf{u} + \Sigma_{\mathbf{b}} \\ &= \mathbf{u}^{\mathrm{T}} \boldsymbol{\Sigma}_{\boldsymbol{\Phi}} \mathbf{u} + \sigma_{\mathbf{b}}^2. \end{aligned} \tag{33}$$

Furthermore, $\mu_{\mathbf{y}}$ and $\Sigma_{\mathbf{y}}$ are scalars, and $\boldsymbol{\Sigma}_{\boldsymbol{\Phi}\mathbf{y}} = \boldsymbol{\Sigma}_{\boldsymbol{\Phi}} \times \mathbf{u}$. The sample mean and the sample covariance matrix, $\boldsymbol{\mu}_{\boldsymbol{\Phi}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\Phi}}$, are respectively

$$\boldsymbol{\mu}_{\boldsymbol{\Phi}} = \frac{1}{N} \sum_{i=1}^{N} \phi(\mathbf{x}_i)$$

and

$$\boldsymbol{\Sigma}_{\boldsymbol{\Phi}} = \frac{1}{N-1} \sum_{i=1}^{N} (\phi(\mathbf{x}_i) - \boldsymbol{\mu}_{\boldsymbol{\Phi}}) (\phi(\mathbf{x}_i) - \boldsymbol{\mu}_{\boldsymbol{\Phi}})^{\mathrm{T}}.$$

For homoscedasticity, this study assumes that the covariance between residues (i.e., b_i) are the same. The intrinsic covariance matrix and the empirical covariance matrix are respectively

$$\Sigma_{\mathbf{b}} = \mathrm{E}\left[(\mathbf{b} - \boldsymbol{\mu}_{\mathbf{b}}) (\mathbf{b} - \boldsymbol{\mu}_{\mathbf{b}})^{\mathrm{T}}\right] = \sigma_{\mathbf{b}}^2$$

and

$$\boldsymbol{\Psi}_{\mathbf{b}} = \begin{bmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_N^2 \end{bmatrix} = \begin{bmatrix} \sigma_{\mathbf{b}}^2 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_{\mathbf{b}}^2 \end{bmatrix} = \sigma_{\mathbf{b}}^2 \mathbf{I}.$$

Consequently,

$$\boldsymbol{\Sigma}_{\mathbf{b}^{\mathrm{T}}} = \boldsymbol{\Psi}_{\mathbf{b}} = \Sigma_{\mathbf{b}} \times \mathbf{I}.$$

This equation is subsequently used in (40).

4.1. Training stage

In Bayesian inference, prior information serves as a model or function parameters to interpret the likelihood probability of observed data $(\mathbf{x}_i, y_i)$. The training stage uses Bayesian inference to estimate the posterior distribution of $\mathbf{u}$, which consists of two parts. One is the likelihood probability "$P(\mathbf{y}^{\mathrm{T}}|\mathbf{u}, \boldsymbol{\Phi})$", and the other is the prior probability "$P(\mathbf{u})$".

It is worth noting that the dimensions of two independent variables should fit their joint probability, i.e., $P([\mathbf{u}\,\mathbf{y}^{\mathrm{T}}])$. Therefore, the transpose of $\mathbf{y}$ is used herein.

$$\begin{aligned} P(\mathbf{u}|\boldsymbol{\Phi}, \mathbf{y}) &= \frac{P\left(\mathbf{y}^{\mathrm{T}}|\mathbf{u}, \boldsymbol{\Phi}\right) P(\mathbf{u})}{P\left(\mathbf{y}^{\mathrm{T}}|\boldsymbol{\Phi}\right)} \\ &\propto P\left(\mathbf{y}^{\mathrm{T}}|\mathbf{u}, \boldsymbol{\Phi}\right) P(\mathbf{u}) \end{aligned} \tag{34}$$

where the marginal likelihood is

$$P\left(\mathbf{y}^{\mathrm{T}}|\boldsymbol{\Phi}\right) = \int P\left(\mathbf{y}^{\mathrm{T}}|\mathbf{u}, \boldsymbol{\Phi}\right) P(\mathbf{u})\, d\mathbf{u}. \tag{35}$$

The following steps establish the posterior distribution by computing the likelihood and prior probabilities.

- Computation of the Likelihood Probability

As $P(b) \sim \mathcal{N}(0, \sigma_{\mathbf{b}}^2)$ and $P(\phi(\mathbf{x}_i)) \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\Phi}}, \boldsymbol{\Sigma}_{\boldsymbol{\Phi}})$, the Gaussian distribution of the likelihood $P(\mathbf{y}|\mathbf{u}, \boldsymbol{\Phi})$ is $\mathcal{N}(\boldsymbol{\mu}_{\mathbf{y}|\mathbf{u},\boldsymbol{\Phi}}, \Sigma_{\mathbf{y}|\mathbf{u},\boldsymbol{\Phi}}) \sim \mathcal{N}(\mathbf{u}^{\mathrm{T}}\boldsymbol{\Phi}, \sigma_{\mathbf{b}}^2)$ based on the following conditional expectation and conditional covariance of a linear Gaussian system [26]. That is,

$$\begin{aligned} \Sigma_{\mathbf{y}|\boldsymbol{\Phi}} &= \Sigma_{\mathbf{y}} - \boldsymbol{\Sigma}_{\mathbf{y}\boldsymbol{\Phi}} \boldsymbol{\Sigma}_{\boldsymbol{\Phi}}^{-1} \boldsymbol{\Sigma}_{\boldsymbol{\Phi}\mathbf{y}} \\ &= \sigma_{\mathbf{b}}^2 \end{aligned} \tag{36}$$

and

$$\begin{aligned} \boldsymbol{\mu}_{\mathbf{y}|\boldsymbol{\Phi}} &= \boldsymbol{\mu}_{\mathbf{y}} + \boldsymbol{\Sigma}_{\mathbf{y}\boldsymbol{\Phi}} \boldsymbol{\Sigma}_{\boldsymbol{\Phi}}^{-1} (\boldsymbol{\Phi} - \boldsymbol{\mu}_{\boldsymbol{\Phi}} \mathbf{e}) \\ &= \mathbf{u}^{\mathrm{T}} \boldsymbol{\Phi} + \boldsymbol{\mu}_{\mathbf{b}} \\ &= \mathbf{u}^{\mathrm{T}} \boldsymbol{\Phi} \end{aligned} \tag{37}$$

where $\boldsymbol{\mu}_{\mathbf{y}|\boldsymbol{\Phi}} = \mu_{\mathbf{y}|\boldsymbol{\Phi}} \times \mathbf{e}$, $\boldsymbol{\mu}_{\mathbf{y}} = \mu_{\mathbf{y}} \times \mathbf{e}$, and $\boldsymbol{\mu}_{\mathbf{b}} = \mu_{\mathbf{b}} \times \mathbf{e}$. Besides, $\mu_{\mathbf{y}|\boldsymbol{\Phi}}$ and $\Sigma_{\mathbf{y}|\boldsymbol{\Phi}}$ are scalars, and $\boldsymbol{\mu}_{\mathbf{y}|\boldsymbol{\Phi}}$ is a 1-by-N vector. Let $\mathrm{tr}(\cdot)$ denote trace operations and $\det(\cdot)$ represent the determinant.

Accordingly,

$$
\begin{aligned}
&P\left(\mathbf{y}|\mathbf{u}, \mathbf{\Phi}\right)\\
&=\prod_{i=1}^{N} P\left(y_i|\mathbf{u}, \mathbf{\Phi}\right)\\
&=\frac{1}{\sqrt{2\pi\cdot\det\left(\Sigma_{\mathbf{y}|\mathbf{u},\mathbf{\Phi}}\right)}}\exp\left(-\frac{1}{2}\sum_{i}\left(y_i-\mu_{\mathbf{y}|\mathbf{u},\mathbf{\Phi}}\right)^{\mathrm{T}}\Sigma^{-1}_{\mathbf{y}|\mathbf{u},\mathbf{\Phi}}\left(y_i-\mu_{\mathbf{y}|\mathbf{u},\mathbf{\Phi}}\right)\right)\\
&=\frac{1}{\left(\sqrt{2\pi\cdot\det\left(\Sigma_{\mathbf{y}|\mathbf{u},\mathbf{\Phi}}\right)}\right)^{N}}\\
&\quad\exp\left(-\frac{1}{2}\mathrm{tr}\left(\left(\mathbf{y}-\mathbf{\mu}_{\mathbf{y}|\mathbf{u},\mathbf{\Phi}}\right)^{\mathrm{T}}\Sigma^{-1}_{\mathbf{y}|\mathbf{u},\mathbf{\Phi}}\left(\mathbf{y}-\mathbf{\mu}_{\mathbf{y}|\mathbf{u},\mathbf{\Phi}}\right)\right)\right)\\
&=\frac{1}{\left(\sqrt{2\pi\cdot\det\left(\sigma_{\mathbf{b}}^{2}\right)}\right)^{N}}\exp\left(-\frac{1}{2}\mathrm{tr}\left(\left(\mathbf{y}-\mathbf{u}^{\mathrm{T}}\mathbf{\Phi}\right)^{\mathrm{T}}\left(\sigma_{\mathbf{b}}^{-2}\right)\left(\mathbf{y}-\mathbf{u}^{\mathrm{T}}\mathbf{\Phi}\right)\right)\right)\\
&=\frac{1}{\left(\sqrt{2\pi\sigma_{\mathbf{b}}^{2}}\right)^{N}}\exp\left(-\frac{1}{2\sigma_{\mathbf{b}}^{2}}\mathrm{tr}\left(\left(\mathbf{y}-\mathbf{u}^{\mathrm{T}}\mathbf{\Phi}\right)^{\mathrm{T}}\left(\mathbf{y}-\mathbf{u}^{\mathrm{T}}\mathbf{\Phi}\right)\right)\right).
\end{aligned}
\tag{38}
$$

- Computation of the Prior Probability

Theoretically, the prior probability can be any distribution. However, such selection would result in posterior distributions without analytical solutions [27]. This benefits no computation. As the likelihood probability is a Gaussian distribution, we can use conjugate prior distributions to model the system for convenience of computation. When the generated posterior distribution and the selected prior distribution belong to the same class of distributions, such a prior distribution is a conjugate prior distribution [28]. There is a systematic analytical model for conjugate prior distributions [29].

To generate a Gaussian posterior distribution, this study selects a Gaussian prior distribution (39) for $P(\mathbf{u}) \sim \mathcal{N}(\mathbf{\mu}_{\mathbf{u}}, \mathbf{\Sigma}_{\mathbf{u}})$. The parameters, $\mathbf{\mu}_{\mathbf{u}}$ and $\mathbf{\Sigma}_{\mathbf{u}}$, can be set to $\mathbf{0}$ and $\sigma_{\mathbf{u}}^{2}\mathbf{I}$, respectively, for simplicity. The dimensions of them are, respectively, J-by-1 and J-by-J.

$$
P\left(\mathbf{u}\right)=\frac{1}{\sqrt{(2\pi)^{J}\cdot\det\left(\mathbf{\Sigma}_{\mathbf{u}}\right)}}\exp\left(-\frac{1}{2}\mathrm{tr}\left[\left(\mathbf{u}-\mathbf{\mu}_{\mathbf{u}}\right)^{\mathrm{T}}\mathbf{\Sigma}_{\mathbf{u}}^{-1}\left(\mathbf{u}-\mathbf{\mu}_{\mathbf{u}}\right)\right]\right). \tag{39}
$$

- Computation of the Posterior Probability

It is worth noting that the likelihood $P(\mathbf{y}|\mathbf{u}, \mathbf{\Phi})$ is $\mathcal{N}(\mathbf{\mu}_{\mathbf{y}|\mathbf{u},\mathbf{\Phi}} = \mathbf{u}^{\mathrm{T}}\mathbf{\Phi}, \mathbf{\Sigma}_{\mathbf{y}|\mathbf{u},\mathbf{\Phi}} = \sigma_{\mathbf{b}}^{2})$, and the prior probability $P(\mathbf{u})$ is $\mathcal{N}(\mathbf{\mu}_{\mathbf{u}}, \mathbf{\Sigma}_{\mathbf{u}})$. According to Gaussian identities in [20,28–30], the posterior distribution is Gaussian.

To fit the joint probability of $\mathbf{y}$ and $\mathbf{u}$, plugging $P\left(\mathbf{y}^{\mathrm{T}}|\mathbf{u}, \mathbf{\Phi}\right) \sim \mathcal{N}\left(\mathbf{y}^{\mathrm{T}}|\mathbf{\mu}_{\mathbf{y}^{\mathrm{T}}|\mathbf{u},\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{y}^{\mathrm{T}}|\mathbf{u},\mathbf{\Phi}}\right) = \mathcal{N}\left(\mathbf{y}^{\mathrm{T}}|\mathbf{\mu}_{\mathbf{y}^{\mathrm{T}}|\mathbf{u},\mathbf{\Phi}}, \mathbf{\Psi}_{\mathbf{y}|\mathbf{u},\mathbf{\Phi}}\right)$ into (34) yields

$$
\begin{aligned}
P\left(\mathbf{u}|\mathbf{\Phi}, \mathbf{y}\right) &\propto P\left(\mathbf{y}^{\mathrm{T}}|\mathbf{u}, \mathbf{\Phi}\right)P\left(\mathbf{u}\right)\\
&=\mathcal{N}\left(\mathbf{y}^{\mathrm{T}}|\mathbf{\mu}_{\mathbf{y}^{\mathrm{T}}|\mathbf{u},\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{y}^{\mathrm{T}}|\mathbf{u},\mathbf{\Phi}}\right)\mathcal{N}\left(\mathbf{u}|\mathbf{\mu}_{\mathbf{u}}, \mathbf{\Sigma}_{\mathbf{u}}\right)\\
&=\mathcal{N}\left(\mathbf{y}^{\mathrm{T}}|\mathbf{\Phi}^{\mathrm{T}}\mathbf{u}, \sigma_{\mathbf{b}}^{2}\mathbf{I}\right)\mathcal{N}\left(\mathbf{u}|\mathbf{\mu}_{\mathbf{u}}, \mathbf{\Sigma}_{\mathbf{u}}\right)\\
&\propto\mathcal{N}\left(\mathbf{u}|\mathbf{\mu}_{\mathbf{u}|\mathbf{y}^{\mathrm{T}},\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{u}|\mathbf{y}^{\mathrm{T}},\mathbf{\Phi}}\right)
\end{aligned}
\tag{40}
$$

where

$$
\begin{aligned}
\mathbf{\Sigma}_{\mathbf{u}|\mathbf{y}^{\mathrm{T}},\mathbf{\Phi}} &= \mathbf{\Sigma}_{\mathbf{u}}-\mathbf{\Sigma}_{\mathbf{u}}\mathbf{\Phi}\left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Sigma}_{\mathbf{u}}\mathbf{\Phi}+\mathbf{\Sigma}_{\mathbf{b}^{\mathrm{T}}}\right)^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Sigma}_{\mathbf{u}}\\
&=\left(\mathbf{\Sigma}_{\mathbf{u}}^{-1}+\sigma_{\mathbf{b}}^{-2}\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}\right)^{-1}
\end{aligned}
\tag{41}
$$

and

$$
\begin{aligned}
\mathbf{\mu}_{\mathbf{u}|\mathbf{y}^{\mathrm{T}},\mathbf{\Phi}} &= \mathbf{\mu}_{\mathbf{u}}+\mathbf{\Sigma}_{\mathbf{u}}\mathbf{\Phi}\left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Sigma}_{\mathbf{u}}\mathbf{\Phi}+\mathbf{\Sigma}_{\mathbf{b}^{\mathrm{T}}}\right)^{-1}\left(\mathbf{y}^{\mathrm{T}}-\mathbf{\mu}_{\mathbf{y}}^{\mathrm{T}}\right)\\
&=\mathbf{\mu}_{\mathbf{u}}+\left(\mathbf{\Sigma}_{\mathbf{u}}^{-1}+\mathbf{\Phi}\mathbf{\Sigma}_{\mathbf{b}^{\mathrm{T}}}^{-1}\mathbf{\Phi}^{\mathrm{T}}\right)^{-1}\mathbf{\Phi}\mathbf{\Sigma}_{\mathbf{b}^{\mathrm{T}}}^{-1}\left(\mathbf{y}^{\mathrm{T}}-\mathbf{\mu}_{\mathbf{y}}^{\mathrm{T}}\right)\\
&=\mathbf{\mu}_{\mathbf{u}}+\mathbf{\Sigma}_{\mathbf{u}|\mathbf{y}^{\mathrm{T}}}\mathbf{\Phi}\mathbf{\Sigma}_{\mathbf{b}^{\mathrm{T}}}^{-1}\left(\mathbf{y}^{\mathrm{T}}-\mathbf{\mu}_{\mathbf{y}}^{\mathrm{T}}\right)\\
&=\mathbf{\Sigma}_{\mathbf{u}|\mathbf{y}^{\mathrm{T}}}\left(\mathbf{\Sigma}_{\mathbf{u}|\mathbf{y}^{\mathrm{T}}}^{-1}\mathbf{\mu}_{\mathbf{u}}+\mathbf{\Phi}\mathbf{\Sigma}_{\mathbf{b}^{\mathrm{T}}}^{-1}\left(\mathbf{y}^{\mathrm{T}}-\mathbf{\mu}_{\mathbf{y}}^{\mathrm{T}}\right)\right)\\
&=\mathbf{\Sigma}_{\mathbf{u}|\mathbf{y}^{\mathrm{T}},\mathbf{\Phi}}\left(\mathbf{\Sigma}_{\mathbf{u}}^{-1}\mathbf{\mu}_{\mathbf{u}}+\sigma_{\mathbf{b}}^{-2}\mathbf{\Phi}\left(\mathbf{y}-\mathbf{b}\right)^{\mathrm{T}}\right).
\end{aligned}
\tag{42}
$$

Moreover, $\mathbf{\mu}_{\mathbf{u}|\mathbf{y}^{\mathrm{T}},\mathbf{\Phi}}$ is a J-by-1 vector, and $\mathbf{\Sigma}_{\mathbf{u}|\mathbf{y}^{\mathrm{T}},\mathbf{\Phi}}$ is a J-by-J matrix. Assume that the prior information changes with time. Subsequently,

$$
\begin{aligned}
&\mathbf{\Sigma}_{\mathbf{u}|\mathbf{y}^{\mathrm{T}},\mathbf{\Phi}}\left[\ell+1\right]\\
&=\left(\mathbf{\Sigma}^{-1}_{\mathbf{u}|\mathbf{y}^{\mathrm{T}},\mathbf{\Phi}}\left[\ell\right]+\sigma_{\mathbf{b}}^{-2}\left(\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}\right)\left[\ell\right]\right)^{-1}\\
&=\mathbf{\Sigma}_{\mathbf{u}|\mathbf{y}^{\mathrm{T}},\mathbf{\Phi}}\left[\ell\right]-\sigma_{\mathbf{b}}^{-2}\mathbf{\Sigma}_{\mathbf{u}|\mathbf{y}^{\mathrm{T}},\mathbf{\Phi}}\left[\ell\right]\\
&\quad\times\left(\mathbf{I}+\sigma_{\mathbf{b}}^{-2}\left(\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}\right)\left[\ell\right]\mathbf{\Sigma}_{\mathbf{u}|\mathbf{y}^{\mathrm{T}},\mathbf{\Phi}}\left[\ell\right]\right)^{-1}\left(\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}\right)\left[\ell\right]\mathbf{\Sigma}_{\mathbf{u}|\mathbf{y}^{\mathrm{T}},\mathbf{\Phi}}\left[\ell\right]
\end{aligned}
\tag{43}
$$

where

$$
\left(\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}\right)\left[\ell\right]=\left(\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}\right)\left[\ell-1\right]+\mathbf{\Phi}_{\mathcal{H}}\mathbf{\Phi}'_{\mathcal{H}}
$$

and

$$
\begin{aligned}
&\mathbf{\mu}_{\mathbf{u}|\mathbf{y}^{\mathrm{T}},\mathbf{\Phi}}\left[\ell+1\right]\\
&=\mathbf{\Sigma}_{\mathbf{u}|\mathbf{y}^{\mathrm{T}},\mathbf{\Phi}}\left[\ell+1\right]\left(\mathbf{\Sigma}^{-1}_{\mathbf{u}|\mathbf{y}^{\mathrm{T}},\mathbf{\Phi}}\left[\ell\right]\mathbf{\mu}_{\mathbf{u}|\mathbf{y}^{\mathrm{T}},\mathbf{\Phi}}\left[\ell\right]\right.\\
&\left.+\sigma_{\mathbf{b}}^{-2}\mathbf{\Phi}\left[\ell\right]\left(\mathbf{y}^{\mathrm{T}}\left[\ell\right]-\mathbf{b}^{\mathrm{T}}\left[\ell\right]\right)\right).
\end{aligned}
\tag{44}
$$

When existing training samples change, $\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}$ and $\mathbf{\Phi}\mathbf{y}^{\mathrm{T}}$ in (43) and (44), respectively, should be accordingly updated. Otherwise, only prior information is updated. The posterior probability reflects the modification in training samples and prior information.

4.2. Predictive stage

As the training stage already generates the posterior distribution and the uncertainty of $\mathbf{u}$, the posterior predictive distribution "$P((y^{*})^{\mathrm{T}}|\phi(\mathbf{x}^{*}), \mathbf{\Phi}, \mathbf{y})$" is then used to model the uncertainty of the predictive output. The posterior predictive distribution can be rewritten as the marginal distribution of "$P((y^{*})^{\mathrm{T}}|\phi(\mathbf{x}^{*}), \mathbf{u})$" and "$P(\mathbf{u}|\mathbf{\Phi}, \mathbf{y})$". Let y^{*} represent the scalar predictive output of the model when an M-by-1 testing sample $\mathbf{x}^{*}$ is input. Applying the product rule and the integral rule of Gaussian identities [30] to the marginal distribution yields the following form.

$$
\begin{aligned}
&P\left(\left(y^{*}\right)^{\mathrm{T}}|\phi\left(\mathbf{x}^{*}\right), \mathbf{\Phi}, \mathbf{y}\right)\\
&=\int P\left(\left(y^{*}\right)^{\mathrm{T}}|\phi\left(\mathbf{x}^{*}\right), \mathbf{u}\right)P\left(\mathbf{u}|\mathbf{\Phi}, \mathbf{y}\right)d\mathbf{u}\\
&=\int\mathcal{N}\left(\left(y^{*}\right)^{\mathrm{T}}|\phi\left(\mathbf{x}^{*}\right)^{\mathrm{T}}\mathbf{u}, \sigma_{\mathbf{b}}^{2}\right)\mathcal{N}\left(\mathbf{u}|\mathbf{\mu}_{\mathbf{u}|\mathbf{y},\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{u}|\mathbf{y},\mathbf{\Phi}}\right)d\mathbf{u}\\
&\propto\mathcal{N}\left(\left(y^{*}\right)^{\mathrm{T}}|\phi\left(\mathbf{x}^{*}\right)^{\mathrm{T}}\mathbf{\mu}_{\mathbf{u}|\mathbf{y},\mathbf{\Phi}}, \sigma_{\mathbf{b}}^{2}+\phi\left(\mathbf{x}^{*}\right)^{\mathrm{T}}\mathbf{\Sigma}_{\mathbf{u}|\mathbf{y},\mathbf{\Phi}}\phi\left(\mathbf{x}^{*}\right)\right)\\
&=\mathcal{N}\left(\left(y^{*}\right)^{\mathrm{T}}|\mu^{*}, \Psi^{*}\right).
\end{aligned}
\tag{45}
$$

Notably, although y^{*} is a scalar, (45) still uses the transpose of y^{*} for clarity. If $P((y^{*})^{\mathrm{T}}|\phi(\mathbf{x}^{*}), \mathbf{u})$ follows the distribution of the training data $P(\mathbf{y}^{\mathrm{T}}|\mathbf{u}, \mathbf{\Phi})$ due to the need for analytical solutions, the predictive distribution becomes Gaussian. Accordingly,

$$
\begin{aligned}
&P\left(\left(y^{*}\right)^{\mathrm{T}}|\phi\left(\mathbf{x}^{*}\right), \mathbf{\Phi}, \mathbf{y}\right)\\
&=\frac{1}{\sqrt{2\pi\cdot\det\left(\Psi^{*}\right)}}\\
&\quad\exp\left(-\frac{1}{2}\mathrm{tr}\left(\left(\left(y^{*}\right)^{\mathrm{T}}-\mu^{*}\right)^{\mathrm{T}}\left(\Psi^{*}\right)^{-1}\left(\left(y^{*}\right)^{\mathrm{T}}-\mu^{*}\right)\right)\right)
\end{aligned}
\tag{46}
$$

where

$$
\Psi^{*}=\sigma_{\mathbf{b}}^{2}+\phi\left(\mathbf{x}^{*}\right)^{\mathrm{T}}\mathbf{\Sigma}_{\mathbf{u}|\mathbf{y},\mathbf{\Phi}}\phi\left(\mathbf{x}^{*}\right) \tag{47}
$$

and

$$
\mu^{*}=\phi\left(\mathbf{x}^{*}\right)^{\mathrm{T}}\mathbf{\mu}_{\mathbf{u}|\mathbf{y},\mathbf{\Phi}}. \tag{48}
$$

Moreover, μ^{*} and Ψ^{*} are scalars. When existing training samples change, $\mathbf{\Sigma}_{\mathbf{u}|\mathbf{y},\mathbf{\Phi}}$ in (47) and $\mathbf{\mu}_{\mathbf{u}|\mathbf{y},\mathbf{\Phi}}$ in (48) need updates, respectively. Subsequently, a new posterior predictive distribution is generated.

When variable prior information is involved in the update, (47) and (48) become

$$
\Psi^{*}=\sigma_{\mathbf{b}}^{2}+\phi\left(\mathbf{x}^{*}\right)^{\mathrm{T}}\mathbf{\Sigma}_{\mathbf{u}|\mathbf{y},\mathbf{\Phi}}\left[\ell\right]\phi\left(\mathbf{x}^{*}\right) \tag{49}
$$

and

$$
\mu^{*}=\phi\left(\mathbf{x}^{*}\right)^{\mathrm{T}}\mathbf{\mu}_{\mathbf{u}|\mathbf{y},\mathbf{\Phi}}\left[\ell\right]. \tag{50}
$$

5. Experimental results

Experiments on open datasets were carried out for evaluating the performance. The information of these datasets is listed in Table 1. The first column shows the name. The rest columns specify the number of classes, samples, and dimensions, respectively. Dataset "MIT/BIH ECG" is available at PhysioNet (www.physionet.org), and "Dorothea (DRT)" was downloaded from the UC Irvine (UCI) Machine Learning Repository (archive.ics.uci.edu/ml/). The datasets show two typical data, where both $N > M$ and $M > N$ are presented, respectively.

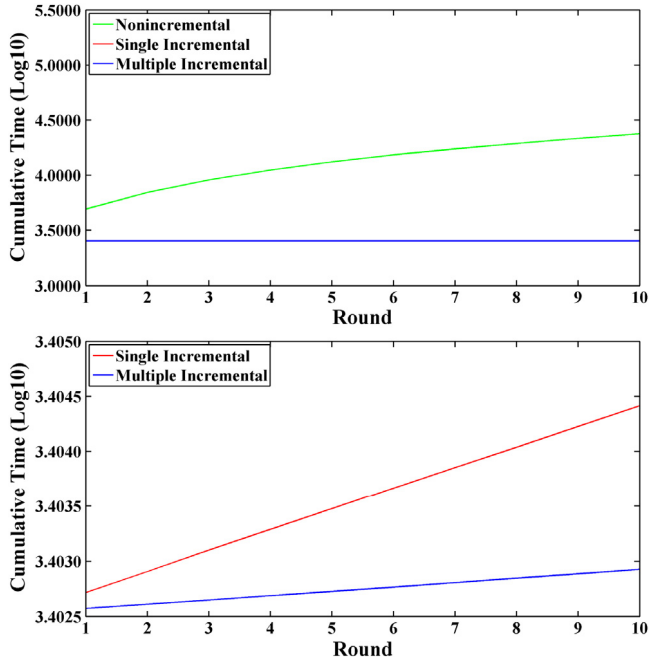

Fig. 2. KRR comparison between multiple incremental (blue), single incremental (red) and nonincremental (green) learning with the use of the ECG dataset and the poly2 kernel. The computational time (log10) was cumulative. The accuracy rates were all 94.71%. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

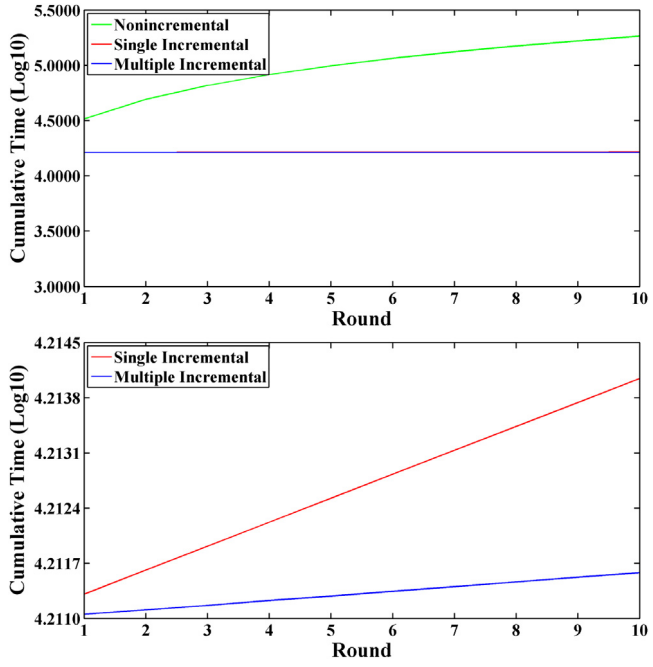

Fig. 3. KRR comparison between multiple incremental (blue), single incremental (red) and nonincremental (green) learning with the use of the ECG dataset and the poly3 kernel. The computational time (log10) was cumulative. The accuracy rates were all 97.37%. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
Attributes of the datasets.

Name	#Classes	#Samples	#Dimensions
ECG	2	104033	21
DRT	2	800	1000000

Table 2
Settings of incremental/decremental computation.

Name	Basic training size	Multiple incremental/decremental size
ECG	83226	+4/−2
DRT	640	+4/−2

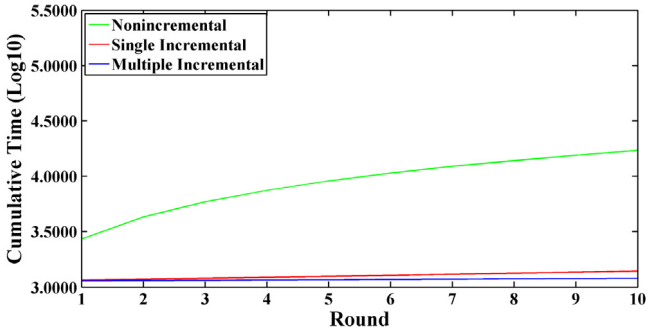

Fig. 4. KRR comparison between multiple incremental (blue), single incremental (red) and nonincremental (green) learning with the use of the DRT dataset and the poly2 kernel. The computational time (log10) was cumulative. The accuracy rates were all 90.00%. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The experiment used approximately 80.00% of the data for training and 20.00% of the data for testing. Furthermore, +4 and −2 samples were randomly selected for incremental and decremental computation at the same time. Tables 2 and 9 summarize the incremental and decremental settings. The algorithmic details are listed in Table 9. Ridges were empirically set in the experiments. Regarding KBR settings, $\boldsymbol{\mu}_{\mathbf{u}}$ and $\mu_{\mathbf{b}}$ were $\mathbf{0}$ and 0, respectively. Besides, both $\sigma_{\mathbf{u}}^2$ and $\sigma_{\mathbf{b}}^2$ were set to 0.01.

Two baselines, "nonincremental analysis" and "single incremental algorithm", along with the proposed method "multiple incremental approach" were used for comparison. In total, ten rounds of data operations (i.e., data insertion and deletion) were evaluated, and computational time in log10 was calculated. For the nonincremental part, it recomputed the weight of the system based on the new dataset after one round of data operations. Regarding the single incremental part, it reanalyzed the new dataset every time when data insertion or deletion occurred. Besides, only when one round of data operations was complete, cumulative computational time was measured.

Figures 2–6 display the incremental/nonincremental results, where the horizontal axis is the round, and the vertical axis represents the cumulative computational time in log10. The unit was seconds. The green curve represents the nonincremental analysis, and the red curve indicates the single incremental method. The proposed approach is shown by the blue curve. For the computational time of a single round, Tables 3–7 display the details of single rounds, and average computational time is summarized in Table 8. Examining the result in Table 8 reveals that the proposed mechanism could improve the efficiency in intrinsic space by more than 3.71 times and the performance in empirical space by more than 2.56 times, compared with the single incremental algorithm.

As for KBR, the same dataset along with the same settings was used for evaluation. The details are listed at the beginning of this section. Figures 7–8 show the cumulative computational time based on the proposed

Table 3
KRR computational time (log10) based on the ECG dataset and the poly2 kernel in a single round.

#Samples	83226	83228	83230	83232	83234	83236	83238	83240	83242	83244
Multiple	−0.537544	−0.665259	−0.659984	−0.635436	−0.651824	−0.645394	−0.634669	−0.622588	−0.643913	−0.623469
Single	0.047783	0.043765	0.050801	0.038683	0.040046	0.041661	0.039198	0.036630	0.042320	0.041475
None	3.376356	3.314288	3.316463	3.317598	3.315914	3.317286	3.317168	3.317430	3.326118	3.331818

Table 4
KRR computational time (log10) based on the ECG dataset and the poly3 kernel in a single round.

#Samples	83226	83228	83230	83232	83234	83236	83238	83240	83242	83244
Multiple	4.211003	0.297297	0.314129	0.314672	0.364845	0.361343	0.334149	0.354265	0.340383	0.337915
Single	4.224946	1.058435	1.056797	1.056978	1.056486	1.059564	1.058412	1.055703	1.057832	1.061875
None	4.211003	4.214649	4.214517	4.219702	4.219394	4.224266	4.226119	4.230048	4.226973	4.241862

Table 5
KRR computational time (log10) based on the DRT dataset and the poly2 kernel in a single round.

#Samples	640	642	644	646	648	650	652	654	656	658
Multiple	3.053674	0.846649	0.720064	0.850986	0.845865	0.853454	0.851205	0.855350	0.856517	0.797779
Single	3.051355	1.373776	1.351769	1.373161	1.400000	1.426793	1.422169	1.445650	1.453737	1.452745
None	3.053674	3.196123	3.196231	3.201359	3.201729	3.206425	3.211160	3.178982	3.217578	3.217389

Table 6
KRR computational time (log10) based on the DRT dataset and the poly3 kernel in a single round.

#Samples	640	642	644	646	648	650	652	654	656	658
Multiple	0.853478	0.718077	0.856490	0.878286	0.862960	0.851657	0.903701	0.898904	0.901134	0.841226
Single	1.373330	1.348596	1.371429	1.393420	1.406572	1.424955	1.421473	1.444416	1.459492	1.454026
None	3.194155	3.198641	3.214183	3.208231	3.209122	3.213412	3.247027	3.212538	3.250765	3.228786

Table 7
KRR computational time (log10) based on the DRT dataset and the RBF in a single round.

#Samples	640	642	644	646	648	650	652	654	656	658
Multiple	0.888406	0.776181	0.852696	0.851705	0.848764	0.853636	0.852904	0.854650	0.858440	0.801611
Single	1.419054	1.394077	1.419183	1.439993	1.450303	1.468781	1.466268	1.478293	1.485936	1.487907
None	3.225958	3.218848	3.201681	3.206244	3.208604	3.207531	3.210940	3.179368	3.217160	3.218175

Table 8
KRR average computational time in a single round.

	Multiple	Single	None	Improvement (Fold)
ECG–Poly2	0.234105	1.102187	2115.546985	3.71
ECG–Poly3	2.160822	11.429962	16743.767084	4.29
DRT–Poly2	6.838521	25.827835	1597.192878	2.78
DRT–Poly3	7.234008	25.777343	1652.188852	2.56
DRT–RBF	6.997008	28.316223	1620.388448	3.05

RBFs are inapplicable to intrinsic space due to infinite dimensions. Improvement is computed based on comparison between multiple and single incremental analyses.

Table 9
Algorithmic settings.

Name	Kernel	Ridge
Intrinsic-space KRR	Poly2 & Poly3	0.5
Empirical-space KRR	Poly2, Poly3, & RBF	0.5

RBFs are inapplicable to intrinsic space due to infinite dimensions. The radius of RBFs is 50.00.

method and the single incremental algorithm. The detailed time is listed in Tables 10–11. The average computational time is listed in Table 12.

6. Conclusion

This work presents an efficient incremental/decremental mechanism for updating the weight vector of KRR functions. The proposed mechanism combines data insertion and deletion together in the same equation, such that operations on data modifications are performed in the same round. This mechanism is conducive to improvement of computational loads, and it becomes more efficient than typical single-instance incremental analysis. Moreover, this work also presents intrinsic-space and empirical-space updates. The former is suitable for the case with $N > M$, whereas the latter fits the case when $N < M$. This study also suggests an appropriate batch size during multiple incremental/decremental analyses in intrinsic and empirical space. For intrinsic space, the mathematical model shows that the size of each batch should be smaller than the feature dimensional size after kernel mapping. Furthermore, in empirical space when decremental computation is performed, the size of the residual data should be larger than that of samples that are about to be removed. Otherwise, both situations save no computation. Finally, this study employed the developed incremental and decremental mechanism for KBR to speed up uncertainty calculation.

Open benchmark datasets, consisting of two typical datasets where $N>M$ and $M>N$, were used to evaluate the computational performance. Compared with the single incremental algorithm, the computational speed of the proposed method for KRR was enhanced by more than 3.71 times in

Table 10
KBR computational time (log10) based on the ECG dataset and the poly2 kernel in a single round.

#Samples	83 226	83 228	83 230	83 232	83 234	83 236	83 238	83 240	83 242	83 244
Multiple	−0.433992	−0.432390	−0.386889	−0.407412	−0.425207	−0.416759	−0.411944	−0.435774	−0.419782	−0.430701
Single	0.316193	0.308310	0.304919	0.306496	0.311477	0.316939	0.309808	0.304631	0.309345	0.308108

Table 11
KBR computational time (log10) based on the ECG dataset and the poly3 kernel in a single round.

#Samples	83 226	83 228	83 230	83 232	83 234	83 236	83 238	83 240	83 242	83 244
Multiple	0.647582	0.670926	0.664598	0.687889	0.675405	0.656299	0.670020	0.650114	0.637448	0.647175
Single	1.385879	1.390621	1.395218	1.395784	1.387192	1.395385	1.390216	1.392410	1.401707	1.388583

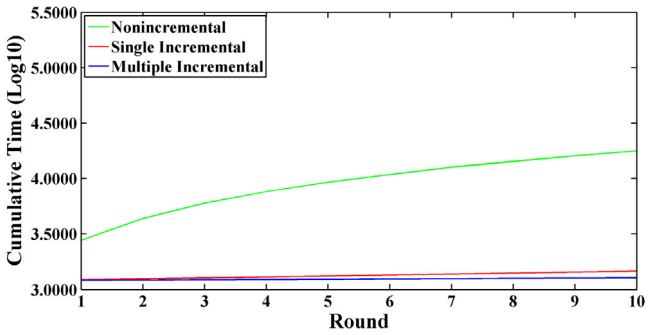

Fig. 5. KRR comparison between multiple incremental (blue), single incremental (red) and nonincremental (green) learning with the use of the DRT dataset and the poly3 kernel. The computational time (log10) was cumulative. The accuracy rates were all 90.00%. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

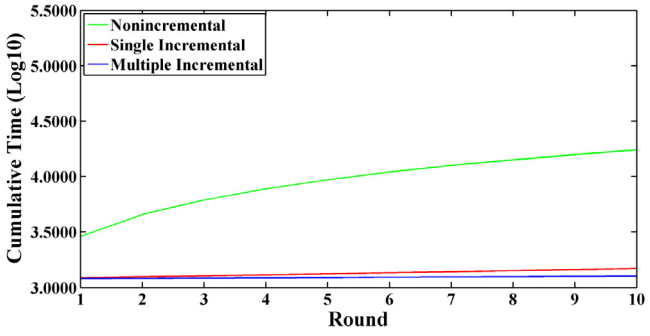

Fig. 6. KRR comparison between multiple incremental (blue), single incremental (red) and nonincremental (green) learning with the use of the DRT dataset and the RBF. The computational time (log10) was cumulative. The accuracy rates were all 90.00%. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

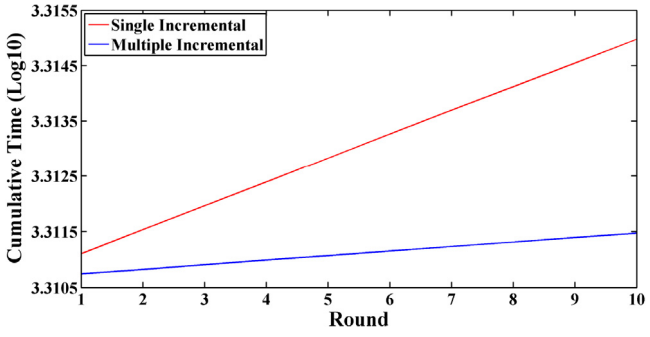

Fig. 7. KBR comparison between multiple incremental (blue) and single incremental (red) learning with the use of the ECG dataset and the poly2 kernel. The computational time (log10) was cumulative. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

intrinsic space and by more than 2.56 times in empirical space. For KBR, computational speed was 3.38-fold faster than the single incremental one on average. Such findings have established the effectiveness of the multiple incremental/decremental analyses.

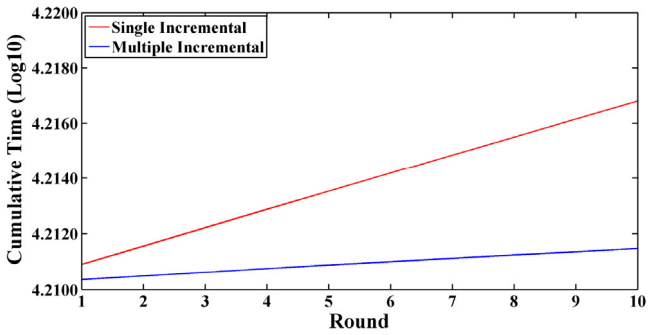

Fig. 8. KBR comparison between multiple incremental (blue) and single incremental (red) learning with the use of the ECG dataset and the poly3 kernel. The computational time (log10) was cumulative. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 12
KBR average computational time in a single round.

	Multiple	Single	Improvement (Fold)
ECG–poly2	0.380326	2.040052	4.36
ECG–poly3	4.581399	24.678768	4.39

RBFs are inapplicable to intrinsic space due to infinite dimensions.

Acknowledgments

This work was supported by 2016 Strategic Large Grants of Monash University with Medtronic Inc.

References

[1] S.-Y. Kung, Kernel Methods and Machine Learning, Cambridge University Press, Cambridge, United Kingdom, 2014.
[2] K. Ericson, S. Pallickara, On the performance of high dimensional data clustering and classification algorithms, Future Gener. Comput. Syst. 29 (4) (2013) 1024–1034.
[3] S.-Y. Kung, P.-Y. Wu, On efficient learning and classification kernel methods, in: Proc. 2012 IEEE International Conference on Acoustics, Speech, and Signal Processing, Kyoto, Japan, Mar. 25–30, 2012, pp. 2065–2068.
[4] S.G. Djorgovski, M.J. Graham, C. Donalek, A.A. Mahabal, A.J. Drake, M. Turmon, T. Fuchs, Real-time data mining of massive data streams from synoptic sky surveys, Future Gener. Comput. Syst. 59 (2016) 95–104.
[5] D. Takaishi, H. Nishiyama, N. Kato, R. Miura, Toward energy efficient big data gathering in densely distributed sensor networks, IEEE Trans. Emerging Top. Comput. 2 (3) (2014) 388–397.
[6] C. Zhang, R.C. Qiu, Massive MIMO as a big data system: Random matrix models and testbed, IEEE Access 3 (2015) 837–851.
[7] X. Luo, D. Zhang, L.T. Yang, J. Liu, X. Chang, H. Ning, A kernel machine-based secure data sensing and fusion scheme in wireless sensor networks for the cyber-physical systems, Future Gener. Comput. Syst. 61 (2016) 85–96.
[8] M. Smit, B. Simmons, M. Litoiu, Distributed, application-level monitoring for heterogeneous clouds using stream processing, Future Gener. Comput. Syst. 29 (8) (2013) 2103–2114.
[9] J. Liu, Y. Liang, N. Ansari, Spark-based large-scale matrix inversion for big data processing, IEEE Access 4 (2016) 2166–2176.
[10] D. Lee, J.-S. Kim, S. Maeng, Large-scale incremental processing with MapReduce, Future Gener. Comput. Syst. 36 (2014) 66–79.
[11] L. Kuang, F. Hao, L.T. Yang, M. Lin, C. Luo, G. Min, A tensor-based approach for big data representation and dimensionality reduction, IEEE Trans. Emerging Top. Comput. 2 (3) (2014) 280–291.
[12] G. Cauwenberghs, T. Poggio, Incremental and decremental support vector machine learning, in: Proc. 14th Annual Conference on Neural Information Processing System, Denver, Colorado, United States, 2000, pp. 409–415, Nov. 28–30.
[13] J.C. Platt, Fast training of support vector machines using sequential minimal optimization, in: B. Schölkopf, C.J.C. Burges, A.J. Smola (Eds.), Advances in Kernel Methods: Support Vector Learning, MIT Press, Cambridge, Massachusetts, United States, 1999.
[14] C.P. Diehl, G. Cauwenberghs, SVM incremental learning, adaptation and optimization, in: Proc. International Joint Conference on Neural Networks, Portland, Oregon, United States, Jul. 20–24, 2003, pp. 2685–2690.
[15] P. Laskov, C. Gehl, S. Krüger, K.-R. Müller, Incremental support vector learning: Analysis, implementation and applications, J. Mach. Learn. Res. 7 (2006) 1909–1936.

[16] M. Karasuyama, I. Takeuchi, Multiple incremental decremental learning of support vector machines, IEEE Trans. Neural Netw. 21 (7) (2010) 1048–1059.
[17] Y. Engel, S. Mannor, R. Meir, The kernel recursive least-squares algorithm, IEEE Trans. Signal Process. 52 (8) (2004) 2275–2285.
[18] S.V. Vaerenbergh, M. Lázaro-Gredilla, I. Santamaría, Kernel recursive least-squares tracker for time-varying regression, IEEE Trans. Neural Netw. Learn. Syst. 23 (8) (2012) 1313–1326.
[19] C.M. Bishop, Pattern Recognition and Machine Learning, second ed., Springer-Verlag, New York City, New York, United States, 2007.
[20] C.E. Rasmussen, C.K.I. Williams, Gaussian Processes for Machine Learning, MIT Press, Cambridge, Massachusetts, United States, 2006.
[21] J.Q. Candela, O. Winther, Incremental gaussian processes, in: Proc. 16th Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, Dec. 09–14, 2002, pp. 1001–1008.
[22] K. Markov, T. Matsui, Music genre and emotion recognition using gaussian processes, IEEE Access 2 (2014) 688–697.
[23] E. Levina, P.J. Bickel, Maximum likelihood estimation of intrinsic dimension, in: Proc. 18th Annual Conference on Neural Information Processing System, Vancouver, British Columbia, Canada, Dec. 13–18, 2004.
[24] A.N. Akansu, M.U. Torun, Toeplitz approximation to empirical correlation matrix of asset returns: A signal processing perspective, IEEE J. Sel. Top. Sign. Proces. 6 (4) (2012) 319–326.
[25] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes: The Art of Scientific Computing, third ed., Cambridge University Press, Cambridge, United Kingdom, 2007.
[26] K.P. Murphy, Machine Learning: A Probabilistic Perspective, MIT Press, Cambridge, Massachusetts, United States, 2012.
[27] P.M. Lee, Bayesian Statistics: An Introduction, fourth ed., Wiley, West Sussex, United Kingdom, 2012.
[28] A. Gelman, J.B. Carlin, H.S. Stern, D.B. Dunson, A. Vehtari, D.B. Rubin, Bayesian Data Analysis, third ed., CRC Press, Boca Raton, Florida, United States, 2013.
[29] H. Raiffa, R. Schlaifer, Applied Statistical Decision Theory, Wiley, Hoboken, New Jersey, United States, 2000.
[30] M. Toussaint, Introduction to Machine Learning, Department of Computer Science, University of Stuttgart, Stuttgart, Germany, 2016 Apr. 19.

Bo-Wei Chen is with the School of Information Technology, Monash University, Australia. During 2014–2015, he worked as a postdoctoral researcher with Princeton University, USA. His research interests include data analytics, machine learning, and audiovisual sensor networks. He serves as the Chair of the Signal Processing Chapter, IEEE Harbin Section (email: dennisbwc@gmail.com).

Nik Nailah Binti Abdullah is with the School of Information Technology, Monash University. She received her DEA (equivalently French Master's Degree) and Doctor of Philosophy degree with Distinction in Informatics from Université Montpellier II Sciences et Techniques du Languedoc in 2002 and 2006, respectively. She was the Technovisionaire speaker for the opening of the Year of Creativity Milano officiated by President Filippo Penati, in 2009. In 2008, she was invited as a Google Tech Talk speaker at Google Mountain View, California in 2008 on her research project. Dr. Nik Nailah Binti Abdullah was also nominated as one of the three finalists for "the Technovisionarie - BlackBerry Women and Technology Award" associated by IFIP in 2008. Dr. Nik Nailah Binti Abdullah has spent 11 years abroad for her graduate studies and professional work in the LIRMM (Laboratory of Informatics, Robotics, Microelectronic), Montpellier, France and at the National Institute Informatics, Tokyo Japan in a cross-disciplinary setting of human cognition and communication, and multiagent system projects. Her current effort is focused on modeling user requirements (e.g., clinicians and patients' needs in healthcare technology) based on the analysis of human activities using communication and cognition theories, such as activity theory and situated cognition, to help build smarter technologies in healthcare.

Sangoh Park is with the School of Computer Engineering, Chungang University, South Korea.

Y. Gu is with Beijing University of Science and Technology, China, and he is also a visiting scholar with Monash University.