Efficient Cross-validation of the Complete Two Stages in KFD Classifier Formulation[*]

Senjian An, Wanquan Liu and Svetha Venkatesh

Department of Computing, Curtin University of Technology, WA 6102, Australia
{senjian,wanquan,svetha}@cs.curtin.edu.au

Abstract

This paper presents an efficient evaluation algorithm for cross-validating the two-stage approach of KFD classifiers. The proposed algorithm is of the same complexity level as the existing indirect efficient cross-validation methods but it is more reliable since it is direct and constitutes exact cross-validation for the KFD classifier formulation. Simulations demonstrate that the proposed algorithm is almost as fast as the existing fast indirect evaluation algorithm and the two-stage cross-validation selects better models on most of the thirteen benchmark data sets.

1 Introduction

Given a training set $\{(x_i, y_i)\}_{i=1}^n$ with input data $x_i \in \mathbb{R}^n$ and class labels $y_i \in \{-1, 1\}$, let us assume that one has n_+ positive samples and thus $n_- = (n - n_+)$ negative samples. Fisher's linear discriminant [7] 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. Kernel Fisher discriminant (KFD) proposed by [9] is a powerful nonlinear version of Fisher's linear discriminant where the kernel trick allows the efficient computation of fisher discriminant in feature space. To complete the KFD classifier formulation, one also needs to find a suitable bias term. If one chooses the bias term such that the projections of all the training data are with zero mean, the KFD can be described as

$$y(x) = \text{sign}\left[F(x) + b\right] \tag{1}$$

where

$$F(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x), \tag{2}$$

[*]This work was supported by Australian Research Council.

and α, b can be obtained by solving the following system of linear equations [15]

$$\begin{bmatrix} n & \mathbf{1}_n^T K \\ K\mathbf{1}_n & KK + \mu I_n \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ K\hat{y} \end{bmatrix}. \tag{3}$$

Here μ is the regularization parameter, $K_{ij} = \varphi(x_i)^T \varphi(x_j)$ and $\varphi(\cdot) : \mathbb{R}^m \to \mathbb{R}^n$ is induced by some kernel function that maps the input space to a high dimensional feature space (the kernel function $K(\cdot, \cdot)$ can typically be linear, polynomial or Gaussian kernels) and $\hat{y}_i$ are weighted labels given as

$$\hat{y}_i = \begin{cases} \frac{n}{n_+} & \text{if } \; y_i = 1; \\ -\frac{n}{n_-}, & \text{else.} \end{cases} \quad i = 1, 2, \cdots, n. \tag{4}$$

However, the bias term b chosen in this way is usually not a good choice for classification performance and one usually needs to adjust the bias term by applying some one-dimensional (1D) classification method (say 1D support vector machines) on the extracted feature $F(x_i), i = 1, 2, \cdots, n$ [9]. Hence, the typical KFD classifier formulation includes two stages: the first stage computes the fisher discriminant in feature space by solving α from (3) and the second stage computes the optimal bias term by applying some 1D classification method on the extracted features. Hence, in model selection for KFD classifiers, we need select three types of hyper-parameters: the kernel parameters, the regularization parameter μ and (possibly) the hyper-parameter for the 1D classification method.

Cross-validation is one of the popular ways for model selection. 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 especially when l is large. Much efforts have been done to reduce the complexity of cross-validation for kernel based classification, see [13, 14, 10, 8, 6] for LOO-CV of support vector machines, [16] for LOO-CV of least square support vector machines, [5] for LOO-CV of KFD, and [1, 2] for general l-fold cross-validation of least square support vector machines and KFD.

In l-fold cross-validation of KFD classifiers, given certain model parameters, for each split, one needs to find the discriminant coefficients α_i by solving (3) and then compute the bias term b by applying some 1-D classification method on the extracted feature $F(x_i)$ (2) and finally compute the validation errors using the classifier (1). However, the efficient cross-validation algorithms in [5] and [2] provide two efficient ways to compute the predicted responses (i.e., $F(x_k) + b$) of the validation data x_k where the bias term b is from the solution of (3) instead of the optimal bias term obtained by applying some 1-D classification method on the extracted feature. Since the bias term selected in this way is usually not used in the KFD classifier, as [5] suggested, one chooses the model parameter by comparing the predicted residual of sum of squares instead of comparing the cross-validation errors. Hence these algorithms are indirect and thus approximate for KFD model selection. For the efficient implementation of the exact l-fold cross-validation of KFD classifiers, one needs to provide efficient ways to compute, for each split, both the predicted response $F(x_k)$ of the validation data x_k and the extracted feature $F(x_i)$ associated with the training data x_i. This paper will present such ways to evaluate the cross-validation of the two-stage KFD classifiers. The proposed algorithm is of the same complexity level as the existing indirect efficient cross-validation methods but it is more reliable since it is direct and constitutes exact cross-validation for the KFD classifiers.

2 Efficient Cross-validation

Since the nonlinear term KK appears in the system matrix of (3), it is difficult to derive the predicted response for l-fold cross-validation by applying the matrix inversion formula. It is fortunate that we can reformulate it as an expanded linear system such that K appears linearly in the system matrix by introducing the training error vector $e = \hat{y} - K\alpha - b\mathbf{1}_n$. Note that $\mathbf{1}_n^T\hat{y} = 0$. We have

$$nb + \mathbf{1}_n^T K\alpha = 0 \Leftrightarrow \mathbf{1}_n^T e = 0 \tag{5}$$

and

$$K\mathbf{1}_n b + (KK + \mu I)\alpha = K\hat{y} \Leftrightarrow Ke = \mu\alpha. \tag{6}$$

Thus, the solutions (α, b) of (3) and that of the following extended linear system are identical:

$$\begin{bmatrix} 0 & 0 & \mathbf{1}_n^T \\ 0 & -\mu I & K \\ \mathbf{1}_n & K & I \end{bmatrix} \begin{bmatrix} b \\ \alpha \\ e \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \hat{y} \end{bmatrix}. \tag{7}$$

Based on this formulation and the well-known matrix inversion formula, this paper will derive the predicted residual formula for efficient l-fold cross-validation.

Let $B = \mu(\mu I + KK)^{-1}, C = (\mu I + KK)^{-1}K$. Then

$$\begin{aligned} \begin{bmatrix} -\mu I & K \\ K & I \end{bmatrix}^{-1} &= \begin{bmatrix} -B/\mu & C \\ C & B \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & \mathbf{1}_n^T \\ 0 & -\mu I & K \\ \mathbf{1}_n & K & I \end{bmatrix}^{-1} &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & -B/\mu & C \\ 0 & C & B \end{bmatrix} \\ &+ \frac{1}{d}\begin{bmatrix} 1 \\ -\beta \\ -\gamma \end{bmatrix}\begin{bmatrix} 1 \\ -\beta \\ -\gamma \end{bmatrix}^T \end{aligned} \tag{8}$$

and therefore the solution of (7) is

$$b = -\gamma^T\hat{y}/d, \alpha = C\hat{y} - b\beta, e = B\hat{y} - b\gamma \tag{9}$$

where

$$d = -\mathbf{1}_n^T B\mathbf{1}_n, \beta = C\mathbf{1}_n, \gamma = B\mathbf{1}_n. \tag{10}$$

In l-fold cross-validation, one splits the data into l subsets $\{x_{k,i}\}_{i=1}^{n_k}$ of (approximately) equal size (n_v), i.e., $n_k \approx n_v$, where $k = 1, 2, \cdots, l$ and $\sum_{k=1}^{l} n_k = n$. The naive implementation of l-fold cross-validation needs to solve the linear system (3) repeatedly for each split. In this paper, we implement it based on the solution of (7) and B, C, β, γ.

Corresponding to the data split, we split $y, \hat{y}, \alpha, e, \beta, \gamma$ into l sub-vectors and split B and C into $l \times l$ blocks, i.e.,

$$\begin{aligned} y \triangleq \begin{bmatrix} y_{(1)} \\ y_{(2)} \\ \vdots \\ y_{(l)} \end{bmatrix}, \hat{y} \triangleq \begin{bmatrix} \hat{y}_{(1)} \\ \hat{y}_{(2)} \\ \vdots \\ \hat{y}_{(l)} \end{bmatrix}, \alpha \triangleq \begin{bmatrix} \alpha_{(1)} \\ \alpha_{(2)} \\ \vdots \\ \alpha_{(l)} \end{bmatrix}, \\ e \triangleq \begin{bmatrix} e_{(1)} \\ e_{(2)} \\ \vdots \\ e_{(l)} \end{bmatrix}, \beta \triangleq \begin{bmatrix} \beta_{(1)} \\ \beta_{(2)} \\ \vdots \\ \beta_{(l)} \end{bmatrix}, \gamma \triangleq \begin{bmatrix} \gamma_{(1)} \\ \gamma_{(2)} \\ \vdots \\ \gamma_{(l)} \end{bmatrix} \end{aligned} \tag{11}$$

and

$$B \triangleq \begin{bmatrix} B_1 & B_{12} & \cdots & B_{1l} \\ B_{12}^T & B_2 & \cdots & B_{2l} \\ \vdots & \vdots & \ddots & \vdots \\ B_{1l}^T & B_{2l}^T & \cdots & B_l \end{bmatrix}, \tag{12}$$

$$C \triangleq \begin{bmatrix} C_1 & C_{12} & \cdots & C_{1l} \\ C_{12}^T & C_2 & \cdots & C_{2l} \\ \vdots & \vdots & \ddots & \vdots \\ C_{1l}^T & C_{2l}^T & \cdots & C_l \end{bmatrix} \tag{13}$$

Note that, in the above notations, we use B_k, instead of B_{kk}, to denote the main diagonal blocks for simplicity since these terms will be frequently used in the rest part of this paper. For convenience, we also introduce $\bar{B}_k$ and $\bar{C}_k$ to denote the k^{th} column blocks of B and C by deleting B_k and C_k respectively, i.e.,

$$\bar{B}_k \triangleq \begin{bmatrix} B_{1k} \\ \vdots \\ B_{(k-1)k} \\ B_{(k+1)k} \\ \vdots \\ B_{lk} \end{bmatrix}, \bar{C}_k \triangleq \begin{bmatrix} C_{1k} \\ \vdots \\ C_{(k-1)k} \\ C_{(k+1)k} \\ \vdots \\ C_{lk} \end{bmatrix} \tag{14}$$

Given a classifier $y(x) = \text{sign}[g(x)]$, we will call $\hat{y}_i - g(x_i)$ the training error (or validation error) if x_i is a training sample (or validation sample respectively).

Now we present the basic formula for the proposed algorithm. Let the kth group be used as validation set and let $\tilde{e}^{(k)}$ and $\tilde{e}_{(k)}$ denote the training error vector and the validation error vector respectively. Then we have (for derivations see Appendix)

$$\begin{aligned} \tilde{e}_{(k)} &= Z_2; \\ \tilde{b}_k &= b + (\beta_{(k)}^T Z_1 + \gamma_{(k)}^T Z_2)/d; \\ \tilde{e}^{(k)} &= e^{(k)} - \bar{C}_k Z_1 - \bar{B}_k Z_2 - (b_k - b)\gamma^{(k)} \end{aligned} \tag{15}$$

where $\tilde{b}_k$ is the bias term of the classifier and (Z_1, Z_2) is the solution of the following system of linear equations

$$\left\{ \begin{bmatrix} -B_k/\mu & C_k \\ C_k & B_k \end{bmatrix} + \frac{1}{d} \begin{bmatrix} \beta_{(k)} \\ \gamma_{(k)} \end{bmatrix} \begin{bmatrix} \beta_{(k)}^T & \gamma_{(k)}^T \end{bmatrix} \right\} \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} = \begin{bmatrix} \alpha_{(k)} \\ e_{(k)} \end{bmatrix}. \tag{16}$$

Based on (15), one can evaluate the l-fold cross-validation of KFD as follows.

1. *Evaluate the kernel matrix K and compute $B = \mu(\mu I + KK)^{-1}$ and $C = \frac{1}{\mu}BK$;*

2. *Compute $d, b, \alpha, \beta, \gamma, e$ using (9,10);*

3. *For $k = 1, 2, \cdots, l$, a) solve the linear system (16); b) compute $\tilde{e}^{(k)}$ using (15) and train the best bias term $\tilde{b}$ by applying 1D-SVM on the 1-d projected data $(\hat{y}^{(k)} - \tilde{e}^{(k)})$; c) compute the predicted labels, $y^{(k)} = \text{sign}[\tilde{y}_{(k)} - \tilde{e}_{(k)} + \tilde{b}]$;*

4. *Repeat Step 3) for a number of times (m), each time permute α, β, γ, e and both rows and columns of B, C with a random order;*

5. *Sum up all incorrect labels.*

Comparing this algorithm with that in [2], the most extra computations come from the evaluation of 1D-SVM in step 3. According to [12], the training of 1D-SVM is of complexity $O(n \lg n)$. Hence the complexity of the proposed algorithm is still of $O(n^3)$. For very large data set, one may apply the incomplete Cholesky decomposition technique [3], to further reduce the computational complexity.

3 Experimental Results

We compare the efficiency of the proposed and the naive cross-validation methods on the benchmark dataset: the Statlog German credit (1000 patterns with dimension 24), from UCI benchmark repository [4]. Fig 1 compares the mean run time of the proposed, the indirect [2] and the naive 10-fold cross-Validation for various number of training examples on the Statlog German credit dataset. It demonstrates that the proposed algorithm is almost as fast as the indirect evaluation algorithm in [2] and much faster than the naive implementation.

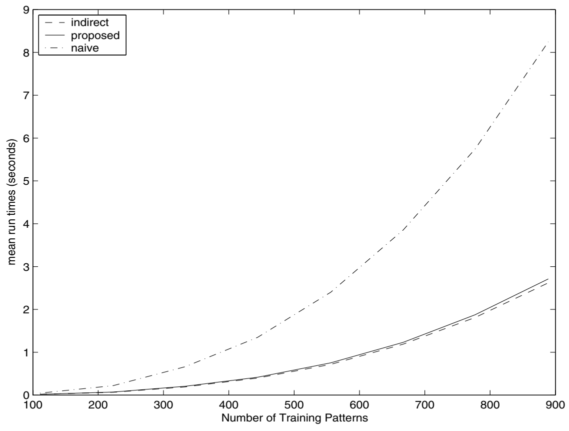

Figure 1. Run time of 10-fold cross validation vs number of training examples.

Second, we report the performances of KFD with model selection by cross-validation in the complete two stages on a suite of 13 data sets from UCI benchmark repository [4]. For model selection and performance evaluation, we use the Gaussian kernel

$$k(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{\sigma^2}}$$

and adopt the experimental procedure used in [11, 9], where 100 different random training and test splits[1] are defined (20

[1]The training and test splits are available from *http://ida.first.gmd.de/ raetsch/data/benchmarks.htm.*

in the case of *image* and *splice* datasets) and the first five training splits are used for model selection. The model is selected by minimizing the mean of five-fold cross-validation errors on the five training splits using grid search. The results are shown in Table 4 with a comparison with the results reported in [9].

Table 1. Performance of KFD with model selection by two-stage cross-validation comparing with the results reported in [9] on 13 benchmark data sets.

Data set	KFD[9]	KFD
Banana	10.8 ± 0.5	$\mathbf{10.5 \pm 0.4}$
B.Cancer	25.8 ± 4.6	$\mathbf{25.6 \pm 4.5}$
Diabetes	$\mathbf{23.2 \pm 1.6}$	23.4 ± 1.6
F.solar	$\mathbf{33.2 \pm 1.7}$	33.5 ± 1.5
German	23.7 ± 2.2	$\mathbf{23.5 \pm 2.2}$
Heart	16.1 ± 3.4	$\mathbf{15.9 \pm 3.5}$
Image	4.8 ± 0.6	$\mathbf{3.2\pm 0.7}$
Ringnorm	$\mathbf{1.5 \pm 0.1}$	1.6 ± 0.1
Splice	10.5 ± 0.6	$\mathbf{10.4 \pm 0.7}$
Thyroid	4.2 ± 2.1	$\mathbf{3.6 \pm 1.8}$
Titanic	23.2 ± 2.0	$\mathbf{22.6 \pm 1.0}$
Twonorm	2.6 ± 0.2	$\mathbf{2.4 \pm 0.1}$
Waveform	9.9 ± 0.4	$\mathbf{9.7 \pm 0.3}$

4 Concluding Remarks

An efficient evaluation algorithm is proposed for cross-validation of the complete two stages in KFD classifier formulation. Comparing with the existing indirect and approximate evaluation algorithms, the proposed algorithm requires some extra computations for evaluating 1D linear support vector machines on the extracted 1D features but it is more reliable since it is direct and exact. For very large data set, one may apply the incomplete Cholesky decomposition technique to further reduce the computational complexity.

Appendix: Derivations of (15)

First, we permute the kth group to be the last one by the following permutation matrix

$$P_k = \begin{bmatrix} I_{n_1} & & & & & & \\ & \ddots & & & & & \\ & & I_{n_{k-1}} & & & & \\ & & & & I_{n_{k+1}} & & \\ & & & & & \ddots & \\ & & & & & & I_{n_l} \\ & & & I_{n_k} & & & \end{bmatrix}. \tag{17}$$

Then

$$\hat{K} \triangleq P_k K P_k^T = \begin{bmatrix} K_{11} & K_{12} \\ K_{12}^T & K_{22} \end{bmatrix}$$
$$P_k\alpha = \begin{bmatrix} \alpha^{(k)} \\ \alpha_{(k)} \end{bmatrix}, P_k e = \begin{bmatrix} e^{(k)} \\ e_{(k)} \end{bmatrix}, P_k\hat{y} = \begin{bmatrix} \hat{y}^{(k)} \\ \hat{y}_{(k)} \end{bmatrix} \tag{18}$$

where $\alpha^{(k)}, e^{(k)}, \hat{y}^{(k)}$ are sub-vectors of $\alpha, e, \hat{y}$ by deleting $\alpha_{(k)}, e_{(k)}$ and $\hat{y}_{(k)}$ respectively , $K_{22} \in \mathbb{R}^{n_k \times n_k}$ is the kernel matrix of the examples in the kth group while $K_{11} \in \mathbb{R}^{(n-n_k)\times(n-n_k)}$ is the kernel matrix of the examples in all the other groups, and $K_{12} \in \mathbb{R}^{(n-n_k)\times n_k}$ with $\{K_{12}\}_{ij} = K(x_i, x_j)$ and x_j being in the kth group while x_i being in the other groups.

One can rewrite (7) as

$$\begin{bmatrix} 0 & 0 & \mathbf{1}^T & 0 & \mathbf{1}^T \\ 0 & -\mu I & K_{11} & 0 & K_{12} \\ \mathbf{1} & K_{11} & I & K_{12} & 0 \\ 0 & 0 & K_{12}^T & -\mu I & K_{22} \\ \mathbf{1} & K_{12}^T & 0 & K_{22} & I \end{bmatrix} \begin{bmatrix} b \\ \alpha^{(k)} \\ e^{(k)} \\ \alpha_{(k)} \\ e_{(k)} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \hat{y}^{(k)} \\ 0 \\ \hat{y}_{(k)} \end{bmatrix} \tag{19}$$

where (and hereafter) $\mathbf{1}$ and I denote all 1 vectors and identity matrices respectively with proper dimensions.

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

$$\begin{bmatrix} 0 & 0 & \mathbf{1}^T \\ 0 & -\mu I & K_{11} \\ \mathbf{1} & K_{11} & I \end{bmatrix} \begin{bmatrix} \tilde{b}_k \\ \tilde{\alpha}^{(k)} \\ \tilde{e}^{(k)} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \hat{y}^{(k)} \end{bmatrix}. \tag{20}$$

The validation error vector writes

$$\tilde{e}_{(k)} = \hat{y}_{(k)} - K_{12}^T\tilde{\alpha}^{(k)} - \tilde{b}_k\mathbf{1}_{n_k}. \tag{21}$$

Denote

$$A_{11} \triangleq \begin{bmatrix} 0 & 0 & \mathbf{1}^T \\ 0 & -\mu I & K_{11} \\ \mathbf{1} & K_{11} & I \end{bmatrix}, A_{12} \triangleq \begin{bmatrix} 0 & \mathbf{1}^T \\ 0 & K_{12} \\ K_{12} & 0 \end{bmatrix},$$
$$A_{22} \triangleq \begin{bmatrix} -\mu I & K_{22} \\ K_{22} & I \end{bmatrix} \tag{22}$$

and let $Z_1 \triangleq -K_{12}^T \tilde{e}^{(k)}$.

By (19) and the block inverse formula,

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{bmatrix}^{-1} = \begin{bmatrix} A_{11}^{-1} + A_{11}^{-1} A_{12} F_{22}^{-1} A_{12}^T A_{11}^{-1} & -A_{11}^{-1} A_{12} F_{22}^{-1} \\ -F_{22}^{-1} A_{12}^T A_{11}^{-1} & F_{22}^{-1} \end{bmatrix} \tag{23}$$

where $F_{22} = A_{22} - A_{12}^T A_{11}^{-1} A_{12}$, we have

$$\begin{bmatrix} 0 \\ \hat{y}_{(k)} \end{bmatrix} - A_{12}^T A_{11}^{-1} \begin{bmatrix} 0 \\ 0 \\ \hat{y}^{(k)} \end{bmatrix} = \begin{bmatrix} Z_1 \\ \tilde{e}_{(k)} \end{bmatrix} \tag{24}$$

and therefore

$$\begin{aligned} \begin{bmatrix} \alpha_{(k)} \\ e_{(k)} \end{bmatrix} &= F_{22}^{-1} \begin{bmatrix} Z_1 \\ \tilde{e}_{(k)} \end{bmatrix} \\ \begin{bmatrix} b \\ \alpha^{(k)} \\ e^{(k)} \end{bmatrix} &= A_{11}^{-1} \begin{bmatrix} 0 \\ 0 \\ \hat{y}^{(k)} \end{bmatrix} - A_{11}^{-1} A_{12} F_{22}^{-1} \begin{bmatrix} Z_1 \\ \tilde{e}_{(k)} \end{bmatrix} \\ &= \begin{bmatrix} \tilde{b}_k \\ \tilde{\alpha}^{(k)} \\ \tilde{e}^{(k)} \end{bmatrix} - A_{11}^{-1} A_{12} F_{22}^{-1} \begin{bmatrix} Z_1 \\ \tilde{e}_{(k)} \end{bmatrix} \end{aligned} \tag{25}$$

Compare (23) with (8) and note the permutation matrix P_k, one can verify that

$$F_{22}^{-1} = \begin{bmatrix} -B_k/\mu & C_k \\ C_k & B_k \end{bmatrix} + \frac{1}{d} \begin{bmatrix} \beta_{(k)} \\ \gamma_{(k)} \end{bmatrix} \begin{bmatrix} \beta_{(k)}^T & \gamma_{(k)}^T \end{bmatrix} \tag{26}$$

and

$$\begin{aligned} -A_{11}^{-1} A_{12} F_{22}^{-1} &= \begin{bmatrix} 0 & 0 \\ -\frac{1}{\mu} \bar{B}_k & \bar{C}_k \\ \bar{C}_k & \bar{B}_k \end{bmatrix} \\ &+ \frac{1}{d} \begin{bmatrix} -1 \\ \beta_{(k)} \\ \gamma_{(k)} \end{bmatrix} \begin{bmatrix} \beta_{(k)}^T & \gamma_{(k)}^T \end{bmatrix}. \end{aligned} \tag{27}$$

Hence, from the above two equations and (25), one can see that (15) holds with (Z_1, Z_2) being the solution of (16). □

References

[1] S. An, W. Liu, and S. Venkatesh. Fast exact cross-validation of least squares support vector machines. In *Proc. Of the Asia-Pacific workshop on visual information processing (VIP05)*. HongKong, China, 11-13, Dec. 2005.

[2] S. An, W. Liu, and S. Venkatesh. Fast cross-validation of kernel fisher discriminant classifiers. In *Proc. Of the fourth international conference on machine learning and applications (ICMLA05)*. Los Angeles, CA, USA, 15-17, Dec. 2005.

[3] F. R. Bach and M. I. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48, 2002.

[4] C. L. Blake and C. J. Merz. *UCI repository of machine learning databases [http://www.ics.uci.edu/ mlearn/MLRepository.html]*. Irvine, CA: University of California, Dept. of Information and Computer Science, 1998.

[5] G. C. Cawley and N. Talbot. Efficient leave-one-out cross-validation of kernel Fisher discriminant classifiers. *Pattern Recognition*, 36:2585–2592, 2003.

[6] O. Chapelle and V. Vapnik. Model selection for support vector machines. In *Advances in Neural Information Processing Systems*, 1999.

[7] R. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[8] T. Jaakkola and D. Haussler. Probabilistic kernel regression models. In *Proceedings of the 1999 Conference on AI and Statistics*, 1999.

[9] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller. Fisher discriminant analysis with kernels. In Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas, editors, *Neural Networks for Signal Processing IX*, pages 41–48. IEEE, 1999.

[10] M. Opper and O. Winter. Gaussian processes and SVM: Mean field and leave-one-out. In A. Smola, P. Barlett, B. Scholkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 311–326. MIT Press, 2000.

[11] G. Rätsch, T. Onoda, and K.-R. Muller. Soft margin for AdaBoost. *Machine Learning*, 42:287–320, 2001.

[12] Y. Su, T. M. Murali, V. Pavlovic, and S. Kasif. Training support vector machines in 1D. URL http://genomics10.bu.edu/yangsu/rankgene/one-svm.pdf, 2002.

[13] V. Vapnik and O. Chapelle. Bounds on error expectation for support vector machines. *Neural Computations*, 12(9), 2000.

[14] G. Wahba, Y. Lin, and H. Zhang. Generalized approximate cross-validation for support vector machines. In A. Smola, P. Barlett, B. Scholkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 297–309. MIT Press, 2000.

[15] J. Xu, X. Zhang, and Y. Li. Kernel MSE algorithm: a unified framework for KFD, LS-SVM and KRR. In *Proceedings of the international joint conference on Neural Networks*, pages 1486–1491. Washington DC, July 2001.

[16] Y. Zhao and C. K. Kwoh. Fast leave-one-out evaluation and improvement on inference for ls-svms. In *Proc. Of the 17th international conference on pattern recognition*, 2004.