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 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 \( T = \{ (x_i, y_i) | i \in \{ 1, \ldots, n \} \}, \ x_i \in \mathbb{R}^d, \ y_i \in \mathbb{R} \) be a given training data set with \( n \) samples, where \( x_i \) and \( y_i \) denote an input vector and the corresponding output value, satisfying
\[
y_i = f(x_i) + n_i, \tag{1}
\]
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
\[
\hat{f}(\cdot) = \sum_{i=1}^{n} \alpha_i K(x_i, \cdot), \tag{2}
\]
where \( \alpha_i \in \mathbb{R} \) denotes the coefficients to be estimated. Let \( \alpha = [\alpha_1, \ldots, \alpha_n]' \in \mathbb{R}^n \), where ‘ stands for the transposition operator. In general, the minimizer of the criterion
\[
J(\alpha) = \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2 + \gamma \| \hat{f} \|_{H_K}^2 \tag{3}
\]
is adopted as the optimal coefficients, where \( \| \cdot \|_{H_K} \) denotes the norm of the reproducing kernel Hilbert space \([4] K \) uniquely corresponding to the kernel \( K \), and \( \gamma \) denotes a positive regularization parameter.

The closed-form minimizer of Eq. (3) is given as
\[
\hat{\alpha} = (G_{XX} + \gamma I_n)^{-1} y, \tag{4}
\]
where \( y = [y_1, \ldots, y_n]' \in \mathbb{R}^n \), \( G_{XX} = (K(x_i, x_j)) \in \mathbb{R}^{n \times n} \) is the Gram matrix of the kernel \( K \) with the set of training input vectors \( X = \{ x_1, \ldots, x_n \} \), and \( I_n \) denotes the identity matrix of degree \( n \).

Note that the output vector corresponding to the set of test input vectors \( Z = \{ z_1, \ldots, z_m \} \) with the optimal coefficients Eq. (4) can be represented by
\[
\hat{y} = [\hat{f}(z_1), \ldots, \hat{f}(z_m)]' = G_{ZX}(G_{XX} + \gamma I_n)^{-1} y. \tag{5}
\]
where \( G_{ZX} = (K(z_i, x_j)) \in \mathbb{R}^{m \times n} \) 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

\[\text{Copyright © 2019 The Institute of Electronics, Information and Communication Engineers}\]
costs of these operations to that of the matrix product, and we represent the computational costs of these matrix operations by $R_L n^3 + O(n^4)$, $R_I n^3 + O(n^4)$, and $R_E n^3 + O(n^4)$. Note that $R_L < R_I < R_E$ holds in general.

4. Cross-Validation in Kernel Ridge Regression

Let $n_K$ be the number of kernel candidates and $\mathcal{K} = \{K_1, \ldots, K_{n_K}\}$ be the set of kernel candidates. Also let $n_T$ be the number of regularization parameter candidates and $\Gamma = \{\gamma_1, \ldots, \gamma_{n_T}\}$ 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 $\ell$-fold cross-validation ($\ell$-fold CV), training data set $T$ is divided into $\ell$ 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 training samples with $k \in \mathcal{K} \times \Gamma$ that achieves a good generalization performance.

The computational cost of the pre-processing is reduced to $R_I n^3 + O(n^4)$. Therefore, when $m_k = m$ and $n = \ell m$, the overall computational cost for all $\gamma \in \Gamma$ is reduced to

$$O_N = nT \ell (\ell - 1)^3 R_I n^3 + O(nT \ell n^2)$$

with a sufficiently large $n$, since it is dominated by solving $\ell$ linear equations of order $((\ell - 1)/\ell) n$ for each $\gamma \in \Gamma$. It should be noted that when $\ell = n$, corresponding to the leave-one-out cross-validation (LOO-CV), $O_N \approx nT R_I n^3 + O(nT n^2)$.

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

**Theorem 1:** [3]

$$d_k(\gamma) = (E_k (G_{XX} + \gamma I_n)^{-1} E_k')^{-1} (E_k \alpha).$$

According to Theorem 1, if we obtain $(G_{XX} + \gamma I_n)^{-1}$ and $\alpha = (G_{XX} + \gamma I_n)^{-1} y$ 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 $R_I n^3 + O(n^4)$. Therefore, when $n_k = m$ and $n = \ell m$, the overall computational cost for all $\gamma \in \Gamma$ is reduced to

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

with a sufficiently large $n$. It is trivial that $O_F$ is much smaller than $O_N$, especially in case that $\ell$ 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

$$d_k(\gamma) = (E_k P (\Lambda + \gamma I_n)^{-1} P')^{-1} \times (E_k P (\Lambda + \gamma I_n)^{-1} P' y).$$

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

$$e(\gamma) = \sum_{k=1}^{\ell} ||d_k(\gamma)||^2,$$

where

$$d_k(\gamma) = y_k - G_{X_k} x_k^c (G_{X_k} x_k^c + \gamma I_n - m_k)^{-1} y_k^c.$$
Since the eigenvalue decomposition \( G_{XX} = P\Lambda P' \) and calculation of \( q = P'y \) are independent from \( \gamma \), 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

\[
\hat{\alpha} = P(\Lambda + \gamma I_n)^{-1} q
\]

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 \( (\Lambda + \gamma I_n)^{-1} q \) is reduced to \( O(n) \).

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

\[
M_k(\gamma) = (E_k P)(\Lambda + \gamma I_n)^{-1}(E_k P)',
\]

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

\[
d_k(\gamma) = M_k(\gamma)^{-1}(E_k \hat{\alpha}),
\]

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

\[
O_P = R_E n^3 + O(n^2) + n_\Gamma(n^2 + O(n)) + n_\Gamma \ell(mn + m^2 n + R_L m^3 + O(m^2)) = \left( R_E + n_\Gamma \ell + R_L \right)n^3 + O(n_\Gamma n^2),
\]

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

\[
n_\Gamma > \frac{\ell R_E}{R_L - 1}
\]

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, \( n_\Gamma = 1 \) 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\(^\dagger\). 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 \([0.1, 1]\).

\(^\dagger\) Setting for the output values does not affect the computational cost.

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 \( R_L \approx 0.72, R_I \approx 1.88 \), and \( R_E \approx 4.69 \), respectively, by averaging 1,000 trails.

Figure 1 demonstrate the averaged computational time of 100 trials with respect to \( n_\Gamma \) 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 \( n_\Gamma \) approximately specified by \( \ell \) 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


