Kernel Learning for Extrinsic Classification of Manifold Features

Raviteja Vemulapalli
Jaishanker K. Pillai
Professor Rama Chellappa

University of Maryland, College Park

Overview

- Motivation
- Problem formulation
- Optimization procedure
- Grassmann manifold
- Symmetric positive definite matrices
- Experimental results

Where are manifold features used in computer vision?

Applications of manifold features

- Diffusion tensor magnetic resonance imaging (DT-MRI)
- Texture classification and segmentation
- Object detection and tracking
- Motion segmentation using structure tensors
- Face and object recognition from image sets
- Human activity recognition using dynamical systems
- Shape analysis

Classification in Euclidean space

For features that lie in Euclidean spaces, classifiers based on discriminative approaches such as linear discriminant analysis (LDA), partial least squares (PLS) and support vector machines (SVM) have been successfully used in various applications.

How can we extend these techniques to manifold features?

How can we extend these techniques to manifold features?

Define kernels on manifolds!

How to find good kernels for classification of manifold features?

How to find good kernels for classification of manifold features?

Try out kernel learning!

Kernel learning for manifolds features

Let $\mathcal{M}$ be a Riemannian manifold (with known geometry) on which the features lie.

Kernel learning for manifolds features

Let $\mathcal{M}$ be a Riemannian manifold (with known geometry) on which the features lie.

Since we are interested in finding a kernel $\mathcal{K}$ (defined on $\mathcal{M}$) for the purpose of classification, we propose to learn both the kernel and classifier jointly based on the following two criteria:

Kernel learning for manifolds features

Let $\mathcal{M}$ be a Riemannian manifold (with known geometry) on which the features lie.

Since we are interested in finding a kernel $\mathcal{K}$ (defined on $\mathcal{M}$) for the purpose of classification, we propose to learn both the kernel and classifier jointly based on the following two criteria:

1. **Risk minimization:** For good classification performance, the risk functional associated with the classifier should be minimized.

Kernel learning for manifolds features

Let $\mathcal{M}$ be a Riemannian manifold (with known geometry) on which the features lie.

Since we are interested in finding a kernel $\mathcal{K}$ (defined on $\mathcal{M}$) for the purpose of classification, we propose to learn both the kernel and classifier jointly based on the following two criteria:

1. **Risk minimization:** For good classification performance, the risk functional associated with the classifier should be minimized.
2. **Structure preservation:** Since the features lie on a Riemannian manifold with well defined structure, the kernel should try to preserve the underlying manifold structure. This criterion acts as a regularizer.

Kernel learning for manifolds features

Our general framework for learning a good kernel-classifier combination can be represented as the following optimization problem

$$\min_{W,\mathcal{K}} \; \lambda \, \Gamma_s(\mathcal{K}) + \Gamma_c(W, \mathcal{K}).$$

Kernel learning for manifolds features

Our general framework for learning a good kernel-classifier combination can be represented as the following optimization problem

$$\min_{W,\mathcal{K}} \ \lambda\, \Gamma_s(\mathcal{K}) + \Gamma_c(W, \mathcal{K}).$$

- $\Gamma_c(W, \mathcal{K})$ is the classifier cost expressed as a function of the classifier parameters W and kernel $\mathcal{K}$.

Kernel learning for manifolds features

Our general framework for learning a good kernel-classifier combination can be represented as the following optimization problem

$$\min_{W,\mathcal{K}} \ \lambda\, \Gamma_s(\mathcal{K}) + \Gamma_c(W, \mathcal{K}).$$

- $\Gamma_c(W, \mathcal{K})$ is the classifier cost expressed as a function of the classifier parameters W and kernel $\mathcal{K}$.
- $\Gamma_s(\mathcal{K})$ is the manifold-structure cost expressed as a function of kernel $\mathcal{K}$.

Kernel learning for manifolds features

Our general framework for learning a good kernel-classifier combination can be represented as the following optimization problem

$$\min_{W,\mathcal{K}} \ \lambda\, \Gamma_s(\mathcal{K}) + \Gamma_c(W, \mathcal{K}).$$

- $\Gamma_c(W, \mathcal{K})$ is the classifier cost expressed as a function of the classifier parameters W and kernel $\mathcal{K}$.
- $\Gamma_s(\mathcal{K})$ is the manifold-structure cost expressed as a function of kernel $\mathcal{K}$.
- λ is a parameter controlling the tradeoff between the two cost functions.

Kernel learning for manifolds features

Since learning the kernel $\mathcal{K}$ in a non-parametric fashion makes the problem transductive, we follow the multiple kernel learning approach and parametrize the kernel $\mathcal{K}$ as a positive linear combination of known base kernels $\mathcal{K}^1, \mathcal{K}^2, \ldots, \mathcal{K}^M$.

$$\mathcal{K} = \sum_{m=1}^{M} \mu_m \mathcal{K}^m, \ \mu_m \geq 0.$$

Classifier cost in kernel learning

SVM cost function:

$$\max_{\vec{\alpha}\in\Omega} \left(\vec{\alpha}^\top \vec{1} - \frac{1}{2}\vec{\alpha}^\top \left(\vec{y}\vec{y}^\top o\, K \right) \vec{\alpha} \right)$$

- $\Omega = \{\vec{\alpha} \in \mathcal{R}^{N_{tr}} \mid 0 \leq \vec{\alpha} \leq C\vec{1},\ \vec{\alpha}^\top \vec{y} = 0\}$.
- $\vec{y}$ is the vector of training labels.
- N_{tr} is the number of training samples.
- The operation o denotes the matrix Hadamard product.

Manifold-structure cost in kernel learning

A simple geodesic distance based manifold-structure cost:

$$\sum_{i=1}^{N}\sum_{i=1}^{N}(K_{ii} + K_{jj} - K_{ij} - K_{ji} - d_{ij}^2)^2$$

- K_{ij} is the kernel value for samples i and j.
- d_{ij} is the geodesic distance between samples i and j.

Kernel learning for manifolds features

Combining both the costs, we get the following optimization problem:

$$\min_{\vec{\zeta},\,\vec{\mu}} \max_{\vec{\alpha}\in\Omega} \lambda\|\vec{\zeta}\|_2^2 + \left(\vec{\alpha}^\top\vec{1} - \frac{1}{2}\vec{\alpha}^\top\left(\vec{y}\vec{y}^\top o \sum_{m=1}^{M}\mu_m K^m\right)\vec{\alpha}\right),$$

$$\text{subject to} \sum_{m=1}^{M}\mu_m(K_{ii}^m + K_{jj}^m - K_{ij}^m - K_{ji}^m) - d_{ij}^2 = \zeta_{ij},$$

$$\text{for } 1 \le i < j \le N_{tr} \text{ and } \vec{\mu} \ge \vec{0},$$

where $\Omega = \{\vec{\alpha} \in \mathcal{R}^{N_{tr}} \mid 0 \le \vec{\alpha} \le C\vec{1},\ \vec{\alpha}^\top\vec{y} = 0\}$.

Kernel learning for manifolds features

Combining both the costs, we get the following optimization problem:

$$\min_{\vec{\zeta},\,\vec{\mu}} \max_{\vec{\alpha}\in\Omega} \lambda\|\vec{\zeta}\|_2^2 + \left(\vec{\alpha}^\top\vec{1} - \frac{1}{2}\vec{\alpha}^\top\left(\vec{y}\vec{y}^\top o \sum_{m=1}^{M}\mu_m K^m\right)\vec{\alpha}\right),$$

$$\text{subject to} \sum_{m=1}^{M}\mu_m(K_{ii}^m + K_{jj}^m - K_{ij}^m - K_{ji}^m) - d_{ij}^2 = \zeta_{ij},$$

$$\text{for } 1 \le i < j \le N_{tr} \text{ and } \vec{\mu} \ge \vec{0},$$

$$\text{where } \Omega = \{\vec{\alpha} \in \mathcal{R}^{N_{tr}} \mid 0 \le \vec{\alpha} \le C\vec{1},\ \vec{\alpha}^\top\vec{y} = 0\}.$$

This is a convex optimization problem and can be solved efficiently using gradient based methods.

Kernel learning for manifolds features

Manifold-structure cost:

$$J_1(\vec{\mu}) = \sum_{i=1}^{N_{tr}} \sum_{j=i+1}^{N_{tr}} \zeta_{ij}^2 = \sum_{i=1}^{N_{tr}} \sum_{j=i+1}^{N_{tr}} \left(\sum_{m=1}^{M} \mu_m p_{ij}^m - d_{ij}^2 \right)^2,$$

where $p_{ij}^m = K_{ii}^m + K_{jj}^m - K_{ij}^m - K_{ji}^m$.

Kernel learning for manifolds features

Manifold-structure cost:

$$J_1(\vec{\mu}) = \sum_{i=1}^{N_{tr}} \sum_{j=i+1}^{N_{tr}} \zeta_{ij}^2 = \sum_{i=1}^{N_{tr}} \sum_{j=i+1}^{N_{tr}} \left(\sum_{m=1}^{M} \mu_m p_{ij}^m - d_{ij}^2 \right)^2,$$

where $p_{ij}^m = K_{ii}^m + K_{jj}^m - K_{ij}^m - K_{ji}^m$.

$J_1(\vec{\mu})$ is convex and differentiable.

Kernel learning for manifolds features

Manifold-structure cost:

$$J_1(\vec{\mu}) = \sum_{i=1}^{N_{tr}} \sum_{j=i+1}^{N_{tr}} \zeta_{ij}^2 = \sum_{i=1}^{N_{tr}} \sum_{j=i+1}^{N_{tr}} \left(\sum_{m=1}^{M} \mu_m p_{ij}^m - d_{ij}^2 \right)^2 ,$$

where $p_{ij}^m = K_{ii}^m + K_{jj}^m - K_{ij}^m - K_{ji}^m$.

$J_1(\vec{\mu})$ is convex and differentiable.

$$\frac{\partial J_1}{\partial \mu_m} = \sum_{i=1}^{N_{tr}} \sum_{j=i+1}^{N_t r} \left(2 p_{ij}^m \left(\sum_{k=1}^{M} \mu_k p_{ij}^k - d_{ij}^2 \right) \right) .$$

Kernel learning for manifolds features

SVM classifier cost:

$$J_2(\vec{\mu}) = \max_{\vec{\alpha} \in \Omega} \left(\vec{\alpha}^\top \vec{1} - \frac{1}{2} \vec{\alpha}^\top (\vec{y}\vec{y}^\top o \sum_{m=1}^{M} \mu_m K^m) \vec{\alpha} \right),$$

where $\Omega = \{\vec{\alpha} \in \mathcal{R}^{N_{tr}} \mid 0 \leq \vec{\alpha} \leq C\vec{1},\ \vec{\alpha}^\top \vec{y} = 0\}$.

Kernel learning for manifolds features

SVM classifier cost:

$$J_2(\vec{\mu}) = \max_{\vec{\alpha} \in \Omega} \left(\vec{\alpha}^\top \vec{1} - \frac{1}{2} \vec{\alpha}^\top (\vec{y}\vec{y}^\top o \sum_{m=1}^{M} \mu_m K^m) \vec{\alpha} \right),$$

where $\Omega = \{\vec{\alpha} \in \mathcal{R}^{N_{tr}} \mid 0 \leq \vec{\alpha} \leq C\vec{1},\ \vec{\alpha}^\top \vec{y} = 0\}$.

$J_2(\vec{\mu})$ is convex and differentiable if $K^m \succ 0$.
[Rakotomamonjy *et al.* JMLR08]

Kernel learning for manifolds features

SVM classifier cost:

$$J_2(\vec{\mu}) = \max_{\vec{\alpha}\in\Omega} \left(\vec{\alpha}^\top \vec{1} - \frac{1}{2}\vec{\alpha}^\top (\vec{y}\vec{y}^\top o \sum_{m=1}^{M} \mu_m K^m)\vec{\alpha} \right),$$

where $\Omega = \{\vec{\alpha} \in \mathcal{R}^{N_{tr}} \mid 0 \le \vec{\alpha} \le C\vec{1},\ \vec{\alpha}^\top \vec{y} = 0\}$.

$J_2(\vec{\mu})$ is convex and differentiable if $K^m \succ 0$.
[Rakotomamonjy *et al.* JMLR08]

$$\frac{\partial J_2}{\partial \mu_m} = -\frac{1}{2}\sum_{i=1}^{N_{tr}}\sum_{j=1}^{N_{tr}} \alpha_i^* \alpha_j^* y_i y_j K_{ij}^m,$$

where $\vec{\alpha}^*$ is the optimal solution for the above SVM dual problem.

Kernel learning for manifolds features

Let $J(\vec{\mu}) = \lambda J_1(\vec{\mu}) + J_2(\vec{\mu})$. Then, the kernel learning optimization becomes

$$\min_{\vec{\mu}} J(\vec{\mu})$$
$$\text{subject to } \vec{\mu} \geq 0.$$

Kernel learning for manifolds features

Let $J(\vec{\mu}) = \lambda J_1(\vec{\mu}) + J_2(\vec{\mu})$. Then, the kernel learning optimization becomes

$$\min_{\vec{\mu}} J(\vec{\mu})$$
$$\text{subject to } \ \vec{\mu} \geq 0.$$

$J(\vec{\mu})$ is convex and differentiable if $K^m \succ 0$.

Kernel learning for manifolds features

Let $J(\vec{\mu}) = \lambda J_1(\vec{\mu}) + J_2(\vec{\mu})$. Then, the kernel learning optimization becomes

$$\min_{\vec{\mu}} J(\vec{\mu})$$
$$\text{subject to } \vec{\mu} \geq 0.$$

$J(\vec{\mu})$ is convex and differentiable if $K^m \succ 0$.

The above optimization problem can be solved using reduced gradient descent method or any other standard algorithm used for solving constrained convex optimization problems.

Computational aspects - Training

First, we need to compute the pairwise geodesic distances d_{ij} between training samples.

Computational aspects - Training

First, we need to compute the pairwise geodesic distances d_{ij} between training samples.

Then, we need to solve the following optimization problem:

$$\min_{\vec{\mu}} \lambda J_1(\vec{\mu}) + J_2(\vec{\mu})$$
$$\text{subject to } \vec{\mu} \geq 0.$$

Computational aspects - Training

Computation of the cost value for a given $\vec{\mu}$:

Computational aspects - Training

Computation of the cost value for a given $\vec{\mu}$:

- Computation of $J_1(\vec{\mu})$:

$$J_1(\vec{\mu}) = \sum_{i=1}^{N_{tr}} \sum_{j=i+1}^{N_{tr}} \zeta_{ij}^2 = \sum_{i=1}^{N_{tr}} \sum_{j=i+1}^{N_{tr}} \left(\sum_{m=1}^{M} \mu_m p_{ij}^m - d_{ij}^2 \right)^2 .$$

Computational aspects - Training

Computation of the cost value for a given $\vec{\mu}$:

- Computation of $J_1(\vec{\mu})$:

$$J_1(\vec{\mu}) = \sum_{i=1}^{N_{tr}} \sum_{j=i+1}^{N_{tr}} \zeta_{ij}^2 = \sum_{i=1}^{N_{tr}} \sum_{j=i+1}^{N_{tr}} \left(\sum_{m=1}^{M} \mu_m p_{ij}^m - d_{ij}^2 \right)^2 .$$

- Computation of $J_2(\vec{\mu})$: Solve an SVM optimization problem

$$J_2(\vec{\mu}) = \max_{\vec{\alpha} \in \Omega} \left(\vec{\alpha}^\top \vec{1} - \frac{1}{2} \vec{\alpha}^\top (\vec{y}\vec{y}^\top o \sum_{m=1}^{M} \mu_m K^m) \vec{\alpha} \right).$$

Computational aspects - Training

Computation of the gradient at a given $\vec{\mu}$: For $m = 1$ to M

Computational aspects - Training

Computation of the gradient at a given $\vec{\mu}$: For $m = 1$ to M

$$\frac{\partial J_1}{\partial \mu_m} = \sum_{i=1}^{N_{tr}} \sum_{j=i+1}^{N_t r} \left(2p_{ij}^m \left(\sum_{k=1}^{M} \mu_k p_{ij}^k - d_{ij}^2 \right) \right),$$

Computational aspects - Training

Computation of the gradient at a given $\vec{\mu}$: For $m = 1$ to M

$$\frac{\partial J_1}{\partial \mu_m} = \sum_{i=1}^{N_{tr}} \sum_{j=i+1}^{N_t r} \left(2p_{ij}^m \left(\sum_{k=1}^{M} \mu_k p_{ij}^k - d_{ij}^2 \right) \right),$$

$$\frac{\partial J_2}{\partial \mu_m} = -\frac{1}{2} \sum_{i=1}^{N_{tr}} \sum_{j=1}^{N_{tr}} \alpha_i^* \alpha_j^* y_i y_j K_{ij}^m.$$

Computational aspects - Testing

Computing the kernel values:

- For each test point, we need to perform $M \times N_{tr}$ kernel evaluations.

Computational aspects - Testing

Computing the kernel values:

- For each test point, we need to perform $M \times N_{tr}$ kernel evaluations.

Decision function evaluation:

$$f(x) = \left(\sum_{i=1}^{N_{tr}} \alpha_i^* y_i \sum_{m=1}^{M} \mu_m K^m(x_i, x) \right) + b.$$

Grassmann manifold

- Grassmann manifold $\mathcal{G}_{n,d}$ is the set of all d-dimensional linear subspaces of $\mathcal{R}^n$.

Grassmann manifold

- Grassmann manifold $\mathcal{G}_{n,d}$ is the set of all d-dimensional linear subspaces of $\mathcal{R}^n$.
- A linear subspace S on Grassmann manifold can be represented by any $n \times d$ orthonormal matrix Y_s whose column space is S. Note that Y_s is not unique.

Grassmann manifold

- Grassmann manifold $\mathcal{G}_{n,d}$ is the set of all d-dimensional linear subspaces of $\mathcal{R}^n$.
- A linear subspace S on Grassmann manifold can be represented by any $n \times d$ orthonormal matrix Y_s whose column space is S. Note that Y_s is not unique.
- The geodesic distance between two subspaces S_1 and S_2 on Grassmann manifold is given by $\|\vec{\theta}\|_2$, where $\theta_1, \ldots, \theta_d$ are the principal angles between S_1 and S_2.

Grassmann manifold

- Grassmann manifold $\mathcal{G}_{n,d}$ is the set of all d-dimensional linear subspaces of $\mathcal{R}^n$.
- A linear subspace S on Grassmann manifold can be represented by any $n \times d$ orthonormal matrix Y_s whose column space is S. Note that Y_s is not unique.
- The geodesic distance between two subspaces S_1 and S_2 on Grassmann manifold is given by $\|\vec{\theta}\|_2$, where $\theta_1, \ldots, \theta_d$ are the principal angles between S_1 and S_2.
- $\vec{\theta}$ can be computed using $\theta_i = \cos^{-1}(\alpha_i) \in [0, \frac{\pi}{2}]$, where α_i are the singular values of $Y_{s1}^\top Y_{s2}$.

Grassmann manifold - Kernels

Let $\Phi_P(S) = Y_s Y_s^\top$. Note that $\Phi_P(S)$ does not depend on the choice of Y_s.

Grassmann manifold - Kernels

Let $\Phi_P(S) = Y_s Y_s^\top$. Note that $\Phi_P(S)$ does not depend on the choice of Y_s.

Projection-RBF kernels:

$$\mathcal{K}_P^{\text{rbf}}(S_1, S_2) = \exp\left(-\gamma \|\Phi_P(S_1) - \Phi_P(S_2)\|_F^2\right).$$

Grassmann manifold - Kernels

Let $\Phi_P(S) = Y_s Y_s^\top$. Note that $\Phi_P(S)$ does not depend on the choice of Y_s.

Projection-RBF kernels:

$$\mathcal{K}_P^{\text{rbf}}(S_1, S_2) = \exp\left(-\gamma \|\Phi_P(S_1) - \Phi_P(S_2)\|_F^2\right).$$

Projection-polynomial kernels:

$$\mathcal{K}_P^{\text{poly}}(S_1, S_2) = \left(\gamma \text{ trace}\left(\Phi_P(S_1)^\top \Phi_P(S_2)\right)\right)^d.$$

Grassmann manifold - Kernels

Let $\Phi_P(S) = Y_s Y_s^\top$. Note that $\Phi_P(S)$ does not depend on the choice of Y_s.

Projection-RBF kernels:

$$\mathcal{K}_P^{\text{rbf}}(S_1, S_2) = \exp\left(-\gamma \|\Phi_P(S_1) - \Phi_P(S_2)\|_F^2\right).$$

Projection-polynomial kernels:

$$\mathcal{K}_P^{\text{poly}}(S_1, S_2) = \left(\gamma \text{ trace}\left(\Phi_P(S_1)^\top \Phi_P(S_2)\right)\right)^d.$$

When $d = \gamma = 1,\ \mathcal{K}_P^{\text{poly}}$ is same as the projection kernel introduced in [Hamm *et al.* ICML08].

Symmetric positive definite matrices

- The set of all $n \times n$ symmetric positive definite (SPD) matrices form a manifold.

Symmetric positive definite matrices

- The set of all $n \times n$ symmetric positive definite (SPD) matrices form a manifold.
- Equipped with the affine-invariant Riemannian metric, the geodesic distance distance between two SPD matrices C_1 and C_2 is given by

$$\sqrt{\sum_{i=1}^{d} \ln^2 \lambda_i(C_1, C_2)},$$

where $\lambda_i(C_1, C_2)$ are the generalized Eigenvalues of C_1 and C_2.

Symmetric positive definite matrices - Kernels

Let $\log(C)$ denote the matrix logarithm of a symmetric positive definite matrix C.

Symmetric positive definite matrices - Kernels

Let $\log(C)$ denote the matrix logarithm of a symmetric positive definite matrix C.

LED-RBF kernels:

$$\mathcal{K}_{\text{log}}^{\text{rbf}}(C_1, C_2) = \exp\left(-\gamma \|\text{log}(C_1) - \text{log}(C_2)\|_F^2\right).$$

Symmetric positive definite matrices - Kernels

Let log(C) denote the matrix logarithm of a symmetric positive definite matrix C.

LED-RBF kernels:

$$\mathcal{K}_{\text{log}}^{\text{rbf}}(C_1, C_2) = \exp\left(-\gamma \|\text{log}(C_1) - \text{log}(C_2)\|_F^2\right).$$

LED-polynomial kernels:

$$\mathcal{K}_{\text{log}}^{\text{poly}}(C_1, C_2) = \left(\gamma \text{ trace}\left(\text{log}(C_1)^\top \text{log}(C_2)\right)\right)^d.$$

Symmetric positive definite matrices - Kernels

Let log(C) denote the matrix logarithm of a symmetric positive definite matrix C.

LED-RBF kernels:

$$\mathcal{K}_{\text{log}}^{\text{rbf}}(C_1, C_2) = \exp\left(-\gamma \|\text{log}(C_1) - \text{log}(C_2)\|_F^2\right).$$

LED-polynomial kernels:

$$\mathcal{K}_{\text{log}}^{\text{poly}}(C_1, C_2) = \left(\gamma \text{ trace}\left(\text{log}(C_1)^\top \text{log}(C_2)\right)\right)^d.$$

When $d = \gamma = 1,\ \mathcal{K}_{\text{log}}^{\text{poly}}$ is same as the LED kernel introduced in [Wang *et al.* CVPR12].

How to represent an image set?

Experiments - Image set based recognition

Linear subspaces:

- Given multiple images of the same face or object, they can be collectively represented using a lower dimensional subspace obtained by applying PCA on the feature vectors representing individual images.

Covariance features:

- Alternatively, the image set can also be represented using its natural second-order statistic, i.e., the covariance matrix.

Experimental results - Image set based recognition

ETH-80 object dataset [Leibe *et al.* CVPR03]

- 8 objects categories with 10 different object instances in each category.
- Each object instance is a set of images of the same object captured under different views.
- For each category, we used 5 sets for training and 5 sets for testing.

Experimental results - Image set based recognition

ETH-80 object dataset [Leibe *et al.* CVPR03]

- 8 objects categories with 10 different object instances in each category.
- Each object instance is a set of images of the same object captured under different views.
- For each category, we used 5 sets for training and 5 sets for testing.

YouTube Celebrities face dataset [Kim *et al.* CVPR08]

- Multiple video clips of 47 subjects collected from YouTube.
- Low resolution and highly compressed videos.
- For each class, we used 3 videos for training and 6 videos for testing.

Experimental results - Image set based recognition

- Image sets were modeled using covariance features and linear subspaces [Wang *et al.* CVPR12].

Experimental results - Image set based recognition

- Image sets were modeled using covariance features and linear subspaces [Wang *et al.* CVPR12].
- In the case of linear subspaces, multiple Projection-RBF and Projection-polynomial kernels were used.

Experimental results - Image set based recognition

- Image sets were modeled using covariance features and linear subspaces [Wang *et al.* CVPR12].
- In the case of linear subspaces, multiple Projection-RBF and Projection-polynomial kernels were used.
- In the case of covariance features, multiple LED-RBF and LED-polynomial kernels were used.

Experimental results - Image set based recognition

dataset	NN	S-MKL	GDA	Proj + PLS	Proposed approach
YouTube	62.8	64.3	65.7	67.7	70.8
ETH80	93.2	93.7	92.8	95.3	96.0

Table: Recognition rates for image set-based face and object recognition tasks using linear subspaces.

dataset	NN	S-MKL	CDL- LDA	CDL- PLS	Proposed approach
YouTube	40.7	69.7	67.5	70.1	73.2
ETH80	92.7	93.7	94.5	96.5	98.2

Table: Recognition rates for image set-based face and object recognition tasks using covariance features.

References

- A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet, "SimpleMKL," *JMLR*, vol. 9, pp. 2491 – 2521, 2008.
- J. Hamm and D. D. Lee, "Grassmann Discriminant Analysis: a Unifying View on Subspace-Based Learning," *In ICML*, 2008.
- R. Wang, H. Guo, L. S. Davis and Q. Dai, "Covariance Discriminative Learning: A Natural and Efficient Approach to Image Set Classification", *In CVPR*, 2012.
- B. Leibe and B. Schiele, "Analyzing Appearance and Contour Based Methods for Object Categorization", *In CVPR*, 2003.
- M. Kim, S. Kumar, V. Pavlovic and H. Rowley, "Face Tracking and Recognition with Visual Constraints in Real-World Videos", *In CVPR*, 2008.