Feature Extraction for
Multiple Kernel Learning

Brent Castle\textsuperscript{a}    Minh Tang\textsuperscript{b}    Michael W. Trosset\textsuperscript{c}

December 21, 2009

Technical Report 09-04
Department of Statistics
Indiana University
Bloomington, IN

\textsuperscript{a}School of Informatics & Computing, Indiana University. Email: bcastle@cs.indiana.edu
\textsuperscript{b}School of Informatics & Computing, Indiana University. Email: mhtang@cs.indiana.edu
\textsuperscript{c}Department of Statistics, Indiana University. Email: mtrosset@indiana.edu
Feature Extraction for
Multiple Kernel Learning

Brent Castle∗  Minh Tang†  Michael W. Trosset‡

December 21, 2009

Abstract

Multiple Kernel Learning (MKL) synthesizes a single kernel from a set of multiple kernels for use in a support vector machine. We propose that MKL be preceded by feature extraction. Given a set of kernels and a vector $y$ of class labels, Multiple Kernel Basis Extraction (MKBE) constructs orthogonal vectors $\{v_1, \ldots, v_m\}$ whose corresponding kernels, $\{v_1 v_1^T, \ldots, v_m v_m^T\}$, are maximally aligned with $yy^T$. Each of these vectors maximizes a Rayleigh quotient with respect to one of the given kernels, subject to orthogonality constraints. Standard MKL techniques can then be applied to the extracted set of rank-one kernels. Theoretical considerations suggest that preliminary feature extraction may improve classifier performance. Examples illustrate that the improvement can be substantial.

1 Introduction

The standard construction of a support vector machine (SVM) requires a kernel (inner product) matrix. In many supervised learning tasks, multiple kernels are available for consideration. Multiple Kernel Learning (MKL) synthesizes a single kernel from a set of multiple kernels for the purpose of constructing an SVM. To date, most techniques for MKL have formed linear (often convex) combinations of the given kernels.

Kernels provided by domain experts are typically of high rank, corresponding to high-dimensional feature spaces. It is widely appreciated that performance on a supervised learning task in a high-dimensional feature space may benefit from preliminary feature selection or extraction. This is a form of regularization: the idea is to use only essential information in constructing the classifier. Stated

∗School of Informatics & Computing, Indiana University, Bloomington, IN 47405. Email: bcastle@cs.indiana.edu
†School of Informatics & Computing, Indiana University, Bloomington, IN 47405. Email: mhtang@cs.indiana.edu
‡Department of Statistics, Indiana University, Bloomington, IN 47405. Email: mtrosset@indiana.edu
another way, statistical models of many dimensions often have low bias but high variance. Dimension reduction may substantially decrease variance while only slightly increasing bias.

A variety of methods exist for supervised feature extraction. Although the rationale for extracting useful low-rank kernels from high-rank kernels is identical, less is known about how to do so. We propose Multiple Kernel Basis Extraction (MKBE), a technique by which the original high-dimensional kernels are replaced by a number of rank-one kernels. A standard MKL technique is then applied to the rank-one kernels to construct an SVM.

2 Preliminaries

Let $S = \{(x_j, y_j)\}_{j=1}^N$ be a set of learning instances, where $x_j \in \mathcal{X}$ and $y_j \in \{-1, 1\}$ is the label for $x_j$. By a classifier, we mean a function $f : \mathcal{X} \to \mathbb{R}$ with the interpretation that $f$ assigns label 1 to $x$ if $f(x) > 0$ and label $-1$ if $f(x) < 0$.

Assume that we are given a collection of $p$ kernels, $\mathcal{K} = \{K_1, \ldots, K_p\}$. A common approach to MKL is to construct a classifier of the form

$$f(x) = \sum_{j=1}^N \alpha_j^* K^*(x, x_j) + b^*,$$

where $K^*$ is a convex combination of the $K_i \in \mathcal{K}$, i.e.,

$$K^*(x, x_j) = \sum_{i=1}^p \mu_i K_i(x, x_j), \quad \mu_i \geq 0, \quad \sum_{i=1}^p \mu_i = 1. \tag{2}$$

Such classifiers can be constructed in a variety of ways. Lanckriet et al. [5] posed the problem as a quadratically constrained quadratic program. Bach et al. [1] showed that the problem formulated in [5] is equivalent to a mixed ($l_2, l_1$) norm regularization on the weight vector and proposed an SMO-like algorithm to solve it. Sonnenburg et al. [8] formulated the problem as a semi-infinite linear program (SILP). Their algorithm is an instance of a class of exchange methods for solving SILPs. Rakotomamonjy et al. [7] used $l_2$ norm regularization on the weight vector and proposed an algorithm based on gradient descent.

All of the above approaches manipulate entire kernels. In contrast, one might write each kernel as the sum of rank-one kernels, $K_i = \sum_{j=1}^N M_{ij}$, then apply any of the above approaches to (a subset of) the collection of rank-one kernels, $\mathcal{M} = \{M_{ij}\}$. This is a form of feature extraction. If most of the discriminating information contained in the $K_i$ can be represented by a small number of $M_{ij}$, then we anticipate that this procedure will produce better classifiers than procedures based on convex combinations of the $K_i$.

An obvious way to proceed is by spectral decomposition, i.e., by writing

$$K_i = \sum_{j=1}^N \lambda_{ij} \nu_{ij} \nu_{ij}^T, \tag{3}$$
where $\lambda_{ij}$ and $\nu_{ij}$ are the eigenvalues and corresponding eigenvectors of $K_i$. Combining (2) and (3), we obtain the representation
\[
K^* = \sum_{i=1}^{p} \sum_{j=1}^{N} \mu_i \lambda_{ij} \nu_{ij} \nu_{ij}^T.
\]

Typically, $Np$ will be prohibitively large. One can limit the number of eigenvectors extracted from each $K_i$, but it won’t do to simply select the eigenvectors that correspond to the largest eigenvalues: if some of the $K_i$ contain similar information (for example, multiple Gaussian kernels with slightly different bandwidths are sometimes used in computer vision [10, 9]), then the eigenvectors extracted from these $K_i$ may be nearly identical. To avoid this difficulty, we propose constructing rank-one kernels from a single set of orthogonal vectors. The following section explains how these vectors are extracted.

## 3 Multiple Kernel Basis Extraction

We seek to extract $V = \{v_1, \ldots, v_m\}$, a set of orthogonal vectors, from $K$, a set of kernels. To ensure that these vectors will have predictive value, we rely on the notion of kernel alignment. The alignment between kernels $K_1$ and $K_2$ is
\[
A(K_1, K_2) = \frac{\text{trace}(K_1 K_2)}{\sqrt{\text{trace}(K_1 K_1)} \sqrt{\text{trace}(K_2 K_2)}},
\]
the cosine of the angle between $K_1$ and $K_2$. This quantity was used for kernel selection in [2]. We might use other criteria, but it turns out that maximizing alignment reduces to simple calculations for maximizing Rayleigh quotients.

We divide the set of learning instances, $S$, into $S_{\text{train}}$ and $S_{\text{test}}$, then use $S_{\text{train}}$ to supervise the construction of $V$. Let $N_{\text{train}} = |S_{\text{train}}|$. Given $x \in \mathbb{R}^N$, we denote the components of $x$ that correspond to elements of $S_{\text{train}}$ by $\hat{x} \in \mathbb{R}^{N_{\text{train}}}$.

We construct $V$ sequentially. Initially, compute
\[
q_{1,i} = \arg\max_{x \neq 0} \frac{x^T K_i x}{x^T x},
\]
for $i = 1, \ldots, p$, then set
\[
v_1 = \arg\max_{q_{1,i}} A \left( [\hat{q}_{1,i} \hat{q}_{1,i}^T], [\hat{y} \hat{y}^T] \right).
\]
Subsequently, given $v_1, \ldots, v_k$, compute
\[
q_{k+1,i} = \arg\max_{x \neq 0, x \perp v_1, \ldots, v_k} \frac{x^T K_i x}{x^T x},
\]
then set
\[
v_{k+1} = \arg\max_{q_{k+1,i}} A \left( [\hat{q}_{k+1,i} \hat{q}_{k+1,i}^T], [\hat{y} \hat{y}^T] \right).
\]
Notice that the alignment between rank-one kernels \( \hat{q} \hat{q}^T \) and \( \hat{y} \hat{y}^T \) is just a Raleigh quotient:

\[
A(\hat{q} \hat{q}^T, \hat{y} \hat{y}^T) = \frac{\text{trace}(\hat{q} \hat{q}^T \hat{y} \hat{y}^T)}{\sqrt{\text{trace}(\hat{q} \hat{q}^T) \text{trace}(\hat{y} \hat{y}^T)}} = \frac{1}{N_{\text{train}}} \frac{\hat{q}^T (\hat{y} \hat{y}^T) \hat{q}}{\hat{q}^T \hat{q}}.
\]

Furthermore, if \( C_k \) denotes the \( k \times N \) matrix with rows \( v_1^T, \ldots, v_k^T \), then the problem of computing \( q_{k+1,i} \) is a constrained Rayleigh quotient problem with constraints \( C_k x = 0 \). The constrained problem can be written as an unconstrained problem,

\[
q_{k+1,i} = \arg\max_{x \neq 0} \frac{x^T P_k K_i P_k x}{x^T x},
\]

where \( P_k \) is the projection matrix \( I_n - C_k^T (C_k C_k^T)^{-1} C_k \). Because the rows of \( C_k \) are orthogonal, \( C_k C_k^T = I_k \) and \( P_k = I_n - C_k^T C_k \). Thus, each step of our procedure for constructing \( V \) involves maximizing a Rayleigh quotient. The entire algorithm, which has a worst case computational complexity of \( O(N^3 pm) \), is summarized below.

---

**Algorithm 1**: Multiple Kernel Basis Extraction (MKBE)

**input**: \( S_{\text{train}} \), a training set of learning instances

**input**: \( \mathcal{K} \), a set of kernels

**input**: \( m \), the number of rank-one kernels to be extracted

**output**: \( V \), a set of \( m \) orthogonal vectors

1. \( V \leftarrow \emptyset \);
2. \( P \leftarrow I_n \);
3. for \( k \leftarrow 1 \) to \( m \) do
   1. \( Q \leftarrow \emptyset \);
   2. for \( K_i \in \mathcal{K} \) do
      1. \( q \leftarrow \) eigenvector associated with largest eigenvalue of \( P K_i P \);
      2. \( Q \leftarrow Q \cup \{ q \} \);
   3. end
   4. \( v_k \leftarrow \arg\max_{q \in Q} A(\hat{q} \hat{q}^T, \hat{y} \hat{y}^T) \);
   5. \( V \leftarrow V \cup \{ v_k \} \);
   6. \( C_k \leftarrow k \times N \) matrix with rows \( v_1^T, \ldots, v_k^T \);
   7. \( P \leftarrow I - C_k C_k^T \);
   8. end

---

4 **Error Bounds for Transduction**

Following Section 5 of [5], we consider the problem of transduction, i.e., of classifying the test data using a classifier constructed from the training data.
The SVM $f$ in (1) classifies $x_i$ according to the sign of $f(x_i)$. The vector $x_i$ is misclassified if $y_if(x_i) \leq 0$. Consider two loss functions,

$$L_1(f(x_i), y_i) = \begin{cases} 1 & y_if(x_i) \leq 0 \\ 0 & y_if(x_i) > 0 \end{cases}$$

and

$$L_2(f(x_i), y_i) = \begin{cases} 1 - y_if(x_i) & y_if(x_i) \leq 1 \\ 0 & y_if(x_i) > 1 \end{cases}.$$  

The former allows us to count misclassification errors; the latter is a margin cost function.

For convenience, assume that $N_{\text{train}} = N/2 = n$. We measure the performance of $f$ on $S_{\text{test}}$ using $L_1$ and the performance of $f$ on $S_{\text{train}}$ using $L_2$. Let

$$\text{er}(f) = \frac{1}{n} \sum_{j \in S_{\text{test}}} L_1(f(x_i), y_i),$$

the proportion of misclassification errors in the test set, and

$$\text{mc}(f) = \frac{1}{n} \sum_{j \in S_{\text{train}}} L_2(f(x_i), y_i),$$

the average cost of margin violation in the training set. We interpret the difference, $\text{gen}(f) = \text{er}(f) - \text{mc}(f)$, as a measure of generalization.

Because $L_1(f(x_i), y_i) \leq L_2(f(x_i), y_i)$, classifiers for which $\text{gen}(f) \gg 0$ have poor generalization. To remove the dependence of $\text{gen}(f)$ on how $S$ is partitioned into $S_{\text{train}}$ and $S_{\text{test}}$, we assume a probability model under which each of the $\binom{N}{n}$ possible partitions is equally likely. One can then measure how well $f$ generalizes by considering the probability that $\text{gen}(f)$ is small.

Consider kernel classifiers with margin $\gamma$ of the form

$$\mathcal{F}_K(\gamma) = \left\{ x_j \mapsto \sum_{i=1}^{2n} \alpha_i K(x_i, x_j) : K \in \mathcal{K}, \alpha^T K \alpha \leq \gamma^{-2} \right\}. $$

Using a proof technique introduced by Koltchinskii and Panchenko [4], Lanckriet et al. [5, Theorem 24] derived the following bound.

**Proposition 1.** Fix $\gamma > 0$ and $\delta \in (0, 1)$. Assume that each partition of $S$ into $S_{\text{train}}$ and $S_{\text{test}}$ is equally likely. Then every $f \in \mathcal{F}_K(\gamma)$ satisfies

$$P\left( \text{gen}(f) \leq \frac{4 + \sqrt{-2 \log \delta}}{\sqrt{n}} + \frac{\mathcal{C}(K)}{n \gamma} \right) \geq 1 - \delta,$$

where

$$\mathcal{C}(K) = \mathbb{E}_{K} \max_{\sigma} \sigma^T K \sigma$$

and the expectation is with respect to $\sigma$ drawn uniformly from $\{\pm 1\}^N$.  

5
Lanckriet et al. refer to $\mathcal{C}(\mathcal{K})$ as the \textit{complexity} of $\mathcal{K}$. Notice that, if $\mathcal{K}_1 \subseteq \mathcal{K}_2$, then $\mathcal{C}(\mathcal{K}_1) \leq \mathcal{C}(\mathcal{K}_2)$. Furthermore, for $\mathcal{K} = \{K_1, \ldots, K_m\}$, let

$$\mathcal{K}_c^+ = \left\{ K = \sum_{j=1}^m \mu_j K_j : K \succeq 0, \mu_j \geq 0, \text{trace}(K) \leq c \right\}.$$ 

Lanckriet et al. [5, Theorem 24] also established that

$$\mathcal{C}(\mathcal{K}_c^+) \leq c \cdot \min \left( m, nm \cdot \max_j \lambda_j \frac{\text{trace}(K_j)}{} \right),$$

where $\lambda_j$ is the largest eigenvalue of $K_j$.

Our concern is with multiple kernel learning from the collection of extracted kernels, $\mathcal{M} = \{M_1, \ldots, M_m\}$, where $M_j = v_j v_j^T$ and therefore $\text{rank}(M_j) = 1$. Accordingly, we apply Proposition 1 to the set of kernels conv($\mathcal{M}$), the convex hull of $\mathcal{M}$. To bound

$$\mathcal{C}(\text{conv}(\mathcal{M})) = \mathbb{E} \max_{M \in \text{conv}(\mathcal{M})} \sigma^T M \sigma = \mathbb{E} \max_{\mu \in \Delta} \sum_{j=1}^m \mu_j \sigma^T M_j \sigma,$$

where $\Delta$ denotes the unit simplex in $\mathbb{R}^m$, let

$$c = \max_{M \in \text{conv}(\mathcal{M})} \text{trace}(M) = \max_j \text{trace}(M_j) = \max_j \text{trace}(v_j v_j^T) = 1.$$

Then $\text{conv}(\mathcal{M}) \subseteq \mathcal{M}_c^+$, so $\mathcal{C}(\text{conv}(\mathcal{M})) \leq \mathcal{C}(\mathcal{M}_c^+)$. Furthermore, because $\text{rank}(M_j) = 1, \lambda_j = \text{trace}(M_j)$ and the complexity bound simplifies to

$$\mathcal{C}(\text{conv}(\mathcal{M})) \leq \min(m, n).$$

Combining the above results, we see that extracting a small number ($m$) of rank-one kernels leads to a kernel classifier that should have good generalization. Hence, when the predictive information in $\mathcal{K} = \{K_1, \ldots, K_p\}$ can be captured by a small number of rank-one kernels, we would expect MKL based on $\mathcal{M} = \{M_1, \ldots, M_m\}$ to outperform MKL based on $\mathcal{K} = \{K_1, \ldots, K_p\}$.

5 Examples

We demonstrate MKBE on several data sets, using SimpleMKL [7] to learn from the extracted rank-one kernels.

5.1 Stars

Sonnenburg et al. [8] considered the problem of discriminating between two concentric star-shaped distributions. Following their lead, we performed four
experiments, fixing the radius of the inner star at 4 and varying the radius of the outer star in \{5, 6, 7, 8\}. For each experiment, we generated 1000 points from each distribution. Scatter plots of the resulting data sets are displayed in Figure 1.

We randomly divided each data set into training and testing sets of \( n = 1000 \) points and constructed seven different Gaussian kernels, \( \{K_1, \ldots, K_7\} \), defined by

\[
K_\sigma (x_i, x_j) = \exp \left( -\frac{\|x_i - x_j\|^2}{2\sigma^2} \right).
\]

We then constructed kernel classifiers by two methods. The first method applied SimpleMKL directly to \( \{K_1, \ldots, K_7\} \). The second method used MKBE to extract \( \{v_1, \ldots, v_{10}\} \), then applied SimpleMKL to \( \{v_1v_1^T, \ldots, v_{10}v_{10}^T\} \). (To conserve memory, we modified the Matlab implementation of SimpleMKL to
operate directly on \( \{v_1, \ldots, v_{10}\} \). We experimented with several choices of SimpleMKL’s SVM hyperparameter \( C \) and discovered that varying it did not greatly affect performance. All of the results reported below were obtained using \( C = 100 \).

Classifier performance was measured by area under the ROC curve. Results are displayed in Table 5.1. For outer radii of 6, 7, 8, replacing the seven Gaussian kernels with the ten rank-one kernels extracted by MKBE resulted in substantially improved classifier performance. Classifier performance did not improve for an outer radius of 5. This is the most difficult case, and it may be that more than \( m = 10 \) rank-one kernels are needed to extract sufficient information from the Gaussian kernels.

<table>
<thead>
<tr>
<th>Outer Radius</th>
<th>7 Gaussian Kernels</th>
<th>10 MKBE Kernels</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.60</td>
<td>0.55</td>
</tr>
<tr>
<td>6</td>
<td>0.59</td>
<td>0.79</td>
</tr>
<tr>
<td>7</td>
<td>0.85</td>
<td>0.95</td>
</tr>
<tr>
<td>8</td>
<td>0.75</td>
<td>0.99</td>
</tr>
</tbody>
</table>

Table 1: Star Experiments. Area under the ROC curve for two sets of kernels, seven Gaussian kernels and ten rank-one kernels extracted from them by MKBE. Multiple kernel learning was performed by SimpleMKL.

### 5.2 Handwritten Digit Recognition

Using the MNIST database of handwritten digits,\(^1\) we attempted to discriminate between the digits 1 and 7. We randomly selected 1000 examples of each digit, then randomly divided these \( N = 2000 \) digits into training and testing sets of \( n = 1000 \) digits. We constructed \( p = 10 \) kernels, five Gaussian with \( \sigma = 0.01, 0.1, 1, 10, 100 \), and five polynomial, i.e.,

\[
K(x_i, x_j) = (x_i^T x_j + 1)^d
\]

with degree \( d = 2, 3, 4, 5, 6 \).

As above, we constructed kernel classifiers by two methods. The first method applied SimpleMKL directly to \( \{K_1, \ldots, K_{10}\} \). The second method used MKBE to extract \( \{v_1, \ldots, v_{10}\} \), then applied SimpleMKL to \( \{v_1 v_1^T, \ldots, v_{10} v_{10}^T\} \). In this experiment, classifier performance varied with the choice \( C \), the SVM hyperparameter in SimpleMLK.

Again, performance was measured by area under the ROC curve. Results are displayed in Table 5.2. The best performance was obtained by applying SimpleMKL with \( C \leq 1 \) to \( \{v_1 v_1^T, \ldots, v_{10} v_{10}^T\} \).

\(^1\)http://yann.lecun.com/exdb/mnist/
Table 2: Digit Recognition (1 v 7) Experiments. Area under the ROC curve for two sets of kernels, ten kernels (five Gaussian, five polynomial) and ten rank-one kernels extracted from them by MKBE. Multiple kernel learning was performed by SimpleMKL with various choices of the SVM hyperparameter $C$.

<table>
<thead>
<tr>
<th>$C$</th>
<th>10 Original Kernels</th>
<th>10 MKBE Kernels</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.54</td>
<td>0.90</td>
</tr>
<tr>
<td>0.5</td>
<td>0.57</td>
<td>0.90</td>
</tr>
<tr>
<td>1</td>
<td>0.56</td>
<td>0.89</td>
</tr>
<tr>
<td>10</td>
<td>0.79</td>
<td>0.79</td>
</tr>
<tr>
<td>100</td>
<td>0.63</td>
<td>0.65</td>
</tr>
<tr>
<td>200</td>
<td>0.57</td>
<td>0.56</td>
</tr>
</tbody>
</table>

6 Discussion

Performance on supervised learning tasks in high-dimensional feature spaces often benefits from preliminary feature extraction. We have proposed a technique (MKBE) that extracts rank-one kernels from high-rank kernels. The extracted kernels are formed from orthogonal vectors that are maximally aligned with the class labels. One then applies standard techniques for multiple kernel learning (MKL) to the rank-one kernels to construct a support vector machine (SVM).

The potential benefits of preliminary feature extraction are based on general statistical considerations. The theory presented in Section 4 hints that MKBE may improve the performance of SVMs constructed by MKL. Section 5 presents some examples for which the improvement is fairly dramatic. We believe that these examples provide a provocative proof-of-concept.

The ideas that we have explored have much in common with the ideas explored by Ma et al. [6], who considered the problem of learning from multiple dissimilarity matrices, $\{\Delta_1, \ldots, \Delta_p\}$. Ma et al. first embedded each $\Delta_i$ by classical multidimensional scaling (CMDS). CMDS exploits a well-known connection between squared Euclidean distances and Euclidean inner products. The matrix of squared dissimilarities is subjected to a linear transformation, resulting in a matrix, $B$, of fallible inner products. The matrix $B$ is then replaced by the inner product matrix, $\bar{B}$, that is nearest (in the sense of Frobenius norm) $B$, subject to a rank restriction. The inner product matrices $\{\bar{B}_1, \ldots, \bar{B}_p\}$ are our kernel matrices $\{K_1, \ldots, K_p\}$.

Our MKBE method of feature extraction differs slightly from Ma et al.’s $J$-function approach. The latter forms the Cartesian product of the $p$ representations constructed by CMDS, performs a principal component analysis on the product, and selects the principal components with the largest values of the $J$-statistic described below. Given univariate samples $u = (u_1, \ldots, u_r)^T$ and $v = (v_1, \ldots, v_s)^T$, let $q^T = (u^T|v^T)$ and let $y^T = (e_r^T|e_s^T)$, where $e_r \in \mathbb{R}^r$ and $e_s \in \mathbb{R}^s$ are vectors with unit entries. Assume that $ue_r^T + ve_s^T = 0$. Then
\[ J = |\bar{u} - \bar{v}|/\lambda, \]

where

\[ \lambda^2 = \sum_{i=1}^{r} u_i^2 + \sum_{j=1}^{s} v_j^2, \]

whereas

\[
\begin{align*}
\text{trace} (qq^T yy^T) &= \text{trace} (q^T yy^T q) = (q^T y)^2 = \left( \sum_{i=1}^{r} u_i - \sum_{j=1}^{s} v_j \right)^2, \\
\text{trace} (qq^T) &= \text{trace} (q^T q) = \sum_{i=1}^{r} u_i^2 + \sum_{j=1}^{s} v_j^2 = \lambda^2, \\
\text{trace} (yy^T) &= \text{trace} (y^T y) = r + s,
\end{align*}
\]

and therefore the alignment statistic

\[
A (qq^T, yy^T) = \frac{\left( \sum_{i=1}^{r} u_i - \sum_{j=1}^{s} v_j \right)^2}{\sqrt{(r + s)\lambda^2}}.
\]

Furthermore, Ma et al. select a subset of the principal components of the product representation, whereas our \( v_1 \) is the first principal component of one of the individual kernels. Finally, we construct an SVM by MKL, whereas Ma et al. perform a linear discriminant analysis using the selected principal components.

**Acknowledgments**

This research was supported by a grant to Indiana University from the Office of Naval Research.

**References**


