Efficient Optimization for Low-Rank Integrated Bilinear Classifiers

Takumi Kobayashi and Nobuyuki Otsu

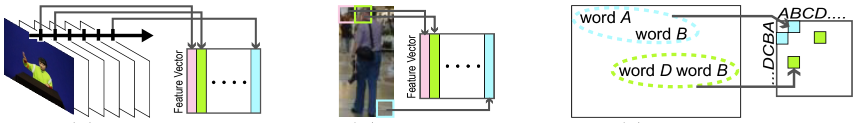
National Institute of Advanced Industrial Science and Technology,
1-1-1, Umezono, Tsukuba, Japan
{takumi.kobayashi,otsu.n}@aist.go.jp

Abstract. In pattern classification, it is needed to efficiently treat two-way data (feature matrices) while preserving the two-way structure such as spatio-temporal relationships, *etc*. The classifier for the feature matrix is generally formulated by multiple bilinear forms which result in a matrix. The rank of the matrix, *i.e.*, the number of bilinear forms, should be low from the viewpoint of generalization performance and computational cost. For that purpose, we propose a low-rank bilinear classifier based on the efficient optimization. In the proposed method, the classifier is optimized by minimizing the trace norm of the classifier (matrix), which contributes to the rank reduction for an efficient classifier without any hard constraint on the rank. We formulate the optimization problem in a tractable convex form and propose the procedure to solve it efficiently with the global optimum. In addition, by considering a kernel-based extension of the bilinear method, we induce a novel multiple kernel learning (MKL), called heterogeneous MKL. The method combines both inter kernels between heterogeneous types of features and the ordinary kernels within homogeneous features into a new discriminative kernel in a unified manner using the bilinear model. In the experiments on various classification problems using feature arrays, co-occurrence feature matrices, and multiple kernels, the proposed method exhibits favorable performances compared to the other methods.

1 Introduction

Classification problems are usually addressed to the tasks to classify feature vectors extracted from the target domain such as images [1,2] and motion sequences [3,4]. The features are often to be represented in a *vector* form (one-way) such as by concatenation, whereas it is inherently defined in a *matrix* form (two-way). For example, the matrix forms are found in image pixels, arrays of (local) feature vectors extracted at spatio-temporal (grid) points such as in HOG [1], and co-occurrence features (Fig. 1). The dimensionality of the concatenated feature vector is the product of two-way's dimensions, resulting in high dimensionality, and the inherent structure of the two-way features is unfortunately collapsed [5].

There are some recent works to directly deal with the features in a matrix form; 2DPCA [6] and 2DLDA [7] are extended from PCA and LDA to matrix-based formulations for dimensionality reduction, and the methods in [8,9] are



(a) One-way (b) Two-way (c) Co-occurrence

Fig. 1. Examples of feature matrices. Arrays of feature vectors extracted from one-way (a) and two-way data (b) and co-occurrence features (c) are defined in a matrix form.

also proposed to factorize the feature matrix. Those methods are formulated mainly to learn effective image representations in a matrix form and are not intended for classification problems in a supervised framework.

For classification, significant research efforts have been made on constructing linear classifiers for feature vectors, *e.g.*, SVM [10] and L_1-SVM [11] in the maximum-margin framework [12], which can be further extended to kernel-based methods [13]. On the other hand, the classifier to deal with feature matrices is naturally defined as a *bilinear* model comprising two kinds (row and column) of weights [5], which forms a matrix in general. Along with the advances of the linear classifiers, some bilinear classifiers have been recently proposed [14,15]. The main concern in the bilinear methods is to construct the bilinear classifier (matrix) of low rank in a manner similar to the maximum-margin framework; The VC-dimension of the low-rank bilinear model is proven to be less than that of the concatenated linear models [15]. In those methods, the approximated optimization approaches are usually employed, resulting in local minima, since the optimization problems are formulated as a biconvex (non-convex) formulations and semi-definite programming (SDP) which is computationally less efficient. In addition, the constraint regarding to the rank is explicitly introduced as a free parameter; that is, users are required to determine the classifier rank in advance.

In this paper, we propose a novel method to optimize the efficient bilinear classifier which results in low rank. Without approximations nor hard constraints on the rank, the proposed method automatically produces the optimal low-rank classifier by minimizing the trace norm of the classifier matrix, while reducing the classification errors.

The contributions of this paper are 1) to formulate a tractable convex optimization problem and propose an efficient optimization procedure to provide the global minimum, and 2) by considering a kernel extension of the bilinear classifier, to apply the proposed method to multiple kernel learning (MKL) [16] and induce a novel MKL, called *heterogeneous MKL*. In the heterogeneous MKL, by using the bilinear model, we can integrate both inter kernels between heterogeneous types of features and the ordinary kernels within homogeneous features into a new discriminative kernel in a unified manner.

2 Bilinear Classifier

Let $\boldsymbol{X}$ be a feature matrix whose dimensions are denoted by h and w ($\boldsymbol{X} \in \mathbb{R}^{h\times w}$). For example, $\boldsymbol{X}$ is regarded as the array of the h-dimensional feature vectors extracted at w points, such as in xy-coordinates for images or along t-axis for time-series signals, as shown in Fig. 1. To deal with the feature matrix, a bilinear classifier is simply formulated as $\hat{y}=\boldsymbol{w}_h^\top \boldsymbol{X}\boldsymbol{w}_w+b$ where $\boldsymbol{w}_h\in\mathbb{R}^h$, $\boldsymbol{w}_w\in\mathbb{R}^w$. This is regarded as a '1-rank' classifier, and by integrating multiple such classifiers, the general bilinear classifier is defined by

$$\hat{y} = \mathrm{tr}(\boldsymbol{W}_h^\top \boldsymbol{X}\boldsymbol{W}_w) + b = \mathrm{tr}(\boldsymbol{W}^\top \boldsymbol{X}) + b, \tag{1}$$

where b is the bias and $\boldsymbol{W} = \boldsymbol{W}_h\boldsymbol{W}_w^\top \in \mathbb{R}^{h\times w}$ is the classifier matrix ($\boldsymbol{W}_h \in \mathbb{R}^{h\times r}$, $\boldsymbol{W}_w\in\mathbb{R}^{w\times r}$ where $r\leq \min[w,h]$). For simplicity, we consider a two-class classification problem, given n samples $\{\boldsymbol{X}_i, y_i\}_{i=1}^n$ where y_i is the class label ($y_i\in\{+1,-1\}$) of the i-th sample. As in the maximum-margin framework [12], we measure the margin of the bilinear classifier in (1) by the matrix trace norm, *i.e.*, sum of singular values, for minimizing the matrix rank, which results in the following optimization problem:

$$\min_{\boldsymbol{W},b}\ ||\boldsymbol{W}||_\Sigma + C\sum_{i=1}^n \max\big[0, 1-y_i\{\mathrm{tr}(\boldsymbol{W}^\top \boldsymbol{X}_i) + b\}\big], \tag{2}$$

where $||\boldsymbol{W}||_\Sigma$ indicates the trace norm of $\boldsymbol{W}$. Let the singular values of $\boldsymbol{W}$ be denoted by $\boldsymbol{\sigma} \in \mathbb{R}^r$. The trace norm is represented by $||\boldsymbol{W}||_\Sigma = ||\boldsymbol{\sigma}||_1$, while the rank is measured by $\mathrm{rank}(\boldsymbol{W})=||\boldsymbol{\sigma}||_0$. Thus, in the formulation (2), the L_1 norm (trace norm) in the objective cost is regarded as a relaxation of the L_0 norm which directly minimizes the rank. Such L_1-norm minimization induces sparsity [11] in the singular values, thereby minimizing the rank. In the followings, we reformulate (2) to the tractable convex problem (10) and propose the procedure to solve it efficiently.

2.1 SDP Problem

We begin with rewriting (2) to semi-definite programming (SDP) [17]. By introducing the augmented variables,

$$\tilde{\boldsymbol{X}}_i = \begin{bmatrix}\boldsymbol{0} & \boldsymbol{X}_i\\ \boldsymbol{X}_i^\top & \boldsymbol{0}\end{bmatrix},\quad \tilde{\boldsymbol{W}} = \begin{bmatrix}\boldsymbol{W}_h\\ \boldsymbol{W}_w\end{bmatrix}\begin{bmatrix}\boldsymbol{W}_h\\ \boldsymbol{W}_w\end{bmatrix}^\top = \begin{bmatrix}\boldsymbol{W}_h\boldsymbol{W}_h^\top & \boldsymbol{W}\\ \boldsymbol{W}^\top & \boldsymbol{W}_w\boldsymbol{W}_w^\top\end{bmatrix},$$

the formulation (2) results in

$$\min_{\tilde{\boldsymbol{W}}\succeq 0,b}\ \frac{1}{2}\mathrm{tr}(\tilde{\boldsymbol{W}}) + C\sum_i \max\Big[0, 1-y_i\Big\{\frac{1}{2}\mathrm{tr}(\tilde{\boldsymbol{W}}^\top\tilde{\boldsymbol{X}}_i)+b\Big\}\Big], \tag{3}$$

where we replace $||\boldsymbol{W}||_\Sigma$ by $\mathrm{tr}(\tilde{\boldsymbol{W}})$ as in [17]. In the convex problem (3), the global optimum is obtained by SDP [18], but it requires significant computational costs, which makes it infeasible for large-scaled samples.

2.2 Efficient Convex Problem

Considering $\tilde{\boldsymbol{W}}=\begin{bmatrix}\boldsymbol{W}_h\\ \boldsymbol{W}_w\end{bmatrix}\begin{bmatrix}\boldsymbol{W}_h\\ \boldsymbol{W}_w\end{bmatrix}^\top$, we further rewrite (3) to

$$\min_{\boldsymbol{W}_w}\Big[\frac{1}{2}\mathrm{tr}(\boldsymbol{W}_w\boldsymbol{W}_w^\top) + \underline{\min_{\boldsymbol{W}_h,b}\Big\{\frac{1}{2}\mathrm{tr}(\boldsymbol{W}_h\boldsymbol{W}_h^\top)+C\sum_i \max\big[0,1-y_i\{\mathrm{tr}(\boldsymbol{W}_h^\top\boldsymbol{X}_i\boldsymbol{W}_w)+b\}\big]\Big\}}\Big]. \tag{4}$$

The formulation (4) is biconvex (non-convex) or bilevel [19], and the iterative approach which optimizes either of $\boldsymbol{W}_h$ or $\boldsymbol{W}_w$ in an alternating manner is applicable as in [14]. Such an approach is tractable in contrast to SDP, but it results in local minima. In this study, we reformulate (4) to a tractable convex problem and propose a procedure to produce the global optimum efficiently.

Here, we suppose the column size is smaller than the row size, $h > w$, without loss of generality. The inner optimization (underlined term) in (4) is regarded as the standard SVM problem with respect to $\boldsymbol{W}_h, b$, and thus it has the dual:

$$\max_{\boldsymbol{\alpha}\in\Theta_{\boldsymbol{y}}} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j K_{ij}, \tag{5}$$

$$\text{where } \Theta_{\boldsymbol{y}} = \left\{\boldsymbol{\alpha} \mid \forall i,\ 0\le\alpha_i\le C,\ \textstyle\sum_i y_i\alpha_i = 0\right\}, \tag{6}$$

$$K_{ij} = \mathrm{tr}\{(\boldsymbol{X}_i\boldsymbol{W}_w)^\top(\boldsymbol{X}_j\boldsymbol{W}_w)\} = \mathrm{tr}(\boldsymbol{W}_w\boldsymbol{W}_w^\top\boldsymbol{X}_i^\top\boldsymbol{X}_j). \tag{7}$$

Thereby, given the optimum $\boldsymbol{W}_w^*$, we get the optimum bilinear classifier as

$$\boldsymbol{W}_h^* = \sum_i \alpha_i^* y_i \boldsymbol{X}_i\boldsymbol{W}_w^*, \quad \hat{y} = \mathrm{tr}(\boldsymbol{W}_w^*\boldsymbol{W}_w^{*\top}\sum_i\alpha_i^* y_i\boldsymbol{X}_i^\top\boldsymbol{X}), \tag{8}$$

where α_i^* are the optimizers in (5). In the forms (4-8), we can see that the key variable is $\boldsymbol{\Sigma}_w \triangleq \boldsymbol{W}_w\boldsymbol{W}_w^\top \succcurlyeq 0$ rather than $\boldsymbol{W}_w$ itself. Since the inner optimization in (4) can be replaced with its dual (5) due to the string duality [20,21], the optimization (4) is reformulated into

$$\min_{\boldsymbol{\Sigma}_w\succcurlyeq 0}\left[\frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}_w) + \max_{\boldsymbol{\alpha}\in\Theta_{\boldsymbol{y}}}\left\{\sum_i\alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j K_{ij(\boldsymbol{\Sigma}_w)}\right\}\right],$$

$$\text{where } K_{ij(\boldsymbol{\Sigma}_w)} = \mathrm{tr}(\boldsymbol{\Sigma}_w\boldsymbol{X}_i^\top\boldsymbol{X}_j). \tag{9}$$

This is still one form of bilevel optimization [19], but by using the *unique* optimizer $\boldsymbol{\alpha}^*$ in (5), we finally obtain our proposed formulation from (2) as

$$\min_{\boldsymbol{\Sigma}_w\succcurlyeq 0}\left[J(\boldsymbol{\Sigma}_w) \triangleq \frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}_w) + \sum_i\alpha^*_{i(\boldsymbol{\Sigma}_w)} - \frac{1}{2}\sum_{i,j}\alpha^*_{i(\boldsymbol{\Sigma}_w)}\alpha^*_{j(\boldsymbol{\Sigma}_w)}y_i y_j K_{ij(\boldsymbol{\Sigma}_w)}\right], \tag{10}$$

$$\text{where } \boldsymbol{\alpha}^*_{(\boldsymbol{\Sigma}_w)} = \arg\max_{\boldsymbol{\alpha}\in\Theta_{\boldsymbol{y}}}\sum_i\alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j K_{ij(\boldsymbol{\Sigma}_w)}. \tag{11}$$

The optimization problem (10) is a single-level form by $\boldsymbol{\Sigma}_w$ and is *convex* (refer to supplementary material for the proof). Thus, we can obtain the global optimum $\boldsymbol{\Sigma}_w^*$, instead of $\boldsymbol{W}_w^*$, by means of the following gradient-descent approach.

By using Lemma 2 in [22] (see supplementary material for the detail), the derivative of J is given, as if $\boldsymbol{\alpha}^*_{(\boldsymbol{\Sigma}_w)}$ do not depend on $\boldsymbol{\Sigma}_w$, by

$$\frac{\partial J}{\partial\boldsymbol{\Sigma}_w} = \frac{1}{2}\left\{\boldsymbol{I} - \sum_{ij}\alpha^*_{i(\boldsymbol{\Sigma}_w)}\alpha^*_{j(\boldsymbol{\Sigma}_w)}y_i y_j\boldsymbol{X}_i^\top\boldsymbol{X}_j\right\}.$$

For optimization of (10), to ensure the positive semi-definiteness of $\boldsymbol{\Sigma}_w$, the projected gradient descent [20,21] is applied via the eigen decomposition of $\boldsymbol{\Sigma}_w - \eta\frac{\partial J}{\partial\boldsymbol{\Sigma}_w}$ and cutting off the negative eigenvalues and their eigenvectors:

$$\boldsymbol{\Sigma}_w^{old} - \eta\frac{\partial J}{\partial\boldsymbol{\Sigma}_w} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^\top = \sum_{i=1}^{w}\lambda_i\boldsymbol{v}_i\boldsymbol{v}_i^\top, \quad \boldsymbol{\Sigma}_w^{new} \leftarrow \sum_{i|\lambda_i>0}\lambda_i\boldsymbol{v}_i\boldsymbol{v}_i^\top = \boldsymbol{V}_+\boldsymbol{\Lambda}_+\boldsymbol{V}_+^\top, \tag{12}$$

where η is the step size determined by the line search [23] and $\lambda_i, \boldsymbol{v}_i$ are the i-th eigenvalue and eigenvector, respectively. The variables $\boldsymbol{W}_w, \boldsymbol{W}_h$ are easily retrieved by using those positive eigenvalues and eigenvectors: $\boldsymbol{W}_w = \boldsymbol{V}_+ \boldsymbol{\Lambda}_+^{\frac{1}{2}}$, and $\boldsymbol{W}_h = \sum_i \alpha_i^* (\boldsymbol{V}_+ \boldsymbol{\Lambda}_+ \boldsymbol{V}_+^\top) y_i \boldsymbol{X}_i \boldsymbol{V}_+ \boldsymbol{\Lambda}_+^{\frac{1}{2}}$.

The essential rank of the classifier $\boldsymbol{W} = \boldsymbol{W}_h \boldsymbol{W}_w^\top$ is usually less than r, and the redundant ranks are eventually eliminated by assigning zero singular values through the optimization in (10) which corresponds to minimization of the trace norm $||\boldsymbol{W}_h \boldsymbol{W}_w^\top||_\Sigma$, even if starting with the full rank $\boldsymbol{\Sigma}_w$.

In the proposed formulation (10), the dimensionality of the variable $\boldsymbol{\Sigma}_w$, $O(w^2)$, is much smaller than that of $\tilde{\boldsymbol{W}}$, $O((h+w)^2)$, in SDP (3). The computational complexity of the above optimization procedure is solely dependent on that of the quadratic programming (QP) in (11). For solving the QP, we can apply the efficient off-the-shelf SVM solver, such as libsvm [24]. The computationally exhaustive step other than the QP is the eigen decomposition in (12). In practice, however, either the dimension h, w is low; e.g., high-dimensional features are extracted at a few points $(h \gg w)$, or either of the dimensionalities can be reduced such as by applying PCA in advance. Thus, the computational cost of the eigen decomposition is negligible compared to those of QP in most cases.

2.3 Comparison to Related Works

The formulation of the linear SVM [10] is actually regarded as the minimization of the Frobenius norm $\|\boldsymbol{W}\|_F = \mathrm{tr}(\boldsymbol{W}^\top \boldsymbol{W})$, but it usually results in a full rank classifier. The Frobenius norm corresponds to the L_2 norm of the singular values $\boldsymbol{\sigma}$ and the obtained classifier tends to have dense singular values, resulting in the full-rank classifier, unlike the trace norm (the L_1 norm of $\boldsymbol{\sigma}$). Therefore, to achieve low-rank SVM, Wolf *et al.* [15] and Pirsiavash *et al.* [14] additionally introduced a hard constraint on the rank, $\mathrm{rank}(\boldsymbol{W}) \leq k$, into the formulation using $\|\boldsymbol{W}\|_F$. Those methods, however, usually produce the classifier of the maximum rank under that constraint, *i.e.*, $\mathrm{rank}(\boldsymbol{W}) = k$, since the objective cost $\|\boldsymbol{W}\|_F$ is the same as in the SVM, and it is difficult to determine the optimal rank k in advance.

The bilinear models are also addressed in the literature of the collaborative filtering [17,25,26]. In those methods, the trace norm $||\boldsymbol{W}||_\Sigma$ was employed as the objective cost, but the authors also mentioned that the resulting SDP is exhaustive and difficult to solve for large-scale samples. Thus, Loeff and Farhadi [26] and Rennie and Srebro [25] applied the approximated optimization approaches, though resulting in local minima.

Compared to the above methods, in terms of optimization, our contribution is to formulate the bilinear classification problem in the tractable convex form (10) and to propose the computationally efficient optimization procedure to solve it. Without any hard constraint on the rank, the proposed method can automatically produce the bilinear classifier of the optimal low rank.

3 Extensions of Bilinear Classification

3.1 Smoothing Regularization

Either or both of the bilinear weights, $\boldsymbol{W}_h, \boldsymbol{W}_w$, are occasionally connected to physical properties; *e.g.*, $\boldsymbol{W}_w$ works as weights on spatio-temporal positions, while $\boldsymbol{W}_h$ is for the feature vector. In such cases, it is useful to take into account the (physical) relationships between the weight components in terms of regularization for improving the generalization performance. For that purpose, we introduce the smoothing regularization. In the case of time-series, the extracted features are not independently drawn but naturally have *continuity* between adjacent features, which expects the smooth weights $\boldsymbol{W}_w$. The smoothing regularization is expressed by using the quadratic form derived from Laplacian of the weights, and the formulation results in

$$\begin{aligned}\min_{\boldsymbol{W}_w,\boldsymbol{W}_h,b} & \frac{1}{2}\mathrm{tr}(\boldsymbol{W}_w\boldsymbol{W}_w^\top)+\frac{1}{2}\mathrm{tr}(\boldsymbol{W}_h\boldsymbol{W}_h^\top)+\frac{1}{2}C_w\mathrm{tr}(\boldsymbol{W}_w^\top\boldsymbol{L}_w\boldsymbol{W}_w)+\frac{1}{2}C_h\mathrm{tr}(\boldsymbol{W}_h^\top\boldsymbol{L}_h\boldsymbol{W}_h)\\ &+C\sum_i\max\left[0, 1-y_i\{\mathrm{tr}(\boldsymbol{W}_h^\top\boldsymbol{X}_i\boldsymbol{W}_w)+b\}\right],\end{aligned}$$

where $\boldsymbol{L}_w, \boldsymbol{L}_h$ are the matrices measuring the smoothness and C_w, C_h are regularization parameters. For time-series (one-way), the matrix $\boldsymbol{L}_w$ is determined based on $\mathrm{tr}(\boldsymbol{W}_w^\top\boldsymbol{L}_w\boldsymbol{W}_w) = \sum_t \| -\boldsymbol{w}_{w,t-1} + 2\boldsymbol{w}_{w,t} - \boldsymbol{w}_{w,t+1}\|^2$. In this study, the regularization parameter C_w is set so as to equally balance the two spectral matrix norms of $\boldsymbol{I}$ and $\boldsymbol{L}_w$ by $C_w = 1/||\boldsymbol{L}_w||_s$, ($C_h = 1/\|\boldsymbol{L}_h\|_s$ for $\boldsymbol{L}_h$), where $\|\boldsymbol{L}_w\|_s$ indicates the spectral matrix norm (the maximum singular value) of $\boldsymbol{L}_w$. Then, the above formulation is rewritten to the following form similar to (4):

$$\begin{aligned}\min_{\bar{\boldsymbol{W}}_w,\bar{\boldsymbol{W}}_h,b} & \frac{1}{2}\mathrm{tr}(\bar{\boldsymbol{W}}_w\bar{\boldsymbol{W}}_w^\top)+\frac{1}{2}\mathrm{tr}(\bar{\boldsymbol{W}}_h\bar{\boldsymbol{W}}_h^\top)+C\sum_i\max\left[0, 1-y_i\{\mathrm{tr}(\bar{\boldsymbol{W}}_h^\top\bar{\boldsymbol{X}}_i\bar{\boldsymbol{W}}_w)+b\}\right],\\ \text{where } & \bar{\boldsymbol{X}}_i = (\boldsymbol{I}+C_h\boldsymbol{L}_h)^{-\frac{1}{2}}\boldsymbol{X}_i(\boldsymbol{I}+C_w\boldsymbol{L}_w)^{-\frac{1}{2}}.\end{aligned}$$

The optimization procedure in Sec.2.2 is applicable to it and we obtain the smoothed classifier weights $\boldsymbol{W}_h = (\boldsymbol{I}+C_h\boldsymbol{L}_h)^{-\frac{1}{2}}\bar{\boldsymbol{W}}_h,\ \boldsymbol{W}_w = (\boldsymbol{I}+C_h\boldsymbol{L}_w)^{-\frac{1}{2}}\bar{\boldsymbol{W}}_w$.

3.2 Heterogeneous Multiple Kernel Learning

By considering a kernel-based extension of the bilinear classifier, we naturally apply the proposed method to multiple kernel learning (MKL) [16], which induces a novel MKL, called *heterogeneous multiple kernel learning.*

For kernelization, we employ the kernels for $\boldsymbol{R}^{ij} \triangleq \boldsymbol{X}_i^\top\boldsymbol{X}_j \in \mathbb{R}^{w\times w}$ in (9). In this case, the optimized $\boldsymbol{\Sigma}_w$ works as weights for the multiple ($w\times w$ types) kernels in $\boldsymbol{R}^{ij}$ to produce a new composite kernel in (9), which is closely related to MKL [16]. In the followings, we consider the column feature vectors $\boldsymbol{x}_{ic}$ ($c=1,\cdots,w$) in $\boldsymbol{X}_i = [\boldsymbol{x}_{i1},\cdots,\boldsymbol{x}_{iw}]$. When the feature vectors $\boldsymbol{x}_{ic}$ are all homogeneous, an identical kernel function is simply applied as $R_{cd}^{ij} = k(\boldsymbol{x}_{ic}, \boldsymbol{x}_{jd})$. The proposed bilinear model can also deal with multiple types of kernels (features), which induces heterogeneous MKL (hMKL), as follows. Given respective types of kernels k_c, the main concern in the hMKL is to construct kernels, especially for the off-diagonal elements of $\boldsymbol{R}^{ij}$; those are inter kernels between heterogeneous types of kernels (features), which is a novel concept in this paper. We propose the following form of the kernels for $\boldsymbol{R}^{ij}$,

$$R^{ij}_{cd} = \boldsymbol{k}_{ic}^{\top}\boldsymbol{K}_c^{-\frac{1}{2}}\boldsymbol{K}_d^{-\frac{1}{2}}\boldsymbol{k}_{id}, \tag{13}$$

where $\boldsymbol{K}_c$ is the c-th type of kernel Gram matrix, $\boldsymbol{K}_c = \{k_c(\boldsymbol{x}_{ic}, \boldsymbol{x}_{jc})\}_{i=1,..,n}^{j=1,..,n} \in \mathbb{R}^{n\times n}$, and the kernel feature vector $\boldsymbol{k}_{ic} = [k_c(\boldsymbol{x}_{ic}, \boldsymbol{x}_{1c}), \cdots, k_c(\boldsymbol{x}_{ic}, \boldsymbol{x}_{nc})]^{\top} \in \mathbb{R}^n$ (the i-th column vector of $\boldsymbol{K}_c$). We give an interpretation to this formulation of the kernel (13) as follows.

For the intra kernels between homogeneous types of kernels, each type of feature vector $\boldsymbol{\phi}_{ic}$ associated with $\boldsymbol{x}_{ic}$ in the reproducing kernel Hilbert space (RKHS) is projected into the subspace by PCA [13],

$$R^{ij}_{cc} = \boldsymbol{k}_{ic}^{\top}\boldsymbol{K}_c^{-\frac{1}{2}}\boldsymbol{K}_c^{-\frac{1}{2}}\boldsymbol{k}_{jc} = \boldsymbol{\phi}_{ic}^{\top}\boldsymbol{V}_c\boldsymbol{V}_c^{\top}\boldsymbol{\phi}_{jc},$$

where $\boldsymbol{\Phi}_c = [\boldsymbol{\phi}_{1c}, \cdots, \boldsymbol{\phi}_{nc}] = \boldsymbol{V}_c\boldsymbol{\Lambda}_c\boldsymbol{U}_c^{\top}$ (SVD), and $\boldsymbol{k}_{ic} = \boldsymbol{\Phi}_c^{\top}\boldsymbol{\phi}_{ic}$. Note that $\boldsymbol{K}_c = \boldsymbol{U}_c\boldsymbol{\Lambda}_c^2\boldsymbol{U}_c^{\top}$ and $\boldsymbol{V}_c$ is the projection matrix by PCA in RKHS. Especially, for the training samples, the diagonal elements of $\boldsymbol{R}^{ij}$ are identical to the original kernels; $R^{ij}_{cc} = k_c(\boldsymbol{x}_{ic}, \boldsymbol{x}_{jc})$. When dealing with only the diagonal matrix of R^{ij}_{cc}, the proposed bilinear method (10) corresponds to the MKL method of [21].

On the other hand, the inter kernels between heterogeneous types of kernels are represented by

$$R^{ij}_{cd} = \boldsymbol{k}_{ic}^{\top}\boldsymbol{K}_c^{-\frac{1}{2}}\boldsymbol{K}_d^{-\frac{1}{2}}\boldsymbol{k}_{jd} = \boldsymbol{\phi}_{ic}^{\top}\boldsymbol{V}_c\boldsymbol{U}_c^{\top}\boldsymbol{U}_d\boldsymbol{V}_d^{\top}\boldsymbol{\phi}_{jd}. \tag{14}$$

In this form, the consistency between the different types of kernels is ensured via canonical correlation analysis (CCA). The CCA provides the projections $\boldsymbol{A}, \boldsymbol{B}$ for the two types of features $\boldsymbol{\Phi}_c, \boldsymbol{\Phi}_d$ in RKHS so as to maximize the correlation coefficient, by solving the following eigenvalue problem:

$$\begin{bmatrix} \boldsymbol{0} & \boldsymbol{\Phi}_c\boldsymbol{\Phi}_d^{\top} \\ \boldsymbol{\Phi}_d\boldsymbol{\Phi}_c^{\top} & \boldsymbol{0} \end{bmatrix}\begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{B} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Phi}_c\boldsymbol{\Phi}_c^{\top} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Phi}_d\boldsymbol{\Phi}_d^{\top} \end{bmatrix}\begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{B} \end{bmatrix}\boldsymbol{\Omega}, \ \therefore \boldsymbol{A} = \boldsymbol{V}_c\boldsymbol{\Lambda}_c^{-1}\boldsymbol{P}, \ \boldsymbol{B} = \boldsymbol{V}_d\boldsymbol{\Lambda}_d^{-1}\boldsymbol{Q},$$

where $\boldsymbol{U}_c^{\top}\boldsymbol{U}_d = \boldsymbol{P}\boldsymbol{\Omega}\boldsymbol{Q}^{\top}$ (SVD). In the CCA, the feature vectors $\boldsymbol{\phi}_{ic}, \boldsymbol{\phi}_{jd}$ are first whitened by $\boldsymbol{V}_c\boldsymbol{\Lambda}_c^{-1}, \boldsymbol{V}_d\boldsymbol{\Lambda}_d^{-1}$ via PCA, and then the PCA axes are rotated by $\boldsymbol{P}, \boldsymbol{Q}$ so as to ensure the consistency and to maximize the correlation coefficient. By using these notations, (14) is rewritten to

$$R^{ij}_{cd} = \boldsymbol{\phi}_{ic}^{\top}\boldsymbol{V}_c\boldsymbol{P}\boldsymbol{\Omega}^{\frac{1}{2}}\boldsymbol{\Omega}^{\frac{1}{2}}\boldsymbol{Q}^{\top}\boldsymbol{V}_d^{\top}\boldsymbol{\phi}_{jd}.$$

This is quite similar to the CCA projection; The feature vectors $\boldsymbol{\phi}_{ic}, \boldsymbol{\phi}_{jd}$ are projected into PCA subspaces by $\boldsymbol{V}_c, \boldsymbol{V}_d$, and then they are rotated by the CCA rotation matrices $\boldsymbol{P}, \boldsymbol{Q}$ with weighting the CCA axes by the correlation coefficients $\boldsymbol{\Omega}^{\frac{1}{2}}$. The differences from the CCA projections are that 1) we use the orthogonal PCA projection, not whitening, for preserving the magnitude (norm) of $\boldsymbol{\phi}_{ic}, \boldsymbol{\phi}_{jd}$ and then 2) we use the weighting by the correlation coefficients $\boldsymbol{\Omega}$ which measure consistency between the heterogeneous kernels.

As a result, in hMKL, we deal with the feature matrix which is explicitly represented by the following form;

$$\boldsymbol{X} = [\boldsymbol{K}_1^{-\frac{1}{2}}\boldsymbol{k}_1, \cdots, \boldsymbol{K}_w^{-\frac{1}{2}}\boldsymbol{k}_w] \in \mathbb{R}^{n\times w}, \ \ \boldsymbol{k}_c = [k_c(\boldsymbol{x}_c, \boldsymbol{x}_{1c}), \cdots, k_c(\boldsymbol{x}_c, \boldsymbol{x}_{nc})]^{\top} \in \mathbb{R}^n.$$

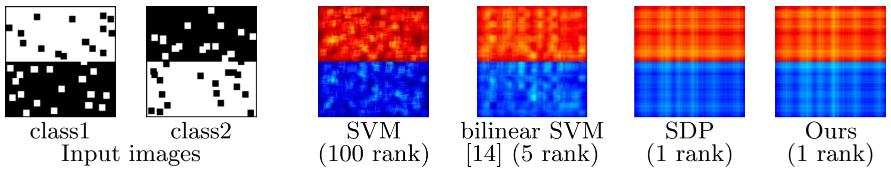


Fig. 2. Classifier weights on toy data. Negative and positive weights are shown by pseudo colors from blue to red.

Thus, the resultant bilinear classifier is also represented in the bilinear form (1) and the proposed optimization procedure (Sec.2.2) is directly applicable even to this hMKL. It should be noted again that $\boldsymbol{\Sigma}_w = \boldsymbol{W}_w \boldsymbol{W}_w^\top$ is regarded as the weights on both the intra and inter kernels.

4 Experimental Results

We applied the proposed bilinear classification method to various classification problems using feature arrays, co-occurrence features and multiple kernels.

For comparison, we also applied linear SVM [10] to the concatenated feature vectors and bilinear SVM [14] to the feature matrices except in multiple kernel learning. The bilinear SVM [14] includes an rank parameter for rank($\boldsymbol{W}$) $\leq k$ which is determined based on two-fold cross validations from $k \in \{5, 10\}$ as in [14]. In all the methods, to cope with multi-class problems, the one-vs-rest approach is employed. All the methods were implemented by MATLAB with libsvm [24] on 3.33GHz PC.

4.1 Toy Example

First, by using toy data, we intuitively demonstrate how the proposed method works on feature matrices. For two-class classification, two types of binary images (100×100) are used as feature matrices ($\boldsymbol{X} \in \mathbb{R}^{100\times100}$), as shown in Fig. 2; the images have basically one rank with salt and pepper noise of size 5×5, and there are 100 samples in each type (class). The obtained classifier ($\boldsymbol{W}$) is shown in Fig. 2, compared to those by the other method including SDP (3) which is feasible in such a small dataset by using SeDuMi solver. In the linear SVM, the concatenated linear classifier (vector) is folded into the intrinsic matrix form and its rank is also measured. The proposed method favorably produces the one-rank classifier which is the same as the global optimum one by SDP, while the classifiers by the methods of linear SVM and bilinear SVM [14] are overly fitted to the data with higher (full) rank. Without any hard constraint about the rank, the proposed method recovers the essential rank in data with a low computational cost (0.4 sec) compared to the exhaustive SDP method (14.0 sec).

4.2 Feature Array

Next, we conducted the practical experiments on motion classification using RWC gesture dataset [3] and image classification (detection) using INRIA person

Table 1. Classification performances for feature arrays

(a) RWC dataset

Method	Rank	Err. (%)
SVM	50	2.11
bilinear SVM [14]	5.59	1.41
Ours	3.27	**0.98**
Ours (smooth)	3.25	1.01
[4]	-	1.9
[3]	-	4.14

(b) INRIA dataset

Method	Rank	EER (%)
SVM	32	0.55
bilinear SVM [14]	10	0.71
Ours	12	0.62
Ours (smooth)	12	**0.53**
[27]	-	0.58
[1]	-	2.25

dataset [1]. In these experiments, the array of the feature vectors extracted at temporal/spatial points forms the feature matrix $\boldsymbol{X}$ as shown in Fig. 1ab.

RWC Gesture Dataset [3] contains 17 types of human gesture as shown in Table 1a. Each of these gestures is performed four times by 48 subjects (23 men and 25 women). We extract 751-dimensional CHLAC motion feature vector [3] at every frame with multiple correlation intervals $\Delta r \in \{1, 3, 5\}$; for details of this feature, refer to [3]. Since the numbers of frames are different across the motion image sequences, we subsample those frame-based features by bilinear interpolation into 50-frame feature vectors. As a result, the feature matrix is formed as feature-vs-time; $\boldsymbol{X} \in \mathbb{R}^{751\times 50}$ (Fig. 1a). The performance is evaluated by three-fold cross validation, and both the error rates and the ranks of the classifiers averaged across classes are shown in Table 1a. The proposed method produces the favorable performance compared to the other methods of SVM and bilinear SVM [14] and the prior works [4,3]. While the rank of the SVM classifier is 50, the proposed classifier has around only three rank, improving the efficiency of information compression and also the generalization performance. In this case, the smoothed classifier (Sec.3.1) that imposes the smoothing regularization on the temporal weight $\boldsymbol{W}_w$ produces slightly inferior performance. This is because the feature matrices are already smoothed by the subsampling procedure and such further smoothing slightly degrades the performance.

In **INRIA Person Dataset** [1], we used 2,416 person and 12,180 person-free images (64×128) for training, and 1,132 person and 13,590 person-free images for test as shown in Table 1b. The image is divided into 4×8 subregions and 324-dimensional GLAC feature vectors [27] are extracted from each region with the same parameter settings as in [27]. Thereby, the feature matrix is formed as feature-vs-space (positions); $\boldsymbol{X} \in \mathbb{R}^{324\times 32}$ (Fig. 1b). The performance results are shown in Table 1b. In the proposed method, the smoothed classifier that imposes the regularization on the (two-way) spatial positions $\boldsymbol{W}_w$ slightly improves the performance. The performance is comparable to SVM and is superior to the prior works [27,1]. It should be noted that the proposed classifier of low rank ($= 12$) significantly reduces the computational cost in the detection stage since our low-rank classifier is decomposed into a few separable filters which require $O(\text{rank}(\boldsymbol{W}) \times \max(h, w))$ compared to $O(hw)$ of the full rank SVM classifier.

4.3 Co-occurrence Feature

The bilinear classifier can also directly deal with the co-occurrence features which are inherently formed as a matrix (Fig. 1c). In the bag-of-features [2], the local features, such as HOG [1], are assigned with (visual) words via clustering, and the simple occurrence of the words are counted to produce the final histogram-based feature vector. The occurrence features are extended to the co-occurrence features of the words in local neighborhoods in the bag-of-features framework [28]. For motion recognition, we follow the framework used in [4] which extracts frame-based motion features at every 10 frames and applies k-means to cluster the features into motion words in Fisher discriminant space; refer to [4] for more details. In these experiments, we count the co-occurrence of the motion words along the time axis, as follows. Given w motion words, the frame-based feature at time t is assigned with multiple motion words of which weights form a vector $\boldsymbol{f}(t) \in \mathbb{R}^w$ [4]. The co-occurrence features are extracted by $\boldsymbol{F}(\varDelta t) = \sum_t \boldsymbol{f}(t)\boldsymbol{f}(t+\varDelta t)^\top$, in which $\varDelta t$ denotes the interval along the t axis, say $\varDelta t \in \{0, 20\}$ in these experiments, and those features are concatenated into the final feature matrix $\boldsymbol{X} = [\boldsymbol{F}(0), \boldsymbol{F}(20)]^\top \in \mathbb{R}^{2w \times w}$. The number of words is simply determined by $w = 10 \times \#\text{class}$. We conducted the motion classification experiments by using UCF sport action dataset [29] and Cambridge hand gesture dataset [30]. In the smoothed classifier (Sec.3.1), the matrices $\boldsymbol{L}_h, \boldsymbol{L}_w$ are set as the graph Laplacian [31] for which the similarities between words are measured by 10 nearest neighbors of word centers.

UCF Sport Action Dataset [29] contains nine types of sport actions as shown in Table 2a. Each action is performed by several (about 17) players, and the total number of sequence is 150. To enlarge the training size, the number of training samples are doubled by adding horizontal mirror images. For evaluation, three-fold cross validation is applied and the averaged error rates across action classes are reported in Table 2a. The performances reported in the prior works [29,14], which are measured in a slightly different protocol (slightly different number of action classes), are also shown as a reference. The proposed methods exhibit superior performances to the other methods, and in particular, the smoothed bilinear classifier improves the performance with quite low rank. In Fig. 3a, we show the performances by increasing the number of words to evaluate the robustness of the methods. In the proposed method, due to the appropriate low rank, the performances are stably high even for larger number of words, while the other methods degrade their performances.

Cambridge Hand Gesture Dataset [30] contains nine hand gestures defined by three primitive hand shapes and three primitive motions, which are performed ten times by two subjects under five different illumination conditions, as shown in Table 2b. We used the sequences acquired under the plain illumination condition for training and those under the remaining four conditions for test. The averaged error rates across all gesture classes over the four test conditions are reported in Table 2b. The proposed method produces superior performances to the others including the prior works [4,30]; the smoothed classifier is the most favorable. Fig. 3b also shows the performances for various numbers of words, demonstrating the robustness of the proposed method as is the case with UCF dataset.

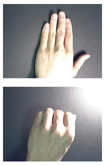
These experimental results show that the proposed method robustly produces high performances, requiring users only to set sufficiently large number of words

Table 2. Classification performances for co-occurrence features

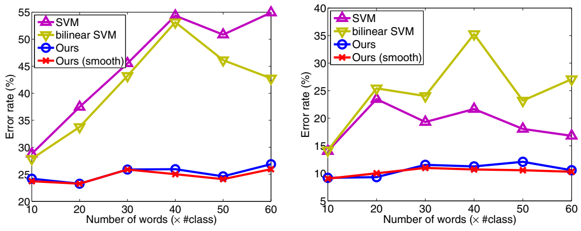
(a) UCF dataset

Method	Rank	Err. (%)
SVM	246.67	28.80
bilinear SVM [14]	9.63	27.87
Ours	1.15	24.19
Ours (smooth)	1.11	**23.73**
[29]	-	35.2
[14]	-	30.8

(b) Cambridge dataset



Method	Rank	Err. (%)
SVM	251	14.03
bilinear SVM [14]	10	14.17
Ours	1.22	9.17
Ours (smooth)	1.33	**9.03**
[4]	-	11
[30]	-	18



(a) UCF dataset (b) Cambridge dataset

Fig. 3. Performances for various numbers of words

without carefully tuning the number of words nor the rank of the classifier for classifying the co-occurrence features.

4.4 Multiple Kernels

At the last, we applied the proposed method to classification using multiple kernels (Sec.3.2) and compared the performance to those of the other MKL methods: simpleMKL [20] and the method of [21]. It should be noted again that the proposed method dealing with only diagonal matrix of $R^{ij}_{cc} = k_c(\boldsymbol{x}_{ic}, \boldsymbol{x}_{jc})$ corresponds to the MKL method [21] and it is denoted by "diagonal ([21])".

To demonstrate the effectiveness of the proposed kernel (13), we conducted the comparative experiment using **PASCAL VOC 2007 dataset**[1] which contains 5,011 images for training and 4,952 images for test in 20 object categories. We used 15 types of precomputed features provided in the website[2] of the authors [32] (for deals of the features, refer to [32]), and employed the RBF kernels of those features as $k_c(\boldsymbol{x}_{ic}, \boldsymbol{x}_{jc}) = \exp(-||\boldsymbol{x}_{ic} - \boldsymbol{x}_{jc}||^2/\gamma)$ where γ is the mean of pairwise distances. The following alternative forms to the proposed kernel (13) are conceivable.

$$\begin{aligned} \textit{Product}:\ R^{ij}_{cd} &= \boldsymbol{k}_{ic}^\top \boldsymbol{k}_{jd}, \\ \textit{PCA}:\ R^{ij}_{cd} &= \boldsymbol{k}_{ic}^\top \boldsymbol{U}_c \boldsymbol{\Lambda}_c^{-1} \boldsymbol{\Lambda}_d^{-1} \boldsymbol{U}_d^\top \boldsymbol{k}_{jd} = \boldsymbol{\phi}_{ic}^\top \boldsymbol{V}_c \boldsymbol{V}_d^\top \boldsymbol{\phi}_{jd}, \\ \textit{Inverse}:\ R^{ij}_{cd} &= \boldsymbol{k}_{ic}^\top \boldsymbol{K}_c^{-1} \boldsymbol{K}_d^{-1} \boldsymbol{k}_{jd} = \boldsymbol{\phi}_{ic}^\top \boldsymbol{V}_c \boldsymbol{\Lambda}_c^{-1} \boldsymbol{U}_c^\top \boldsymbol{U}_d \boldsymbol{\Lambda}_d^{-1} \boldsymbol{V}_d^\top \boldsymbol{\phi}_{jd}. \end{aligned}$$

Compared to (13, 14), the *PCA* kernel loses the CCA rotation matrix derived from $\boldsymbol{U}_c$, while the *inverse* kernel additionally introduces the whitening by $\boldsymbol{\Lambda}_c^{-1}$

[1] `http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html`
[2] `http://lear.inrialpes.fr/people/guillaumin/data.php`

Table 3. PASCAL VOC 2007

(a) Kernel types

Kernel	mAP (%)
Product	42.30
PCA	45.60
Inverse	37.88
Proposed	**48.64**

(b) Comparison

Method	mAP (%)
simpleMKL [20]	48.04
diagonal ([21])	45.83
Ours	**48.64**

Table 4. Flower dataset

Method	Acc. (%)
simpleMKL [20]	81.08
simpleMKL (multiclass) [20]	82.55
diagonal ([21])	85.10
[33]	85.5
Ours	**86.47**

Table 5. Butterfly dataset

Method	Acc. (%)
simpleMKL [20]	69.60
simpleMKL (multiclass) [20]	67.95
diagonal ([21])	74.42
Ours	**76.54**

Table 6. Bird dataset

Method	Acc. (%)
simpleMKL [20]	69.21
simpleMKL (multiclass) [20]	68.71
diagonal ([21])	72.85
Ours	**74.50**

to normalize the magnitudes (norms) of features. Table 3a shows the mean of average precision rates (mAP) across the categories. The proposed kernel (13) is superior to the other types of kernels, showing that both the rotation matrix by CCA and magnitude preserving by PCA are effective for classifications. Table 3b shows that the proposed method produces the favorable performance compared to the others.

We further conducted the MKL experiments using Oxford flower dataset [34], Butterfly dataset [35] and Bird dataset [36]. We used the RBF kernel in the same manner as described above. The **Oxford flower dataset** [34] is composed of 80 images of 17 flower categories. We used the seven types of precomputed pairwise distances provided in the website[3] of the authors [37]; for the details of the distances, refer to [34,37]. Table 4 shows the performances evaluated by three-fold cross validations using the same predefined splits as in [34]. For comparison, the performance of the method [33] which uses the same features and cross validation splits is also shown. The **Butterfly dataset** [35] has 619 images of seven butterfly classes, and the **Bird dataset** [36] contains six bird classes with 100 images per class. In these datasets, we used the three types of precomputed pairwise distances provided in the website[4] of the authors [38]; for the details of the distances, refer to [38]. The classification accuracies are evaluated by three-fold cross validations and are shown in Table 5 and Table 6. In those three datasets, the proposed method produces superior performances to the others. Especially, in comparison to diagonal [21], the inter kernels between heterogeneous kernels effectively contribute to improve the performances.

As shown in above experimental results, the proposed heterogeneous MKL method, which combines both intra (diagonal) and inter (off-diagonal) kernels by using the bilinear model, is effective compared to the standard MKL methods.

[3] `http://www.robots.ox.ac.uk/~vgg/research/flowers/index.html`

[4] `http://www.eecs.berkeley.edu/Research/Projects/CS/vision/shape/msorec/`

5 Conclusion

We have proposed an efficient method to optimize a bilinear classifier for feature matrices which results in low rank without any hard constraint on the rank. The classifier is optimized by minimizing the trace norm of the classifier matrix, which contributes to the rank reduction. The optimization is formulated in a tractable convex form and an computationally efficient optimization procedure is proposed. In addition, by considering a kernel-based extension of the bilinear method, a novel multiple kernel learning, called heterogeneous multiple kernel learning (hMKL), is induced. In the hMKL, we can combine not only the ordinary kernels of homogeneous kernels (features) but also the inter kernels between heterogeneous kernels (features) into a new composite kernel by using the bilinear model. In the experiments on various classification problems using feature arrays, co-occurrence feature matrices and multiple kernels, the proposed method exhibited favorable performances compared to the other methods.

References

1. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR (2005)
2. Csurka, G., Bray, C., Dance, C., Fan, L.: Visual categorization with bags of keypoints. In: ECCV Workshop on Statistical Learning in Computer Vision (2004)
3. Kobayashi, T., Otsu, N.: A three-way auto-correlation based approach to motion recognition. Pattern Recognition Letters 30, 185–192 (2009)
4. Kobayashi, T., Otsu, N.: Motion recognition using local auto-correlation of space-time gradients. Pattern Recognition Letters 33, 1188–1195 (2012)
5. Tenenbaum, J.B., Freeman, W.T.: Separating style and content with bilinear models. Neural Computation 12, 1247–1283 (2000)
6. Yang, J., Zhang, D., Frangi, A.F., Yang, J.: Two-dimensional pca: A new approach to appearance-based face representation and recognition. IEEE Trans. on Pattern Analysis and Machine Intelligence 26, 131–137 (2004)
7. Ye, J., Janardan, R., Li, Q.: Two-dimensional linear discriminant analysis. In: Advances in Neural Information Processing System, vol. 17, pp. 1569–1576 (2005)
8. Lee, D.D., Seung, H.S.: Learning the parts of objects by non-negative matrix factorization. Nature 401, 788–791 (1999)
9. Eriksson, A., van den Hengel, A.: Efficient computation of robust low-rank matrix approximations in the presence of missing data using the l_1 norm. In: CVPR (2010)
10. Vapnik, V.N.: Statistical Learning Theory. Wiley (1998)
11. Graepel, T., Herbrich, R., Schölkopf, B., Smola, A., Bartlett, P., Muüller, K.-R., Obermayer, K., Williamson, R.: Classification on proximity data with lp-machines. In: ICANN (1999)
12. Bartlett, P.J., Schölkopf, B., Schuurmans, D., Smola, A.J.: Advances in Large-Margin Classifiers. MIT Press (2000)
13. Schölkopf, B., Smola, A.J.: Learning with Kernels. MIT Press (2001)
14. Pirsiavash, H., Ramanan, D., Fowlkes, C.: Bilinear classifiers for visual recognition. In: Advances in Neural Information Processing Systems (2009)
15. Wolf, L., Jhuang, H., Hazan, T.: Modeling appearances with low-rank svm. In: CVPR (2007)

16. Lanckriet, G., Cristianini, N., Bartlett, P., Ghaoui, L.E., Jordan, M.I.: Learning the kernel matrix with semidefinite programming. Journal of Machine Learning Research 5, 27–72 (2004)
17. Srebro, N., Rennie, J.D.M., Jaakkola, T.S.: Maximum-margin matrix factorization. In: Advances in Neural Information Processing Systems, vol. 17, pp. 1329–1336 (2005)
18. Boyd, S.P., Vandenberghe, L.: Convex optimization. Cambridge university press, Cambridge (2004)
19. Dempe, S.: Foundations of Bilevel Programming. Kluwer Academic Publishers (2002)
20. Rakotomamonjy, A., Bach, F.R., Canu, S., Grandvalet, Y.: SimpleMKL. Journal of Machine Learning Research 9, 2491–2521 (2008)
21. Varma, M., Ray, D.: Learning the discriminative power-invariance trade-off. In: ICCV (2007)
22. Chapelle, O., Vapnik, V., Bousquet, O., Mukherjee, S.: Choosing multiple parameters for support vector machines. Machine Learning 46, 131–159 (2002)
23. Nocedal, J., Wright, S.J.: Numerical optimization. Springer, New York (1999)
24. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines (2001)
25. Rennie, J.D.M., Srebro, N.: Fast maximum margin matrix factorization for collaborative prediction. In: ICML (2005)
26. Loeff, N., Farhadi, A.: Scene Discovery by Matrix Factorization. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part IV. LNCS, vol. 5305, pp. 451–464. Springer, Heidelberg (2008)
27. Kobayashi, T., Otsu, N.: Image Feature Extraction Using Gradient Local Auto-Correlations. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part I. LNCS, vol. 5302, pp. 346–358. Springer, Heidelberg (2008)
28. Ling, H., Soatto, S.: Proximity distribution kernels for geometric context in category recognition. In: ICCV (2007)
29. Rodriguez, M.D., Ahmed, J., Shah, M.: Action mach a spatio-temporal maximum average correlation height filter for action recognition. In: CVPR (2008)
30. Kim, T.-K., Wong, S.-F., Cipolla, R.: Tensor canonical correlation analysis for action classification. In: CVPR (2007)
31. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Computation 15, 1373–1396 (2003)
32. Guillaumin, M., Verbeek, J., Schmid, C.: Multimodal semi-supervised learning for image classification. In: CVPR (2010)
33. Gehler, P., Nowozin, S.: On feature combination for multiclass object classification. In: ICCV (2009)
34. Nilsback, M.E., Zisserman, A.: A visual vocabulary for flower classification. In: CVPR (2006)
35. Lazebnik, S., Schmid, C., Ponce, J.: Semi-local affine parts for object recognition. In: BMVC (2004)
36. Lazebnik, S., Schmid, C., Ponce, J.: A maximum entropy framework for part-based texture and object recognition. In: CVPR (2005)
37. Nilsback, M.E., Zisserman, A.: Automated flower classification over a large number of classes. In: ICVGIP (2008)
38. Mario Christoudias, C., Urtasun, R., Salzmann, M., Darrell, T.: Learning to Recognize Objects from Unseen Modalities. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part I. LNCS, vol. 6311, pp. 677–691. Springer, Heidelberg (2010)