IEICE TRANSACTIONS

on Fundamentals of Electronics, Communications and Computer Sciences

VOL. E102-A NO. 9
SEPTEMBER 2019

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

LETTER

A Fast Cross-Validation Algorithm for Kernel Ridge Regression by Eigenvalue Decomposition

Akira TANAKA[†a)] ***and*** **Hideyuki IMAI**[†,††]**,** ***Members***

SUMMARY A fast cross-validation algorithm for model selection in kernel ridge regression problems is proposed, which is aiming to further reduce the computational cost of the algorithm proposed by An et al. by eigenvalue decomposition of a Gram matrix.

key words: *kernel ridge regression, model selection, hyperparameter, cross-validation*

1. Introduction

A kernel ridge regressor (KRR) [1] is still one of useful function estimators in the field of machine learning. In order to obtain a good performance in the KRR, selection of hyperparameters, such as a kernel parameter and a regularization parameter, is crucial. The cross-validation (CV) technique [2] is widely used for the selection of these hyperparameters. In general, a naive algorithm of the CV requires large computational cost. In [3], An et al., proposed a fast CV algorithm for the least squares support vector machine (LS-SVM) and the KRR, in which the computational cost was drastically reduced. In this paper, we further reduce the computational cost of the An's algorithm by incorporating a pre-processor based on the eigenvalue decomposition of a Gram matrix. Numerical examples are also shown to verify the efficacy of the proposed algorithm.

2. Overview of Kernel Ridge Regression Problems

In this section, we give an overview of the KRR [1]. Let $T = \{(\boldsymbol{x}_i, y_i) \mid i \in \{1, \ldots, n\},\ \boldsymbol{x}_i \in \mathbf{R}^d,\ y_i \in \mathbf{R}\}$ be a given training data set with n samples, where $\boldsymbol{x}_i$ and y_i denote an input vector and the corresponding output value, satisfying

$$y_i = f(\boldsymbol{x}_i) + n_i, \tag{1}$$

where f denotes the unknown function to be estimated and n_i denotes an additive noise. In the KRR, the unknown function f is modeled as

$$\hat{f}(\cdot) = \sum_{i=1}^{n} \alpha_i K(\boldsymbol{x}_i, \cdot), \tag{2}$$

where $\alpha_i \in \mathbf{R}$ denotes the coefficients to be estimated. Let $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_n]' \in \mathbf{R}^n$, where $'$ stands for the transposition operator. In general, the minimizer of the criterion

$$J(\boldsymbol{\alpha}) = \sum_{i=1}^{n} (y_i - \hat{f}(\boldsymbol{x}_i))^2 + \gamma ||\hat{f}||^2_{\mathcal{H}_K} \tag{3}$$

is adopted as the optimal coefficients, where $||\cdot||_{\mathcal{H}_K}$ denotes the norm of the reproducing kernel Hilbert space [4] $\mathcal{H}_K$ uniquely corresponding to the kernel K, and γ denotes a positive regularization parameter.

The closed-form minimizer of Eq. (3) is given as

$$\hat{\boldsymbol{\alpha}} = (G_{XX} + \gamma I_n)^{-1} \boldsymbol{y}, \tag{4}$$

where $\boldsymbol{y} = [y_1, \ldots, y_n]' \in \mathbf{R}^n$, $G_{XX} = (K(\boldsymbol{x}_i, \boldsymbol{x}_j)) \in \mathbf{R}^{n \times n}$ is the Gram matrix of the kernel K with the set of training input vectors $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, and I_n denotes the identity matrix of degree n.

Note that the output vector corresponding to the set of test input vectors $Z = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_m\}$ with the optimal coefficients Eq. (4) can be represented by

$$\hat{\boldsymbol{y}} = [\hat{f}(\boldsymbol{z}_1), \ldots, \hat{f}(\boldsymbol{z}_m)]' = G_{ZX}(G_{XX} + \gamma I_n)^{-1} \boldsymbol{y}, \tag{5}$$

where $G_{ZX} = (K(\boldsymbol{z}_i, \boldsymbol{x}_j)) \in \mathbf{R}^{m \times n}$ from Eq. (2).

3. Preliminaries for Computational Cost Analysis

In this section, we give preliminaries for computational costs of some matrix operations.

It is well known that the computational order of matrix product of two $n \times n$ matrices is $O(n^3)$[†]. Similarly, the computational orders of

- solving linear equation with an $n \times n$ coefficient matrix,
- the inverse of an $n \times n$ matrix, and
- the eigenvalue decomposition of an $n \times n$ matrix

are also $O(n^3)$. However, the actual computational cost of these matrix operations may differ. Since we can not ignore these differences in the computational cost analyses given in the following sections, we introduce the constants R_L, R_I, and R_E for linear equations, inverses, and eigenvalue decompositions, which are the ratios of the computational

Manuscript received April 16, 2019.
Manuscript revised May 23, 2019.

[†]The authors are with the Faculty of Information Science and Technology, Hokkaido University, Sapporo-shi, 060-0814 Japan.
[††]The author is with the Global Station for Big Data and Cybersecurity, Hokkaido University, Sapporo-shi, 060-0814 Japan.
a) E-mail: takira@ist.hokudai.ac.jp

DOI: 10.1587/transfun.E102.A.1317

[†]Today, the computational order of matrix product of two $n \times n$ matrices has been reduced to $O(n^{2.376})$ as shown in [5]. However, the naive $O(n^3)$ algorithm is used in practical cases in terms of actual computational time as stated in [6] (p.71).

costs of these operations to that of the matrix product, and we represent the computational costs of these matrix operations by $R_L n^3 + O(n^2)$, $R_I n^3 + O(n^2)$, and $R_E n^3 + O(n^2)$. Note that $R_L < 1 < R_I < R_E$ holds in general.

4. Cross-Validation in Kernel Ridge Regression

Let $n_{\mathcal{K}}$ be the number of kernel candidates and $\mathcal{K} = \{K_1, \ldots, K_{n_{\mathcal{K}}}\}$ be the set of kernel candidates. Also let n_Γ be the number of regularization parameter candidates and $\Gamma = \{\gamma_1, \ldots, \gamma_{n_\Gamma}\}$ be the set of regularization parameter candidates. The aim of the model selection in the KRR is to find a pair $(K, \gamma) \in \mathcal{K} \times \Gamma$ that achieves a good generalization performance.

In the ℓ-fold cross-validation (ℓ-fold CV), training data set T is divided into ℓ subsets T_k, $(k = 1, \ldots, \ell)$ that includes m_k training samples with $\sum_{k=1}^{\ell} m_k = n$, and one subset is used as a test data set and the others are used for training. Then, the minimizer of the cross-validation error (CV-error), which is defined by the sum of the error between the estimated output values for each test input data set and the corresponding test output values, is adopted as the optimal model. Since the computational costs of all the algorithms considered in this paper are proportional to $n_{\mathcal{K}}$, we focus on the computational cost under a fixed $K \in \mathcal{K}$. The CV-error is represented as follows.

Let

$$E_k = \left[\begin{array}{ccc} O_{m_k, \sum_{j=1}^{k-1} m_j} & I_{m_k} & O_{m_k, \sum_{j=k+1}^{\ell} m_j} \end{array} \right] \in \mathbf{R}^{m_k \times n}, \tag{6}$$

where $O_{\alpha,\beta}$ denotes the $\alpha \times \beta$ zero matrix and let $E_k^c \in \mathbf{R}^{(n-m_k)\times n}$ be the matrix obtained by excluding the rows of E_k from I_n. Also let X_k be the set of training input data set in T_k and let Y_k be the set of training output data set in T_k. Without loss of generality, we can assume that

$$\boldsymbol{y} = [y_1, \ldots, y_n]' = [\boldsymbol{y}_1', \ldots, \boldsymbol{y}_\ell']', \tag{7}$$

where $\boldsymbol{y}_k = E_k \boldsymbol{y}$ is a vector consisting of the elements in Y_k by rearranging and renumbering the components of $\boldsymbol{y}$. In addition, we define

$$\boldsymbol{y}_k^c = E_k^c \boldsymbol{y} = [\boldsymbol{y}_1', \ldots, \boldsymbol{y}_{k-1}', \boldsymbol{y}_{k+1}', \ldots, \boldsymbol{y}_\ell']', \tag{8}$$

which is a vector consisting of the elements in $Y_k^c = Y \setminus Y_k$. Similarly, we assume that

$$G_{XX} = \left[\begin{array}{ccc} G_{X_1X_1} & \cdots & G_{X_1,X_\ell} \\ \vdots & \ddots & \vdots \\ G_{X_\ell X_1} & \cdots & G_{X_\ell X_\ell} \end{array} \right] \tag{9}$$

holds, where $G_{X_k X_j} = E_k G_{XX} E_j'$, and we define

$$G_{X_k^c X_k^c} = E_k^c G_{XX} (E_k^c)', \tag{10}$$

which is the Gram matrix of the kernel K with $X_k^c = X \setminus X_k$. Note that $G_{X_k X_k^c} = E_k G_{XX} (E_k^c)'$ is immediately obtained.

Under these preparations, the CV-error for a certain $\gamma \in \Gamma$ is represented by

$$e(\gamma) = \sum_{k=1}^{\ell} \|\boldsymbol{d}_k(\gamma)\|^2, \tag{11}$$

where

$$\boldsymbol{d}_k(\gamma) = \boldsymbol{y}_k - G_{X_k X_k^c} (G_{X_k^c X_k^c} + \gamma I_{n-m_k})^{-1} \boldsymbol{y}_k^c. \tag{12}$$

When $m_k = m$ and $n = \ell m$, which is a general setting, the computational cost of a naive calculation of Eq. (11) for all $\gamma \in \Gamma$ is reduced to

$$O_N = n_\Gamma \frac{(\ell - 1)^3}{\ell^2} R_L n^3 + O(n_\Gamma \ell n^2) \tag{13}$$

with a sufficiently large n, since it is dominated by solving ℓ linear equations of order $(\ell-1)m_k = ((\ell-1)/\ell)n$ for each $\gamma \in \Gamma$. It should be noted that when $\ell = n$, corresponding to the leave-one-out cross-validation (LOO-CV), $O_N \simeq n_\Gamma R_L n^4 + O(n_\Gamma n^3)$.

In [3], An et al. proposed a fast calculation algorithm for $\boldsymbol{d}_k(\gamma)$ in Eq. (12), and succeeded to drastically reduce the computational cost. The following theorem is the main result of [3].

Theorem 1: [3]

$$\boldsymbol{d}_k(\gamma) = (E_k (G_{XX} + \gamma I_n)^{-1} E_k')^{-1} (E_k \hat{\boldsymbol{\alpha}}). \tag{14}$$

According to Theorem 1, if we obtain $(G_{XX} + \gamma I_n)^{-1}$ and $\hat{\boldsymbol{\alpha}} = (G_{XX} + \gamma I_n)^{-1} \boldsymbol{y}$ in advance for each $\gamma \in \Gamma$, we just have to solve a linear equation of order m_k for each $k \in \{1, \ldots \ell\}$, while a linear equation of order $n - m_k$ must be solved for each $k \in \{1, \ldots \ell\}$ in the naive implementation.

The computational cost of the pre-processing is reduced to $R_I n^3 + O(n^2)$. Therefore, when $m_k = m$ and $n = \ell m$, the overall computational cost for all $\gamma \in \Gamma$ is reduced to

$$O_F = n_\Gamma \left(R_I + \frac{R_L}{\ell^2} \right) n^3 + O(n_\Gamma n^2) \tag{15}$$

with a sufficiently large n. It is trivial that O_F is much smaller than O_N especially in case that ℓ is large since R_I and R_L are constant.

5. Proposed Algorithm

In the fast algorithm [3] by An et al., we have to calculate $(G_{XX} + \gamma I_n)^{-1}$ for each $\gamma \in \Gamma$, which dominates the computational cost Eq. (15). In this section, we introduce a pre-processor based on the eigenvalue decomposition of G_{XX} in order to further reduce the computational cost of the algorithm given in [3].

Let $G_{XX} = P \Lambda P'$ be the eigenvalue decomposition of G_{XX}, then, we have

$$\begin{aligned} \boldsymbol{d}_k(\gamma) = {} & (E_k P (\Lambda + \gamma I_n)^{-1} P' E_k')^{-1} \\ & \times (E_k P (\Lambda + \gamma I_n)^{-1} P' \boldsymbol{y}). \end{aligned} \tag{16}$$

Since the eigenvalue decomposition $G_{XX} = P\Lambda P'$ and calculation of $\boldsymbol{q} = P'\boldsymbol{y}$ are independent from γ, these calculations are needed only once. Note that the computational cost of these calculations is $R_E n^3 + O(n^2)$.

For each $\gamma \in \Gamma$, we can obtain

$$\hat{\boldsymbol{\alpha}} = P(\Lambda + \gamma I_n)^{-1}\boldsymbol{q} \tag{17}$$

independent from $k \in \{1, \ldots, \ell\}$, whose computational cost is $n^2 + O(n)$, which is dominated by a matrix-vector multiplication. Note that the computational cost of $(\Lambda + \gamma I_n)^{-1}\boldsymbol{q}$ is reduced to $O(n)$.

For a certain fixed $\gamma \in \Gamma$ and for each $k \in \{1, \ldots, \ell\}$, we have to calculate

$$M_k(\gamma) = (E_k P)(\Lambda + \gamma I_n)^{-1}(E_k P)', \tag{18}$$

whose computational cost is $mn + m^2 n$ when $m_k = m$ and $n = \ell m$, where each one of them corresponds to $(\Lambda + \gamma I_n)^{-1}(E_k P)'$, and $(E_k P)(\Lambda + \gamma I_n)^{-1}(E_k P)'$; and we have to solve

$$\boldsymbol{d}_k(\gamma) = M_k(\gamma)^{-1}(E_k \hat{\boldsymbol{\alpha}}), \tag{19}$$

whose computational cost is $R_L m^3 + O(m^2)$. Thus, the overall computational cost of the proposed method under the conditions $m_k = m$ and $n = \ell m$ is given by

$$\begin{aligned} O_P &= R_E n^3 + O(n^2) + n_\Gamma(n^2 + O(n)) \\ &\quad + n_\Gamma \ell (mn + m^2 n + R_L m^3 + O(m^2)) \\ &= \left(R_E + n_\Gamma \frac{\ell + R_L}{\ell^2}\right) n^3 + O(n_\Gamma n^2), \end{aligned} \tag{20}$$

with a sufficiently large n. According to Eqs. (15) and (20), it is expected that $O_F > O_P$ if

$$n_\Gamma > \frac{\ell R_E}{\ell R_I - 1} \tag{21}$$

holds in terms of the approximated computational costs.

6. Numerical Examples

In this section, we give numerical examples to confirm the behavior of the proposed algorithm with the popular Gaussian kernel $K(x, y) = \exp\left(-(x - y)^2\right)$ with the setting $d = 1$. Note that our interest is the computational cost of the CV-error calculation for all candidates of $\gamma \in \Gamma$ as mentioned before. Thus, $n_{\mathcal{K}} = 1$ is assumed.

We use training data set T with $n = 2,000$ samples, in which training input values are randomly chosen from the i.i.d. uniform distribution on the interval $[-10, 10]$, and training output values are also randomly chosen from the i.i.d. standard normal distribution[†]. We adopt $\Gamma = \{0.1 + (k-1)\delta \mid k \in \{1, \ldots, n_\Gamma\}\}$ with $\delta = 0.9/(n_\Gamma - 1)$, which implies that all candidates are in the interval $[0.1, 1]$.

[†]Setting for the output values does not affect the computational cost.

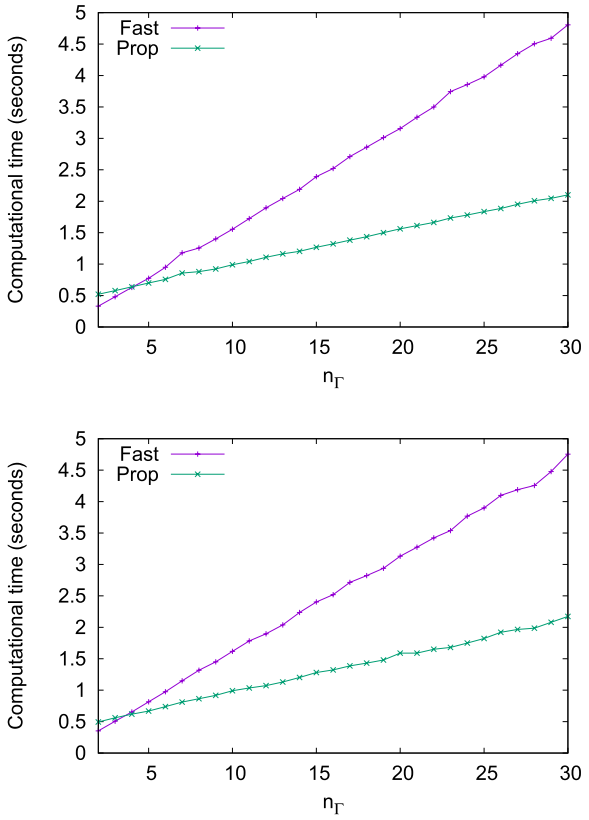

Fig. 1 Averaged computational time with respect to n_Γ (upper: 5-fold CV/lower: LOO-CV).

All the results shown below are obtained by using MATLAB R2019a on the system equipped with Intel Core i7-5960X and 64 GB main memory.

Under these setting, we obtained $R_L \simeq 0.72$, $R_I \simeq 1.88$, and $R_E \simeq 4.69$, respectively, by averaging 1,000 trails.

Figure 1 demonstrate the averaged computational time of 100 trials with respect to n_Γ for 5-fold CV and LOO-CV, in which 'Fast' denotes the algorithm proposed in [3] and 'Prop' denotes the proposed algorithm.

According to these results, it is confirmed that the proposed algorithm is faster than the algorithm proposed in [3] except the case of a small n_Γ approximately specified by ℓ as Eq. (21).

7. Conclusion

In this paper, we proposed a fast cross-validation algorithm for the kernel ridge regression, which is an improved version of the algorithm by An et al., by incorporating the preprocessor based on the eigenvalue decomposition of a Gram matrix.

References

[1] J. Shawe-Taylor and N. Cristianini, Kernel Methods for Pattern Recognition, Cambridge University Press, Cambridge, 2004.
[2] S. Arlot and A. Celisse, "A survey of cross-validation procedures for model selection," Statist. Surv., vol.4, pp.40–79, 2010.

[3] S. An, W. Liu, and S. Venkatesh, "Fast cross-validation algorithms for least squares support vector machine and kernel ridge regression," Pattern Recogn., vol.40, no.8, pp.2154–2162, 2007.
[4] N. Aronszajn, "Theory of reproducing kernels," Trans. Am. Math. Soc., vol.68, no.3, pp.337–404, 1950.
[5] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions," J. Symb. Comput., vol.9, no.3, pp.251–280, 1990.
[6] G. Strang, Introduction to Linear Algebra, 5th ed., Wellesley-Cambridge Press, Wellesley, 2016.