Neurocomputing 73 (2010) 1501–1512

Contents lists available at ScienceDirect

Neurocomputing

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

Multiclass support vector classification via coding and regression

Pei-Chun Chen [a,*], Kuang-Yao Lee [b], Tsung-Ju Lee [c], Yuh-Jye Lee [d], Su-Yun Huang [e]

[a] Bioinformatics and Biostatistics Core, Research Center for Medical Excellence, National Taiwan University, 7F., No. 2, Syu-jhou Road, Taipei 10055, Taiwan, ROC
[b] Department of Statistics, Pennsylvania State University, USA
[c] Computer Science and Engineering, National Chiao-Tung University, Taiwan
[d] Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taiwan
[e] Institute of Statistical Science, Academia Sinica, Taipei, Taiwan

ARTICLE INFO

Article history:
Received 23 December 2008
Received in revised form
3 September 2009
Accepted 2 November 2009
Communicated by J. Tin-Yau Kwok
Available online 24 November 2009

Keywords:
Kernel map
Multiclass classification
Output code
Regularization
Support vector machine
Support vector regression

ABSTRACT

The multiclass classification problem is considered and resolved through coding and regression. There are various coding schemes for transforming class labels into response scores. An equivalence notion of coding schemes is developed, and the regression approach is adopted for extracting a low-dimensional discriminant feature subspace. This feature subspace can be a linear subspace of the column span of original input data or kernel-mapped feature data. The classification training and prediction are carried out in this feature subspace using a linear classifier, which lead to a simple and computationally light but yet powerful toolkit for classification. Experimental results, including prediction ability and CPU time comparison with LIBSVM, show that the regression-based approach is a competent alternative for the multiclass problem.

1. Introduction

Over the last decade, there have been a great interest and successful development of support vector machines (SVMs) for classification [7,5,11]. SVMs are originally designed for binary classification. There are two commonly seen multiclass extensions for SVMs. One is the composition type methods built upon a series of binary classifiers, e.g., the one-against-one, one-against-rest and error correcting output codes [12,1,8,10,18], and the other is the single machine type methods, often huge and solved in one optimization formulation [37,4,9,24,28]. Comparison studies and discussions on compositions of binary classifiers and single machine approaches can be found in [22,33]. Based on their findings, there is no universally dominant classification rule for the multiclass problems, and different methods have their own merits and advantages. Thus, it allows the room for exploration of alternative approaches.

In this article, we propose an alternative approach based on regression concept. The time-honored Fisher linear discriminant analysis (FLDA) separates data classes by projecting input attributes to the eigen-space of the between-class covariance (scaled by the within-class covariance). In the binary classification case, FLDA can be solved via a multiresponse regression [14,30,2] by encoding the binary class labels into numeric responses $\{n_1/n, -n_2/n\}$, where n_1 and n_2 are class sizes and $n=n_1+n_2$. Hastie et al. [20] have further extended the FLDA to the nonlinear and multiclass classification via a penalized regression setting, and they named it the "flexible discriminant analysis (FDA)". The FDA is based on encoding the class labels into response scores and then a nonparametric regression technique, such as MARS (multivariate additive regression splines), neural networks, or else, is used to fit the response scores. A particular encoding scheme "optimal scoring" is proposed in FDA. Later [31,34] have adopted the same FDA approach, but have replaced the conventional nonparametric regression technique with a kernel trick, and their approach is named kernel discriminant analysis (KDA). The KDA [34] is solved by an EM algorithm.

Inspired by the above-mentioned regression approaches and the successful development of SVMs, we take both ideas and adopt the multiresponse support vector regression (mSVR) for the multiclass classification problem. Some preceding works on support vector classification can also be interpreted as ridge regression applied to classification problems, e.g., the proximal SVM [16,17] and the regularized least squares SVM [36]. Our mSVR for classification consists of three major steps. The first is to encode the class labels into multiresponse scores. Next, the

* Corresponding author.
E-mail addresses: d93842005@ntu.edu.tw (P.-C. Chen), kxl280@psu.edu (K.-Y. Lee), freeman1217@gmail.com (T.-J. Lee), yuh-jye@mail.ntust.edu.tw (Y.-J. Lee), syhuang@stat.sinica.edu.tw (S.-Y. Huang).

0925-2312/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2009.11.005

regression fit of scores on kernel-transformed feature inputs is carried out to extract a low-dimensional discriminant feature subspace. The final step is a classification rule to transform (i.e., to decode) the mapped values in this low-dimensional discriminant subspace back to class labels. The standard kernel Fisher discriminant analysis and also SVM solve the classification problems on a high-dimensional feature space. Through the mSVR, we can extract the information of the attributes into a $(J-1)-$dimensional feature subspace (J is the number of classes), which can accelerate the speed of classification training and decrease the influence due to noise. We will give a unified view of different coding schemes. The low-dimensional discriminant feature subspace generated by different coding schemes with long enough code length will be identical, which introduces the notion of equivalence of codes. We will prove this equivalence theoretically and also confirm it by our numerical experiments. The regression step can be viewed as a feature extraction to make the final classification (decoding) step computationally light and easy. The nonlinear structure between different classes of data patterns will be embedded in the extracted features because of the kernel transformation. Thus, we can apply any feasible linear learning algorithm to the data images in this low-dimensional discriminant feature subspace. Numerical tests and comparisons show that our framework, regression setup combined with a linear discriminant algorithm, is an easy and yet competent alternative to the multiclass classification problem.

The rest of the article is organized as follows. In Section 2, we introduce the framework of mSVR for classification. We describe the major steps of our regression approach including kernelization, encoding the class labels, a regularized least-square-based mSVR algorithm and the principle for decoding. In Section 3, we develop some notions and properties of equivalent codes and scores in the discriminant analysis context. In Section 4, implementation issues including model selection and the choice of base classifiers are discussed. Experimental results are provided to demonstrate the efficiency of our proposal and to illustrate numerical properties of different coding schemes. Concluding remarks are in Section 5. All proofs are in the Appendix.

2. Classification by multiresponse regression

Consider the problem of multiclass classification with J classes based on d measurements of input attributes $x \in \Re^{d\times 1}$. Denote the membership set by $\mathcal{J}=\{1,2,\ldots,J\}$ and each individual membership by $g \in \mathcal{J}$. Suppose we have training data $\{(x_i,g_i) \in \Re^{d\times 1} \times \mathcal{J}\}_{i=1}^{n}$. Our goal is to construct a classification rule which, given a new input, can correctly predict the associated class label of this new input. Aside from various support vector approaches mentioned in Section 1 originating from the machine learning community, regression-based methods for classification have some long history in statistical literature [14,30,20,2]. The two articles [20,19] have an in-depth discussion of using multiresponse regression for discriminant purpose. In general, the regression approach for classification consists of three major steps: encoding, linear regression and decoding. Nonlinear extension can be done by conventional adaptive nonparametric regression strategies, for instance, MARS [15], or by the kernel trick [34]. Hastie et al. [20] have used a particular scoring scheme, namely the optimal scoring, for encoding class labels. For regression-based classification, it is natural to ask if different coding schemes for transforming class labels into response scores lead to different classification results. To answer this question, we use general output codes to encode the class labels and give an equivalence criterion on coding and scoring schemes. Also, we adopt the kernel trick for nonlinear extension and employ mSVR algorithm for solving the regression problem. The mSVR is computationally more feasible than traditional nonparametric regression techniques. The regression step aims to extract low-dimensional discriminant features. Then, the classification training and prediction can be done in this low-dimensional feature subspace spanned by the regression coefficients variates. The classification prediction is a decoding step by pulling the fitted response at a test point back to a class label. Any reasonable linear algorithm can be used as a base classifier for decoding. We have tested the hybrid of mSVR with FLDA, linear smooth support vector machine (SSVM) [26] and linear Lagrangian SVM [29]. Our regression-based classification is invariant to encoding. More details on the equivalence and invariance of coding schemes will be given in Section 3.

Below we start with a summary of our regression setup, followed by encoding schemes, the extraction of a low-dimensional discriminant subspace and the criterion of decoding for classification prediction.

2.1. Regression framework: linear and kernel generalization

For the i th input observation x_i with membership $g_i \in \mathcal{J}=\{1,2,\ldots,J\}$, an indicator multiresponse score row vector, $y_i \in \Re^{1\times J}$, is formed as follows:

$$y_i=[y_{i1},\ldots,y_{iJ}], \quad \text{where } y_{ij}=\begin{cases}1, & j=g_i,\\ 0, & j\neq g_i.\end{cases} \tag{1}$$

Combine these n indicator multiresponses into an $n\times J$ score matrix

$$Y=\begin{bmatrix}y_1\\ \vdots\\ y_n\end{bmatrix}=\begin{bmatrix}y_{11} & \cdots & y_{1J}\\ \vdots & & \vdots\\ y_{n1} & \cdots & y_{nJ}\end{bmatrix}.$$

Also gather the input attributes, $x_1,\ldots,x_n$, into an $n\times d$ matrix $A=[x_1,\ldots,x_n]'$. The j th column of Y, denoted by Y_j, partitions the training data into class j and the rest. A multivariate linear regression, of multiresponses Y on input data matrix A with additive intercept, is considered. That is, $y=[y_1,\ldots,y_J]$ is modeled by regression function $\eta(x)=[\eta_1(x),\ldots,\eta_J(x)]$ with the j th response given by

$$\eta_j(x)=x'w_j+b_j, \quad j=1,\ldots,J, \tag{2}$$

where $w_j \in \Re^{d\times 1}$ is the normal vector of the j th regression plane and b_j is a scalar offset. Let $W=[w_1,\ldots,w_J] \in \Re^{d\times J}$ and let $\mathcal{C}(W)$ denote the column space spanned by $\{w_1,\ldots,w_J\}$. We name columns of W the *regression coefficients variates*. By fitting the multiresponse linear regression, we get an estimate of the column space $\mathcal{C}(\hat{W})$, named the *fitted discriminant subspace*. Training inputs (rows of A) are mapped to this fitted subspace for discriminant purpose. Generating the classification rules in this fitted discriminant subspace is the decoding step.

In many real applications, data are complicated and cannot be well separated by hyperplanes, and nonlinear methods are called for. Inspired by the idea of support vector methods, the input measurements can be transformed into a high dimensional feature Hilbert space via a certain positive definite kernel. By the kernel trick, kernel values can be seen as inner products of the training input images in the high dimensional feature space. The most popular kernel function is the Gaussian kernel given by $\kappa(x,x_i)=\exp(-\gamma\|x-x_i\|_2^2)$, where γ is the width parameter, which controls the similarity measure of x_i and x. (We do not care to normalize κ here to make the kernel integrate to one, as the normalization has no effect on support vector classification.) The selection of γ should be carefully done to avoid the phenomenon of overfitting or underfitting. The response surfaces in the feature

space become

$$\eta_j(x) = \kappa(x,A)w_j + b_j, \quad j = 1,\ldots,J, \tag{3}$$

where $w_j = [w_{1j}, w_{2j}, \ldots, w_{nj}]'$ and $\kappa(x,A)w_j := \sum_{i=1}^{n} \kappa(x,x_i)w_{ij}$.

In large data problems there are two major computational difficulties. One is the huge memory space required to store the large kernel data. The other is the complicated mathematical programming problem encountered in solving the regression problem. For large data problems, the reduced kernel technique [26,25] is adopted by replacing the full kernel matrix by a reduced kernel approximation. In the reduced kernel framework the response surfaces are given by

$$\eta_j(x) = \kappa(x,\tilde{A})w_j + b_j, \ \ j = 1,\ldots,J, \tag{4}$$

where $\tilde{A}$ is a reduced set consisting of partial observations, say $\tilde{n}$ many, from A to form a reduced regression model: that is, for the j th response surface

$$\eta_j(x) = \sum_{\tilde{x}_i \in \tilde{A}} \kappa(x,\tilde{x}_i)w_{ij} + b_j, \quad w_j = [w_{1j}, w_{2j}, \ldots, w_{\tilde{n}j}]'.$$

The reduced kernel approximation is to adopt a compressed model, while it still uses the entire data for model fitting. From now on and throughout the rest of the article, we use, for notation simplicity, a unified notation $\tilde{K}$ to represent either the reduced kernel data matrix

$$\kappa(A,\tilde{A}) = [\kappa(x_i,\tilde{x}_j)]_{x_i \in A, \tilde{x}_j \in \tilde{A}} \text{ of size } n \times \tilde{n},$$

or the raw data matrix A of size $n \times d$ (with $\tilde{n} = d$), or the full kernel matrix

$$\kappa(A,A) = [\kappa(x_i,x_j)]_{x_i,x_j \in A} \text{ of size } n \times n \text{ (with } \tilde{n} = n).$$

Likewise, we can write a unified program for implementing linear and nonlinear, as well as full and reduced kernel, regression models. All we need to do is to change the input data matrix $\tilde{K}$.

We have used the indicator output variables for class labels (1). The indicator responses are of dimensionality J. In Section 3 we will further introduce other alternative scoring schemes to encode the class labels into multiresponses of dimensionality, say, k. The response surface model (4) is still valid, but with a different dimensionality k instead of J, i.e., $\eta(x) = [\eta_1(x), \ldots, \eta_k(x)]$. Model (4) is a special case with $k = J$.

2.2. Regularized least-squares support vector regression

In this subsection we introduce the regularized least-squares (RLS) approach [16,35] for mSVR. Let k be the code length of a scoring schemes to encode the class labels. In the classical regression approach, we often estimate the regression coefficients by least-squares for simple linear systems or by regularized least-squares for complex linear systems. The RLS for mSVR in this article is similar to the multicategory proximal SVM [16], but without the modification for unbalancedness of class sizes. Later, in the discussion of equivalence of codes, centering for code columns is done by subtracting from each column a weighted average of code rows, where the weights have taken class sizes into account, and thus there is no need for further modification of unbalancedness. The classification rule in proximal SVM is a softmax-like rule, i.e., a test input is assigned to the class so that the point lies deepest (i.e., lies farthest away from the separating hyperplanes, or has the largest fitted value). Our classification rule can be flexible choices among many possible candidate discriminant algorithms. Here we specifically have experimented with the FLDA and SSVM in a later numerical study.

The main idea of least-squares approach is to minimize the squared errors of regression. The sum-of-squared residuals for multiresponse regression is given by

$$SSR(W,b) = \sum_{j=1}^{k} \|Y_{\cdot j} - \tilde{K}w_j - b_j\mathbf{1}_n\|_2^2,$$

where $\mathbf{1}_n$ denotes the column vector of ones with length n and $\|Y_{\cdot j} - \tilde{K}w_j - b_j\mathbf{1}_n\|_2^2 = \sum_{i=1}^{n}(y_{ij} - \kappa(x_i,\tilde{A})w_j - b_j)^2$. The unique solution to the optimization

$$\min_{W \in \Re^{\tilde{n}\times k}, b \in \Re^{1\times k}} SSR(W,b)$$

is usually good enough. However, if columns of the matrix $\tilde{K}$ are highly correlated, then W will be poorly determined and exhibit high variation. It is due to that the information matrix $[\tilde{K}\ \mathbf{1}_n]'[\tilde{K}\ \mathbf{1}_n]$ is ill-conditioned. This situation often happens in kernel data. The ridge regression is a method to resolve this kind of problem. It shrinks the regression coefficients by imposing a penalty on their norm. The regression coefficients are derived from the following regularized least-squares problem[1]

$$\min_{W \in \Re^{\tilde{n}\times k}, b \in \Re^{1\times k}} \sum_{j=1}^{k} \left\{ \frac{C}{2}\|Y_{\cdot j} - \tilde{K}w_j - b_j\mathbf{1}_n\|_2^2 + \frac{1}{2}\|w_j\|_2^2 \right\}. \tag{5}$$

Notice that each pair of w_j and b_j can be determined independently from the rest. The problem (5) can be decomposed into k regularized least-squares subproblems, and each one solves for one individual w_j and b_j. These k subproblems share a common information matrix on the left hand side of the normal equations and only differ on the right hand side of the equations. Thus, we only need to do one time of any direct method for solving the linear systems of equations. In contrast to the multicategory proximal SVM [16], which requires to run k times of the direct method for solving the linear systems, our version of RLS-mSVR in Eq. (5) needs only *one run time* of a same-sized linear system.

2.3. Decoding and classification rules

We have introduced the indicator coding scheme (1) for encoding the class labels into multivariate scores. In our regression setting, the scores Y are then rendered to a regression fit on input data $\tilde{K}$. The regression coefficients $\hat{W} = [\hat{w}_1, \hat{w}_2, \ldots, \hat{w}_J] \in \Re^{\tilde{n}\times J}$ solved by RLS-mSVR provide a way of mapping input data $\tilde{K}$ of dimensionality $\tilde{n}$ to a low-dimensional discriminant feature subspace of dimensionality Rank $(\hat{W}) \le J$. The decoding step is to map the fitted scores back to class labels, which can be done via incorporating a classifier, such as FLDA or SSVM. For indicator scores, the classification of a new test input x by the column with the largest fitted value is the so called "*softmax*". In an equal-sized binary classification, softmax is exactly the same as FLDA. However, when the class number is larger than two, softmax does not work as well as FLDA due to the unbalanced class sizes and the nonspherical data distribution. Also softmax does not apply to nonindicator scores. Therefore, we need a more general and accurate classification rule to convert images in the discriminant feature subspace back to class labels. Below we describe a general principle for choice of discriminant rules. After extracting the regression coefficient variates $\hat{W}$, kernel data $\tilde{K}$ are then projected onto the subspace $\mathcal{C}(\hat{W})$. Denote the kernel data projection by $P_{\mathcal{C}(\hat{W})}(\tilde{K})$. A general principle for classification is to pick a linear discriminant algorithm to act on the kernel

[1] For the full kernel case, a direct ridge-type regularization, though solves the ill-condition problem, leads to an inferior result due to the phenomenon that full kernel matrix has much lower effective rank than its size. Thus, our suggestion is to preprocess the kernel data by a principal component analysis for dimension reduction before entering the regression step. In this case, $\tilde{K}$ in (5) is a reduced kernel by PCA instead of by random subset.

data projection:

Any reasonable linear discriminant algorithm on $P_{\mathcal{C}(\hat{W})}(\tilde{K})$ will do, (6)

where $P_{\mathcal{C}(\hat{W})}(\tilde{K}) = \tilde{K}\hat{W}_{\text{on}}$ with $\hat{W}_{\text{on}}$ the orthonormalized columns of $\hat{W}$. In this article we choose FLDA and linear SSVM as our base algorithms for numerical experimental demonstration. FLDA assigns a test instance to the class of closest centroid. Distance measure used in FLDA is the Mahalanobis distance. The FLDA is also the Bayes rule with the uniform prior on class probabilities. The FLDA can further extend to a general Bayes rule, which incorporates prior class probabilities π_j to produce posterior probability outputs given by

$$prob(g_j|x) = \frac{\pi_j \eta_j(x)}{\pi_1\eta_1(x) + \cdots + \pi_J\eta_J(x)}, \quad j = 1, \ldots, J. \tag{7}$$

As for SSVM, it is one of many SVM variants, which focuses on the primal problems and solves them directly by smoothing techniques. A Newton–Armijo algorithm is developed to solve the smooth optimization problem. SSVM implementation code takes the advantage of sparsity of the Hessian matrix and has used the limit values of the sigmoid function for smooth approximation in computing the gradient vector and the Hessian matrix. We refer the reader to the original SSVM article [27] for details. Note that the choice of linear discriminant algorithms is not limited to FLDA and SSVM. We have also tried the Lagrangian SVM [29] on benchmark data sets used in Section 4, which has the same formulation as SSVM but solved in the dual space instead of in the primal. The resulting prediction accuracies of Lagrangian SVM are same as those of SSVM but with a bit longer CPU run time, and thus are omitted from the numerical report.

3. Encoding and equivalence class of codes

In this section we introduce several existing coding schemes to encode the class labels. We also unify them under the notions of equivalence of codes and scores in the context of discriminant analysis. We refer the reader to [21] for general theory of discrete codes and to [8,10] for continuous codes.

3.1. Coding and scoring schemes

For an arbitrary matrix $\Theta \in \Re^{J\times k}$, it can be used as a code matrix, where each row corresponds to a class in $\mathcal{J} = \{1, \ldots, J\}$ and each column defines an one-dimensional discriminant subspace for partitioning the class membership set $\mathcal{J}$. Below we introduce several coding and scoring schemes commonly seen in the literature of statistics and/or coding theory.

3.1.1. Discrete codes

Indicator code. We have introduced the encoding of each class label in $\{1, \ldots, J\}$ by a J-dimensional indicator score (1). This coding scheme has been commonly used in literature and is probably the simplest and the most convenient way of encoding the class labels into multivariate indicator output variables. The corresponding code matrix is simply the $J \times J$ identity matrix, denoted by $\Theta_{IS} = I_J$, whose j th row is used to represent the j th class and whose j th column is used to separate class j from the rest. We use the notation Y_{IS}, of size $n \times J$, for corresponding indicator scores (IS) of observational class labels. Notice that rows of Y_{IS} are copies of rows from the code matrix Θ_{IS}.

One-vs-baseline code. This coding scheme picks a baseline class and the rest are contrasted with this baseline class. Without loss of generality, take J as the baseline class. For each $j = 1, \ldots, J-1$, we encode the class label j by a row vector given by $[0, \ldots, 0, 1, 0, \ldots, 0] \in \Re^{1\times(J-1)}$, where the nonzero entry is in the j th place; for $j = J$ we encode the class label J by a row vector of all "-1" as $[-1, \ldots, -1] \in \Re^{1\times(J-1)}$. Denote this code matrix by Θ_{1B}, which is of size $J \times (J-1)$. The j th column of Θ_{1B} contrasts the j th class with the baseline class J. The corresponding response scores are given by $Y_{1B} = Y_{IS}\Theta_{1B}$, which are copies of rows from the code matrix Θ_{1B}.

3-Way code. Consider a code matrix of size $J \times k$ with $k = J(J-1)/2$, whose columns are constructed as follows. For any pair of $j, j' \in \mathcal{J}$ with $j < j'$, a J-column-vector is formed with 1 in the j th place, -1 in the j' th place and 0's elsewhere. Denote this code matrix by Θ_{3W} and the corresponding scores by Y_{3W}, i.e., $Y_{3W} = Y_{IS}\Theta_{3W}$.

Exhaustive code. It looks for a matrix of size $J \times k$ with entries 0 or 1 to encode the class membership set $\mathcal{J}$. There are 2^J different ways of constructing J-column-vector using 0 and 1. However, complements, e.g., "0001" and "1110", are considered the same, and the column of all zeros or all ones is considered useless. Then, dividing 2^J by 2 and subtracting 1, there are $k = (2^{J-1} - 1)$ many ways of constructing useful and different columns. Denote the code matrix by Θ_{EX} and the corresponding scores by Y_{EX}.

Random code. For a random code with a given length k, it is a $J \times k$ matrix with each entry uniformly sampled from $\{-1, 0, 1\}$ (or, more generally, from a finite field).

3.1.2. Continuous codes

Codes considered above are discrete codes defined over the finite field $\{-1, 0, 1\}$. Below we will introduce continuous codes over $\Re$.

Balanced code. This coding scheme takes the class sizes into account for its name. For a binary problem, this scoring scheme is well-known and can be used in a regression setup for classification [14,2]. Here we give an extension for this scoring scheme to the multiclass setting. For each $j = 1, \ldots, J-1$, we encode the class label j by a row vector given by $[0, \ldots, 0, n/n_j, 0, \ldots, 0] \in \Re^{1\times(J-1)}$, where the nonzero entry is in the j th place; for $j = J$ we encode the class label J by a row vector of all "$-n/n_J$" as shown below

$$[-n/n_J, \ldots, -n/n_J, \ldots, -n/n_J] \in \Re^{1\times(J-1)}.$$

Denote the code matrix by Θ_{BS}, which is of size $J \times (J-1)$. We call the J th class the baseline. However, it does not matter in terms of the equivalence notion given later in Definition 1 what class has been chosen as the baseline. The j th column of Θ_{BS} contrasts class j with class J and balances their sizes. Likewise, the j th row is used to encode the j th class. The corresponding balanced scores (BS) of observational class labels are given by $Y_{BS} = Y_{IS}\Theta_{BS}$. Again, rows of Y_{BS} are copies of rows from Θ_{BS}.

Optimal scoring. For a given dimensionality k, consider the collection of all k-dimensional linear scores given by $\mathcal{S}_k := \{Y_{IS}\Theta : \Theta \in \Re^{J\times k}\} \subset \Re^{n\times k}$, $k < J$. The optimal scoring [20] is to search for linear scoring scheme in $\mathcal{S}_k$ that minimizes the average squared residuals (ASR) by regression of scores on input attributes subject to zero mean, unitary variance and being uncorrelated. That is, the scheme is to find $Y \in \mathcal{S}_k$ with constraints $\mathbf{1}_n' Y = \mathbf{0}_k'$ (zero mean) and $Y'Y/n = I_k$ (unitary variance and being uncorrelated), that solves the following minimization problem:

$$\min_{Y \in \mathcal{S}_k, \mathbf{1}_n' Y = \mathbf{0}_k', Y'Y/n = I_k} ASR.$$

In the mSVR context, the ASR translates to

$$ASR := \min_{w_j \in \Re^q, b_j \in \Re} \frac{1}{n}\sum_{j=1}^{k} \|Y_j - \tilde{K}w_j - b_j\mathbf{1}_n\|_2^2, \tag{8}$$

where $Y_j = [y_{1j}, y_{2j}, \ldots, y_{nj}]'$ is the j th column of scores Y. Let $\mathbf{1}_k$ and $\mathbf{0}_k$ denote, respectively, a column vector of ones and a column

vector of zeros with length k. Let $D=\mathrm{diag}(n_1,n_2,\ldots,n_J)$ be a diagonal matrix and $\mathbf{e}_J=D\mathbf{1}_J=[n_1,n_2,\ldots,n_J]'$. Notice that for any score matrix $Y\in\mathcal{S}_k$, we have

$$\mathbf{1}_n' Y=\mathbf{1}_n' Y_{IS}\Theta=(n_1,n_2,\ldots,n_J)\Theta=\mathbf{e}_J'\Theta=\mathbf{1}_J' D\Theta. \tag{9}$$

The minimal ASR in (8) occurs at the least squares fit, then the optimal scoring problem converts to

$$\min_{\Theta\in\Re^{J\times k}}\frac{1}{n}\mathrm{tr}\{\Theta' Y_{IS}'(I_n-P_{\tilde{K}})Y_{IS}\Theta\}\quad \text{s.t. } \mathbf{e}_J'\Theta=\mathbf{0}_k\quad\text{and}\quad \Theta' D\Theta/n=I_k, \tag{10}$$

where $P_{\tilde{K}}=\tilde{K}(\tilde{K}'\tilde{K})^{-}\tilde{K}'$ with $(\tilde{K}'\tilde{K})^{-}$ a generalized inverse. We denote the resulting Θ by Θ_{OS}, which is a $J\times k$ code matrix, and denote the corresponding optimal scores (OS) of observational class labels by Y_{OS}, i.e., $Y_{OS}=Y_{IS}\Theta_{OS}$ of size $n\times k$. Again, rows of Y_{OS} are copies of rows from Θ_{OS}. The first column of Θ_{OS} leads to the optimal one-dimensional discriminant feature subspace, where classes of data projections have the best separation, i.e., the least overlap. The second column leads to the second optimal one-dimensional discriminant feature subspace, and so on. This nice property of column ranking provides a further dimension reduction by using only leading columns. In particular, when there is a limitation placed on the code length, the optimal scoring will be superior to other coding schemes.

3.2. Equivalence class of codes

In this subsection, we give a unified view of the above-mentioned codes and scores by developing some notions of equivalence. We will show that all the coding and scoring schemes introduced in Section 3.1 are equivalent in the context of discriminant analysis. We will also show that, fixing a linear discriminant algorithm for decoding, equivalent codes and scores will lead to exactly the same classification results for class membership assignment. The choice of underlying linear discriminant algorithms can be general, such as FLDA, Bayes rules, linear SVMs, etc. First, we define the equivalency of two code or score matrices.

Definition 1. Two code matrices $\Theta_1\in\Re^{J\times k_1}$ and $\Theta_2\in\Re^{J\times k_2}$ are said to be equivalent if and only if $\mathcal{C}(Q\Theta_1)=\mathcal{C}(Q\Theta_2)$, where $\mathcal{C}(\cdot)$ denotes the linear space spanned by columns of the underlying matrix, and where $Q=I_J-(1/n)\mathbf{1}_J\mathbf{e}_J'$. Similarly, two score matrices $Y_1\in\Re^{n\times k_1}$ and $Y_2\in\Re^{n\times k_2}$ are said to be equivalent if and only if $\mathcal{C}(RY_1)=\mathcal{C}(RY_2)$, where $R=I_n-(1/n)\mathbf{1}_n\mathbf{1}_n'$.

In other words, if two centered code (or score) matrices span the same column space, these two coding (or scoring) schemes are equivalent, and vice versa. Note that $(1/n)\mathbf{1}_n\mathbf{1}_n'$, Q and R are all projection matrices. Also note that the centering of code columns is a weighted centering using weights $(n_1/n,\ldots,n_J/n)$. Define the inner products of the column space of codes and scores, respectively, by

$$\langle\theta_1,\theta_2\rangle_c:=\theta_1' D\theta_2,\quad \theta_1,\theta_2\in\Re^J;\quad \langle y_1,y_2\rangle_s:=y_1' y_2,\quad y_1,y_2\in\Re^n, \tag{11}$$

where $D=\mathrm{diag}(n_1,\ldots,n_J)$. Let $\Theta\in\Re^{J\times k}$ be an arbitrary code matrix and $Y=Y_{IS}\Theta$ be the corresponding scores of observational class labels. Let $Y_{\cdot j}$ and $\Theta_{\cdot j}$ be the j th columns of Y and Θ, respectively. Then, $\langle\Theta_{\cdot j},\Theta_{\cdot j'}\rangle_c=\Theta_{\cdot j}' D\Theta_{\cdot j'}=\Theta_{\cdot j}' Y_{IS}' Y_{IS}\Theta_{\cdot j'}=\langle Y_{\cdot j},Y_{\cdot j'}\rangle_s$. That is, the transformation between codes and scores will preserve their inner products. Moreover, since that $\mathbf{1}_n' Y_{\cdot j}=\mathbf{1}_n' Y_{IS}\Theta_{\cdot j}=\mathbf{e}_J'\Theta_{\cdot j}=\mathbf{1}_J' D\Theta_{\cdot j}$, the orthogonality of $\mathbf{1}_n$ to $Y_{\cdot j}$ is equivalent to the orthogonality of $\mathbf{1}_J$ to $\Theta_{\cdot j}$ under the inner products given by (11).These properties are summarized in the following Proposition.

Proposition 1. *For an arbitrary code matrix $\Theta\in\Re^{J\times k}$ and its corresponding scores of observational class labels $Y=Y_{IS}\Theta$, columns of Θ and Y are inner product preserving, and $\Theta_{\cdot j}$ and $\Theta_{\cdot j'}$ are orthogonal if and only if $Y_{\cdot j}$ and $Y_{\cdot j'}$ are orthogonal. Furthermore, $\mathbf{1}_n$ is orthogonal to $Y_{\cdot j}$ if and only if $\mathbf{1}_J$ is orthogonal to $\Theta_{\cdot j}$. The orthogonality here is with respect to the inner products given by* (11).

By Proposition 1, the centering of the code matrix Θ and the centering of the score matrix Y are equivalent. Therefore, the optimal scoring problem (10) is equivalent to the following generalized eigenvalue problem:

$$\max_{\Theta\in\Re^{J\times k}}\mathrm{tr}\{\Theta' Y_{IS,c}' P_{\tilde{K}} Y_{IS,c}\Theta\}\text{ subject to }\Theta' D\Theta/n=I_k, \tag{12}$$

where $Y_{IS,c}$ are the centered indicator scores. This simplifies the optimization problem (10) by removing the constraint $\mathbf{e}_J'\Theta=\mathbf{0}_k$.

Notice that both the score matrices Y_{BS} and Y_{OS} and their associated code matrices Θ_{BS} and Θ_{OS} are already centered to $\mathbf{1}_n$ and $\mathbf{1}_J$, respectively. By Definition 1, it is obvious that Proposition 2 below holds.

Proposition 2. *Given two code matrices $\Theta_1\in\Re^{J\times k_1}$ and $\Theta_2\in\Re^{J\times k_2}$ such that $\mathcal{C}(\Theta_1)=\mathcal{C}(\Theta_2)$, then Θ_1 and Θ_2 are equivalent. Similarly, if $\mathcal{C}(Y_1)=\mathcal{C}(Y_2)$, then Y_1 and Y_2 are equivalent.*

From Proposition 2, any two code matrices (or score matrices) spanning the same column space are equivalent. Definition 1 gives a more general equivalence class (in terms of central column span) than Proposition 2. Notice that the column centerization for codes, i.e., making its column space orthogonal to $\mathbf{1}_J$, depends on the class sizes $(n_1,\ldots,n_J)$ and so does the equivalence class of codes, i.e., the definition and propositions of equivalence are sample properties taking class sizes into account for balancedness. Also notice that, by letting $Y=Y_{IS}\Theta$, we have $D^{-1}Y_{IS}' Y=\Theta$. This gives the interplay of codes and scores.

Proposition 3. *Let $\Theta_1\in\Re^{J\times k_1}$ and $\Theta_2\in\Re^{J\times k_2}$ be two code matrices and let $Y_1=Y_{IS}\Theta_1$ and $Y_2=Y_{IS}\Theta_2$ be the corresponding scores, respectively. If Θ_1 and Θ_2 are equivalent, then $\mathcal{C}(RY_1)=\mathcal{C}(RY_2)$, i.e., Y_1 and Y_2 are equivalent. Conversely, if Y_1 and Y_2 are equivalent, then Θ_1 and Θ_2 are equivalent as well.*

Theorem 4. *Let $\Theta_1\in\Re^{J\times k_1}$ and $\Theta_2\in\Re^{J\times k_2}$ be two equivalent code matrices and let $Y_1=Y_{IS}\Theta_1$ and $Y_2=Y_{IS}\Theta_2$ be the corresponding scores, respectively. By ridge regression* (5) *of Y_1 and Y_2, respectively, on $\tilde{K}$, we get that $\mathcal{C}(\hat{W}_1)=\mathcal{C}(\hat{W}_2)$. That is, equivalent codes will lead to the same linear discriminant feature subspace. Thus, given a choice of base linear discriminant algorithm, equivalent codes Θ_1 and Θ_2 (or equivalent scores Y_1 and Y_2) will lead to the same classification results of class membership assignment.*

Theorem 4 says that, fixing a choice of linear algorithm, the resulting classifier is the same regardless of coding schemes as long as they span the same central column space $\mathcal{C}(Q\Theta)$.

Theorem 5. *The indicator, one-vs-baseline, 3-way, exhaustive, optimal, and balanced codes in their respective full lengths are all equivalent in the sense of Definition* 1, *and hence Theorem* 4 *applies to them. In other words, given a linear discriminant rule, all the above-mentioned coding schemes lead to the same multiclass classifier.*

Remark 1. An all-ones code column, $\mathbf{1}_J$, or an all-ones score column, $\mathbf{1}_n$, is considered useless in classification. Thus, the act of centering will not change the classification result. By Definition 1, the code matrix centering is a weighted average over rows and the weights depend on sample data class sizes. By Proposition 1, the centered linear scores are orthogonal to $\mathbf{1}_n$ and the centered codes are orthogonal to $\mathbf{1}_J$. For centered scores, the common grand mean is subtracted from the scores. A noncentered score matrix

Y_{nc} can be transformed to a centered one by $Y_c = (I_n - \mathbf{1}_n\mathbf{1}_n'/n)Y_{nc}$. To augment a centered score matrix Y_c to a noncentered one, it can be done by $Y_{nc} = [Y_c\ \mathbf{1}_n]$. Similarly, a noncentered code matrix Θ_{nc} can be transformed to a centered one by $\Theta_c = (I_n - \mathbf{1}_J\mathbf{e}_j'/n)\Theta_{nc}$. To augment a centered code matrix Θ_c to a noncentered one, it can be done by $\Theta_{nc} = [\Theta_c\ \mathbf{1}_J]$.

Remark 2. Though all the scoring schemes in their full lengths discussed above are equivalent, the optimal scoring provides an extra feature of easy dimension reduction. As the optimal scoring is obtained by solving a generalized eigenvalues problem, its columns are ranked according to descending eigenvalues and are uncorrelated. Dropping one or a few the right most columns from the score matrix Y_{OS} will not dramatically change the classification accuracy. On the contrary, dropping one column from Y_{BS}, or two columns from Y_{IS} (IS has one redundant column from the equivalence point of view), will fail to classify a particular pair of classes. Some pictorial illustrations can be found in Section 4. However, if we are allowed to use long enough code length, the IS and BS codes are the simplest to implement and to interpret, and they have the same prediction ability as that of OS code. Moreover, longer codes, such as 3-way code, exhaustive code and random codes with code length $k > J$, have the extra ability of error correcting.

4. Experimental study

Data sets and experimental setting. The following data sets are used for experimental study: ionosphere, Iris, wine, glass, segment, image, dna, satimage and pendigits, where ionosphere, Iris, wine, glass, image and pendigits are from the UCI Repository machine learning databases [3], and segment, dna and satimage are from the UCI statlog collection. Data structures are characterized in Table 1. For data sets without a given training/testing split, we divide them into 10 folds for cross-validation (CV). The splitting into CV folds is stratified over classes. Before making kernel data, we standardize the training inputs to have coordinatewise zero mean and unitary variance, and the same centering and scaling are applied to the testing set, where the center location and the coordinatewise scale factors are determined by the training set. Our proposal will be compared with the benchmark nonlinear SVM software LIBSVM [6], which has the reputation of being efficient in computation and having reliable prediction accuracy.

Model selection. Gaussian kernel is used for nonlinearly transforming the input instances. A selection method based on nested uniform designs (UDs) [23] is adopted for finding the parameter γ in the Gaussian kernel and the weight parameter C in SVM. The key advantage of the UD model selection over the grid search is that the UD points are "*far more uniform*" and "*far more space filling*" than lattice grid points. The number-theoretic based UD methodology [13] is a deterministic analogue of random search known as the quasi-Monte Carlo. It is known that a quasi-Monte Carlo method with judiciously chosen deterministic points usually leads to a faster rate of convergence than the lattice grid method [32]. The phenomena of better uniformity and more space-filling make the UD points very economic by avoiding wasteful function evaluations of similar patterns. Throughout this experimental study, we have used a 2-stage nested UDs search with the first stage of a 13-point UD and the second stage of a 9-point UD. The center UD point of the second stage is the best parameter setting from the first stage and needs not be re-trained. That is, we have to tune only 21 different combinations of γ and C instead of an exhaustive grid search. In [23] and our experience , the results by using the nested-UDs are usually good enough with much less computational cost as compared to the grid search for parameters tuning.

As all the scoring schemes in their full lengths are equivalent, we use BS in our tuning procedure via a 10-fold CV for all training data sets. In our experience, C is not as influential as γ. Hence, to save tuning time, we conveniently fix C at a large number, 10^6, for large data sets (segment, dna, satimge and pendigits), while still tune for C for smaller sets. For the case of a fixed C, the UD method then becomes a one-dimensional 21-grid-points search for γ. Note that the reduced kernel technique is employed for large data sets, i.e., a compressed model is used. In such cases, we are allowed for a large C without over-fitting. We provide 2 options for LIBSVM. One is their default setting $C = 1$ and $\gamma = 1/t$, where t is the number of attributes. The other is to apply our UD-based tuning method to LIBSVM. All resulting parameter settings are recorded in Table 2.

Experimental results. To demonstrate our classification procedure, we use IS, BS and OS to run the experiment by using RLS-mSVR to extract a discriminant feature subspace, and then perform a linear algorithm for classification. Here we particularly have used FLDA and linear SSVM as our choice of base classifiers. Other linear algorithms, acting on this low-dimensional fitted feature subspace, can be feasible as well. Note that this fitted feature subspace is a subspace of the Hilbert space generated by the underlying kernel. The FLDA can be easily generalized to a Bayes rule incorporating prior class probabilities to produce posterior probability outputs given in (7). For data without a given test set the reported accuracy is the mean accuracy over 20 replicate runs of 10-fold CV. Within each 10-fold partition, accuracy is averaged over 10 folds. Numbers in parentheses are standard deviations of the mean accuracies for 10-fold CV over 20 replicate runs. We have also used the reduced kernel approximations for large data sets, such as segment, dna, satimage and

Table 1
Data structures.

Data	# Training data	# Testing data	# Classes	# Attributes
ionosphere	351	0	2	33
Iris	150	0	3	4
wine	178	0	3	13
glass	214	0	6	9
segment	2310	0	7	18
image	210	2100	7	19
dna	2000	1186	3	180
satimage	4435	2000	6	36
pendigits	7494	3498	10	16

Table 2
Parameter settings.

Data	FLDA		SSVM		LIBSVM	
	C	γ	C	γ	C	γ
ion	4.217	0.0517	6.8129	0.0586	10	0.0324
Iris	17 413	0.0458	21 544	0.0334	0.7499	0.3405
wine	215.44	0.0150	139.905	0.025	1	0.0035
glass	146 780	0.0361	139.905	0.0815	3162.278	0.0045
segment[a]	10^6	0.0525	10^6	0.0608	10^6	0.0049
image	22 792	0.0458	3162.3	0.0425	100	0.002
dna[a]	10^6	0.0025	10^6	0.0051	10 000	0.08
satimage[a]	10^6	0.03	10^6	0.0608	3.1623	0.1045
pendigits[a]	10^6	0.0475	10^6	0.05	3.1623	0.0712

[a] Indicates the use of a reduced kernel with reduction rates 0.3, 0.4, 0.3 and 0.05 for segment, dna, satimage and pendigits data sets, respectively.

Table 3
Comparison of testing accuracy.

Data	FLDA	SSVM	LIBSVM	LIBSVM (default)
ionosphere	0.9556	0.9516	0.9472	0.9417
(std)	(0.0049)	(0.0048)	(0.0045)	(0.0045)
Iris	0.9773	0.9777	0.9633	0.9636
(std)	(0.0057)	(0.0064)	(0.0049)	(0.0045)
wine	0.9849	0.9912	0.9834	0.9826
(std)	(0.0040)	(0.0046)	(0.0038)	(0.0035)
glass	0.6753	0.7232	0.7152	0.7109
(std)	(0.0223)	(0.0156)	(0.0142)	(0.0127)
segment[a]	0.9729	0.9761	0.9715	0.9451
(std)	(0.0012)	(0.0011)	(0.0022)	(0.0011)
image	0.9214	0.9343	0.9062	0.8857
dna[a]	0.9545	0.9570	0.9520	0.9503
satimage[a]	0.9085	0.9125	0.9170	0.8955
pendigits[a]	0.9783	0.9813	0.9826	0.9791

[a] Indicates the use of a reduced kernel with reduction rates 0.3, 0.4, 0.3 and 0.05 for segment, dna, satimage and pendigits data sets, respectively.

Table 4
Comparison of training time in CPU seconds.

Data	IS		BS		OS		LIBSVM	
	FLDA	SSVM	FLDA	SSVM	FLDA	SSVM	UD	Default
image	0.01	0.03	0.01	0.03	0.01	0.03	0.01	0.01
dna[a]	0.60	0.59	0.57	0.59	0.60	0.62	2.64	2.05
satimage[a]	2.83	2.59	2.74	2.82	2.34	2.55	1.02	0.71
pendigits[a]	0.63	0.83	0.66	0.85	0.60	0.81	0.62	0.66

[a] Indicates the use of a reduced kernel with reduction rates 0.4, 0.3 and 0.05 for dna, satimage and pendigits data sets, respectively.

Table 5
Comparison of testing time in CPU seconds.

Data	IS		BS		OS		LIBSVM	
	FLDA	SSVM	FLDA	SSVM	FLDA	SSVM	UD	Default
image	0.01	0.02	0.01	0.02	0.01	0.02	0.06	0.08
dna[a]	0.15	0.15	0.15	0.15	0.15	0.15	2.50	1.61
satimage[a]	0.38	0.33	0.38	0.33	0.38	0.33	0.92	0.88
pendigits[a]	0.20	0.22	0.20	0.22	0.20	0.21	0.84	0.96

[a] Indicates the use of a reduced kernel with reduction rates 0.4, 0.3 and 0.05 for dna, satimage and pendigits data sets, respectively.

pendigits, where the reduction rates are 0.3, 0.4, 0.3, 0.05, respectively. All else have used the full kernel. As a reference, we also include LIBSVM in our study. The LIBSVM (default) columns indicate the results are from LIBSVM default setting. The LIBSVM (UD) columns indicate the results are from LIBSVM with UD-tuned parameters. The accuracies obtained by LIBSVM are all with full kernel. We did individually run our multiclass procedures, i.e., RLS-mSVR combined either with FLDA or one-one SSVM, using IS, BS, OS and 3-way scores. Since the accuracies we empirically obtained are all the same, which numerically confirm Theorem 4, they are combined into FLDA and SSVM columns and are compared with LIBSVM in Table 3. We also report the training and testing time in CPU seconds in Tables 4 and 5 for data sets with separate test sets. We have also tried the Lagrangian SVM [29], which has the same formulation as SSVM but solved in the dual space instead of in the primal. The resulting prediction accuracies of Lagrangian SVM are same as those of SSVM but with a bit longer CPU run time, and thus are omitted from the numerical report. We have used ASUS notebook with Core Duo T7200 CPU and 2 GB RAM in all our experiments.

From Table 3, we see that testing accuracies of SSVM are often better than FLDA. The testing accuracies via regression setting combined with SSVM are comparable to LIBSVM, and often a bit better. From the accuracy report for segment, dna, satimage and pendigits data sets, we see that the reduced kernel approximation works well. Table 4 shows the training time of our approach is comparable to LIBSVM (UD and default). All our programming codes (for making kernel, doing regression, FLDA, SSVM, etc.) are written in Matlab, while LIBSVM is in C. The training time for RLS-mSVR based methods has included the time for generating scores and kernel data. SSVM often takes a bit more CPU time than FLDA, but not necessarily always so. It solves $J(J-1)/2-$many binary linear SVM sub-problems, while FLDA solves one bigger problem on the same subspace. The training time difference for different scoring schemes is not significant. Though OS involves extra computation of a generalized eigenvalue problem, the problem size is often not large. Here for the image data example, we have to solve for the leading 6 eigenvectors in a problem of size 210; and for the dna data example, we solve for the leading 2 eigenvectors in a problem of size 800. It is noticeable that the RLS-mSVR approach provides a fast and efficient way for feature extraction and next a convenient linear discriminant algorithm can act on top of these very low-dimensional feature subspace (with dimensionality at most $J-1$) for fast computation. The testing CPU time of our approaches, RLS-mSVR with FLDA or SSVM, is much less than LIBSVM. The reason is that the regression approach has extracted a very low-dimensional discriminant feature subspace and the testing is extremely fast in this low-dimensional feature subspace, while the testing time of LIBSVM depends on the number of support vectors, which is usually much bigger than the number of classes.

Time complexity. All the coding schemes introduced in this article have very little computing load except possibly for OS in large data problems with quite some number of classes. OS is solved by a generalized eigenvalue problem of a data projection (12). Time complexity for solving the OS-related eigenvalue problem is $O(J^3)$ and for forming the underlying projection matrix is $O(n\,\tilde{n}^2)$. The time complexity for solving the RLS-mSVR is $O(n\,\tilde{n}^2)$ regardless of the number of score columns k.

The OS scheme provides an extra nice feature. Its columns are ranked by descending eigenvalues as commented in Remark 2. Some experiments are done to gain insights into the optimal scoring scheme as compared with others. Figs. 1 and 2 depict the effect of the code length on rates of test accuracy using pendigits and letter data sets, respectively. The left most columns from Θ_{OS}, Θ_{BS} and Θ_{IS} are sequentially added to the regression model fitting, while for the random code, *J independent* random code columns are drawn one at a time and sequentially added to the regression model fitting. The experiment of random code is repeated 6 times. The superiority of OS is clear when the code length is short, but when the code length reaches $J-1$, the schemes, OS, BS, IS and random code, become equivalent.

The performance of random code actually has better generalization ability with higher test accuracy than the OS, when the code length is of intermediate size. The OS is optimal in the training accuracy and in the case when the code length is small, such as 1–3 in length in the pendigits data set and 1–5 in length in the letter data set. Figs. 3 and 4 provide a pictorial explanation why OS performs better. Fig. 3 gives the data scatters and the

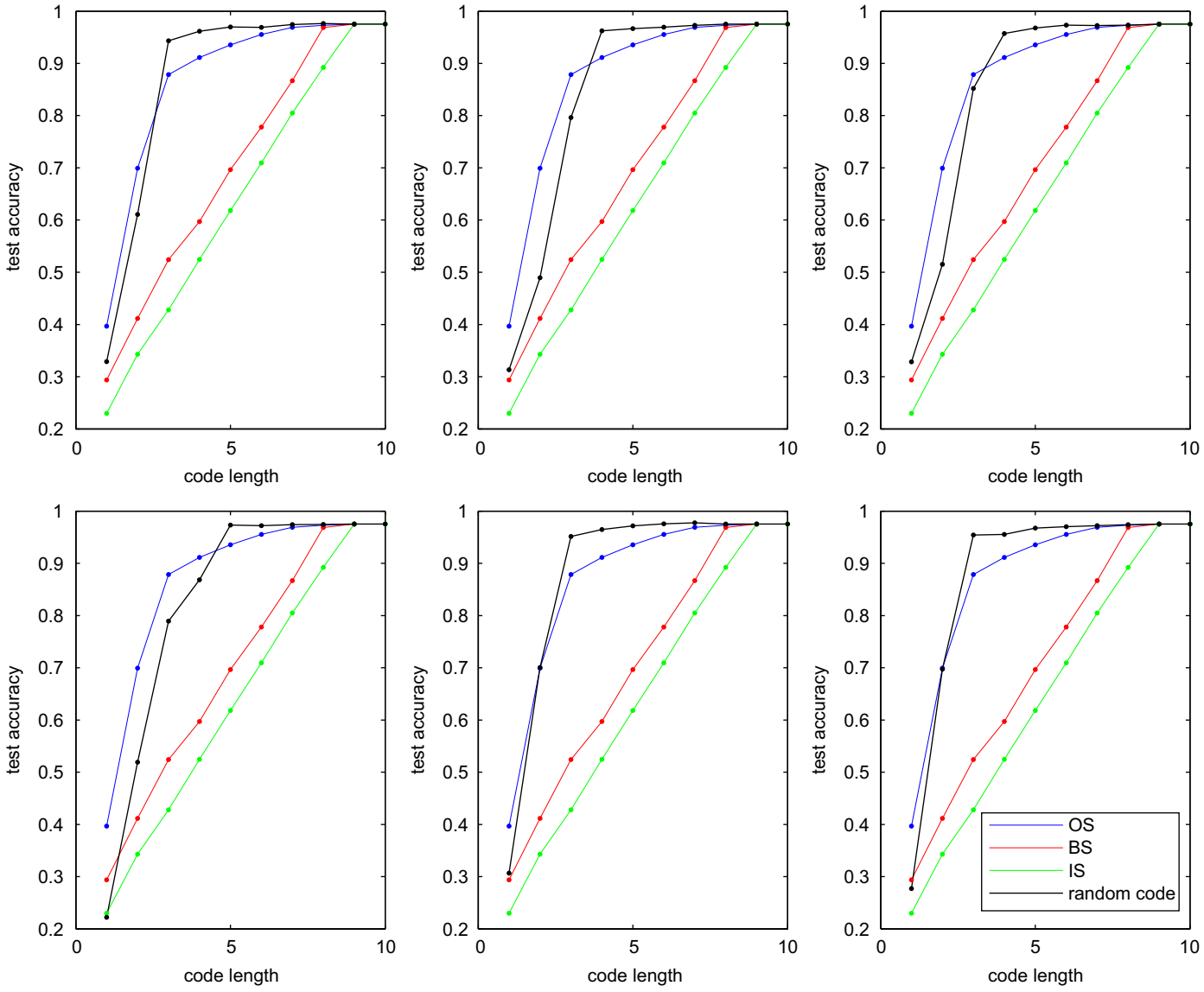

Fig. 1. Comparison of scoring schemes using pendigits data set in six replicate runs.

one-dimensional discriminant feature subspace by OS (in solid line) and by IS (in dashed line). Fig. 4 presents the data projections onto these 2 lines and their fitted normal probability density functions. It is clear that, for data projected along the OS-found direction, the class centroids are far apart, while the spread within each class is small. On the contrary, for data projected along the IS-found direction, classes 2 and 3 are collapsed together and the within class spread is larger than projections along the OS-direction. We further provide some 2D views of OS-found and IS-found discriminant feature subspaces in Figs. 5 and 6. They are, respectively, pendigits data scatters over the 2D discriminant feature subspace using the first 2 columns of OS-code and IS-code, respectively. The 2D-OS and the 2D-IS discriminant feature subspace are obtained based on the entire training set, while only 10 points per digit group from test set are used to plot the data scatter to avoid excessive ink. Data projections onto the 2D-OS discriminant subspace have much better class separation than data projections onto the 2D-IS discriminant subspace. All programming codes are available upon email request to the last author.

5. Concluding remarks

In this article the mSVR is proposed for the multiclass classification problem. The class labels are encoded into multi-response scores and then the regression of scores on kernel inputs is used to extract a low-dimensional discriminant feature subspace, which is spanned by the regression coefficient variates. The discriminant feature subspace generated by different coding schemes with long enough code length will be identical, which introduces the notion of equivalence of codes. Data are then mapped to this low-dimensional feature subspace and classification is carried out therein. The regression step can be viewed as a feature extraction to make the final classification (decoding) step computationally light and easy. We can apply any feasible linear learning algorithm to the data images in this low-dimensional discriminant feature subspace. Even there is a nonlinear structure between different classes in the input space, the linear algorithm still perform very well in the kernel-based feature subspace. Though this feature subspace is linear in the Hilbert space sense, it is nonlinear from the viewpoint of original input attributes. Two

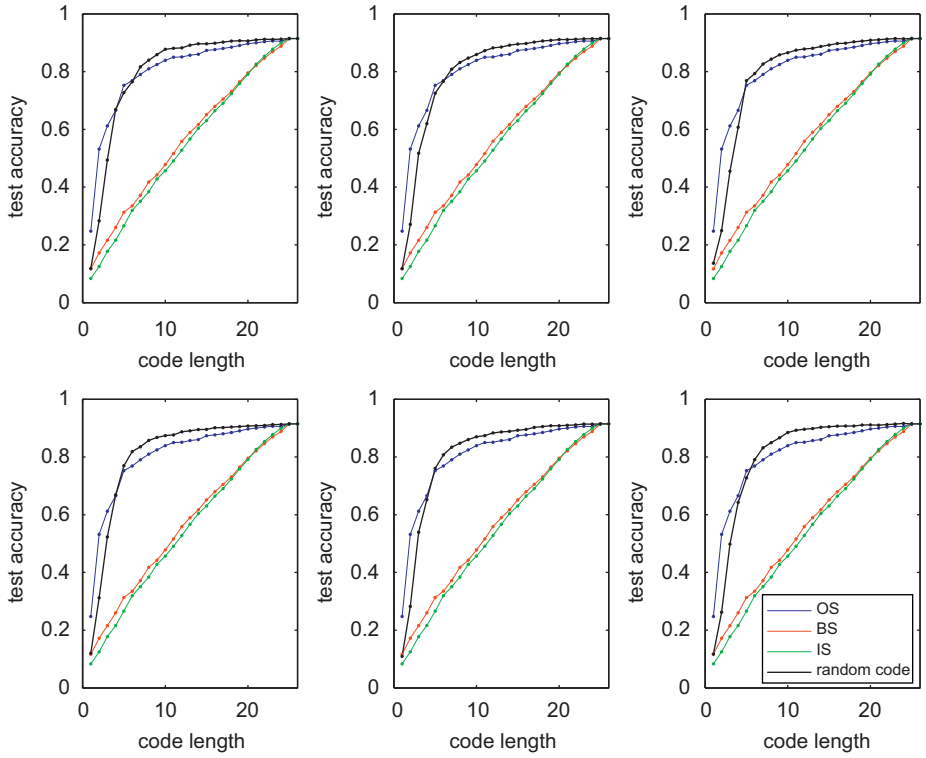

Fig. 2. Comparison of scoring schemes using letter data set in six replicate runs.

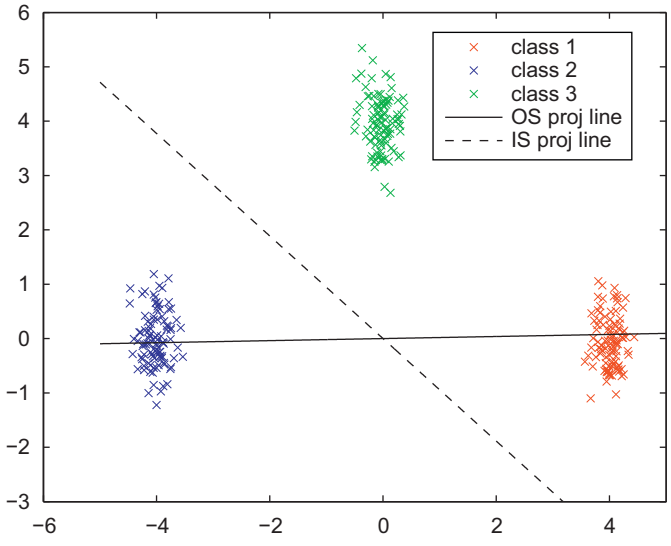

Fig. 3. Data scatters and projection lines by OS and IS.

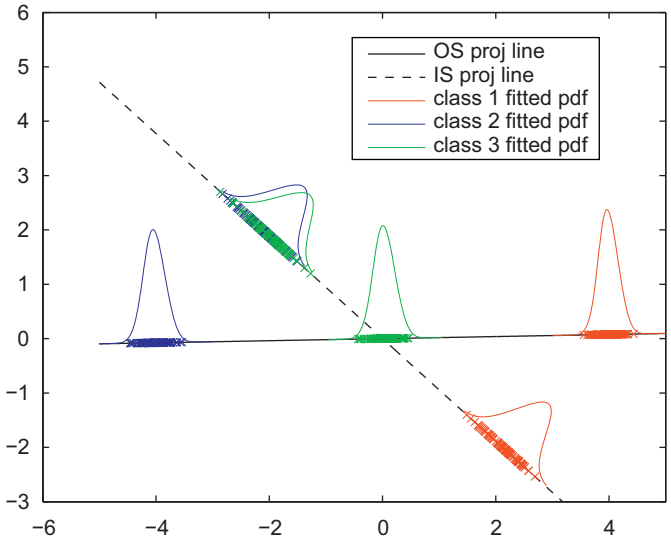

Fig. 4. Data projected onto projection lines and fitted normal density functions.

underlying linear algorithms, FLDA and one-one SSVM, are used for experimental study. However, our proposed method is not limited to these two particular algorithms. We also include LIBSVM for comparison. The RLS-mSVR solves a linear system and provides an economic implementation for multiresponse regression, and it can be combined with any reasonable linear discriminant algorithms for satisfactory prediction accuracy. Results show that our mSVR setup combined with a feasible linear discriminant algorithm is an easy and yet competent alternative to the multiclass classification problem. One thing is worth to be pointed out. By the equivalence of codes, if we allow the code length to reach $J-1$, the maximum dimensionality of linear separation for J classes, then BS will be the best choice for its simplicity in computation and interpretation. It is not necessary to look for the optimal scores for encoding. However, if there is a limitation on the code length, which is much shorter than $J-1$, the optimal scoring will have an extra nice feature. Its columns are ranked by descending eigenvalues, and hence, the OS can provide a further dimension reduction by using only leading columns. For intermediate code length, random code

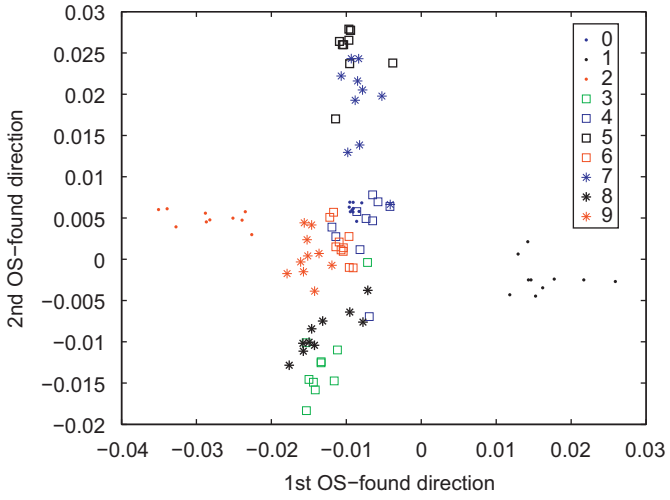

Fig. 5. Pendigits data scatter over 2D-OS discriminant feature subspace.

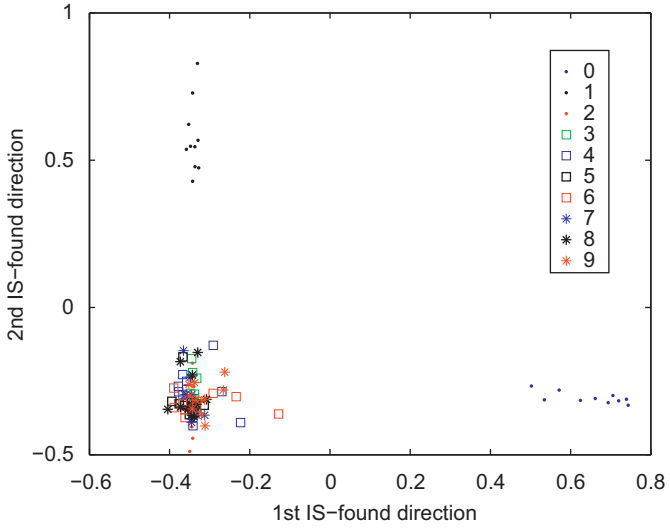

Fig. 6. Pendigits data scatter over 2D-IS discriminant feature subspace.

(with independent code columns enforced) is recommended for its simplicity in forming the code matrix and for its better generalization ability. The BS, which has taken the class sizes into account and has contracted one class with the baseline and with the rest as reference, is a better choice than the IS with one less column of code length. The indicator scoring has the simplest form and is easiest to interpret especially for users not familiar with coding. As for the 3-way or random codes with longer length than J, they provide the extra ability of error-detecting and correcting. Also, given a fixed code length, one can design a deterministic selection of columns from Θ_{3W} or random code to optimize its prediction performance. This is a design problem and may deserve some further study. The reduced kernel technique for dealing with massive data sets and the nested uniform designs for selecting parameters have been incorporated successfully into the mSVR framework. Results show that our regression setup combined with a feasible linear discriminant algorithm is an easy and yet competent alternative to the multiclass classification problem.

Acknowledgments

The authors thank Chuhsing Kate Hsiao and Chii-Ruey Hwang for helpful comments.

Appendix

Proof of Proposition 3. Assume that Θ_1 and Θ_2 are equivalent. For $i=1,2$,

$$\begin{aligned} RY_i &= \left(I_n-\frac{\mathbf{1}_n\mathbf{1}_n'}{n}\right)Y_{IS}\Theta_i \\ &= \left(I_n-\frac{\mathbf{1}_n\mathbf{1}_n'}{n}\right)Y_{IS}Q\Theta_i+\left(I_n-\frac{\mathbf{1}_n\mathbf{1}_n'}{n}\right)Y_{IS}\frac{\mathbf{1}_J\mathbf{e}_j'}{n}\Theta_i \\ &= \left(I_n-\frac{\mathbf{1}_n\mathbf{1}_n'}{n}\right)Y_{IS}Q\Theta_i+\left(Y_{IS}-\frac{\mathbf{1}_n\mathbf{e}_j'}{n}\right)\frac{\mathbf{1}_J\mathbf{e}_j'}{n}\Theta_i \\ &= \left(I_n-\frac{\mathbf{1}_n\mathbf{1}_n'}{n}\right)Y_{IS}Q\Theta_i+\left(\frac{\mathbf{1}_n\mathbf{e}_j'}{n}-\frac{\mathbf{1}_n\mathbf{e}_j'}{n}\right)\Theta_i \\ &= \left(I_n-\frac{\mathbf{1}_n\mathbf{1}_n'}{n}\right)Y_{IS}Q\Theta_i. \end{aligned}$$

Since $Q\Theta_1$ and $Q\Theta_2$ have the same column span, so do $RY_{IS}Q\Theta_1$ and $RY_{IS}Q\Theta_2$. Hence, $\mathcal{C}(RY_1)=\mathcal{C}(RY_2)$, and Y_1 and Y_2 are equivalent. Conversely, assume that Y_1 and Y_2 have the same central column space, we want to show that Θ_1 and Θ_2 are equivalent. Since that $Y_{IS}'Y_{IS}=D$, we have $\Theta_i=D^{-1}Y_{IS}'Y_i$. Then, for $i=1,2$

$$\begin{aligned} Q\Theta_i &= QD^{-1}Y_{IS}'Y_i \\ &= QD^{-1}Y_{IS}'\left(I_n-\frac{\mathbf{1}_n\mathbf{1}_n'}{n}\right)Y_i+QD^{-1}Y_{IS}'\frac{\mathbf{1}_n\mathbf{1}_n'}{n}Y_i \qquad (13) \\ &= QD^{-1}Y_{IS}'\left(I_n-\frac{\mathbf{1}_n\mathbf{1}_n'}{n}\right)Y_i, \qquad (14) \end{aligned}$$

as $QD^{-1}Y_{IS}'\mathbf{1}_n=(I_J-\mathbf{1}_J\mathbf{e}_j'/n)D^{-1}\mathbf{e}_J=D^{-1}\mathbf{e}_J-\mathbf{1}_J=\mathbf{0}_J$. Since that Y_1 and Y_2 have the same central column space, so do $QD^{-1}Y_{IS}'(I_n-\mathbf{1}_n\mathbf{1}_n'/n)Y_i$ for $i=1,2$. Hence, $\mathcal{C}(Q\Theta_1)=\mathcal{C}(Q\Theta_2)$, and Θ_1 and Θ_2 are equivalent. □

The following lemma is for Theorem 4. Let $\hat{W}^{(IS)}=[\hat{w}_1^{(IS)},\ldots,\hat{w}_J^{(IS)}]$, $\hat{w}_j^{(IS)}\in\Re^{\tilde{n}}$, and $\hat{b}^{(IS)}=[\hat{b}_1^{(IS)},\ldots,\hat{b}_J^{(IS)}]\in\Re^{1\times J}$ be the RLS estimates for W and b using scores Y_{IS}.

Lemma 1. *The RLS estimate of W using scores Y_{IS}, denoted by $\hat{W}^{(IS)}$, satisfies the condition that $\hat{W}^{(IS)}\mathbf{1}_J=\mathbf{0}_{\tilde{n}}$.*

Proof of Lemma 1. Let

$$M=\begin{bmatrix}M_{11}&M_{12}\\M_{21}&M_{22}\end{bmatrix}=\begin{bmatrix}\tilde{K}'\tilde{K}+\lambda I_{\tilde{n}}&\tilde{K}'\mathbf{1}_n\\\mathbf{1}_n'\tilde{K}&n\end{bmatrix},\quad \text{where } \lambda=1/C.$$

Then,

$$M^{-1}=\begin{bmatrix}M_{11\cdot2}^{-1}&-M_{11\cdot2}^{-1}M_{12}M_{22}^{-1}\\-M_{22}^{-1}M_{21}M_{11\cdot2}^{-1}&M_{22}^{-1}M_{21}M_{11\cdot2}^{-1}M_{12}M_{22}^{-1}+M_{22}^{-1}\end{bmatrix},$$

where $M_{11\cdot2}=M_{11}-M_{12}M_{22}^{-1}M_{21}$ [2] . By solving the RLS we have that

$$\begin{bmatrix}\hat{W}^{(IS)}\\\hat{b}^{(IS)}\end{bmatrix}_{(\tilde{n}+1)\times J}=M^{-1}\begin{bmatrix}\tilde{K}'\\\mathbf{1}_n'\end{bmatrix}Y_{IS}.$$

Thus,

$$\hat{W}^{(IS)}\mathbf{1}_J=[M_{11\cdot2}^{-1}\quad -M_{11\cdot2}^{-1}M_{12}M_{22}^{-1}]\begin{bmatrix}\tilde{K}'\\\mathbf{1}_n'\end{bmatrix}Y_{IS}\mathbf{1}_J$$

$$= [M_{11\cdot 2}^{-1} \quad -M_{11\cdot 2}^{-1} M_{12} M_{22}^{-1}] \begin{bmatrix} \tilde{K}' \mathbf{1}_n \\ \mathbf{1}_n' \mathbf{1}_n \end{bmatrix} = M_{11\cdot 2}^{-1} (\tilde{K}' \mathbf{1}_n - n M_{12} M_{22}^{-1})$$

$$= M_{11\cdot 2}^{-1} \cdot \mathbf{0}_{\tilde{n}}. \quad \square$$

Proof of Theorem 4. For $i=1,2$, let $\hat{W}^{(i)} = [\hat{w}_1^{(i)}, \ldots, \hat{w}_{k_i}^{(i)}] \in \Re^{\tilde{n} \times k_i}$ and $\hat{b}^{(i)} = [\hat{b}_1^{(i)}, \ldots, \hat{b}_{k_i}^{(i)}] \in \Re^{1 \times k_i}$ be the RLS estimates of regression coefficients and intercepts using scores $Y_1 = Y_{IS}\Theta_1$ and $Y_2 = Y_{IS}\Theta_2$, respectively. Then, as $Y_i = Y_{IS}\Theta_i$, it is straightforward to check that

$$\hat{W}^{(1)} = \hat{W}^{(IS)}\Theta_1, \quad \hat{W}^{(2)} = \hat{W}^{(IS)}\Theta_2, \quad \hat{b}^{(1)} = \hat{b}^{(IS)}\Theta_1 \text{ and } \hat{b}^{(2)} = \hat{b}^{(IS)}\Theta_2. \tag{15}$$

Below we show that $\hat{W}^{(1)}$ and $\hat{W}^{(2)}$ have the same column span. Express $\hat{W}^{(i)}$ as follows:

$$\hat{W}^{(i)} = \hat{W}^{(IS)} Q \Theta_i + \hat{W}^{(IS)} \frac{\mathbf{1}_J \mathbf{e}_j'}{n} \Theta_i. \tag{16}$$

Since $Q\Theta_1$ and $Q\Theta_2$ span the same column space, so do $\hat{W}^{(IS)}Q\Theta_1$ and $\hat{W}^{(IS)}Q\Theta_2$. Also notice that $\hat{W}^{(IS)}\mathbf{1}_J = \mathbf{0}_{\tilde{n}}$ by Lemma 1. Thus, we have $\mathcal{C}(\hat{W}^{(1)}) = \mathcal{C}(\hat{W}^{(2)})$. In other words, scores Y_1 and Y_2 lead to the same discriminant subspace. $\square$

Proof of Theorem 5. We can either check that $\mathcal{C}(Q\Theta)$ for $\Theta = \Theta_{IS}$, Θ_{OS}, Θ_{BS}, Θ_{3W} and Θ_{EX} are all the same, or equivalently, check that $\mathcal{C}((I_n - \mathbf{1}_n\mathbf{1}_n'/n)Y)$ for $Y = Y_{IS}$, Y_{OS}, Y_{BS}, Y_{3W} and Y_{EX} are all the same. Both ways are straightforward to verify by using Definition 1 or Proposition 2. $\square$

References

[1] E.L. Allwein, R.E. Schapire, Y. Singer, Reducing multiclass to binary: a unifying approach for margin classifiers, J. Mach. Learn. Res. 1 (2001) 113–141.
[2] T.W. Anderson, An Introduction to Multivariate Statistical Analysis, 3rd ed., Wiley, New York, 2003.
[3] A. Asuncion, D.J. Newman, UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences, 2007, ⟨http://www.ics.uci.edu/∼ mlearn/MLRepository.htm⟩.
[4] E.J. Bredensteiner, K.P. Bennett, Multicategory classification by support vector machines, Comput. Optim. Appl. 12 (1999) 53–79.
[5] C.J.C. Burges, A tutorial on support vector machines for pattern recognition, Data Min. Knowl. Disc. 2 (1998) 121–167.
[6] C.C. Chang, C.J. Lin, LIBSVM: a library for support vector machines, 2001, ⟨http://www.csie.ntu.edu.tw/∼cjlin/libsvm⟩.
[7] C. Cortes, V. Vapnik, Support vector networks, Mach. Learn. 20 (1995) 273–297.
[8] K. Crammer, Y. Singer, Improved output coding for classification using continuous relaxation, in: Proceeding of the Thirteenth Annual Conference on Neural Information Processing Systems, 2000.
[9] K. Crammer, Y. Singer, On the algorithmic implementation of multiclass kernel-based vector machines, J. Mach. Learn. Res. 2 (2001) 265–292.
[10] K. Crammer, Y. Singer, On the learnability and design of output codes for multiclass problems, Mach. Learn. 47 (2002) 201–233.
[11] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines, Cambridge University Press, Cambridge, 2000.
[12] T.G. Dietterich, G. Bakiri, Solving multiclass learning problems via error-correcting output codes, J. Artif. Intell. Res. 2 (1995) 263–286.
[13] K.T. Fang, Y. Wang, Number-Theoretic Methods in Statistics, Chapmman & Hall, London, 1994.
[14] R.A. Fisher, The use of multiple measurements in taxonomic problems, Ann. Eugen. 7 (1936) 179–188.
[15] J.H. Friedman, Multivariate adaptive regression splines (with discussion), Ann. Stat. 19 (1991) 1–141.
[16] G.M. Fung, O.L. Mangasarian, Proximal support vector machine classifiers, in: Proceedings KDD-2001: Knowledge Discovery and Data Mining, 2001, pp. 77–86.
[17] G.M. Fung, O.L. Mangasarian, Multicategory proximal support vector machine classifiers, Mach. Learn. 59 (2005) 77–97.
[18] J. Fürnkranz, Round robin classification, J. Mach. Learn. Res. 2 (2002) 721–747.
[19] T. Hastie, A. Buja, R. Tibshirani, Penalized discriminant analysis, Ann. Stat. 23 (1995) 73–102.
[20] T. Hastie, R. Tibshirani, A. Buja, Flexible discriminant analysis by optimal scoring, J. Am. Stat. Assoc. 89 (1994) 1255–1270.
[21] D.G. Hoffman, D.A. Leonard, C.C. Lindner, K.T. Phelps, C.A. Rodger, J.R. Wall, Coding Theory: The Essentials, Marcel Dekker, New York, 1991.
[22] C.W. Hsu, C.J. Lin, A comparison on methods for multi-class support vector machines, IEEE Trans. Neural Network 13 (2002) 415–425.
[23] C.M. Huang, Y.J. Lee, D.K.J. Lin, S.Y. Huang, Model selection for support vector machines via uniform design, Comput. Stat. Data Anal. 52 (2007) 335–346.
[24] Y. Lee, Y. Lin, G. Wahba, Multicategory support vector machines: theory and application to the classification of microarray data and satellite radiance data, J. Am. Stat. Assoc. 99 (2004) 67–81.
[25] Y.J. Lee, S.Y. Huang, Reduced support vector machines: a statistical theory, IEEE Trans. Neural Network 18 (2007) 1–13.
[26] Y.J. Lee, O.L. Mangasarian, RSVM: reduced support vector machines, in: First SIAM International Conference on Data Mining, Chicago, 2001.
[27] Y.J. Lee, O.L. Mangasarian, SSVM: a smooth support vector machine for classification, Comput. Optim. Appl. 20 (2001) 5–22.
[28] Y. Liu, Z. You, L. Cao, A novel and quick SVM-based multi-class classifier, Pattern Recognition 39 (2006) 2258–2264.
[29] O.L. Mangasarian, D.R. Musicant, Lagrangian support vector machines, J. Mach. Learn. Res. 1 (2001) 161–177.
[30] K.V. Mardia, J.T. Kent, J.M. Bibby, Multivariate Analysis, Academic Press, New York, 1979.
[31] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, K.R. Müller, Fisher discriminant analysis with kernels, in: Neural Networks for Single Processing IX, 1999, pp. 41–48.
[32] H. Niederreiter, Random number generation and quasi-Monte Carlo methods, in: Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 1999.
[33] R. Rifkin, A. Klautau, In defense of one-vs-all classification, J. Mach. Learn. Res. 5 (2004) 101–141.
[34] V. Roth, V. Steinhage, Nonlinear discriminant analysis using kernel functions, in: Advances in Neural Information Processing System, vol. 12, 1999, pp. 568–574.
[35] J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle, Least Squares Support Vector Machines, World Scientific, New Jersey, 2002.
[36] J.A.K. Suykens, J. Vandewalle, Least squares support vector machine classifiers, Neural Process. Lett. 9 (1999) 293–300.
[37] V.N. Vapnik, Statistical Learning Theory, Wiley, New York, 1998.

Pei-Chun Chen received her B.S. degree from National Dong-Hwa University in 2001, and the M.S. and Ph.D. degrees from Graduate Institute of Epidemiology, National Taiwan University in 2003 and 2008, respectively. Currently, she is a Post-doctoral Fellow in Bioinformatics and Biostatistics Core, Research Center for Medical Excellence, National Taiwan University. Her research interests are in biostatistics, Bayesian statistics, bioinformatics and machine learning.

Kuang-Yao Lee received the B.S. and M.S. degrees in the Mathematics Department of National Taiwan University, in 2002 and 2005. Currently a Ph.D. student in the Department of Statistics in Pennsylvania State University. Research interests include linear and non-linear dimensionality reduction methods and machine learning.

Tsung-Ju Lee received a B.S. degree in Mathematics from TungHai University, Taiwan in 2000 and the M.S. degree in Applied Mathematics from National Chiao Tung University, Taiwan in 2002. Currently, he is working towards the Ph.D. degree in the Department of Computer Science, National Chiao Tung University, Taiwan. His current research interests include machine learning, data mining and various applications, especially in network security, e-learning and computational biology.

Yuh-Jye Lee received his Master degree in Applied Mathematics from the National Tsing Hua University, Taiwan in 1992 and Ph.D. degree in computer sciences from the University of Wisconsin, Madison in 2001. He is an associate professor in the Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology. His research interests are in machine learning, data mining, optimization, information security and operations research. Dr. Lee developed new algorithms for large data mining problems such as classification and regression problem, abnormal detection and dimension reduction. Using the methodologies such as support vector machines, reduced kernel method, chunking and smoothing techniques allow us to get a very robust solution (prediction) for a large dataset. These methods have been applied to solve many real world problems such as intrusion detection system (IDS), face detection, microarray gene expression analysis and breast cancer diagnosis and prognosis.

Su-Yun Huang received the B.S. and M.S. degrees from Department of Mathematics, National Taiwan University, in 1983 and 1985, respectively, and the Ph.D. degree from Department of Statistics, Purdue University in 1990. She is currently a Research Fellow in the Institute of Statistical Science, Academia Sinica, Taiwan. Her research interests are mainly on mathematical statistics.