Fast Cross-validation of Kernel Fisher Discriminant Classifiers

Senjian An, Wanquan Liu and Svetla Venkatesh
Department of Computing
Curtin University of Technology
WA 6102, Australia
senjian,wanquan,svetha@cs.curtin.edu.au

Abstract

Given \( n \) training examples, the training of a Kernel Fisher Discriminant (KFD) classifier corresponds to solving a linear system of dimension \( n \). In cross-validating KFD, the training examples are split into 2 distinct subsets for a number of times (\( L \)) wherein a subset of \( m \) examples is used for validation and the other subset of \((n-m)\) examples is used for training the classifier. In this case \( L \) linear systems of dimension \((n-m)\) need to be solved. We propose a novel method for cross-validation of KFD in which instead of solving \( L \) linear systems of dimension \((n-m)\), we compute the inverse of an \( n \times n \) matrix and solve \( L \) linear systems of dimension \( 2m \), thereby reducing the complexity when \( L \) is large and/or \( m \) is small. For typical 10-fold and leave-one-out cross-validations, the proposed algorithm is approximately 4 and \((\frac{4}{9}n)\) times respectively as efficient as the naive implementations. Simulations are provided to demonstrate the efficiency of the proposed algorithms.

1 Introduction

As an nonlinear version of Fisher linear discriminant analysis [4], Kernel Fisher discriminant (KFD) has been proposed by [6] and it demonstrates the state-of-the-art performance on a range of benchmark data sets. In training KFD classifiers, the model includes some hyper-parameters such as the kernel parameter and the regularization parameter that govern the generalization performance of the classifiers. Finding the hyper-parameters with a good generalization performance is crucial for successful application of KFD. A popular way to estimate the generalization performance of a model is cross-validation. In \( l \)-fold cross-validation, one divides the data into \( l \) subsets of (approximately) equal size and trains the classifier \( l \) times, each time leaving out one of the subsets from training, but using only the omitted subset to compute the classification errors. If \( l \) equals the sample size, this is called leave-one-out cross-validation (LOO-CV). The naive implementation of \( l \)-fold cross-validation trains a classifier for each split of the data and is thus computationally expensive if \( l \) is large especially for LOO-CV where \( l = n \). Many works have been done to reduce the computational complexity of LOO-CV (but none for general \( l \)-fold cross-validation), see [9, 10, 7, 5, 3] for support vector machines, [12] for least square support vector machines and [2] for KFD. It should be noted that all these methods except for [12] computes the approximate LOO errors.

In cross-validating KFD, the classifiers for each training set is not really of interest. One is only concerned with the predicted responses of the left-out examples. In this paper, we present a new formula for calculating the predicted responses of the left-out data and propose algorithms using this formula for evaluating \( l \)-fold cross-validation and LOO-CV of KFD. For \( l \)-fold cross-validation, the proposed algorithm performs generally faster than the naive implementation and the reductions in computation increases with \( l \). An interesting property of the proposed \( l \)-fold cross-validation is that its computation complexity decreases with \( l \) while the naive implementation involves more computation with larger \( l \). Comparing with [2] for LOO-CV, the proposed algorithm provides a more accurate approximation of the exact LOO errors while the computational complexity is slightly reduced but with the same order \( O(n^3) \).

The layout of this paper is as follows. In Section 2, we briefly review the formulation of the KFD classifiers. In Section 3, we develop the cross-validation formula for calculating the predicted responses of the left-out data and develop the algorithms for \( l \)-fold and LOO-CV evaluation. Section 4 provides experimental examples to illustrate the performance of the proposed algorithm with a comparison to naive implementations and the method in [2].
2 Kernel Fisher Discriminant Analysis

Given a training set \( \{(x_i, y_i)\}_{i=1}^n \) with input data \( x_i \in \mathbb{R}^n \) and class labels \( y_i \in \{-1, 1\} \). Assume that one have \( n_+ \) positive samples and thus one have \( n_- = n - n_+ \) negative samples. Fisher’s linear discriminant attempts to find a linear projection such that the classes are well separated and this is achieved by maximizing the ratio of the between and within class variance, that is,

\[
J(w) = \frac{w^T S_B w}{w^T S_W w}
\]

where \( S_B = (m_1 - m_2)(m_1 - m_2)^T \),

\[
S_W = \sum_{k=\pm} \sum_{i \in \mathcal{I}_k} (x_i - m_k)(x_i - m_k)^T
\]

and \( m_k, \mathcal{I}_k \) denote the sample mean and the index set for class \( k \), respectively.

In formulating KFD, the projection coefficient vector \( w \) is expressed in terms of mapped training patterns, i.e.,

\[
w = \sum_{i=1}^n \alpha_i \varphi(x_i)
\]

and the optimization problem for KFD can then be written as [6]

\[
J(\alpha) = \frac{\alpha^T M \alpha}{\alpha^T N \alpha}
\]

where

\[
M = \left( \frac{1}{n_+} K_+ 1_{n_+} - \frac{1}{n_-} K_- 1_{n_-} \right) \left( \frac{1}{n_+} K_+ 1_{n_+} - \frac{1}{n_-} K_- 1_{n_-} \right)^T
\]

\[
N = K K - \sum_{k=\pm} n_k K_k 1_{n_k} 1_{n_k} K_k
\]

\[
K_k = [K_{ij}], i=1,2,\cdots,n; \ j \in \mathcal{I}_k, k=\pm.
\]

Here \( 1_{n_k} \) is a vector with \( n_k \) ones, \( K_{ij} = \varphi(x_i)^T \varphi(x_j) \) and \( \varphi(\cdot) : \mathbb{R}^m \rightarrow \mathbb{R}^n \) is usually induced by a kernel function that maps the input space to a high dimensional feature space. The kernel function \( K(\cdot, \cdot) \) can typically be either linear, polynomial or Gaussian kernels.

Since \( N \) is likely to be ill-conditioned, it is suggested [6] to regularize \( N \) as \( N_\mu = N + \mu I_n \). The coefficients, \( \alpha \), is given by the eigenvector of \( N_\mu^{-1} M \) with the largest eigenvalue. The projection of a test point onto the discriminant is given by

\[
f(x) = w^T \varphi(x) = \sum_{i=1}^n \alpha_i K(x_i, x).
\]

To use this projection in classification, one needs to find a suitable bias, \( b \), which is usually chosen such that the projections of the two class data is with zero mean, i.e.,

\[
b = -1^n K\alpha/n. \] The final KFD classifier is formulated as

\[
y(x) = \text{sign} \left[ \sum_{i=1}^n \alpha_i K(x_i, x) + b \right].
\]

It is shown that \( \alpha, b \) can also be obtained by solving the following system of linear equations [11]

\[
\begin{bmatrix}
    n \\
    K_{1n} \\
    K K + \mu I_n
\end{bmatrix}
\begin{bmatrix}
    b \\
    \alpha
\end{bmatrix}
= 
\begin{bmatrix}
    0 \\
    K \hat{y}
\end{bmatrix}
\]

where

\[
\hat{y}_i = \begin{cases}
    \frac{n}{n_+}, & \text{if } y_i = 1; \\
    \frac{n}{n_-}, & \text{else.}
\end{cases}
\]

3 The Cross-Validation Algorithms

In this section, we provide the formulas and algorithms for fast cross-validation of KFD with the derivations and proofs being delegated in Appendix.

First, we introduce some notations. Denote

\[
B \triangleq (\mu I + K K)^{-1}, \ C \triangleq B K, \ \xi \triangleq B K 1_n, \ \eta \triangleq \mu B 1_n, \ d \triangleq -\mu 1_n^T B 1_n
\]

and let \( b^*, \alpha^* \) be the solution of (8). Then

\[
b^* = -\frac{1}{d} \eta^T \hat{y}, \alpha^* = B K \hat{y} + \frac{1}{d} \xi \eta^T \hat{y}.
\]

Also, we use \( e^* \) to denote the training error vector of KFD classifier, that is

\[
e^* = \hat{y} - K \alpha^* - b^* 1_n.
\]

In \( l \)-fold cross-validation, one splits the data into \( l \) subsets \( \{x_k\}_{i=1}^{n_k} \) of (approximately) equal size \( (n_k, i) \), i.e., \( n_k \approx n \), where \( k = 1, 2, \cdots, l \) and \( \sum_{k=1}^l n_k = n \). Correspondingly, we split \( y, \hat{y}, \alpha^*, e^*, \xi, \eta \) into \( l \) sub-vectors and split \( B \) and \( B K \) into \( l \times l \) blocks. That is

\[
y \triangleq \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(l) \end{bmatrix}, \ \hat{y} \triangleq \begin{bmatrix} \hat{y}(1) \\ \hat{y}(2) \\ \vdots \\ \hat{y}(l) \end{bmatrix}, \ \alpha^* \triangleq \begin{bmatrix} \alpha^*(1) \\ \alpha^*(2) \\ \vdots \\ \alpha^*(l) \end{bmatrix}
\]

\[
e^* \triangleq \begin{bmatrix} e^*(1) \\ e^*(2) \\ \vdots \\ e^*(l) \end{bmatrix}, \ \xi \triangleq \begin{bmatrix} \xi(1) \\ \xi(2) \\ \vdots \\ \xi(l) \end{bmatrix}, \ \eta \triangleq \begin{bmatrix} \eta(1) \\ \eta(2) \\ \vdots \\ \eta(l) \end{bmatrix}
\]

\[
\hat{y}_i = \begin{cases}
    \frac{n}{n_+}, & \text{if } y_i = 1; \\
    \frac{n}{n_-}, & \text{else.}
\end{cases}
\]

\[
i = 1, 2, \cdots, n.
\]
Define

\[
D \triangleq \begin{bmatrix}
D_{11} & D_{12} & \cdots & D_{1l} \\
D_{21}^T & D_{22} & \cdots & D_{2l} \\
\vdots & \vdots & \ddots & \vdots \\
D_{l1}^T & D_{l2} & \cdots & D_{ll}
\end{bmatrix} \quad \triangleq -B + \frac{1}{2} \xi \xi^T
\]

\[
E = \begin{bmatrix}
E_{11} & E_{12} & \cdots & E_{1l} \\
E_{21} & E_{22} & \cdots & E_{2l} \\
\vdots & \vdots & \ddots & \vdots \\
E_{l1} & E_{l2} & \cdots & E_{ll}
\end{bmatrix} \quad \triangleq \mu B + \frac{1}{2} \eta \eta^T
\]

\[
C \triangleq \begin{bmatrix}
C_{11} & C_{12} & \cdots & C_{1l} \\
C_{21}^T & C_{22} & \cdots & C_{2l} \\
\vdots & \vdots & \ddots & \vdots \\
C_{l1}^T & C_{l2}^T & \cdots & C_{ll}
\end{bmatrix} \quad \triangleq BK + \frac{1}{2} \xi \eta^T.
\]

Now, we are ready to present the main result.

**Theorem 1** Let \( y^{(k)}(x) = \text{sign}[g_k(x)] \) denote the classifier formulated by leaving the \( k \)th group out and let \( \hat{e}^{(k)} = [\hat{e}_{k,1}, \hat{e}_{k,2}, \cdots, \hat{e}_{k,n}] \), \( \hat{e}_{k,i} = y_{k,i} - g_k(x_{k,i}) \). Assume that the ratio of the numbers of positive and negative samples for training is the same as that for the entire training set, that is \( n_p/n_- \). Then \( \hat{e}^{(k)} \) can be obtained by solving the following system of linear equations

\[
\begin{bmatrix}
D_{kk} & C_{kk} \\
C_{kk}^T & E_{kk}
\end{bmatrix}
\begin{bmatrix}
Z_k \\
\hat{e}^{(k)}
\end{bmatrix}
= \begin{bmatrix}
\alpha^*_k \\
\epsilon^*_k
\end{bmatrix}
\]

where \( Z_k \in \mathbb{R}^{nk} \) is an auxiliary variable.

To ensure that the ratio of the numbers of positive and negative training samples remains \( n_p/n_- \) for each split, one can divide the data in each class separately and then combine them such that the ratio of the positive and negative samples is \( n_p/n_- \) in each fold. On the other hand, if the sample size \( n \) is large and the data is put in a random order, then the ratio of the positive and negative samples for training will be approximately equal for each split. In particular, for LOO-CV, this ratio is \( (n_p-1)/n_- \) (or \( n_p/n_- - 1 \)) if a positive (or negative respectively) sample is set for validation and thus approximately equals \( n_p/n_- \) if both \( n_p \) and \( n_- \) are large.

Based on Theorem 1, one can evaluate the \( l \)-fold cross-validation of KFD as follows.

1. Evaluate the kernel matrix \( K \) and compute \( B = (I + KK)^{-1} \) and \( C = BK \);
2. Compute \( \alpha^* \);
3. Solve the linear system (16) for \( k = 1, 2, \cdots, l \);
4. Compute the predicted labels, \( y^{(k)} = \text{sign}[y^{(k)} - \hat{e}^{(k)}] \);
5. Sum up all incorrect labels \( N_{err} = \frac{1}{2} \sum_{k=1}^{n_k} \sum_{i=1}^{n_k} |y_{k,i} - y^{(k,i)}| \)

In the naive implementation of \( l \)-fold cross-validation, one trains the KFD classifiers \( l \) times, each time leaving out one of the subsets from training, and using the omitted subset to compute the classification errors. For each classifier, the training involves one multiplication of two \((n - n_k) \times (n - n_k)\) matrices and the solution of a linear system of dimension \((n - n_k)\). Note that multiplication of two \( m \times m \) matrices and solving a linear system with dimension \( m \) have the complexity of \( m^3 \) and \( \frac{1}{3} m^3 \) respectively [8] and \( n_v \approx \frac{n}{4} \), the complexity of the naive \( l \)-fold CV is \( \frac{n^4}{l} (n - n_v)^3 \approx \frac{4}{3} (l-1) n^3 \). In the special case when \( n_v = 1, l = n \), this reduces to the LOO-CV and the computational complexity is \( \frac{2}{3} n(n-1)^3 \approx \frac{4}{3} n^4 \).

On the other hand, the proposed algorithm involves the inverse of a \( n \times n \) matrix, two multiplications of \( n \times n \) matrices, and the solutions of \( l \) linear systems with dimension \( 2n_v \). The total complexity is approximately \( 3n^3 + \frac{2}{3} n^3 \approx 3 + \frac{8}{3} n^3 \). Hence, the proposed algorithm is approximately \( 4(l-1)^3 \) times as efficient as the naive implementation. For any \( l \geq 5 \), \( (4l-1)^3 > 8 + 9l^2 \) and thus the proposed algorithm is more efficient. For typical 10-fold cross-validation, the proposed algorithm is approximately 4 times as efficient as the naive implementations. It is interesting to note that the computations of this algorithm decrease with increasing \( l \) while the naive implementations involves more computations with larger \( l \). Therefore, the computational reductions increase with \( l \).

In the case that \( l = n \), this reduces to LOO-CV and the complexity is \( 3n^3 \) while the naive implementation has much larger complexity \( \frac{4}{3} n^4 \). To apply the formula following (10) in [2] for LOO-CV, one need evaluate three multiplications of \((n + 1) \times (n + 1)\) matrices, an inverse of an \((n + 1) \times (n + 1)\) matrix and a solution of a linear system with dimension \((n + 1)\). Hence the total complexity is approximately \( \frac{1}{3} n^3 \) which is larger than our proposed algorithm.

## 4 Experimental Results

We compare the performance of the proposed and the naive cross-validation methods on two benchmark datasets from UCI benchmark repository [1]: the Statlog German credit (1000 patterns with dimension 24) and the Johns Hopkins university ionosphere (351 patterns with dimension 33).

Fig 1 compares the mean run time of the proposed and the naive \( l \)-fold cross-validation for various folds on the Statlog German credit dataset. The number of training examples is \( 2/3 \) of the all samples, that is 666.
Fig 2 compares the mean run time of the proposed and the naive 10-fold cross-Validation for various number of training examples on the Statlog German credit dataset.

Note that in Fig. 1 and 2, the time is for one pair of $\sigma$ and $\mu$. In model selection of KFD, one need try many possible pairs of $\sigma$ and $\mu$ and then find the best pair with minimal validation errors.

Fig 3 and Fig 4 compare the relative approximation errors of the predicted responses and LOO errors respectively computed by the method in [2] and the proposed algorithms on the dataset ionosphere with $\mu = 1, \sigma^2 = 10$. The relative approximation error is defined as $\|\hat{r} - r\|/\|r\|$ where $r$ is the vector of the predicted responses computed directly by the LOO procedure and $\hat{r}$ is obtained by using the proposed algorithm or the method in [2] respectively.

5 Conclusions

Based on the inverse of the system matrix, this paper presents a new formula for computing the predicted responses of the left-out examples in $i$-fold and LOO cross-validation of KFD. A novel cross-validation algorithm is developed using this formula. The proposed algorithms are generally more efficient than the naive implementations especially for LOO-CV. Comparing with the recently developed efficient method in [2], the proposed algorithm is slightly more efficient and with less approximation errors.

Appendix: Proof of Theorem 1

Let $e = \hat{y} - K\alpha - b1_n$, which is the predicted residual vector. Then

$$nb + 1_n^T K \alpha = 0 \Leftrightarrow 1_n^T e = 0 \quad (17)$$

and

$$K1_nb + (KK + \mu I)\alpha = K\hat{y} \Leftrightarrow Ke + \mu \alpha = 0. \quad (18)$$

Thus, we have the extended linear system

$$
\begin{bmatrix}
0 & 0 & 1_n^T \\
0 & -\mu I & K \\
1_n & K & I
\end{bmatrix}
\begin{bmatrix}
b \\
\alpha \\
e
\end{bmatrix}
= \begin{bmatrix}
0 \\
0 \\
\hat{y}
\end{bmatrix}.
\quad (19)
$$

Let $A \triangleq \begin{bmatrix}
-\mu I & K \\
K & I
\end{bmatrix}$. Since $\mu I + KK$ is positive definite and thus invertible, one have

$$A^{-1} = \begin{bmatrix}
-(\mu I + KK)^{-1} & (\mu I + KK)^{-1}K \\
K(\mu I + KK)^{-1} & \mu(\mu I + KK)^{-1}
\end{bmatrix}. \quad (20)$$

Recall the notations in (10) and (15), and note that $d \neq 0$, one have

$$
\begin{bmatrix}
0 & 0 & 1_n^T \\
0 & -\mu I & K \\
1_n & K & I
\end{bmatrix}
\begin{bmatrix}
b \\
\alpha \\
e
\end{bmatrix}
= \begin{bmatrix}
-\frac{1}{d} [0, 1_n^T] A^{-1} \\
-\frac{1}{d} A^{-1} [0, 1_n^T]^T A^{-1} \\
-\frac{1}{d} A^{-1} [0, 1_n^T]^T [0, 1_n^T] A^{-1}
\end{bmatrix}
= \begin{bmatrix}
-\frac{1}{d} \xi - B + \frac{1}{d} \xi \xi^T BK + \frac{1}{d} \xi \eta^T \\
-\frac{1}{d} \eta K B + \frac{1}{d} \eta \xi^T \mu B + \frac{1}{d} \eta \eta^T \\
-\frac{1}{d} \xi^T - \frac{1}{d} \eta \xi^T - \frac{1}{d} \eta \eta^T
\end{bmatrix}.
\quad (21)
$$

Let $b^*, \alpha^*, e^*$ be the solution of (19). Then

$$b^* = -\frac{1}{d} \eta^T \hat{y}, \alpha^* = BK\hat{y} + \frac{1}{d} \xi \eta^T \hat{y}, e^* = \mu B\hat{y} + \frac{1}{d} \eta \eta^T \hat{y}. \quad (22)$$

Note that the solution $b^*, \alpha^*$ of (8) satisfy (19) and thus (6) holds.

Next, we prove (16). By proper permutation, one can rewrite (19) as follows.

$$
\begin{bmatrix}
0 & 0 & 1_n^T \\
0 & -\mu I_n & K_{11} \\
1_{n_k} & K_{11} & I_{n_k}
\end{bmatrix}
\begin{bmatrix}
b \\
\alpha^{(k)} \\
e^{(k)}
\end{bmatrix}
= \begin{bmatrix}
0 \\
0 \\
\hat{y}^{(k)}
\end{bmatrix}
\quad (23)
$$

where $\hat{n} = n - n_k, K_{11} \in \mathbb{R}^{(n-n_k) \times (n-n_k)}, K_{12} \in \mathbb{R}^{(n-n_k) \times n_k}, K_{22} \in \mathbb{R}^{n_k \times n_k}, \alpha^{(k)}, e^{(k)}, \hat{y}^{(k)}$ are subvectors of $\alpha, e, \hat{y}$ respectively by deleting $\alpha^{(k)}, e^{(k)}, \hat{y}^{(k)}$.

From the definition of the weighted labels $\hat{y}$ in (9), it depends on the ratio of numbers of the positive and negative training samples. Under the assumption that the ratio of the positive and negative training samples remains $n_+/n_-$ when $k$th group is omitted in training, the weighted labels remains the same as that for entire set being training samples.

Thus, to train the classifier after leaving the $k$th group out, one need solve the following system of linear equations

$$
\begin{bmatrix}
0 & 0 & 1_{n-n_k}^T \\
0 & -\mu I_{n-n_k} & K_{11} \\
1_{n-n_k} & K_{11} & I_{n-n_k}
\end{bmatrix}
\begin{bmatrix}
b \\
\alpha^{(k)} \\
e^{(k)}
\end{bmatrix}
= \begin{bmatrix}
0 \\
0 \\
\hat{y}^{(k)}
\end{bmatrix}
\quad (24)
$$

Let $\tilde{b}^{(k)}, \tilde{\alpha}^{(k)}, \tilde{e}^{(k)}$ be the solution of the above system of linear equations. Then the predicted residual vector, denoted
by \( \hat{e}_k \), of the cross-validation data is then
\[
\hat{e}_k = \hat{y}(k) - K_{12}^{T} \alpha_{n2}^{(k)} - \beta b_{nk}
\]
\[
= \begin{bmatrix} 1_{nk} & K_{12}^{T} \end{bmatrix} A_{11}^{-1} \begin{bmatrix} 0 \\
0 \end{bmatrix}
\]
(25)
where
\[
A_{11} \triangleq \begin{bmatrix} 0 & 0 & 1_{n2}^{T} \\
0 & -\mu I_{n} & K_{11} \\
1_{nk} & K_{12} & I_{n}
\end{bmatrix}
\]
\[
A_{12} \triangleq \begin{bmatrix} 0 & 1_{n2}^{T} \\
0 & K_{12} \\
K_{22} & I_{n}
\end{bmatrix}
\]
\[
A_{22} \triangleq \begin{bmatrix} -\mu I_{nk} & K_{22} \\
K_{22} & I_{n}
\end{bmatrix}
\]
(26)
The inverse of a block matrix is given by
\[
\begin{bmatrix} A_{11} & A_{12} \\ A_{12}^{T} & A_{22} \end{bmatrix}^{-1} = \begin{bmatrix} F_{11}^{-1} & -A_{11}^{-1} A_{12} F_{22}^{-1} \\ -F_{22}^{-1} A_{12}^{T} A_{11}^{-1} & F_{22}^{-1} \end{bmatrix}
\]
(27)
where
\[
F_{11} = A_{11} - A_{12} A_{22}^{T} A_{12}^{T}
\]
\[
F_{22} = A_{22} - A_{12} A_{11}^{-1} A_{12}^{T}
\]
(28)
Let \( Z_{k} \triangleq K_{12}^{T} \alpha_{n2}^{(k)} \). From (23) and (27), we have
\[
\begin{bmatrix} \alpha_{n2}^{(k)} \\
\hat{e}_{n2}^{(k)} \end{bmatrix} = \begin{bmatrix} -F_{22}^{-1} A_{12}^{T} A_{11}^{-1} & F_{22}^{-1} \end{bmatrix} \begin{bmatrix} 0 \\
0 \end{bmatrix}
\]
\[
= F_{22}^{-1} \begin{bmatrix} Z_{k} \\
\hat{e}_{n2}^{(k)} \end{bmatrix}
\]
(29)
From (21) and the notations in (15), it is easy to verify that
\[
F_{22}^{-1} = \begin{bmatrix} D_{kk} & C_{kk} \\ C_{kk}^{T} & E_{kk} \end{bmatrix}
\]
(30)
and thus (16) is proved.

References


Figure 1. Run time of l-fold cross validation vs number of folds.
Figure 2. Run time of 10-fold cross validation vs number of training examples.

Figure 3. The approximation errors of the proposed LOO algorithm and the method in [2].

Figure 4. The LOO errors computed by the naive, the proposed algorithms and the method in [2].