CVPR 2015 Tutorial on

Distance Metric Learning for Visual Recognition

Tutors: Jiwen Lu, Ruiping Wang, Wei-Shi Zheng, and Weihong Deng




URL: http://vipl.ict.ac.cn/homepage/CVPR15Metric/

Outline

- Part 1: Introduction (Jiwen Lu)
- Part 2: Metric Learning for Face Recognition (Jiwen Lu)
- Part 3: Metric Learning for Person Re-identification (Wei-Shi Zheng)

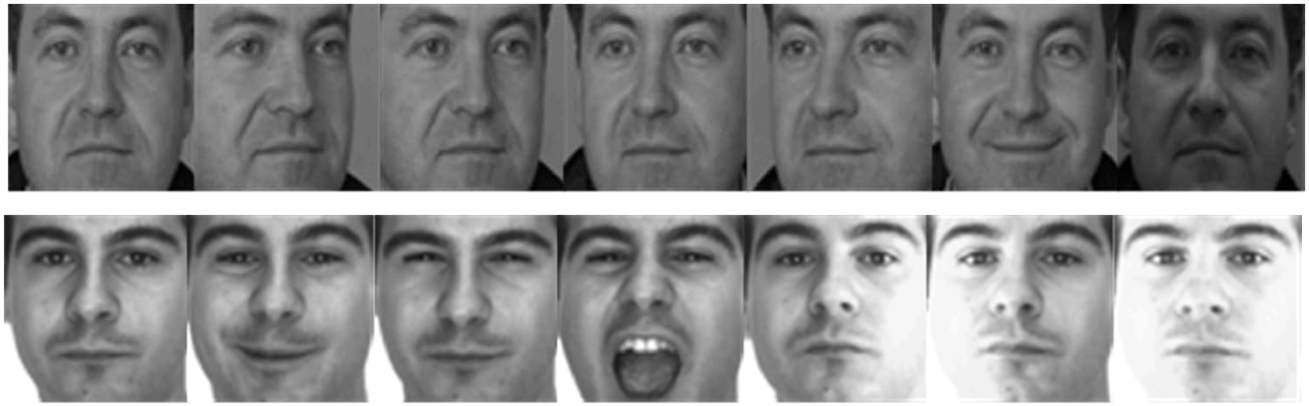
---------- Coffee Break: 45 minutes ----------

- Part 4: Metric Learning for Image Set Classification (Ruiping Wang)
- Part 5: Conclusion and Future Work (Jiwen Lu and Ruiping Wang)

Part 1: Introduction

Visual Recognition Examples

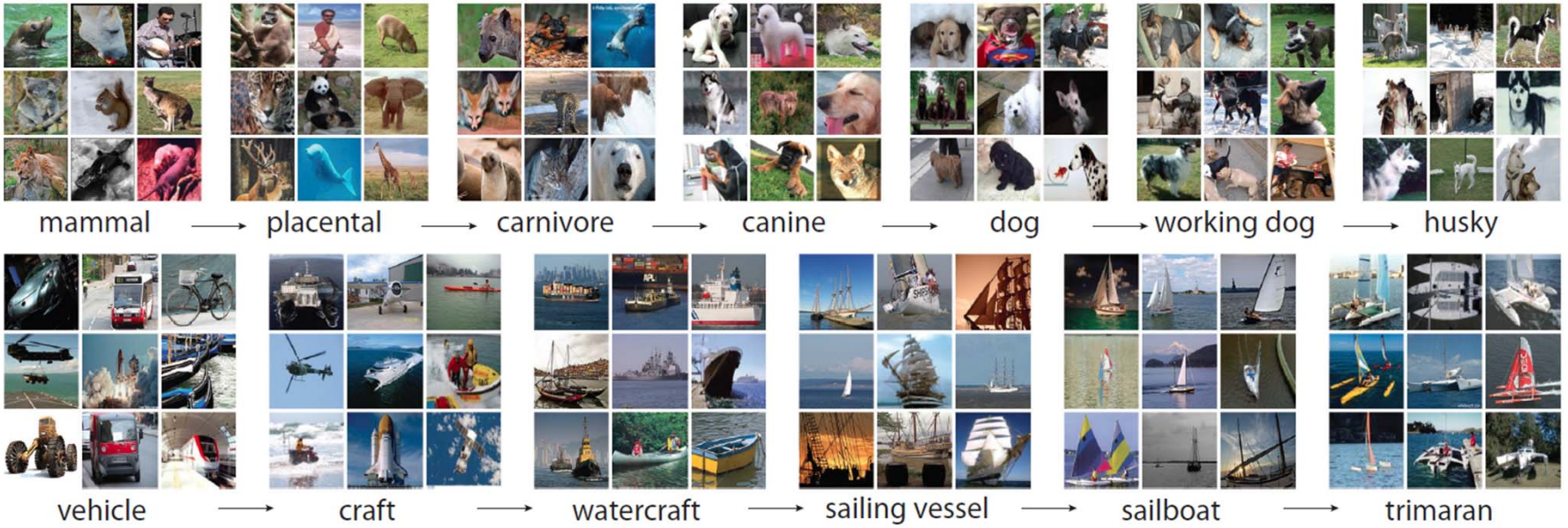
Face Identification



Face Verification

Visual Recognition Examples

Image Classification



Visual Recognition Examples

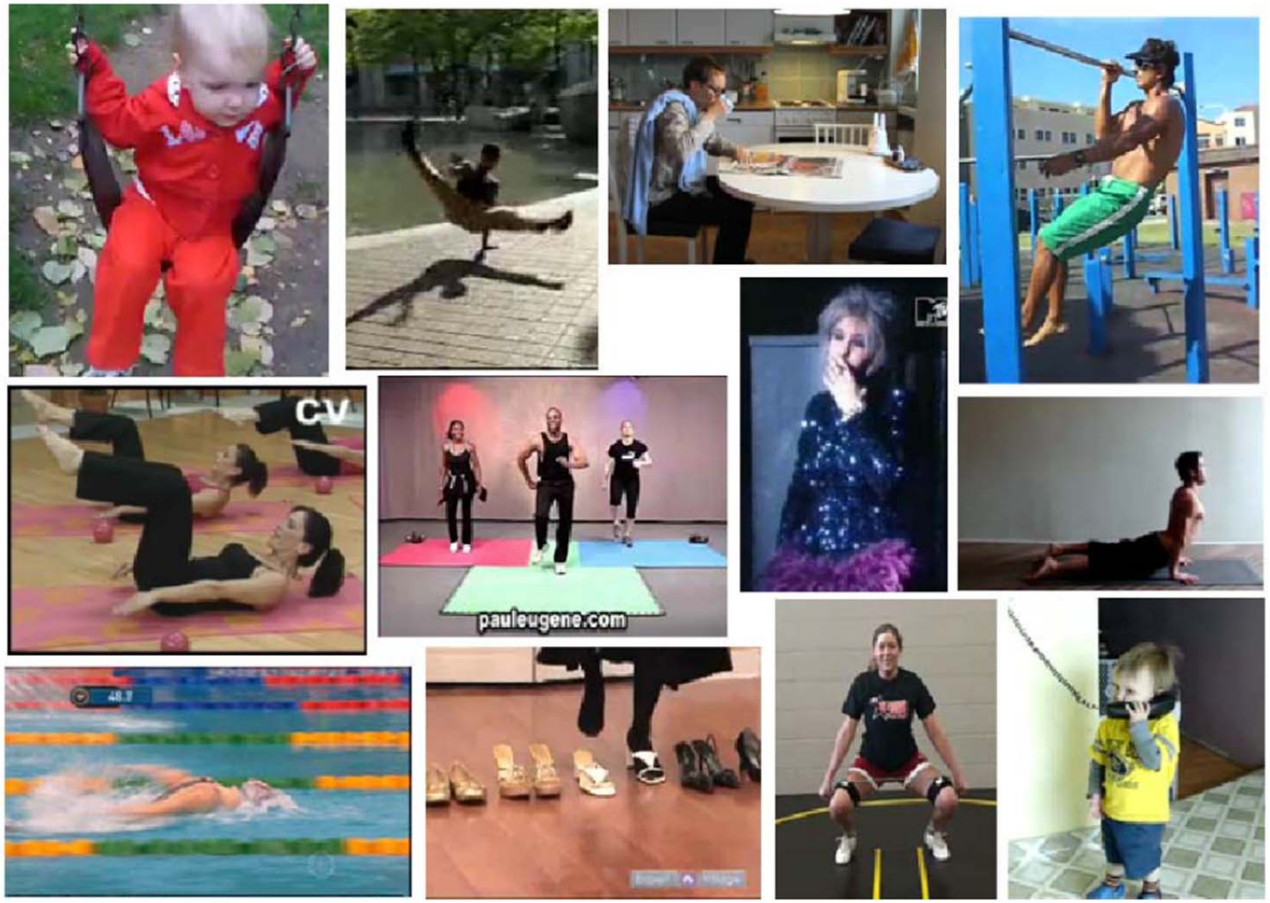
RGB-D Object Recognition

Visual Recognition Examples

Person Re-Identification

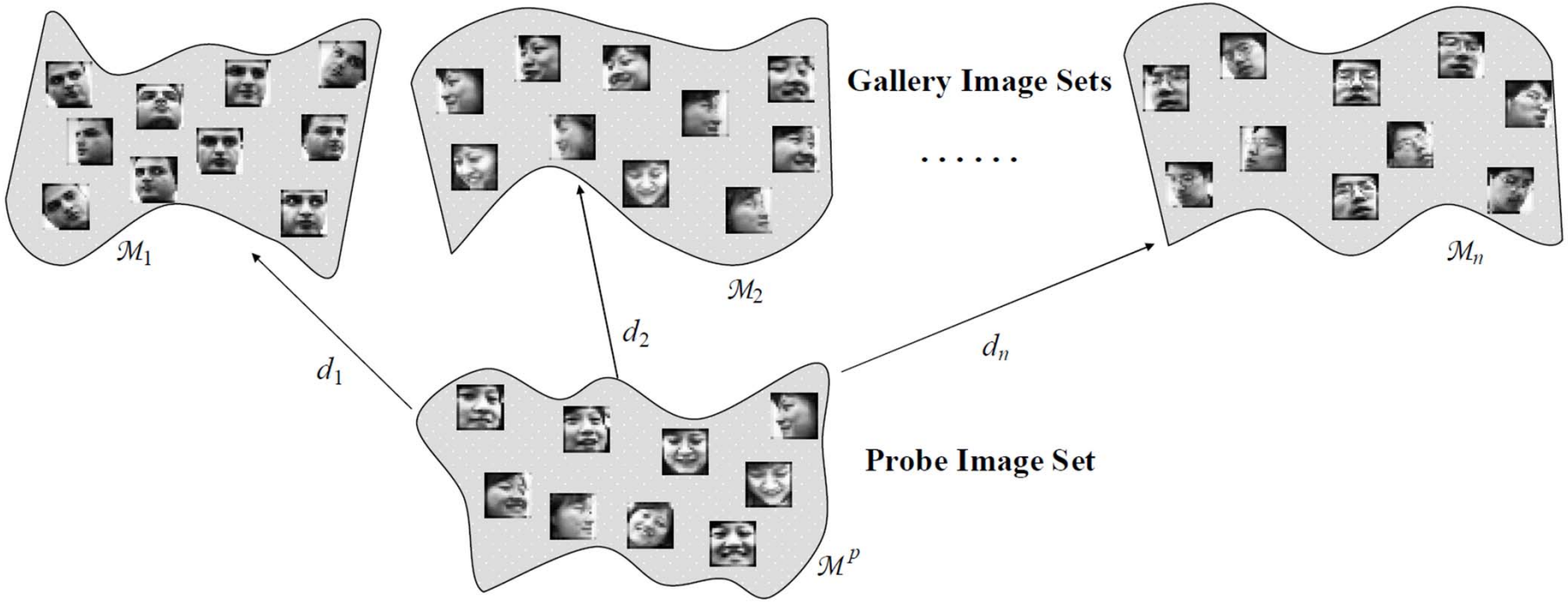
Visual Recognition Examples

Activity Recognition



Visual Recognition Examples

Image Set Classification



Visual Recognition Pipeline

Feature Representation

- LBP
- Gabor
- HOG
- SIFT
- STIP
- Deep features

Classification Model

- Support Vector Machine
- Nearest Neighbor
- Sparse Representation
- Neural Network
- Regression
- Metric Learning

Sample Similarity

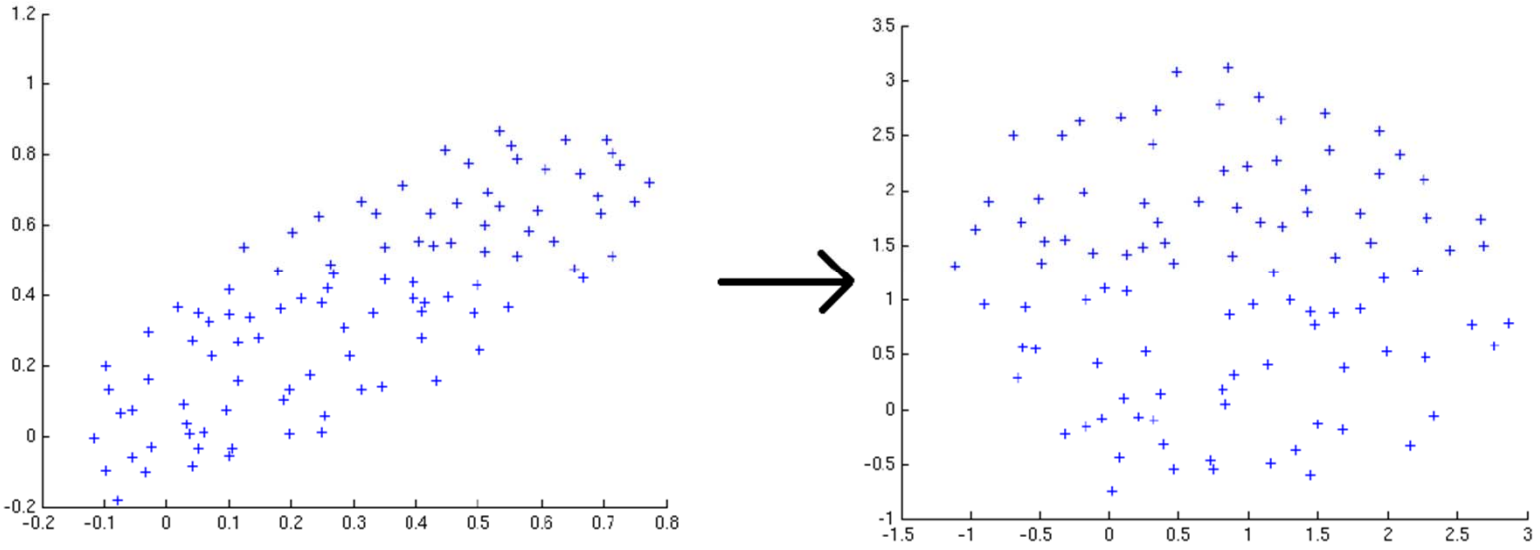
The squared Euclidean distance

$$\begin{aligned} d(\mathbf{x}_1, \mathbf{x}_2) &= \|\mathbf{x}_1 - \mathbf{x}_2\|_2^2 \\ &= (\mathbf{x}_1 - \mathbf{x}_2)^T (\mathbf{x}_1 - \mathbf{x}_2) \end{aligned}$$

Let $\Sigma = \sum_{i,j} (\mathbf{x}_i - \mu)(\mathbf{x}_j - \mu)^T$

The Mahalanobis distance

$$d_M(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 - \mathbf{x}_2)^T \Sigma^{-1} (\mathbf{x}_1 - \mathbf{x}_2)$$



Metric Learning

Applying Mahalanobis distance to learn a semi-positive definite (SPD) matrix

$$d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M} (\mathbf{x}_i - \mathbf{x}_j)}$$

Relationship with subspace learning

$$\begin{aligned} d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) &= \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M} (\mathbf{x}_i - \mathbf{x}_j)} \\ &= \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{W}^T \mathbf{W} (\mathbf{x}_i - \mathbf{x}_j)} \\ &= \|\mathbf{W}\mathbf{x}_i - \mathbf{W}\mathbf{x}_j\|_2 \end{aligned}$$

where $\mathbf{M} = \mathbf{W}^T \mathbf{W}$.

Categorization

Data structure: Linear/Kernel/Tensor

Label type: Supervised/Unsupervised/Semi-supervised

Supervision Type: Weakly-supervised/Strongly-supervised

Architecture model: Shallow/Deep

Metric number: Single-metric/Multi-metric

Recognition task: Sample-based/Set-based

Representative Metric Learning Algorithms

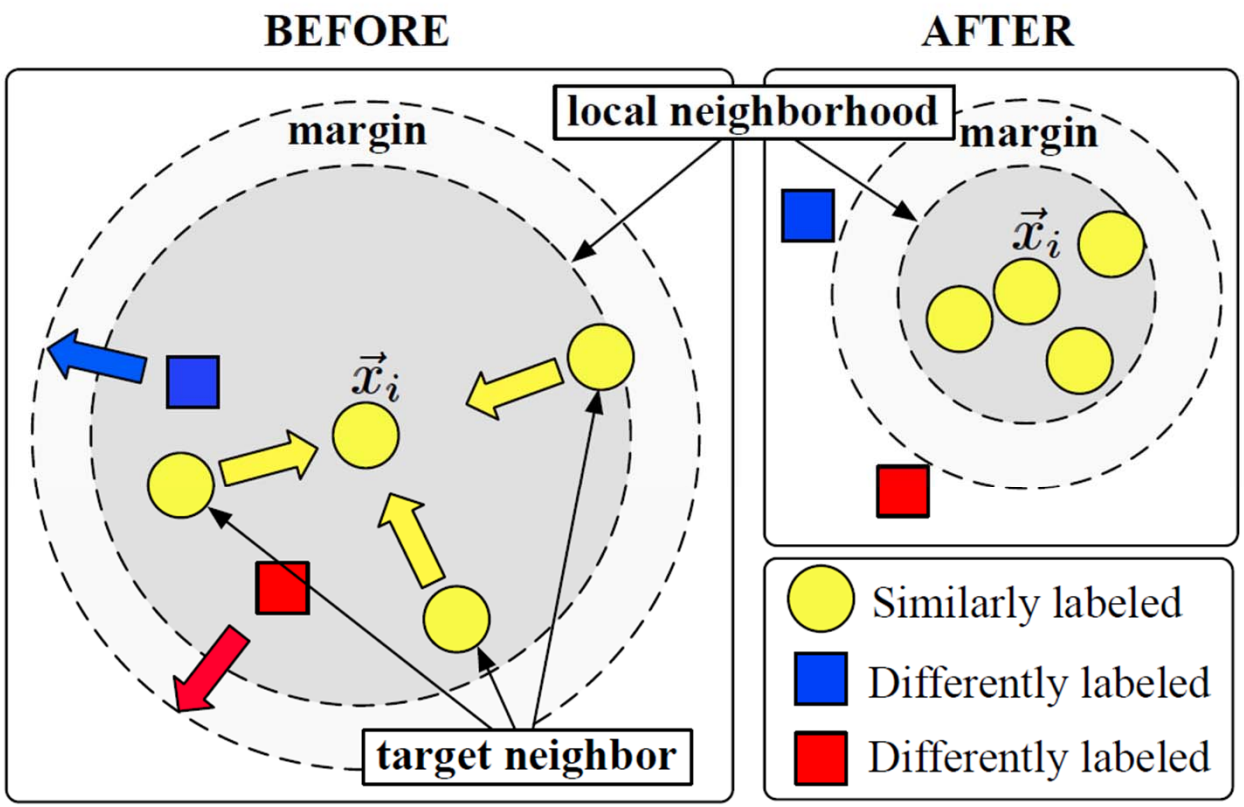
Large Margin Nearest Neighborhood (LMNN)

Minimize $\sum_{ij} \eta_{ij} (\vec{x}_i - \vec{x}_j)^\top \mathbf{M} (\vec{x}_i - \vec{x}_j) + c \sum_{ij} \eta_{ij} (1 - y_{il}) \xi_{ijl}$ **subject to:**

(1) $(\vec{x}_i - \vec{x}_l)^\top \mathbf{M} (\vec{x}_i - \vec{x}_l) - (\vec{x}_i - \vec{x}_j)^\top \mathbf{M} (\vec{x}_i - \vec{x}_j) \geq 1 - \xi_{ijl}$

(2) $\xi_{ijl} \geq 0$

(3) $\mathbf{M} \succeq 0$.



[Weinberger et al, NIPS2005]

Representative Metric Learning Algorithms

Information-Theoretic Metric Learning (ITML)

$$\begin{aligned} \min_{A} \quad & \mathrm{KL}(p(\boldsymbol{x}; A_0) \| p(\boldsymbol{x}; A)) \\ \text{subject to} \quad & d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq u \qquad (i,j) \in S, \\ & d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq \ell \qquad (i,j) \in D. \end{aligned}$$

where $\mathrm{KL}(p(\boldsymbol{x}; A_o) \| p(\boldsymbol{x}; A)) = \int p(\boldsymbol{x}; A_0) \log \frac{p(\boldsymbol{x}; A_0)}{p(\boldsymbol{x}; A)} \, d\boldsymbol{x}$.

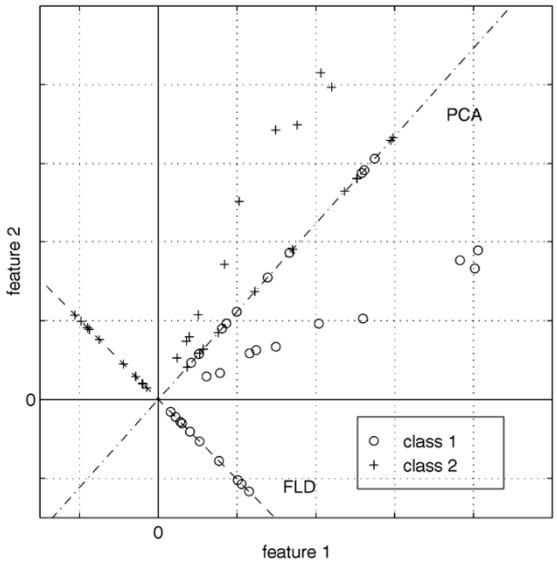
The optimization function can be re-formulated as

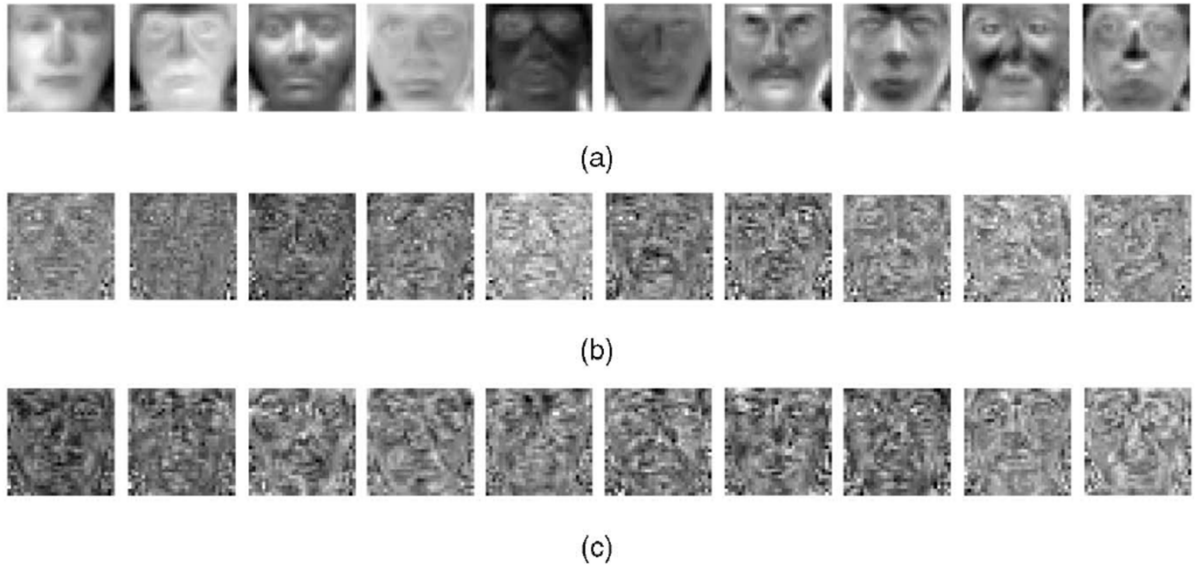
$$\begin{aligned} \min_{A \succeq 0} \quad & D_{\ell \mathrm{d}}(A, A_0) \\ \text{s.t.} \quad & \mathrm{tr}(A(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T) \leq u \qquad (i,j) \in S, \\ & \mathrm{tr}(A(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T) \geq \ell \qquad (i,j) \in D, \end{aligned}$$

[Davis et al, ICML2007]

Subspace Learning

Given a dataset $X = [x_1, x_2, \cdots, x_n]$, subspace learning aims to seek a low-dimensional subspace W to project each x_i to y_i, where $y_i = W^T x_i$, such that some characteristics are preserved.





Part 2: Distance Metric Learning for Face Recognition

Distance Metric Learning for Face Recognition

- Cost-Sensitive Metric Learning
- Deep Metric Learning
- Hamming Metric Learning

2.1 Cost-Sensitive Metric Learning

[1] **Jiwen Lu** and Yap-Peng Tan, Cost-sensitive subspace learning for face recognition, *CVPR*, 2010.

[2] **Jiwen Lu** and Yap-Peng Tan, Cost-sensitive subspace analysis and extensions for face recognition, *TIFS*, 2013.

[3] **Jiwen Lu**, Junlin Hu, Xiuzhuang Zhou *et al*, Neighborhood repulsed metric learning for kinship verification, C*VPR*, 2012.

[4] **Jiwen Lu**, Xiuzhuang Zhou, Yap-Peng Tan *et al*, Neighborhood repulsed metric learning for kinship verification, *PAMI*, 2014.

[5] **Jiwen Lu**, Yap-Peng Tan, Gang Wang and Gao Yang, Image-to-set face recognition using locality repulsion projections and sparse reconstruction-based similarity measure, *T-CSVT,* vol. 23, no. 6, pp. 1070-1080, 2013.

Cost-Sensitive Metric Learning

Motivation

- Most face recognition systems are cost-sensitive.

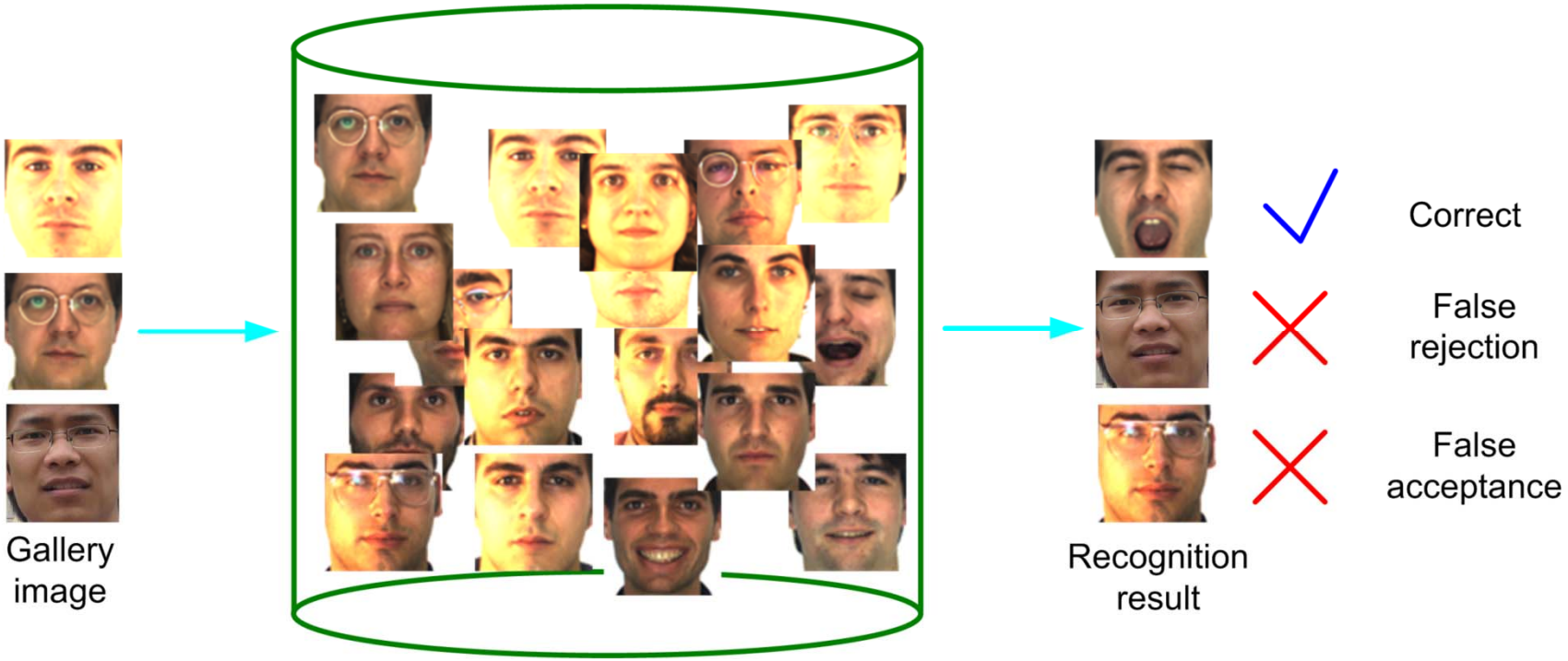


Illustration of different misclassifications of a practical face recognition system.

Cost-Sensitive Metric Learning

Basie idea: Construct a cost matrix to characterize different errors to learn cost-sensitive similarity measure.

	l_1	$\cdots$	l_j	$\cdots$	l_c
l_1	0	$\cdots$	C_{1j}	$\cdots$	C_{1c}
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
l_i	C_{i1}	$\cdots$	C_{ij}	$\cdots$	C_{i1}
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
l_c	C_{c1}	$\cdots$	C_{cj}	$\cdots$	0

Cost matrix of a practical face recognition system.

Cost-Sensitive Metric Learning

Cost-Sensitive Principal Component Analysis (CSPCA)

$$\max_w J_T(w) = \frac{1}{2}\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} cost(y_i, y_j)\|y_i - y_j\|^2$$

$$= \frac{1}{2}\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} cost(x_i, x_j)\|w^T x_i - w^T x_j\|^2$$

$$= w^T[\frac{1}{2}\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} cost(x_i, x_j)(x_i - x_j)(x_i - x_j)^T]w$$

$$= w^T S_T w$$

$$S_T = \frac{1}{2}\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} cost(x_i, x_j)(x_i - x_j)(x_i - x_j)^T$$

Cost-Sensitive Metric Learning

Cost-Sensitive Linear Discriminant Analysis (CSLDA)

$$\max_{w} J_T(w) = \frac{w^T S_b w}{w^T S_w w}$$

$$S_b = \sum_{k_1=1}^{c+1} \sum_{k_2=1}^{c+1} cost(k_1, k_2)(m_{k_1} - m_{k_2})(m_{k_1} - m_{k_2})^T$$

$$S_w = \sum_{k=1}^{c+1} \sum_{l(x_i)=k} f(k)(x_i - m_k)(x_i - m_k)^T$$

$$f(k) = \sum_{j=1}^{c} C_{kj}$$

Cost-Sensitive Metric Learning

Cost-Sensitive Locality Preserving Projections (CSLPP)

$$\min J(w) = \sum_{i=1}^{N}\sum_{j=1}^{N} cost(y_i, y_j)\|y_i - y_j\|^2 S_{ij}$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{N} cost(x_i, x_j) w^T (x_i - x_j)(x_i - x_j)^T w S_{ij}$$

$$= -2w^T A w$$

$$A = \sum_{i=1}^{N}\sum_{j=1, i\neq j}^{N} cost(x_i, x_j) x_i S_{ij} x_j^T$$

$$\max_w w^T A w$$

$$s.t. w^T w = 1.$$

Cost-Sensitive Metric Learning

Cost-Sensitive Marginal Fisher Analysis (CSMFA)

$$\arg\max_{w} F(w) = F_1(w) - F_2(w)$$

$$F_1(w) = \sum_{ij=1}^{N} cost(x_i, x_j) \|w^T x_i - w^T x_j\|^2 G_{ij}^1$$

$$F_2(w) = \sum_{ij=1}^{N} cost(x_i, x_j) \|w^T x_i - w^T x_j\|^2 G_{ij}^2$$

$$G_{ij}^1 = \begin{cases} 1 & \text{if } x_i \in N_{k_2}(x_j) \text{ or } x_j \in N_{k_2}(x_i) \\ & \text{and } l(x_i) \neq l(x_j) \\ 0 & \text{otherwise} \end{cases}$$

$$G_{ij}^2 = \begin{cases} 1 & \text{if } x_i \in N_{k_1}(x_j) \text{ or } x_j \in N_{k_1}(x_i) \\ & \text{and } l(x_i) = l(x_j) \\ 0 & \text{otherwise} \end{cases}$$

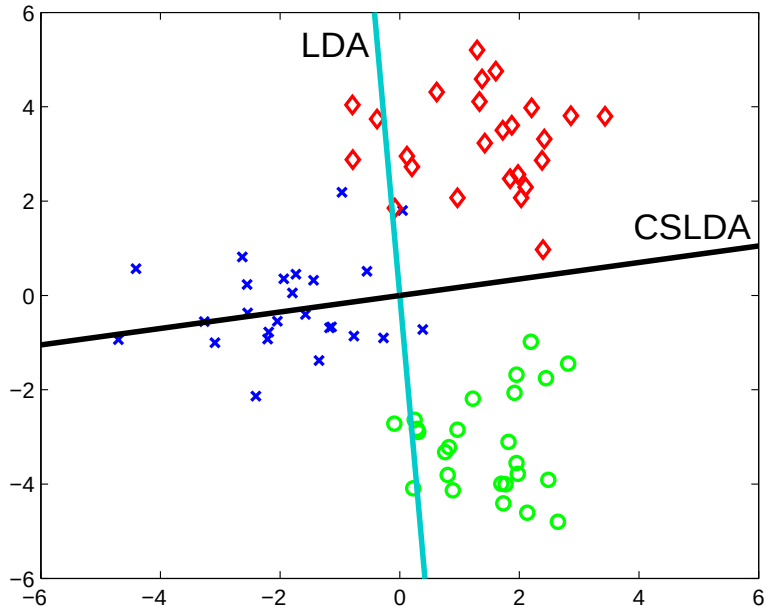
Experimental Results

Toy Example

$$\begin{cases} x_k^i = x_0^i + \delta_1(k) \\ y_k^i = y_0^i + \delta_2(k) \end{cases}$$

	Class 1	Class 2	Class 3
Class 1	0	100	100
Class 2	10	0	1
Class 3	1	10	0

Method	cost	TER (%)
LDA	804	**16.0**
CSLDA	**646**	40.0



Experimental Results

Face Recognition

- Accepting any impostor causes the same loss.
- Mis-recognizing a gallery person as another gallery person or an impostor causes different amounts of loss.

	G_1	$\cdots$	G_c	I
G_1	0	$\cdots$	C_{GG}	C_{GI}
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
G_c	C_{GG}	$\cdots$	0	C_{GI}
I	C_{IG}	$\cdots$	C_{IG}	0

Cost matrix of a face recognition-based access control system.

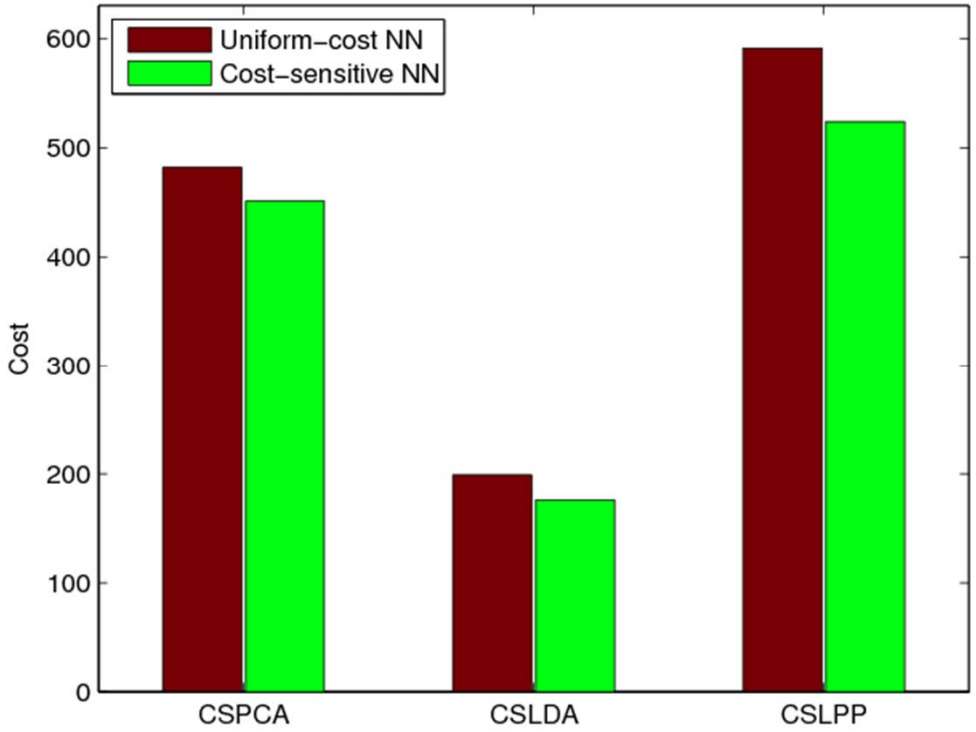
Experimental Results

Face Representation

First 10 basis images of (a) PCA, (b) LDA, (c) LPP, (d) CSPCA, (e) CSLDA, and (f) CSLPP.

Experimental Results

Face Recognition



Comparisons with cost-sensitive and uniform-cost classifiers.

Experimental Results

Face Recognition

- Dataset---AR/FERET face database

Dataset	M	N_G	N_I	$C_{IG} : C_{GI} : C_{GG}$	
				access control	criminal search
AR	30	7	350	20:2:1	1:100:10
FERET	70	3	350	20:2:1	1:100:10

- Results

Method	AR				FERET			
	cost	err (%)	err_{GI} (%)	err_{IG} (%)	cost	err (%)	err_{GI} (%)	err_{IG} (%)
LDA	275	6.07	10.48	3.43	129	6.79	16.67	0.86
CSLDA	195	6.35	11.43	2.29	100	7.21	18.23	0.57
MFA	265	**5.95**	**10.24**	3.37	123	**6.65**	**16.55**	0.82
CSMFA	**190**	6.28	11.26	**2.25**	**98**	7.13	18.12	**0.52**

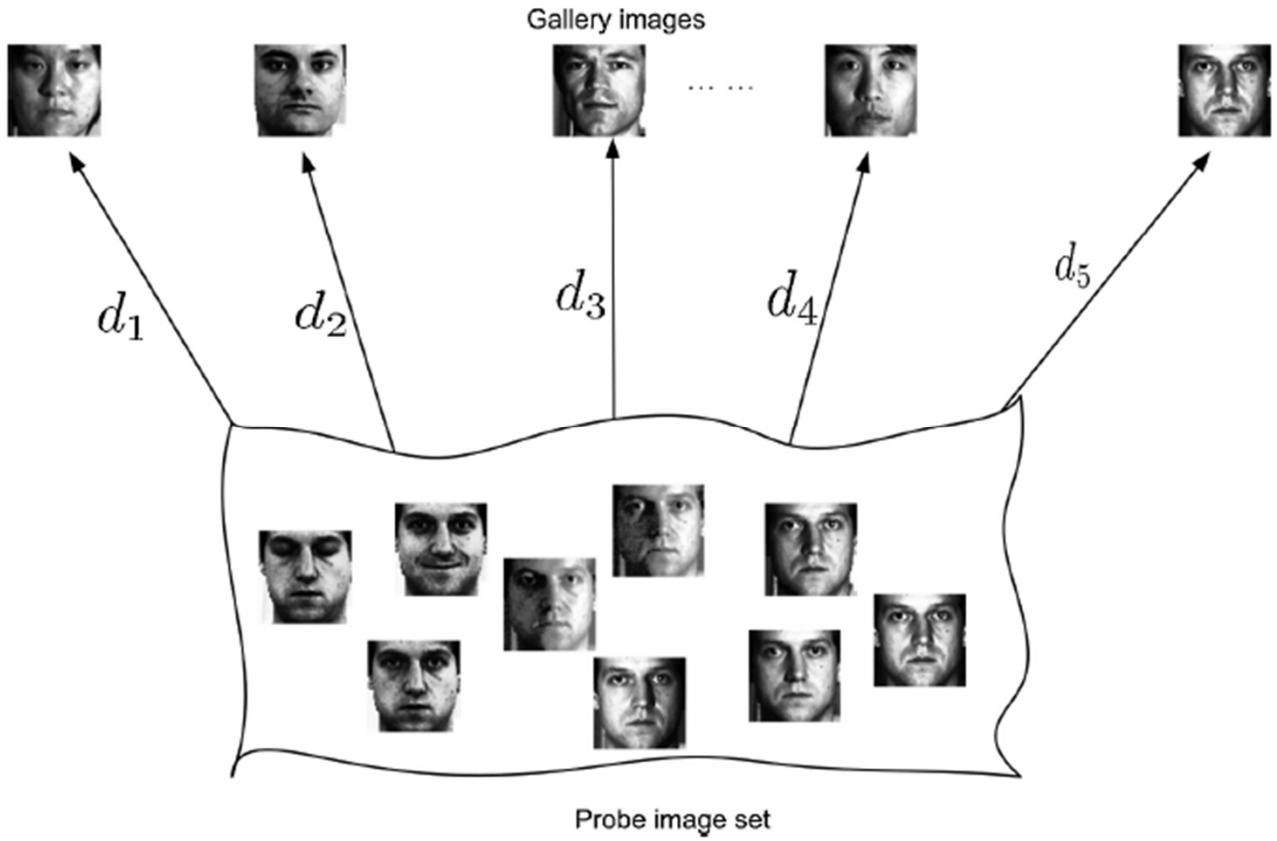
Comparisons for the access control application.

Method	AR				FERET			
	cost	err (%)	err_{GI} (%)	err_{IG} (%)	cost	err (%)	err_{GI} (%)	err_{IG} (%)
LDA	1339	6.07	10.48	3.43	624	6.79	16.67	0.86
CSLDA	1171	6.61	12.86	2.86	364	7.32	17.32	0.29
MFA	1320	**5.95**	**10.02**	3.21	615	**6.43**	**16.32**	0.82
CSMFA	**1152**	6.25	12.35	**2.34**	**352**	7.14	16.98	**0.24**

Comparisons for the criminal search application.

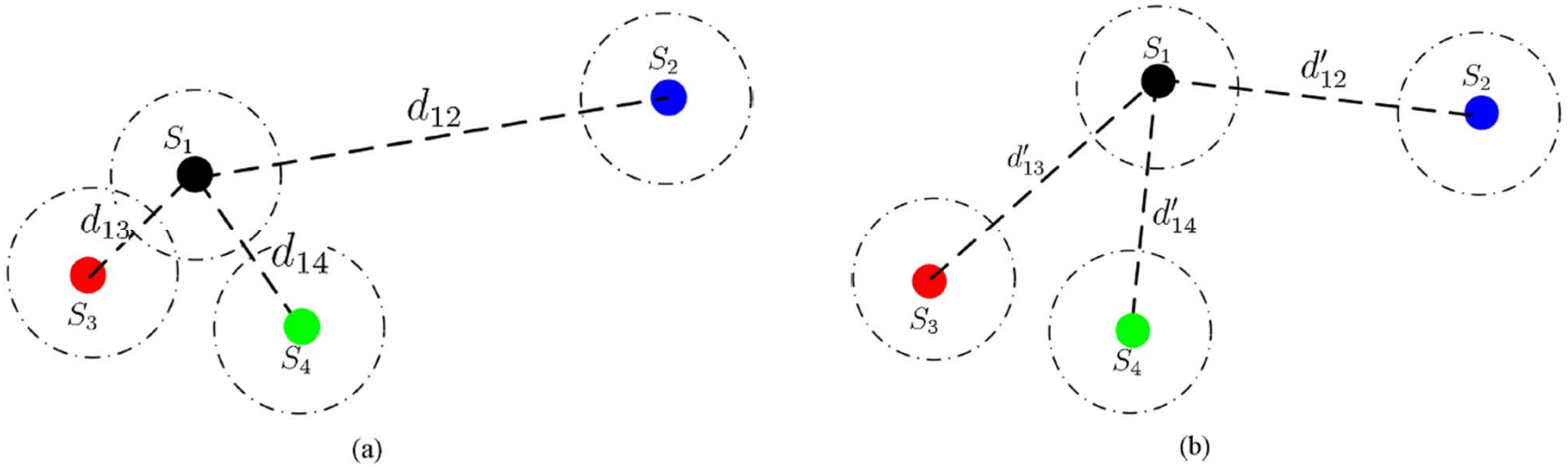
Locality Repulsed Metric Learning

Image-to-Set Face Recognition



Locality Repulsed Metric Learning

Locality Repulsed Projections (LRP)



Locality Repulsed Metric Learning

Objective function

$$\max_W \frac{J_1(W)}{J_2(W)} = \frac{\sum_{i=1}^{N} \| W^T x_i - \sum_{t=1}^{k} \alpha_{it} W^T x_{it} \|^2}{\sum_{i=1}^{N} \sum_{j=1}^{N} \| W^T x_i - W^T x_j \|^2 S_{ij}}$$

$$S_{ij} = \begin{cases} 0, & \text{if } x_i \in N_k(x_j) \text{ or } x_j \in N_k(x_i) \\ 1, & \text{otherwise} \end{cases}$$

$$\begin{aligned} J_1(W) &= \sum_{i=1}^{N} \| W^T x_i - \sum_{t=1}^{k} \alpha_{it} W^T x_{it} \|^2 \\ &= tr\left(\sum_{i=1}^{N} W^T (x_i - \sum_{t=1}^{k} \alpha_{it} x_{it})(x_i - \sum_{t=1}^{k} \alpha_{it} x_{it})^T W \right) \\ &= W^T tr\left(\sum_{i=1}^{N} (x_i - \sum_{t=1}^{k} \alpha_{it} x_{it})(x_i - \sum_{t=1}^{k} \alpha_{it} x_{it})^T \right) W \\ &= W^T H_1 W \end{aligned}$$

$$H_1 \triangleq tr\left(\sum_{i=1}^{N} (x_i - \sum_{t=1}^{k} \alpha_{it} x_{it})(x_i - \sum_{t=1}^{k} \alpha_{it} x_{it})^T \right)$$

Locality Repulsed Metric Learning

Objective function

$$
\begin{aligned}
J_2(W) &= \sum_{i=1}^{N}\sum_{j=1}^{N} \|W^T x_i - W^T x_j\|^2 S_{ij} \\
&= tr\left(\sum_{i=1}^{N}\sum_{j=1}^{N} W^T (x_i - x_j)(x_i - x_j)^T S_{ij} W\right) \\
&= W^T tr\left(\sum_{i=1}^{N}\sum_{j=1}^{N} (x_i - x_j)(x_i - x_j)^T S_{ij}\right) W \\
&= W^T H_2 W
\end{aligned}
$$

$$
H_2 \triangleq tr\left(\sum_{i=1}^{N}\sum_{j=1}^{N} (x_i - x_j)(x_i - x_j)^T S_{ij}\right)
$$

The projection can be obtained

$$
H_1 w = \lambda H_2 w
$$

Locality Repulsed Metric Learning

Objective function

$$\max_W \frac{J_1(W)}{J_2(W)} = \frac{\sum_{i=1}^{N} \| W^T x_i - \sum_{t=1}^{k} \alpha_{it} W^T x_{it} \|^2}{\sum_{i=1}^{N} \sum_{j=1}^{N} \| W^T x_i - W^T x_j \|^2 S_{ij}}$$

$$S_{ij} = \begin{cases} 0, & \text{if } x_i \in N_k(x_j) \text{ or } x_j \in N_k(x_i) \\ 1, & \text{otherwise} \end{cases}$$

$$\begin{aligned} J_1(W) &= \sum_{i=1}^{N} \| W^T x_i - \sum_{t=1}^{k} \alpha_{it} W^T x_{it} \|^2 \\ &= tr \left(\sum_{i=1}^{N} W^T (x_i - \sum_{t=1}^{k} \alpha_{it} x_{it})(x_i - \sum_{t=1}^{k} \alpha_{it} x_{it})^T W \right) \\ &= W^T tr \left(\sum_{i=1}^{N} (x_i - \sum_{t=1}^{k} \alpha_{it} x_{it})(x_i - \sum_{t=1}^{k} \alpha_{it} x_{it})^T \right) W \\ &= W^T H_1 W \end{aligned}$$

$$H_1 \triangleq tr \left(\sum_{i=1}^{N} (x_i - \sum_{t=1}^{k} \alpha_{it} x_{it})(x_i - \sum_{t=1}^{k} \alpha_{it} x_{it})^T \right)$$

Experimental Results

Image-to-Set Face Recognition

Method	AR	CMU PIE	Yale B	FERET	LFW
PCA	86.2	93.7	44.7	75.5	18.5
LPP	87.4	94.7	57.9	76.8	19.9
NPE	86.9	94.2	56.8	77.5	20.5
LRP	**89.6**	**95.7**	**73.9**	**80.2**	**21.2**

Single-Sample Face Recognition

Method	Fb	Fc	Dup1	Dup2	Year
Block FLD [5]	73.3	50.0	41.3	33.8	2004
LBP [1]	93.0	51.0	61.0	50.0	2004
LGBP [47]	94.0	97.0	68.0	53.0	2005
HGPP [43]	97.6	98.9	77.7	76.1	2007
Results of [48]	99.5	99.5	85.0	79.5	2007
SVD-LDA [11]	84.5	54.0	45.0	35.4	2008
Adapted FLD [31]	88.5	71.6	53.3	35.0	2010
POEM [35]	97.0	95.0	77.6	76.2	2010
LDP [42]	94.0	83.0	62.0	53.0	2010
G-LDP [42]	97.0	95.0	71.0	69.0	2010
RAS [29]	95.7	99.0	80.3	80.3	2010
GV-LBP [22]	98.1	98.5	80.9	81.2	2011
Results of [36]	99.7	100	91.7	90.6	2011
DMMA [24]	98.1	98.5	81.6	83.2	2012
Our method	98.1	98.0	82.2	82.4	

Neighborhood Repulsed Metric Learning

Kinship verification via Face Images

Father-Son (F-S)

Father-Daughter (F-D)

Mother-Son (M-S)

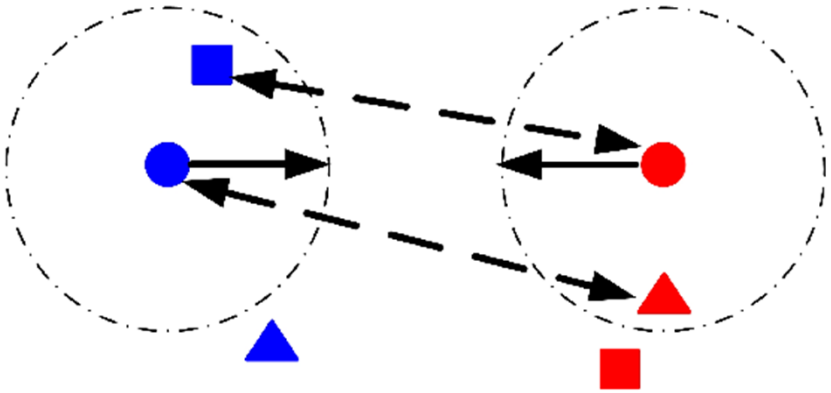
Mother-Daughter (M-D)

Neighborhood Repulsed Metric Learning

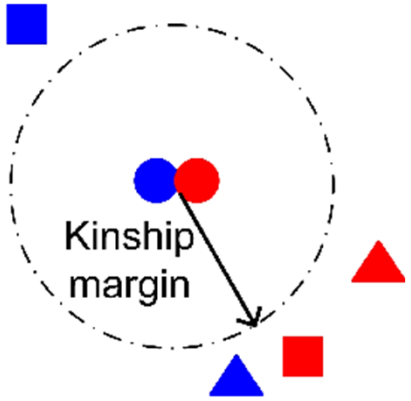
Motivation

- For the verification task, the number of negative pairs is larger than the number of positive pairs if we know the exact label information of each sample.
- The importance of different negative pairs is different. Some negative pairs are very discriminative and some are not so discriminative.
- It is desirable to identify the most informative negative pairs and ignore the less informative negative pairs to learn a discriminative metric for verification.

Neighborhood Repulsed Metric Learning



(a)



(b)

$$\max_A J(A) = J_1(A) + J_2(A) - J_3(A)$$

$$= \frac{1}{Nk}\sum_{i=1}^{N}\sum_{t_1=1}^{k} d^2(x_i, y_{it_1}) + \frac{1}{Nk}\sum_{i=1}^{N}\sum_{t_2=1}^{k} d^2(x_{it_2}, y_i)$$

$$-\frac{1}{N}\sum_{i=1}^{N} d^2(x_i, y_i)$$

$$= \frac{1}{Nk}\sum_{i=1}^{N}\sum_{t_1=1}^{k}(x_i - y_{it_1})^T A(x_i - y_{it_1})$$

$$+\frac{1}{Nk}\sum_{i=1}^{N}\sum_{t_2=1}^{k}(x_{it_2} - y_i)^T A(x_{it_2} - y_i)$$

$$-\frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^T A(x_i - y_i)$$

Algorithm 1: NRML

Input: Training images: $S = \{(x_i, y_i)|i = 1, 2, \cdots, N\}$, Parameters: neighborhood size k, iteration number T, and convergence error ε (set as 0.0001).

Output: Distance metric W.

Step 1 (Initialization):

Search the k-nearest neighbors for each x_i and y_i by using the conventional Euclidean metric.

Step 2 (Local optimization):

For $r = 1, 2, \cdots, T$, repeat

2.1. Compute H_1, H_2 and H_3, respectively.

2.2. Solve the eigenvalue problem in Eq. (9).

2.3. Obtain $W^r = [w_1, w_2, \cdots, w_l]$.

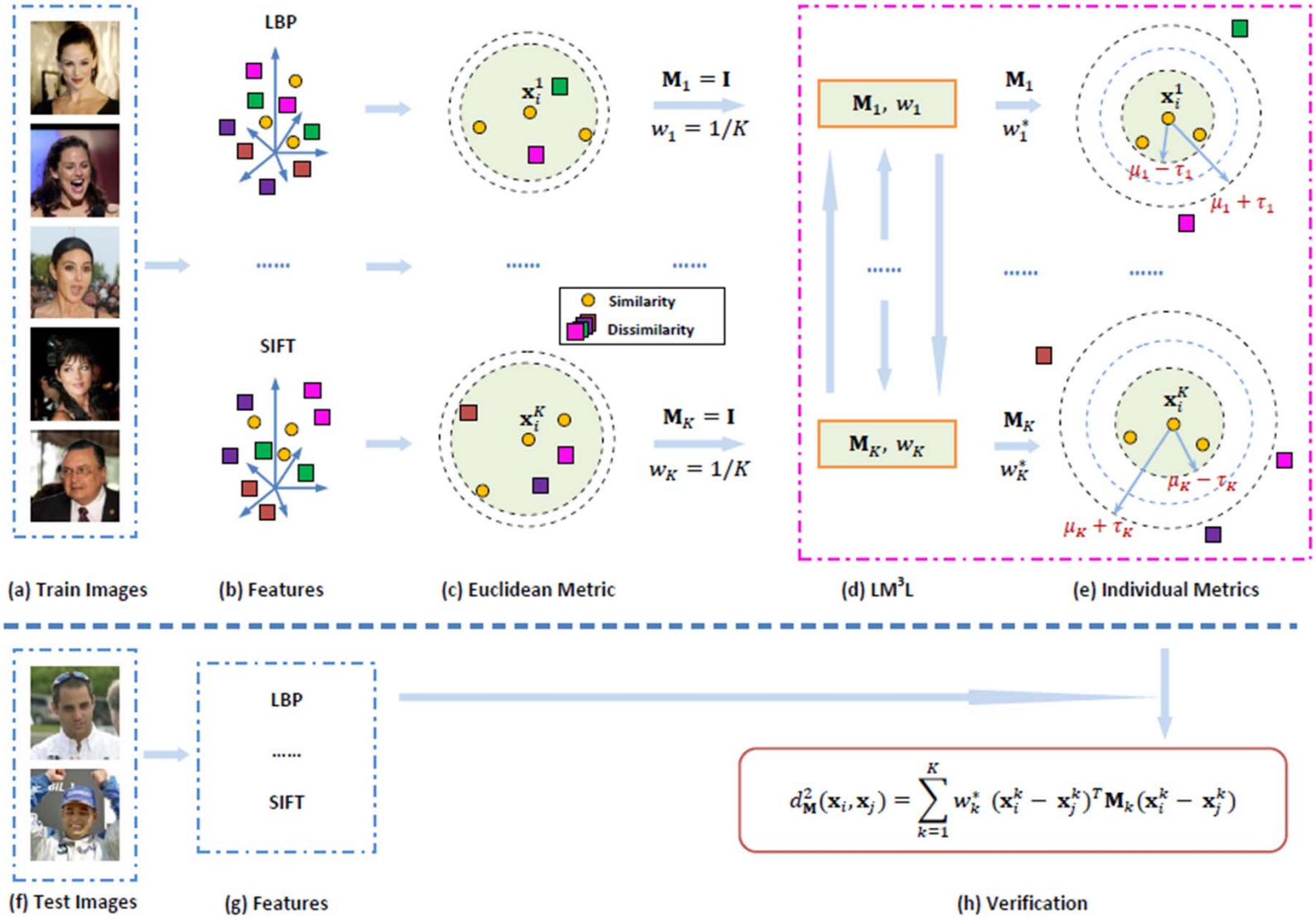
2.4. Update the k-nearest neighbors of x_i and y_i by W^r.

2.5. If $r > 2$ and $|W^r - W^{r-1}| < \varepsilon$, go to Step 3.

Step 3 (Output distance metric):

Output distance metric $W = W^r$.

Multi-view Neighborhood Repulsed Metric Learning



Multi-view Neighborhood Repulsed Metric Learning

$$\max_{W,\beta} \sum_{p=1}^{K} \beta_p tr[W^T(H_1^p + H_2^p - H_3^p)W]$$

$$\text{subject to} \quad W^T W = I, \sum_{p=1}^{K} \beta_p = 1, \beta_p \geq 0.$$

$$\max_{W,\beta} \sum_{p=1}^{K} \beta_p^q tr[W^T(H_1^p + H_2^p - H_3^p)W]$$

$$\text{subject to} \quad W^T W = I, \sum_{p=1}^{K} \beta_p = 1, \beta_p \geq 0.$$

Algorithm 2: MNRML

Input: Training images: $S^p = \{(x_i^p, y_i^p)|i = 1, 2, \cdots, N\}$ be the pth view set of N pairs of kinship images, Parameters: neighborhood size k, iteration number T, tuning parameter q, and convergence error ε (set as 0.0001).

Output: Distance metric W.

Step 1 (Initialization):

1.1. Set $\beta = [1/K, 1/K, \cdots, 1/K]$;

1.2. Obtain W^0 by solving Eq. (18).

Step 2 (Local optimization):

For $r = 1, 2, \cdots, T$, repeat

2.1. Compute β by using Eq. (16).

2.2. Obtain W^r by solving Eq. (18).

2.3. If $r > 2$ and $|W^r - W^{r-1}| < \varepsilon$, go to Step 3.

Step 3 (Output distance metric):

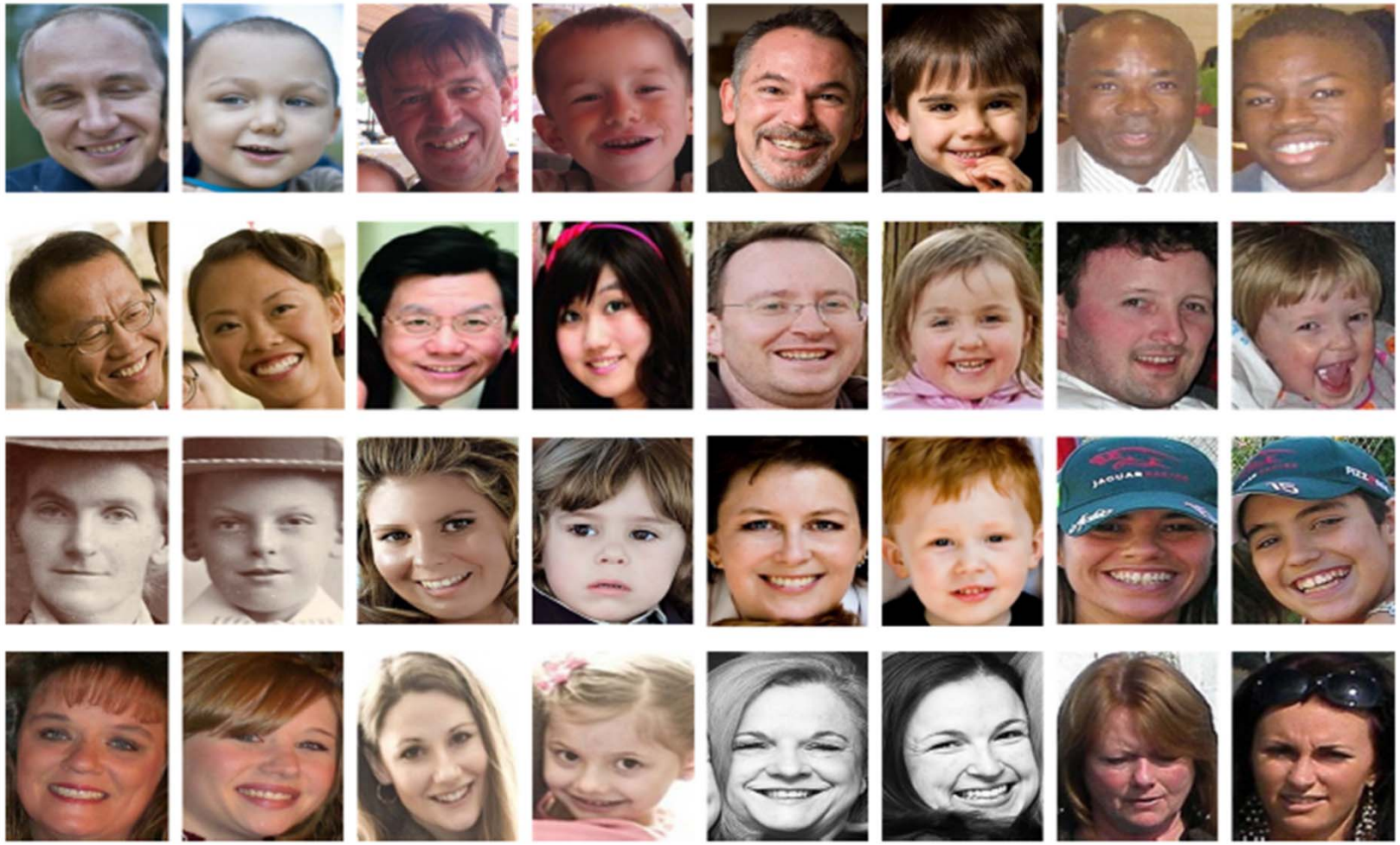
Output distance metric $W = W^r$.

Datasets

- KinFaceW-I: 500 kinship image face pairs

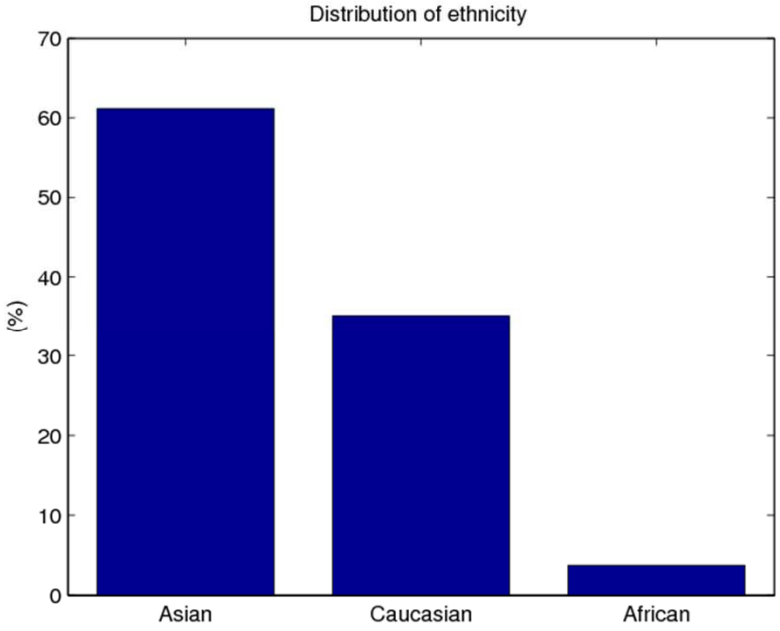
Datasets

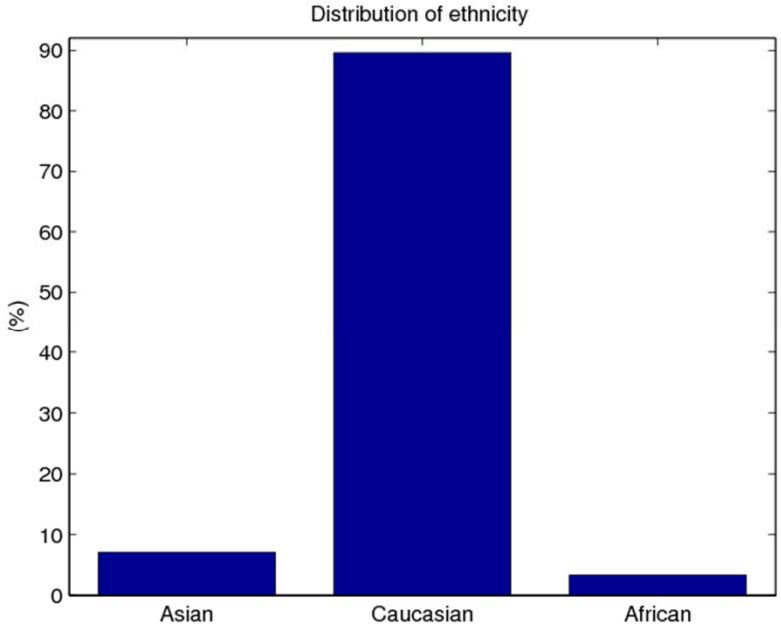
- KinFaceW-II:1000 kinship image face pairs



Datasets

- Statistics





Experimental Results

- Baseline Results

Feature	F-S	F-D	M-S	M-D	Mean
LBP	62.7	60.2	54.4	61.4	59.7
LE	66.1	59.1	58.9	68.0	63.0
SIFT	65.5	59.0	55.5	55.4	58.8
TPLBP	56.3	60.5	56.0	62.2	58.7

Correct verification accuracy on the KinFaceW-I dataset.

Feature	F-S	F-D	M-S	M-D	Mean
LBP	64.0	63.5	62.8	63.0	63.3
LE	69.8	66.1	72.8	72.0	69.9
SIFT	60.0	56.9	54.8	55.4	56.8
TPLBP	64.4	60.6	60.8	62.9	62.2

Correct verification accuracy on the KinFaceW-II dataset.

Experimental Results

- Comparisons with existing metric learning methods

Method	Feature	F-S	F-D	M-S	M-D	Mean
CSML	LBP	63.7	61.2	55.4	62.4	60.7
	LE	61.1	58.1	60.9	70.0	62.5
	SIFT	66.5	60.0	60.0	56.4	59.8
	TPLBP	57.3	61.5	63.2	57.0	59.7
NCA	LBP	61.7	62.2	56.4	62.4	60.7
	LE	62.1	57.1	61.9	69.0	62.3
	SIFT	67.5	61.0	61.0	57.4	60.8
	TPLBP	56.3	60.5	62.2	56.0	58.7
LMNN	LBP	62.7	63.2	57.4	63.4	61.7
	LE	63.1	58.1	62.9	70.0	63.3
	SIFT	69.5	63.0	63.0	59.4	62.8
	TPLBP	57.3	61.5	63.2	57.0	59.7
NRML	LBP	64.7	65.2	59.4	65.4	63.7
	LE	64.1	59.1	63.9	71.0	64.3
	SIFT	70.5	64.0	64.0	60.4	63.8
	TPLBP	59.3	63.5	65.2	60.0	62.9
MNRML	All	**72.5**	**66.5**	**66.2**	**72.0**	**69.9**

Correct verification accuracy on the KinFaceW-I dataset.

Experimental Results

- Comparisons with existing metric learning methods

Method	Feature	F-S	F-D	M-S	M-D	Mean
CSML	LBP	66.0	65.5	64.8	65.0	65.3
	LE	71.8	68.1	73.8	74.0	71.9
	SIFT	62.0	58.9	56.8	57.4	58.8
	TPLBP	66.4	62.6	62.8	64.9	64.2
NCA	LBP	67.0	66.5	65.8	66.0	66.3
	LE	73.8	70.1	74.8	75.0	73.5
	SIFT	63.0	59.9	58.8	59.4	60.4
	TPLBP	67.4	63.6	63.8	66.9	66.5
LMNN	LBP	68.0	68.5	68.8	67.0	68.2
	LE	74.8	71.1	75.8	76.0	74.5
	SIFT	65.0	57.9	58.8	59.4	60.4
	TPLBP	68.4	65.6	65.8	67.9	68.1
NRML	LBP	69.0	69.5	69.8	69.0	69.5
	LE	76.8	73.1	76.8	77.0	75.7
	SIFT	68.0	60.9	60.8	61.4	62.8
	TPLBP	70.4	67.6	67.8	69.9	70.1
MNRML	All	**76.9**	**74.3**	**77.4**	**77.6**	**76.5**

Correct verification accuracy on the KinFaceW-II dataset.

Experimental Results

- Comparisons with human observers

Method	F-S	F-D	M-S	M-D	Mean
HumanA	61.00	58.00	66.00	70.00	63.75
HumanB	67.00	65.00	75.00	77.00	71.00

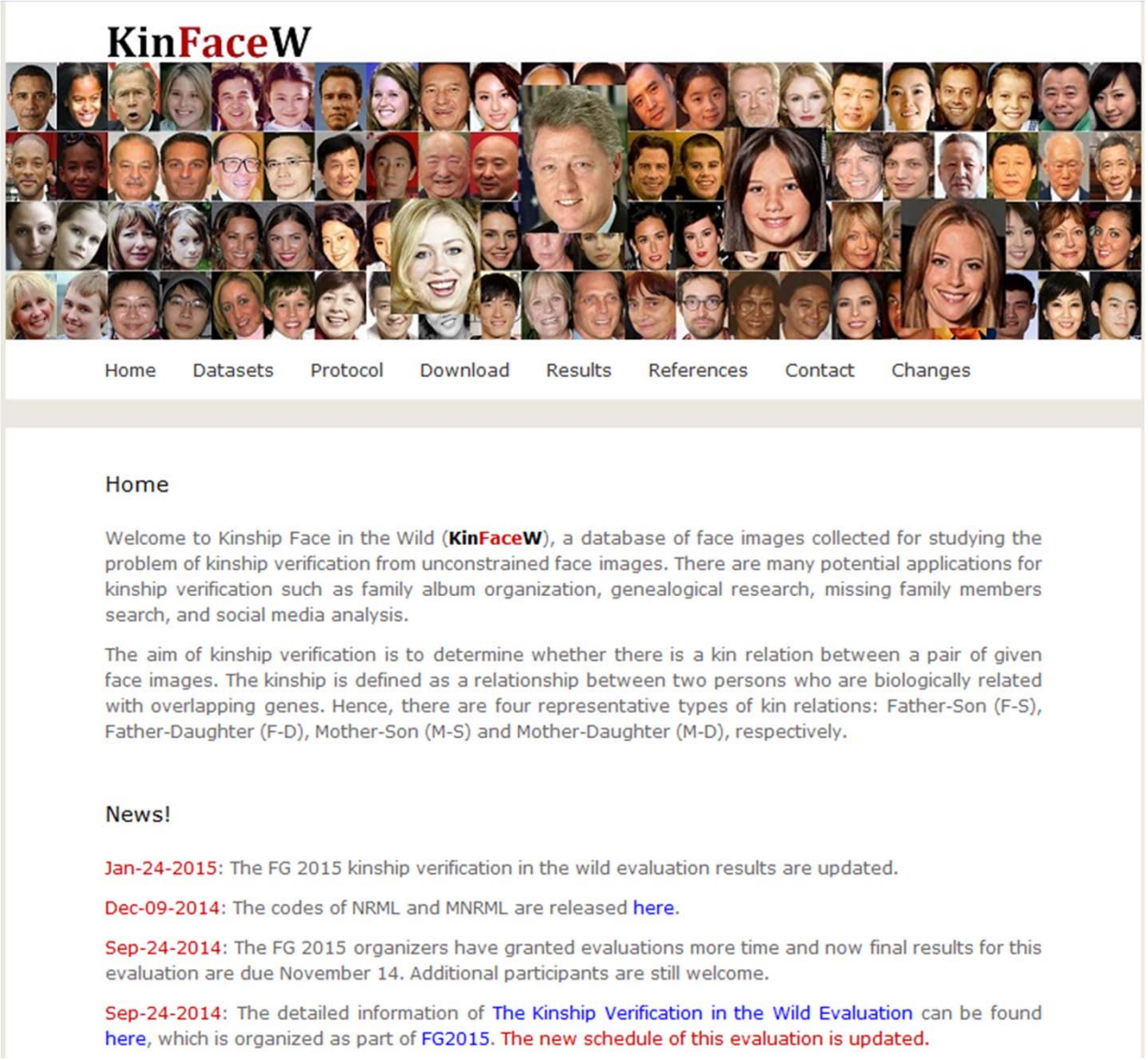
Correct verification accuracy on the KinFaceW-I dataset.

Method	F-S	F-D	M-S	M-D	Mean
HumanA	61.00	61.00	69.00	73.00	66.75
HumanB	70.00	68.00	78.00	80.00	74.00

Correct verification accuracy on the KinFaceW-II dataset.

Webpage

- www.kinfacew.com



2.2 Deep Metric Learning

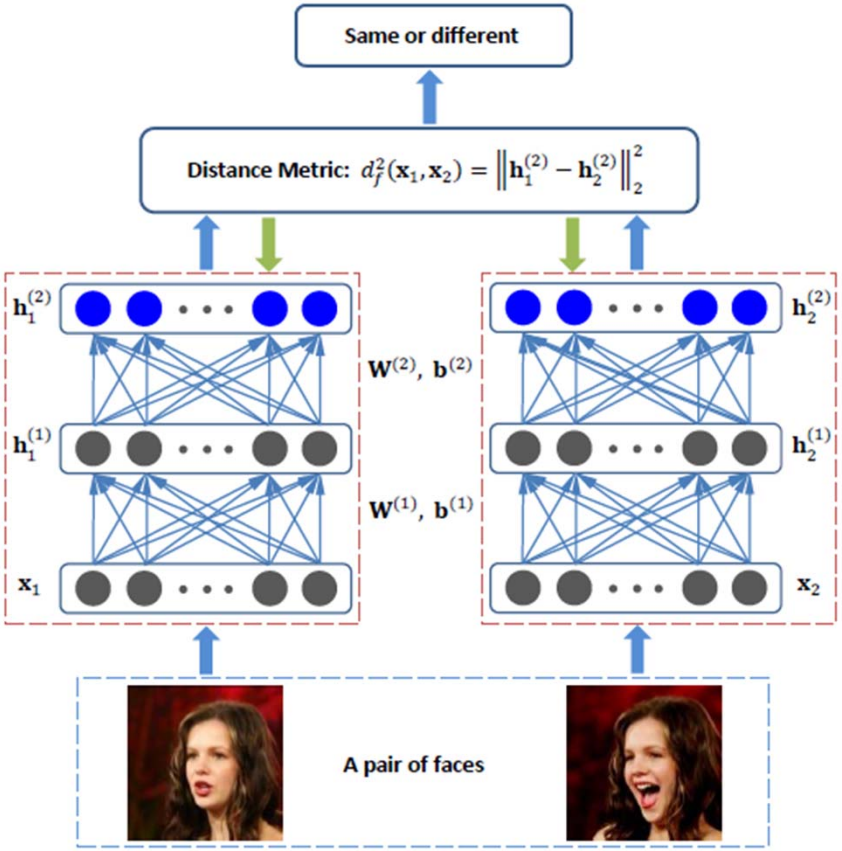
[6] Junlin Hu, **Jiwen Lu**, Yap-Peng Tan, Discriminative deep metric learning for face verification in the wild, *CVPR*, 2014.
[7] Junlin Hu, **Jiwen Lu**, and Yap-Peng Tan, Deep transfer metric learning, *CVPR*, 2015.
[8] **Jiwen Lu**, Gang Wang, Weihong Deng, Pierre Moulin, and Jie Zhou, Multi-manifold deep metric learning for image set classification, *CVPR,* 2015.
[9] Venice Erin Liong, **Jiwen Lu**, Gang Wang *et al*, Deep hashing for compact binary codes learning, *CVPR*, 2015.

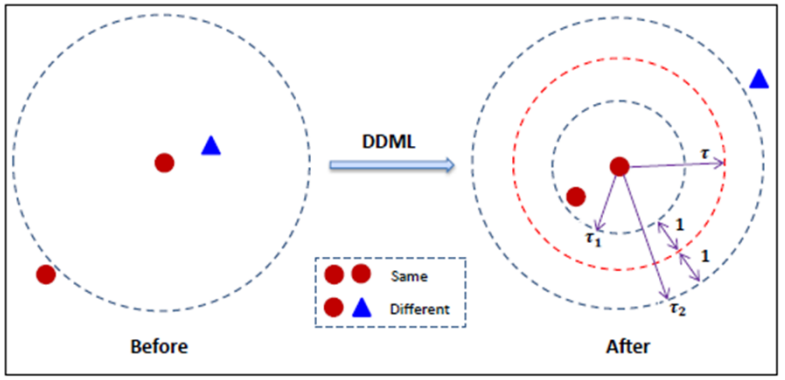
Discriminative Deep Metric Learning

$$
\begin{aligned}
d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) &= \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M} (\mathbf{x}_i - \mathbf{x}_j)} \\
&= \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{W}^T \mathbf{W} (\mathbf{x}_i - \mathbf{x}_j)} \\
&= \|\mathbf{W}\mathbf{x}_i - \mathbf{W}\mathbf{x}_j\|_2
\end{aligned}
$$

- Conventional metric learning methods only seek a linear mapping, which cannot capture the nonlinear manifold where face images usually lie on.
- The kernel trick can be employed to implicitly map face samples into a high-dimensional feature space and then learn a distance metric in the high-dimensional space. However, these methods cannot explicitly obtain the nonlinear mapping functions, which usually suffer from the scalability problem.

Discriminative Deep Metric Learning





$$\ell_{ij}\left(\tau - d_f^2(\mathbf{x}_i, \mathbf{x}_j)\right) > 1.$$

$$\begin{aligned}\arg\min_f J &= J_1 + J_2 \\ &= \frac{1}{2}\sum_{i,j} g\left(1 - \ell_{ij}\left(\tau - d_f^2(\mathbf{x}_i, \mathbf{x}_j)\right)\right) \\ &+ \frac{\lambda}{2}\sum_{m=1}^{M}\left(\left\|\mathbf{W}^{(m)}\right\|_F^2 + \left\|\mathbf{b}^{(m)}\right\|_2^2\right)\end{aligned}$$

$$f(\mathbf{x}) = \mathbf{h}^{(M)} = s\left(\mathbf{W}^{(M)}\mathbf{h}^{(M-1)} + \mathbf{b}^{(M)}\right) \in \mathbb{R}^{p^{(M)}}$$

Discriminative Deep Metric Learning

$$\begin{aligned}
\frac{\partial J}{\partial \mathbf{W}^{(m)}} &= \sum_{i,j} \left(\Delta_{ij}^{(m)} \mathbf{h}_i^{(m-1)^T} + \Delta_{ji}^{(m)} \mathbf{h}_j^{(m-1)^T} \right) \\
&+ \lambda \, \mathbf{W}^{(m)} \\
\frac{\partial J}{\partial \mathbf{b}^{(m)}} &= \sum_{i,j} \left(\Delta_{ij}^{(m)} + \Delta_{ji}^{(m)} \right) + \lambda \, \mathbf{b}^{(m)}
\end{aligned}$$

where

$$\begin{aligned}
\Delta_{ij}^{(M)} &= g'(c) \ell_{ij} \left(\mathbf{h}_i^{(M)} - \mathbf{h}_j^{(M)} \right) \odot s' \left(\mathbf{z}_i^{(M)} \right) \\
\Delta_{ji}^{(M)} &= g'(c) \ell_{ij} \left(\mathbf{h}_j^{(M)} - \mathbf{h}_i^{(M)} \right) \odot s' \left(\mathbf{z}_j^{(M)} \right) \\
\Delta_{ij}^{(m)} &= \left(\mathbf{W}^{(m+1)^T} \Delta_{ij}^{(m+1)} \right) \odot s' \left(\mathbf{z}_i^{(m)} \right) \\
\Delta_{ji}^{(m)} &= \left(\mathbf{W}^{(m+1)^T} \Delta_{ji}^{(m+1)} \right) \odot s' \left(\mathbf{z}_j^{(m)} \right) \\
c &\triangleq 1 - \ell_{ij} \left(\tau - d_f^2(\mathbf{x}_i, \mathbf{x}_j) \right) \\
\mathbf{z}_i^{(m)} &\triangleq \mathbf{W}^{(m)} \mathbf{h}_i^{(m-1)} + \mathbf{b}^{(m)}
\end{aligned}$$

Discriminative Deep Metric Learning

$$\mathbf{W}^{(m)} = \mathbf{W}^{(m)} - \mu \frac{\partial J}{\partial \mathbf{W}^{(m)}}$$
$$\mathbf{b}^{(m)} = \mathbf{b}^{(m)} - \mu \frac{\partial J}{\partial \mathbf{b}^{(m)}}$$

Activation function:

$$s(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$
$$s'(z) = \tanh'(z) = 1 - \tanh^2(z)$$

Initialization:

$$\mathbf{W}^{(m)} \sim U\Big[-\frac{\sqrt{6}}{\sqrt{p^{(m)} + p^{(m-1)}}}, \frac{\sqrt{6}}{\sqrt{p^{(m)} + p^{(m-1)}}}\Big]$$

Algorithm 1: DDML

Input: Training set: $\mathbf{X} = \{(\mathbf{x}_i, \mathbf{x}_j, \ell_{ij})\}$, number of network layers $M+1$, threshold τ, learning rate μ, iterative number I_t, parameter λ, and convergence error ε.

Output: Weights and biases: $\{\mathbf{W}^{(m)}, \mathbf{b}^{(m)}\}_{m=1}^{M}$.

// Initialization:

Initialize $\{\mathbf{W}^{(m)}, \mathbf{b}^{(m)}\}_{m=1}^{M}$ according to Eq. (20).

// Optimization by back prorogation:

for $t = 1, 2, \cdots, I_t$ **do**

  Randomly select a sample pair $(\mathbf{x}_i, \mathbf{x}_j, \ell_{ij})$ in $\mathbf{X}$.

  Set $\mathbf{h}_i^{(0)} = \mathbf{x}_i$ and $\mathbf{h}_j^{(0)} = \mathbf{x}_j$, respectively.

  // Forward propagation

  for $m = 1, 2, \cdots, M$ **do**

    Do forward propagation to get $\mathbf{h}_i^{(m)}$ and $\mathbf{h}_j^{(m)}$.

  end

  // Computing gradient

  for $m = M, M-1, \cdots, 1$ **do**

    Obtain gradient by back propagation according to Eqs. (8) and (9).

  end

  // Back propagation

  for $m = 1, 2, \cdots, M$ **do**

    Update $\mathbf{W}^{(m)}$ and $\mathbf{b}^{(m)}$ according to Eqs. (16) and (17).

  end

  Calculate J_t using Eq (7).

  If $t > 1$ and $|J_t - J_{t-1}| < \varepsilon$, go to **Return**.

end

Return: $\{\mathbf{W}^{(m)}, \mathbf{b}^{(m)}\}_{m=1}^{M}$.

Experimental Results

Deep vs. Shallow Metric Learning

Feature	DDML	DSML
DSIFT (original)	86.78 ± 2.09	83.68 ± 2.06
DSIFT (square root)	87.25 ± 1.62	84.42 ± 1.80
LBP (original)	85.47 ± 1.85	81.88 ± 1.90
LBP (square root)	87.02 ± 1.62	84.08 ± 1.21
SSIFT (original)	86.98 ± 1.37	84.02 ± 1.47
SSIFT (square root)	$\mathbf{87.83 \pm 0.93}$	84.52 ± 1.38
All features	$\mathbf{90.68 \pm 1.41}$	87.45 ± 1.45

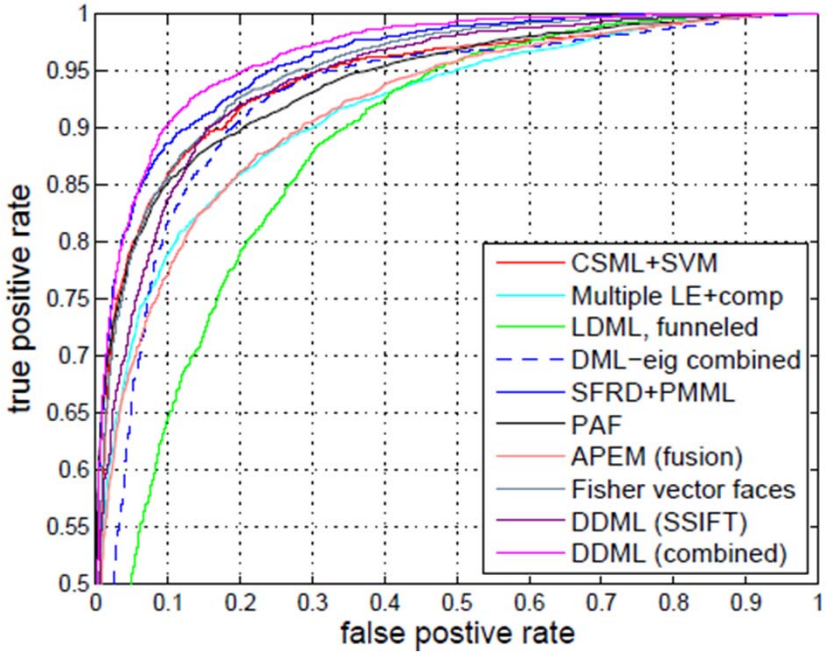
Comparison of the mean verification rate and standard error (%) with the shadow metric learning method on the LFW dataset under the image restricted setting.

Experimental Results

Comparison With State-of-the-Art Methods

Method	NoD	Accuracy
PCCA (SIFT) [25]	1	83.80 ± 0.40
CSML+SVM [26]	6	88.00 ± 0.37
PAF [39]	1	87.77 ± 0.51
STFRD+PMML [6]	8	89.35 ± 0.50
Fisher vector faces [28]	1	87.47 ± 1.49
DDML (SSIFT)	1	$\mathbf{87.83 \pm 0.93}$
DDML (combined)	6	$\mathbf{90.68 \pm 1.41}$

Verification rate (%) of different methods.



ROC curves of different methods.

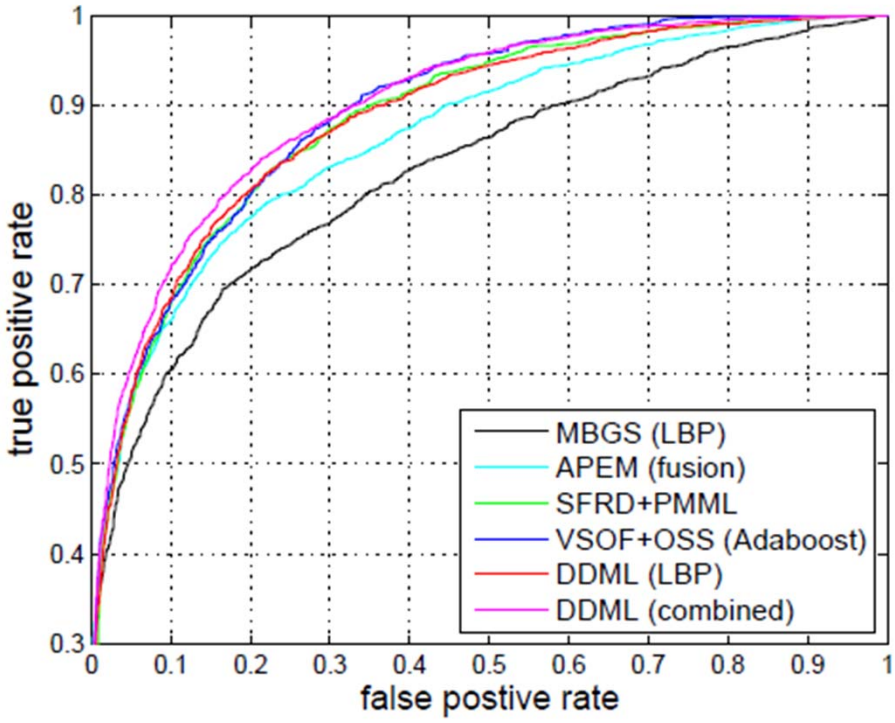
Experimental Results

Video-based Face Verification on YTF

Feature	DDML	DSML
CSLBP	75.98 ± 0.89	73.26 ± 0.99
FPLBP	76.60 ± 1.71	73.46 ± 1.66
LBP	$\mathbf{81.26 \pm 1.63}$	78.14 ± 0.94
All features	$\mathbf{82.34 \pm 1.47}$	79.36 ± 1.22

Method	Accuracy
MBGS (LBP) [34]	76.40 ± 1.80
APEM (LBP) [21]	77.44 ± 1.46
APEM (fusion) [21]	79.06 ± 1.51
STFRD+PMML [6]	79.48 ± 2.52
MBGS+SVM⊖ (LBP) [37]	79.48 ± 2.52
VSOF+OSS (Adaboost) [24]	79.70 ± 1.80
PHL+SILD (LBP) [16]	80.20 ± 1.30
DDML (LBP)	$\mathbf{81.26 \pm 1.63}$
DDML (combined)	$\mathbf{82.34 \pm 1.47}$

Verification rate (%) of different methods.



ROC curves of different methods.

Experimental Results

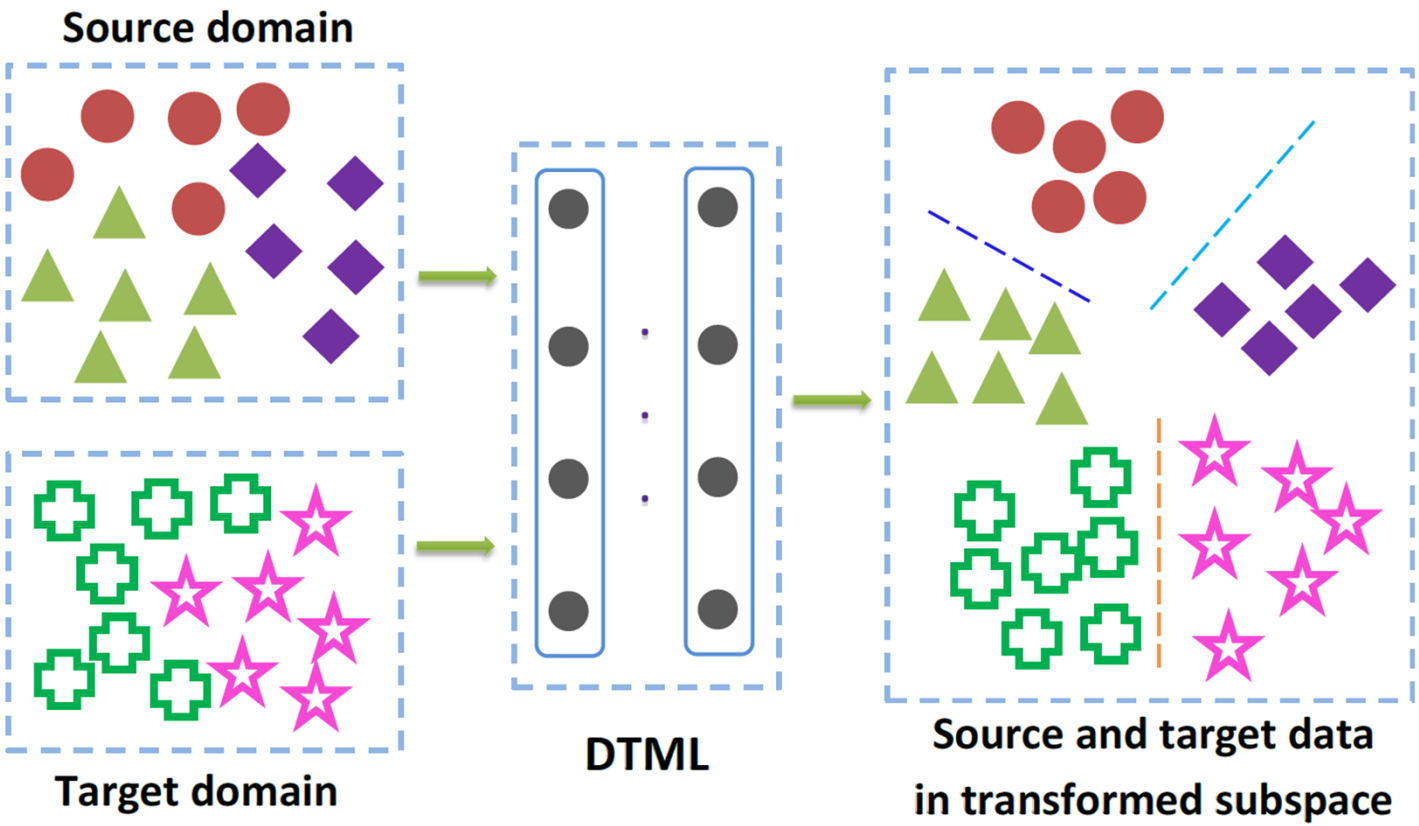
Effects of different activation functions

Dataset	sigmoid	ns-sigmoid	tanh
LFW	77.18 ± 1.82	85.80 ± 1.39	$\mathbf{87.83 \pm 0.93}$
YTF	70.20 ± 1.26	80.78 ± 1.15	$\mathbf{81.26 \pm 1.63}$

Deep Transfer Metric Learning

- DDML assume that the training and testing samples are collected from the same dataset, which are not suitable for cross-dataset visual recognition where the training and testing samples are captured from different datasets, so that the metric learned from one dataset cannot be applied to other datasets directly due to different distributions.
- It is desirable to employ transfer learning techniques to improve DDML to make it suitable for cross-dataset visual recognition.

Deep Transfer Metric Learning



Basic idea of the proposed DTML method.

Deep Transfer Metric Learning

Objective function

$$\min_{f^{(M)}} J = S_c^{(M)} - \alpha\, S_b^{(M)} + \beta\, D_{ts}^{(M)}(\mathcal{X}_t, \mathcal{X}_s)$$

$$+ \gamma \sum_{m=1}^{M} \left(\left\| \mathbf{W}^{(m)} \right\|_F^2 + \left\| \mathbf{b}^{(m)} \right\|_2^2 \right),$$

where $$D_{ts}^{(m)}(\mathcal{X}_t, \mathcal{X}_s) = \left\| \frac{1}{N_t} \sum_{i=1}^{N_t} f^{(m)}(\mathbf{x}_{ti}) - \frac{1}{N_s} \sum_{i=1}^{N_s} f^{(m)}(\mathbf{x}_{si}) \right\|_2^2$$

$$S_c^{(m)} = \frac{1}{N k_1} \sum_{i=1}^{N} \sum_{j=1}^{N} P_{ij}\, d_{f^{(m)}}^2(\mathbf{x}_i, \mathbf{x}_j)$$

$$S_b^{(m)} = \frac{1}{N k_2} \sum_{i=1}^{N} \sum_{j=1}^{N} Q_{ij}\, d_{f^{(m)}}^2(\mathbf{x}_i, \mathbf{x}_j)$$

Deep Transfer Metric Learning

Optimization

$$
\begin{aligned}
&\frac{\partial J}{\partial \mathbf{W}^{(m)}} \\
&= \frac{2}{Nk_1}\sum_{i=1}^{N}\sum_{j=1}^{N} P_{ij}\left(\mathbf{L}_{ij}^{(m)}\mathbf{h}_i^{(m-1)^T} + \mathbf{L}_{ji}^{(m)}\mathbf{h}_j^{(m-1)^T}\right) \\
&- \frac{2\alpha}{Nk_2}\sum_{i=1}^{N}\sum_{j=1}^{N} Q_{ij}\left(\mathbf{L}_{ij}^{(m)}\mathbf{h}_i^{(m-1)^T} + \mathbf{L}_{ji}^{(m)}\mathbf{h}_j^{(m-1)^T}\right) \\
&+ 2\beta\left(\frac{1}{N_t}\sum_{i=1}^{N_t}\mathbf{L}_{ti}^{(m)}\mathbf{h}_{ti}^{(m-1)^T} + \frac{1}{N_s}\sum_{i=1}^{N_s}\mathbf{L}_{si}^{(m)}\mathbf{h}_{si}^{(m-1)^T}\right) \\
&+ 2\gamma\,\mathbf{W}^{(m)},
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial J}{\partial \mathbf{b}^{(m)}} &= \frac{2}{Nk_1}\sum_{i=1}^{N}\sum_{j=1}^{N} P_{ij}\left(\mathbf{L}_{ij}^{(m)} + \mathbf{L}_{ji}^{(m)}\right) \\
&- \frac{2\alpha}{Nk_2}\sum_{i=1}^{N}\sum_{j=1}^{N} Q_{ij}\left(\mathbf{L}_{ij}^{(m)} + \mathbf{L}_{ji}^{(m)}\right) \\
&+ 2\beta\left(\frac{1}{N_t}\sum_{i=1}^{N_t}\mathbf{L}_{ti}^{(m)} + \frac{1}{N_s}\sum_{i=1}^{N_s}\mathbf{L}_{si}^{(m)}\right) \\
&+ 2\gamma\,\mathbf{b}^{(m)},
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{L}_{ij}^{(M)} &= \left(\mathbf{h}_i^{(M)} - \mathbf{h}_j^{(M)}\right) \odot \varphi'\left(\mathbf{z}_i^{(M)}\right), \\
\mathbf{L}_{ji}^{(M)} &= \left(\mathbf{h}_j^{(M)} - \mathbf{h}_i^{(M)}\right) \odot \varphi'\left(\mathbf{z}_j^{(M)}\right), \\
\mathbf{L}_{ij}^{(m)} &= \left(\mathbf{W}^{(m+1)^T}\mathbf{L}_{ij}^{(m+1)}\right) \odot \varphi'\left(\mathbf{z}_i^{(m)}\right), \\
\mathbf{L}_{ji}^{(m)} &= \left(\mathbf{W}^{(m+1)^T}\mathbf{L}_{ji}^{(m+1)}\right) \odot \varphi'\left(\mathbf{z}_j^{(m)}\right), \\
\mathbf{L}_{ti}^{(M)} &= \left(\frac{1}{N_t}\sum_{j=1}^{N_t}\mathbf{h}_{tj}^{(M)} - \frac{1}{N_s}\sum_{j=1}^{N_s}\mathbf{h}_{sj}^{(M)}\right) \odot \varphi'\left(\mathbf{z}_{ti}^{(M)}\right), \\
\mathbf{L}_{si}^{(M)} &= \left(\frac{1}{N_s}\sum_{j=1}^{N_s}\mathbf{h}_{sj}^{(M)} - \frac{1}{N_t}\sum_{j=1}^{N_t}\mathbf{h}_{tj}^{(M)}\right) \odot \varphi'\left(\mathbf{z}_{si}^{(M)}\right), \\
\mathbf{L}_{ti}^{(m)} &= \left(\mathbf{W}^{(m+1)^T}\mathbf{L}_{ti}^{(m+1)}\right) \odot \varphi'\left(\mathbf{z}_{ti}^{(m)}\right), \\
\mathbf{L}_{si}^{(m)} &= \left(\mathbf{W}^{(m+1)^T}\mathbf{L}_{si}^{(m+1)}\right) \odot \varphi'\left(\mathbf{z}_{si}^{(m)}\right),
\end{aligned}
$$

Deep Transfer Metric Learning

Iteration

$$\mathbf{W}^{(m)} = \mathbf{W}^{(m)} - \lambda \frac{\partial J}{\partial \mathbf{W}^{(m)}},$$

$$\mathbf{b}^{(m)} = \mathbf{b}^{(m)} - \lambda \frac{\partial J}{\partial \mathbf{b}^{(m)}},$$

Algorithm 1: DTML

Input: Training set: labeled source domain data $\mathcal{X}_s$ and unlabeled target domain data $\mathcal{X}_t$;
Parameters: α, β, γ, M, k_1, k_2, learning rate λ, convergence error ε, and total iterative number T.

for $k = 1, 2, \cdots, T$ **do**
- Do forward propagation to all data points;
- Compute compactness $S_c^{(M)}$ by (4);
- Compute separability $S_b^{(M)}$ by (5);
- Obtain MMD term $D_{ts}^{(M)}(\mathcal{X}_t, \mathcal{X}_s)$ by (6);
- **for** $m = M, M-1, \cdots, 1$ **do**
  - Compute $\partial J/\partial \mathbf{W}^{(m)}$ and $\partial J/\partial \mathbf{b}^{(m)}$ by back-propagation using (8) and (9);
- **end**
- *// Updating weights and biases*
- **for** $m = 1, 2, \cdots, M$ **do**
  - $\mathbf{W}^{(m)} \longleftarrow \mathbf{W}^{(m)} - \lambda\ \partial J/\partial \mathbf{W}^{(m)}$;
  - $\mathbf{b}^{(m)} \longleftarrow \mathbf{b}^{(m)} - \lambda\ \partial J/\partial \mathbf{b}^{(m)}$;
- **end**
- $\lambda \longleftarrow 0.95 \times \lambda$; *// Reducing the learning rate*
- Obtain J_k by (7);
- If $|J_k - J_{k-1}| < \varepsilon$, go to **Output**.

end

Output: Weights and biases $\{\mathbf{W}^{(m)}, \mathbf{b}^{(m)}\}_{m=1}^{M}$.

Deeply Supervised Transfer Metric Learning

Objective function

$$\min_{f^{(M)}} J = J^{(M)} + \sum_{m=1}^{M-1} \omega^{(m)}\, h\big(J^{(m)} - \tau^{(m)}\big)$$

$$J^{(m)} = S_c^{(m)} - \alpha\, S_b^{(m)} + \beta\, D_{ts}^{(m)}(\mathcal{X}_t, \mathcal{X}_s) + \gamma \left(\|\mathbf{W}^{(m)}\|_F^2 + \|\mathbf{b}^{(m)}\|_2^2 \right),$$

Motivation: DTML considers supervised information at the top layer of the network, and ignores discriminative information of the outputs at the hidden layers. To better exploit such information, DSTML considers outputs of all layers to learn the deep metric network.

Experimental Results

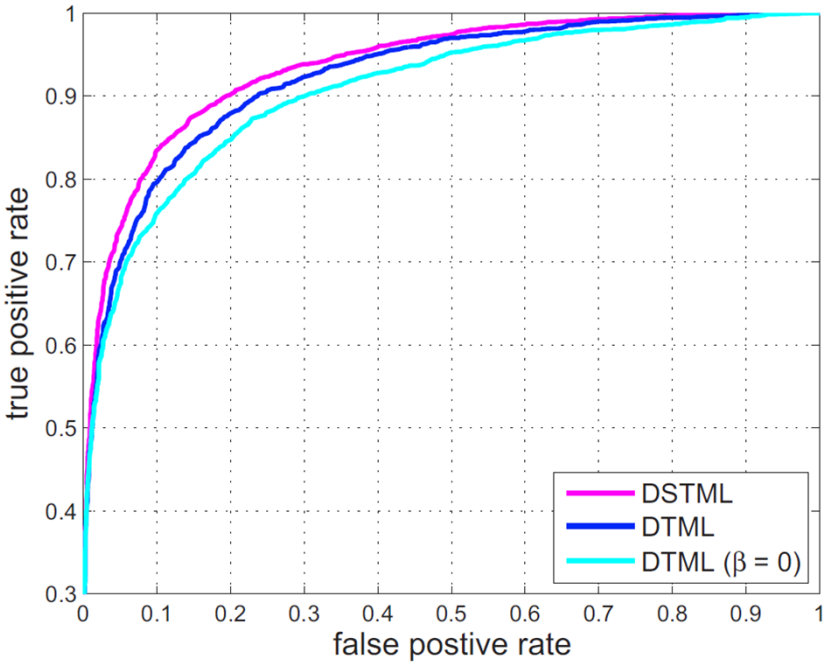
Cross-Dataset Face Verification

- ✓ Feature representation: LBP
- ✓ Target domain: Labeled Faces in the Wild (LFW)
- ✓ Source Domain: Wide and Deep Reference (WDRef)

Experimental Results

Method	Transfer	Accuracy (%)
DDML [16]	*no*	83.16 ± 0.80
STML	*yes*	83.60 ± 0.75
STML ($\beta = 0$)	*no*	82.57 ± 0.81
DTML	*yes*	85.58 ± 0.61
DTML ($\beta = 0$)	*no*	83.80 ± 0.55
DSTML	*yes*	$\mathbf{87.32 \pm 0.67}$

Verification rate (%) of different methods.



ROC curves of different methods.

Experimental Results

Cross-Dataset Person Re-identification

- ✓ Feature representation: LBP and color histogram
- ✓ Datasets: VIPER, i-LIDS, CAVIAR, 3DPeS

Experimental Results

Method	Source	$r = 1$	$r = 5$	$r = 10$	$r = 30$
L_1	-	3.99	8.73	12.59	25.32
L_2	-	4.24	8.92	12.66	25.35
DDML [16]	i-LIDS	5.63	12.91	21.71	41.80
	CAVIAR	5.91	13.53	19.86	37.92
	3DPeS	6.67	17.16	23.87	41.65
DTML ($\beta = 0$)	i-LIDS	5.88	13.72	21.03	41.49
	CAVIAR	6.02	13.81	20.33	38.46
	3DPeS	7.20	18.04	25.96	43.80
DTML	i-LIDS	6.68	15.73	23.20	46.42
	CAVIAR	6.17	13.10	19.65	37.78
	3DPeS	8.51	**19.40**	**27.59**	**47.91**
DSTML	i-LIDS	6.11	16.01	23.51	45.35
	CAVIAR	6.61	16.93	24.40	41.55
	3DPeS	**8.58**	19.02	26.49	46.77

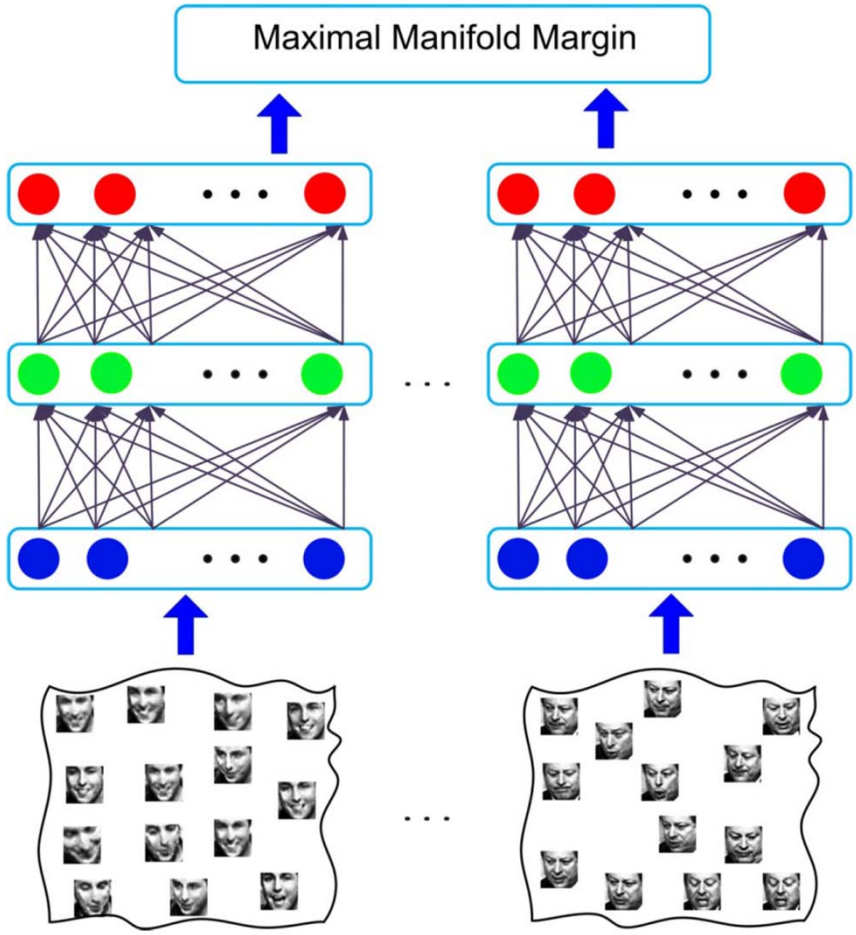
Top r matched results of different methods on the VIPeR dataset

Experimental Results

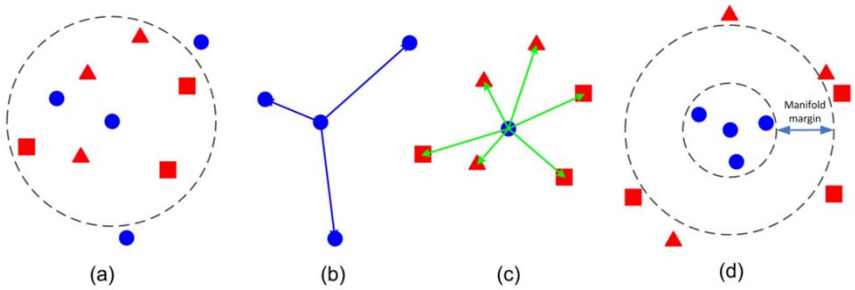
Method	Source	$r = 1$	$r = 5$	$r = 10$	$r = 30$
L_1	-	20.65	36.44	48.52	88.34
L_2	-	20.19	36.43	48.55	87.69
DDML [16]	VIPeR	23.80	42.15	55.61	90.73
	i-LIDS	22.72	41.36	56.92	90.06
	3DPeS	23.85	44.30	57.81	90.27
DTML ($\beta = 0$)	VIPeR	23.71	42.57	56.15	90.55
	i-LIDS	23.09	42.81	58.43	90.41
	3DPeS	25.11	46.71	59.69	91.99
DTML	VIPeR	23.88	42.36	55.60	92.12
	i-LIDS	26.06	47.37	61.70	**94.23**
	3DPeS	26.10	47.80	61.31	93.02
DSTML	VIPeR	26.05	44.33	57.02	92.80
	i-LIDS	25.91	44.47	58.88	93.33
	3DPeS	**28.18**	**49.96**	**63.67**	94.13

Top r matched results of different methods on the CAVIAR dataset

Multi-Manifold Deep Metric Learning



$$h_{ci}^L = s(W_c^L h_{ci}^{L-1} + b_c^L)$$



$$D_1(h_{ci}^L) = \frac{1}{K_1}\sum_{p=1}^{K_1} \|h_{ci}^L - h_{cip}^L\|_2^2$$

$$D_2(h_{ci}^L) = \frac{1}{K_2}\sum_{q=1}^{K_2} \|h_{ci}^L - h_{ciq}^L\|_2^2$$

Objective function

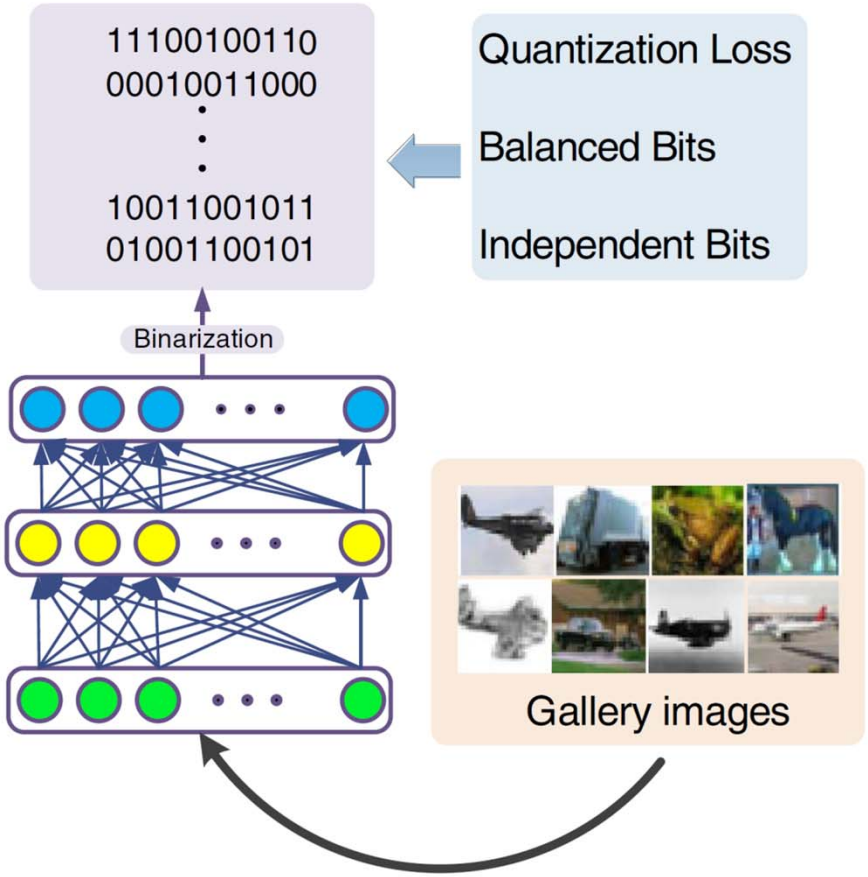
$$\begin{aligned}\min_{f_1,f_2,\cdots,f_C} H &= H_1 + \frac{\lambda}{2}H_2 \\ &= \sum_{c=1}^{C}\sum_{i=1}^{N_c} g\left(D_1(h_{ci}^L) - D_2(h_{ci}^L)\right) \\ &+ \frac{\lambda}{2}\sum_{c=1}^{C}\sum_{l=1}^{L}\left(\|W_c^l\|_F^2 + \|b_c^l\|_2^2\right)\end{aligned}$$

Experimental Results

Method	Honda	Mobo	YTC	PubFig	ETH-80	Year
MSM [38]	92.5 ± 2.3	96.5 ± 2.0	61.7 ± 4.3	57.4 ± 1.7	75.5 ± 4.9	1998
DCC [16]	92.6 ± 2.5	88.9 ± 2.5	65.8 ± 4.5	45.5 ± 1.5	91.8 ± 3.7	2006
MMD [36]	92.1 ± 2.3	92.5 ± 2.9	67.7 ± 3.8	46.3 ± 1.5	86.5 ± 4.5	2008
MDA [34]	94.5 ± 3.2	94.4 ± 2.5	68.1 ± 4.3	48.6 ± 1.6	89.2 ± 3.7	2009
AHISD [2]	91.5 ± 1.8	94.1 ± 1.5	66.5 ± 4.5	62.1 ± 1.4	78.6 ± 4.7	2010
CHISD [2]	93.7 ± 1.9	95.8 ± 1.3	67.4 ± 4.7	64.5 ± 1.5	79.7 ± 4.3	2010
SANP [13]	95.3 ± 3.1	96.1 ± 1.5	68.3 ± 5.2	78.5 ± 1.4	80.5 ± 4.7	2011
CDL [35]	97.4 ± 1.3	92.5 ± 2.9	69.7 ± 4.5	65.5 ± 1.5	86.5 ± 3.7	2012
DFRV [5]	97.4 ± 1.9	94.4 ± 2.3	74.5 ± 4.5	74.5 ± 1.4	87.5 ± 2.7	2012
LMKML [27]	98.5 ± 2.5	94.5 ± 2.5	75.2 ± 3.9	72.5 ± 1.5	92.5 ± 4.5	2013
SSDML [40]	93.5 ± 2.8	95.1 ± 2.2	74.3 ± 4.5	65.5 ± 1.7	87.5 ± 4.7	2013
SFDL [26]	98.5 ± 1.5	96.5 ± 2.3	75.7 ± 3.4	78.5 ± 1.7	90.5 ± 4.7	2014
MMDML	$\mathbf{100.0 \pm 0.0}$	$\mathbf{97.8 \pm 1.0}$	$\mathbf{78.5 \pm 2.8}$	$\mathbf{82.5 \pm 1.2}$	$\mathbf{94.5 \pm 3.5}$	

Average classification rates of different methods on different datasets

Deep Hashing



$$
\begin{aligned}
\min_{\mathbf{W},\mathbf{c}} J &= J_1 - \lambda_1 J_2 + \lambda_2 J_3 + \lambda_3 J_4 \\
&= \frac{1}{2}\|\mathbf{B} - \mathbf{H}^M\|_F^2 - \frac{\lambda_1}{2N}\mathrm{tr}((\mathbf{H}^M\mathbf{H}^M)^T) \\
&+ \frac{\lambda_2}{2}\sum_{m=1}^{M}\|\mathbf{W}^m(\mathbf{W}^m)^T - \mathbf{I}\|_F^2 \\
&+ \frac{\lambda_3}{2}(\|\mathbf{W}^m\|_F^2 + \|\mathbf{c}^m\|_2^2)
\end{aligned}
$$

$$
\begin{aligned}
\arg\min_{\mathbf{W},\mathbf{c}} J &= \frac{1}{2}\|\mathbf{B} - \mathbf{H}^M\|_F^2 \\
&- \frac{\lambda_1}{2}(\mathrm{tr}(\frac{1}{N}\mathbf{H}^M(\mathbf{H}^M)^T) + \alpha\mathrm{tr}(\Sigma_B - \Sigma_W)) \\
&+ \frac{\lambda_2}{2}\sum_{m=1}^{M}\|\mathbf{W}^m(\mathbf{W}^m)^T - \mathbf{I}\|_F^2 \\
&+ \frac{\lambda_3}{2}\sum_{m=1}^{M}(\|\mathbf{W}^m\|_F^2 + \|\mathbf{c}^m\|_2^2)
\end{aligned}
$$

Deep Hashing

Algorithm 1: DH

Input: Training set $\mathbf{X}$, network layer number M, learning rate η, iterative number R, parameters λ_1, λ_2 and λ_3, and convergence error ε.

Output: Parameters $\{\mathbf{W}^m, \mathbf{c}^m\}_{m=1}^{M}$.

Step 1 (Initialization):

Initialize $\mathbf{W}^1$ by getting the top p^1 eigenvectors from the covariance matrix.
Initialize $\{\mathbf{W}^m\}_{m=2}^{m=M} = \mathbf{I}^{p_{m-1} \times p_m}$ and $\{\mathbf{c}^m\}_{m=1}^{m=M} = \mathbf{1}^{p_m \times 1}$.

Step 2 (Optimization by back propagation):

for $r = 1, 2, \cdots, R$ **do**
- Set $\mathbf{H}^0 = \mathbf{X}$
- **for** $m = 1, 2, \cdots, M$ **do**
  - Compute $\mathbf{H}^m$ using the deep networks from (3).
- **end**
- **for** $m = M, M-1, \cdots, 1$ **do**
  - Obtain the gradients according to (6)-(7).
- **end**
- **for** $m = 1, 2, \cdots, M$ **do**
  - Update $\mathbf{W}^m$ and $\mathbf{c}^m$ according to (10)-(11).
- **end**
- Calculate J_t using (5).
- If $r > 1$ and $|J_r - J_{r-1}| < \varepsilon$, go to **Return**.

end

Return: $\{\mathbf{W}^m, \mathbf{c}^m\}_{m=1}^{M}$.

Algorithm 2: SDH

Input: Training set $\mathbf{X}$, pairwise sample indices, network layer number M, learning rate η, iterative number R, parameter λ_1, λ_2, λ_3 and α, and convergence error ϵ.

Output: Parameters $\{\mathbf{W}^m, \mathbf{c}^m\}_{m=1}^{M}$.

Step 1 (Initialization):

Initialize $\mathbf{W}^1$ by getting the top p^1 eigenvectors from a semi-supervised hashing method in [33].
Initialize $\{\mathbf{W}^m\}_{m=2}^{m=M} = \mathbf{I}^{p_{m-1} \times p_m}$ and $\{\mathbf{c}^m\}_{m=1}^{m=M} = \mathbf{1}^{p_m \times 1}$.

Step 2 (Optimization by back propagation):

for $r = 1, 2, \cdots, R$ **do**
- Set $\mathbf{H}^0 = \mathbf{X}$, $\{\mathbf{H}_{s1}^0, \mathbf{H}_{s2}^0\}$ and $\{\mathbf{H}_{d1}^0, \mathbf{H}_{d2}^0\}$ for pairwise samples in $\mathcal{S}$ and $\mathcal{D}$ respectively from set $\mathbf{X}$
- **for** $m = 1, 2, \cdots, M$ **do**
  - Compute $\mathbf{H}^m$, $\mathbf{H}_{s1}^m$, $\mathbf{H}_{s2}^m$, $\mathbf{H}_{d1}^m$, and $\mathbf{H}_{d2}^m$ using the deep networks.
- **end**
- **for** $m = M, M-1, \cdots, 1$ **do**
  - Obtain the gradients according to (15)-(16).
- **end**
- **for** $m = 1, 2, \cdots, M$ **do**
  - Update $\mathbf{W}^m$ and $\mathbf{c}^m$ according to (10)-(11).
- **end**
- Calculate J_t using (12).
- If $r > 1$ and $|J_r - J_{r-1}| < \varepsilon$, go to **Return**.

end

Return: $\{\mathbf{W}^m, \mathbf{c}^m\}_{m=1}^{M}$.

Experimental Results

Method	Hamming ranking (mAP, %)			precision (%) @ sample = 500			precision (%) @ r=2	
	16	32	64	16	32	64	16	32
PCA-ITQ [6]	15.67	16.20	16.64	22.46	25.30	27.09	22.60	14.99
KMH [8]	13.59	13.93	14.46	20.28	21.97	22.80	22.08	5.72
Spherical [9]	13.98	14.58	15.38	20.13	22.33	25.19	20.96	12.50
SH [36]	12.55	12.42	12.56	18.83	19.72	20.16	18.52	**20.60**
Semantic [26]	12.95	14.09	13.89	14.79	17.87	18.27	11.49	13.78
PCAH [34]	12.91	12.60	12.10	18.89	19.35	18.73	21.29	2.68
LSH [1]	12.55	13.76	15.07	16.21	19.10	22.25	16.73	7.07
DH	**16.17**	**16.62**	**16.96**	**23.79**	**26.00**	**27.70**	**23.33**	15.77
SPLH [34]	17.61	20.20	20.98	25.32	29.43	32.22	23.05	30.47
MLH [21]	18.37	20.49	21.89	24.43	29.60	33.01	23.52	28.72
BRE [15]	14.42	15.14	15.88	20.68	22.86	25.14	20.89	20.29
SDH	**18.80**	**20.83**	**22.51**	**26.32**	**30.42**	**33.60**	**23.26**	**31.48**

Results on CIFAR-10

Experimental Results

Method	Hamming ranking (mAP, %)			precision (%) @ sample = 500			precision (%) @ r=2	
	16	32	64	16	32	64	16	32
PCA-ITQ [6]	41.18	43.82	45.37	66.39	74.04	77.42	65.73	73.14
KMH [8]	32.12	33.29	35.78	60.43	67.19	72.65	61.88	68.85
Spherical [9]	25.81	30.77	34.75	49.48	61.27	69.85	51.71	64.26
SH [36]	26.64	25.72	24.10	56.29	61.29	61.98	57.52	65.31
PCAH [34]	27.33	24.85	21.47	56.56	59.99	57.97	36.36	65.54
LSH [1]	20.88	25.83	31.71	37.77	50.16	61.73	25.10	55.61
DH	**43.14**	**44.97**	**46.74**	**67.89**	**74.72**	**78.63**	**66.10**	**73.29**
SPLH [34]	44.20	48.29	48.34	62.98	67.89	67.99	63.71	74.06
BRE [15]	33.34	35.09	36.80	60.72	68.86	**73.08**	34.09	64.21
SDH	**46.75**	**51.01**	**52.50**	**65.19**	**70.18**	72.33	**63.92**	**77.07**

Results on MNIST

2.3 Hamming Metric Learning

[10] **Jiwen Lu**, Venice Erin Liong, Xiuzhuang Zhou, and Jie Zhou, Learning compact binary face descriptor for face recognition, *PAMI*, 2015.

Compact Binary Metric Learning

Motivation

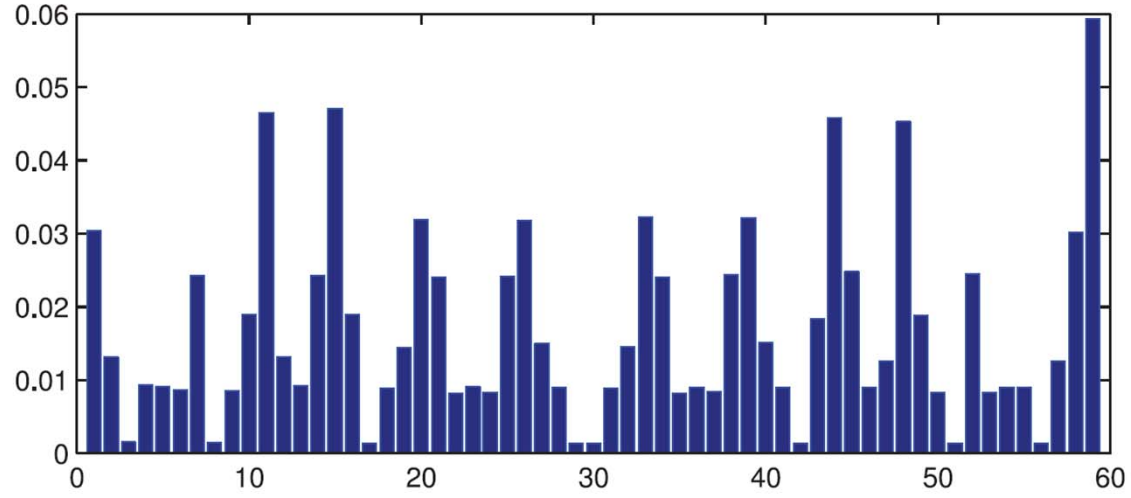
- Existing face descriptors are hand-crafted, which require strong prior knowledge to engineer them.
- Learning-based feature representation methods show promising performance because they exploit more data-adaptive information.
- Binary features are robust to local variations and have fast computational cost.

Compact Binary Metric Learning

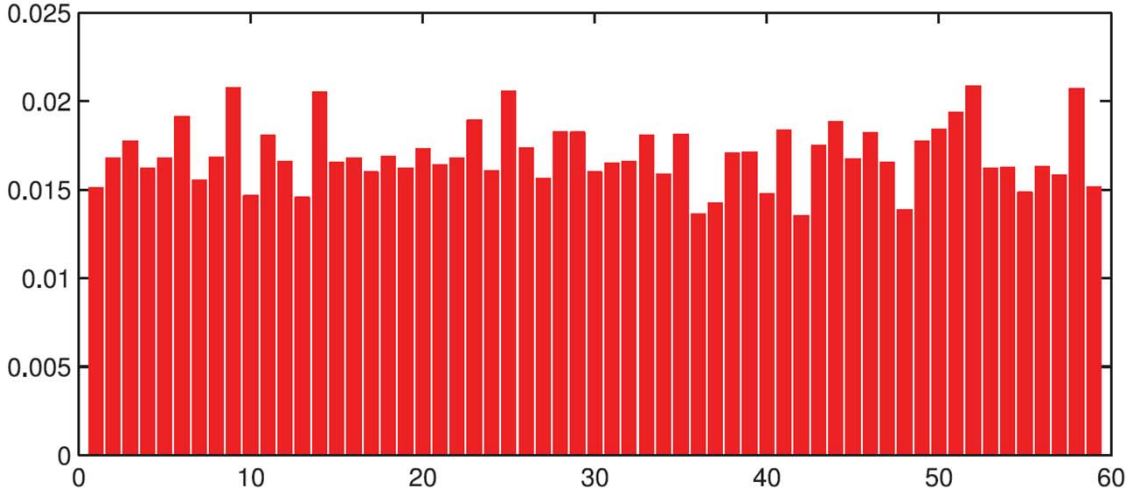
Limitation of LBP and its variations

- It is difficult to sample large size neighborhoods for LBP and its variations in feature encoding due to the high computational burden.
- It is difficult to manually design optimal feature encoding methods for LBP and its variations.
- Binary codes usually unevenly distribute in LBP and its variations, which means that some bins in the histogram are less informative and compact.

Compact Binary Metric Learning

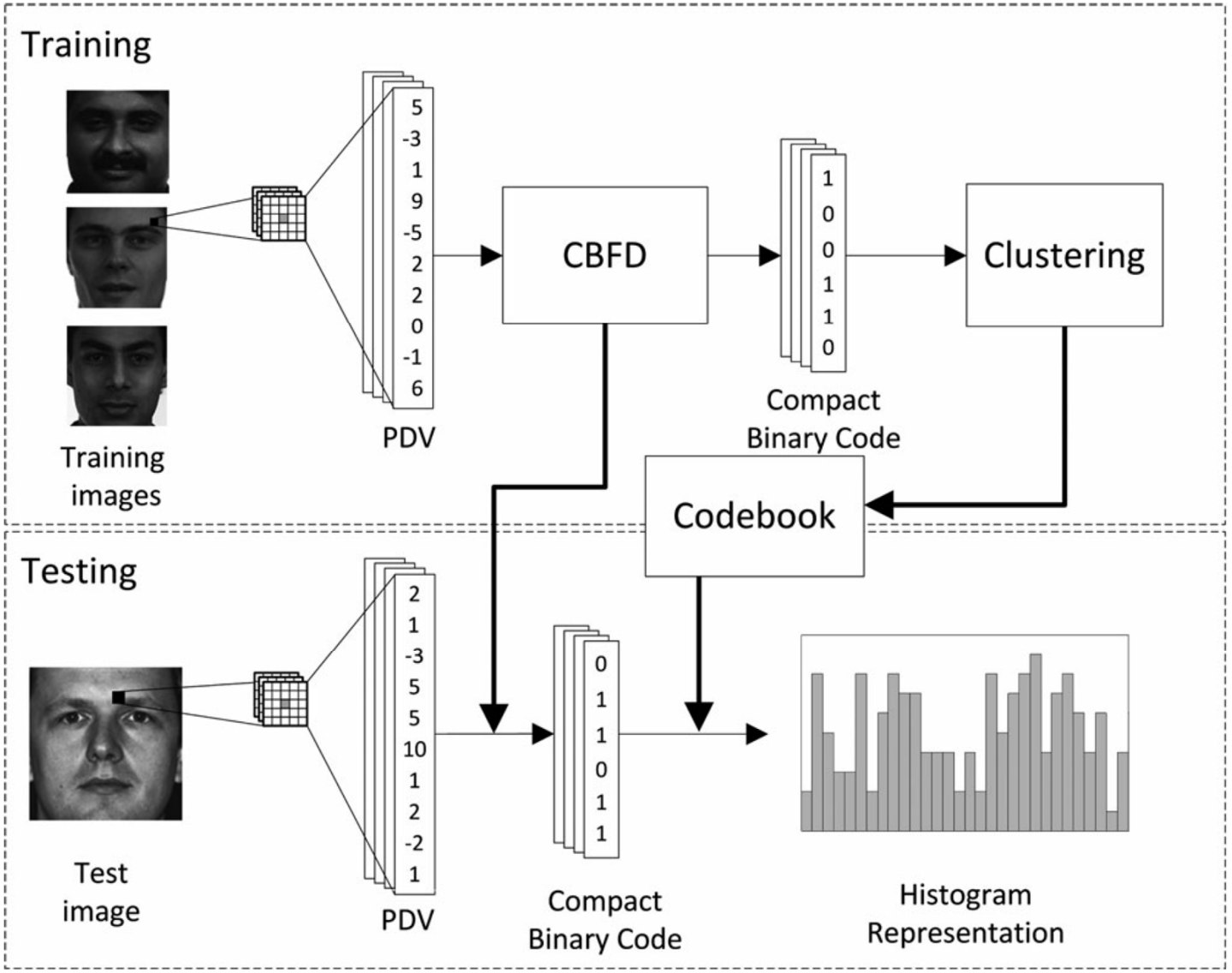


(a)



(b)

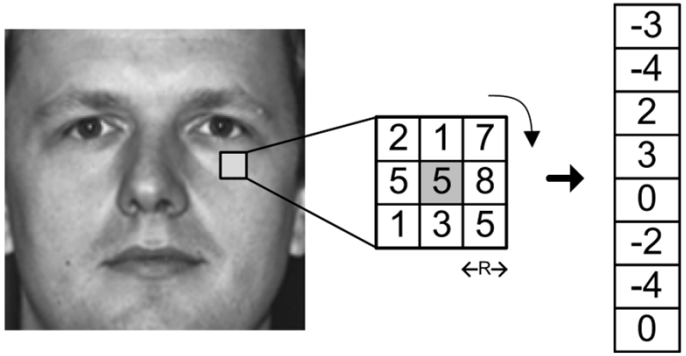
Compact Binary Metric Learning



Pipeline of our approach

Compact Binary Metric Learning

PDV (Pixel Difference Vector) Extraction



Binary Codes Learning

$$b_{nk} = 0.5 \times \left(\mathrm{sgn}(w_k^T x_n) + 1\right)$$

Compact Binary Metric Learning

Learning Criterion

- The learned binary codes are compact, so that the redundancy can be well reduced.
- The learned binary codes well preserve the energy o the original samples vectors, so that less information is missed in the binary codes learning step.
- The learned binary codes evenly distribute so that each bin in the histogram conveys more discriminative.

Compact Binary Metric Learning

Objective Function

$$\min_{w_k} J(w_k) = J_1(w_k) + \lambda_1 J_2(w_k) + \lambda_2 J_3(w_k)$$

$$= -\sum_{n=1}^{N} \|b_{nk} - \mu_k\|^2$$

$$+ \lambda_1 \sum_{n=1}^{N} \|(b_{nk} - 0.5) - w_k^T x_n\|^2$$

$$+ \lambda_2 \| \sum_{n=1}^{N} (b_{nk} - 0.5)\|^2,$$

Let $W = [w_1, w_2, \ldots w_K] \in \mathbb{R}^{d \times K}$ be the projection matrix

$$\min_{W} J(W) = J_1(W) + \lambda_1 J_2(W) + \lambda_2 J_3(W)$$

$$= -\frac{1}{N} \times \mathrm{tr}((B - U)^T (B - U))$$

$$+ \lambda_1 \|(B - 0.5) - W^T X\|_F^2$$

$$+ \lambda_2 \|(B - 0.5) \times \mathbf{1}^{N \times 1}\|_F^2,$$

where $B = 0.5 \times \left(\mathrm{sgn}(W^T X) + 1\right) \in \{0, 1\}^{N \times K}$

Compact Binary Metric Learning

Relaxation

$$J_1(W) = -\frac{1}{N} \times (\mathrm{tr}(W^T X X^T W)) - 2 \times \mathrm{tr}(W^T X M^T W) + \mathrm{tr}(W^T M M^T W),$$

$$\begin{aligned} J_3(W) &= \|(W^T X - 0.5) \times \mathbf{1}^{N\times 1}\|_2^2 \\ &= \|W^T X \times \mathbf{1}^{N\times 1}\|_2^2 \\ &\quad - N \times \mathrm{tr}(\mathbf{1}^{1\times K} \times W^T X \times \mathbf{1}^{N\times 1}) \\ &\quad + 0.5 \times \mathbf{1}^{1\times N} \times \mathbf{1}^{N\times 1} \times 0.5 \\ &= \mathrm{tr}(W^T X \mathbf{1}^{N\times 1} \mathbf{1}^{1\times N} X^T W) \\ &\quad - N \times \mathrm{tr}(\mathbf{1}^{1\times K} W^T X \mathbf{1}^{N\times 1}) \\ &\quad + H, \end{aligned}$$

Rewritten as follows:

$$\min_W J(W) = \mathrm{tr}(W^T Q W) + \lambda_1 \|(B - 0.5) - W^T X\|_2^2 - \lambda_2 \times N \times \mathrm{tr}(\mathbf{1}^{1\times K} W^T X \mathbf{1}^{N\times 1})$$

$$\text{subject to :} \quad W^T W = I,$$

where

$$Q \triangleq -\frac{1}{N} \times (X X^T - 2 X M^T + M M^T) + \lambda_2 X \mathbf{1}^{N\times 1} \mathbf{1}^{1\times N} X^T$$

Compact Binary Metric Learning

Algorithm 1. CBFD

Input: Training set $X = [x_1, x_2, \ldots, x_N]$, iteration number T, parameters λ_1 and λ_2, binary code length K, and convergence parameter ϵ.

Output: Feature projection matrix W.

Step 1 (Initialization):

Initialize W to be the top K eigenvectors of XX^T corresponding to the K largest eigenvalues.

Step 2 (Optimization):

For $t = 1, 2, \ldots, T$, repeat

2.1. Fix W and update B using (10).

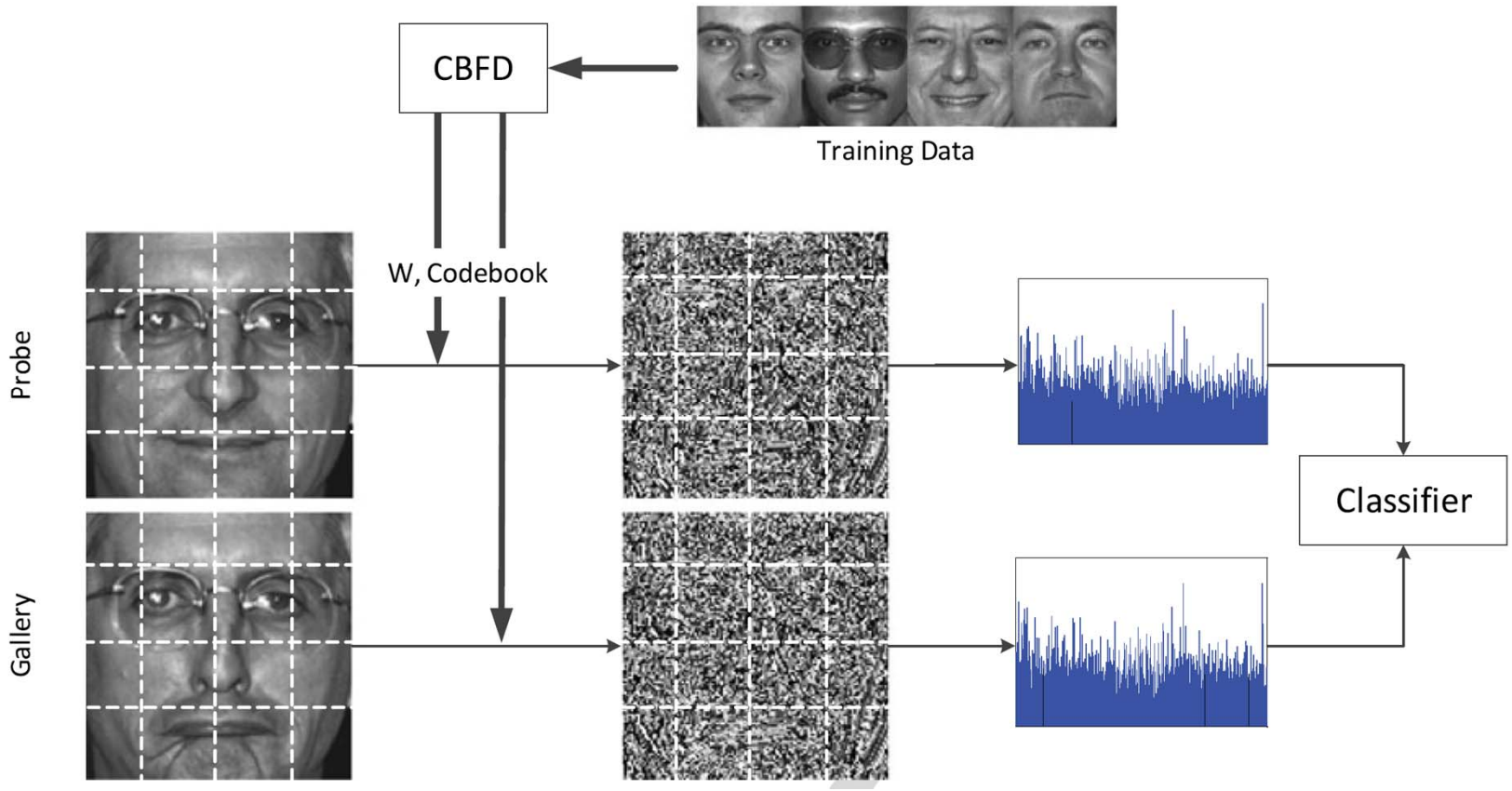
2.2. Fix B and update W using (11).

2.3. If $|W^t - W^{t-1}| < \epsilon$ and $t > 2$, go to Step 3.

Step 3 (Output):

Output the matrix W.

Compact Binary Metric Learning



The flow-chart of the CBFD-based face representation and recognition method

Experimental Results

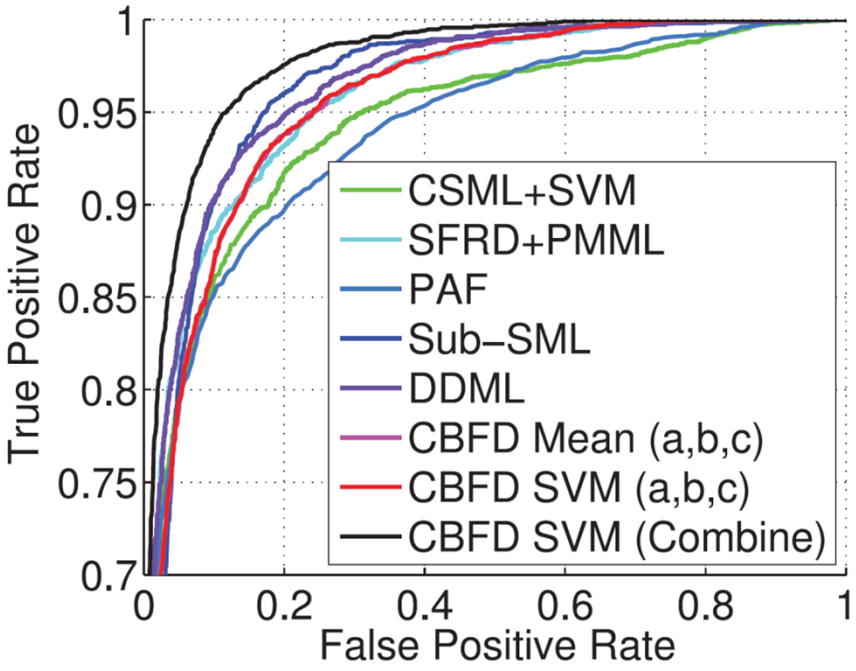
Method	fb	fc	dup1	dup2
LBP [26]	93.0	51.0	61.0	50.0
LBP+WPCA [26]	98.5	84.0	79.4	70.0
LGBP [70]	94.0	97.0	68.0	53.0
LGBP+WPCA [70]	98.1	99.0	83.8	85.0
LVP [41]	97.0	70.0	66.0	50.0
LGT [30]	97.0	90.0	71.0	67.0
HGGP [69]	97.6	98.9	77.7	76.1
HOG [42]	90.0	74.0	54.0	46.6
DT-LBP [39]	99.0	**100.0**	84.0	80.0
LDP [68]	94.0	83.0	62.0	53.0
GV-LBP-TOP [31]	98.4	99.0	82.0	81.6
DLBP [40]	99.0	99.0	86.0	85.0
GV-LBP [31]	98.1	98.5	80.9	81.2
LQP+WPCA [23]	99.8	94.3	85.5	78.6
POEM [59]	97.0	95.0	77.6	76.2
POEM+WPCA [59]	99.6	99.5	88.8	85.0
s-POEM+WPCA [58]	99.4	**100.0**	91.7	90.2
DFD [32]	99.2	98.5	85.0	82.9
DFD+WPCA [32]	99.4	**100.0**	91.8	92.3
CBFD	98.2	**100.0**	86.1	85.5
CBFD+WPCA	**99.8**	**100.0**	**93.5**	**93.2**

Rank-one recognition rate on FERET

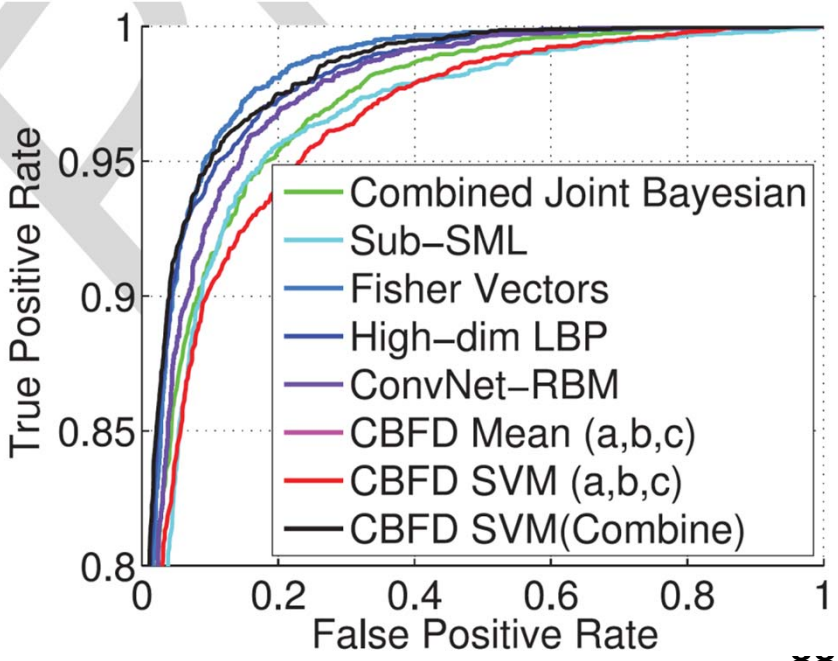
Experimental Results

Method	Accuracy
CSML+SVM, aligned+WPCA [43]	88.00 ± 0.37
PAF [67]	87.77 ± 0.51
SFRD+PMML+WPCA [11]	89.35 ± 0.50
Sub-SML [7]	89.73 ± 0.38
VMRS+WPCA [3]	91.10 ± 0.59
DDML+WPCA [19]	90.68 ± 1.41
CBFD+WPCA(a)	87.33 ± 2.42
CBFD+WPCA(b)	87.57 ± 1.43
CBFD+WPCA(c)	87.23 ± 1.68
CBFD+WPCA(mean: a, b, c)	89.05 ± 1.51
CBFD+WPCA(svm: a, b, c)	$\mathbf{89.07 \pm 1.51}$
CBFD+WPCA(combine)	$\mathbf{92.62 \pm 1.08}$

Method	Accuracy
Combined Joint Bayesian [9]	90.90 ± 1.48
Sub-SML [7]	90.75 ± 0.64
ConvNet - RBM [53]	91.75 ± 0.48
VMRS+WPCA [3]	92.05 ± 0.45
Fisher vector faces+WPCA [52]	93.03 ± 1.05
High-dim LBP [10]	93.18 ± 1.07
CBFD+WPCA(a)	87.87 ± 1.86
CBFD+WPCA(b)	88.90 ± 1.81
CBFD+WPCA(c)	88.35 ± 1.61
CBFD+WPCA(mean: a, b, c)	90.90 ± 1.40
CBFD+WPCA(svm: a, b, c)	$\mathbf{90.75 \pm 1.10}$
CBFD+WPCA(combine)	$\mathbf{93.80 \pm 1.31}$



Results on LFW under image-restricted setting



Results on LFW under image-unrestricted setting

Summary

- We introduce three types of metric learning approaches: cost-sensitive metric learning, deep metric learning and hamming metric learning for different visual recognition tasks.
- Experimental results have demonstrated the efficacy of these proposed approaches.