Towards Incremental and Large Scale Face Recognition

Junjie Yan Zhen Lei Dong Yi Stan Z.Li*
Center for Biometrics and Security Research & National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences
95 Zhongguancun Donglu, Beijing 100190, China
{jjyan,zlei,dyi,szli}@cbsr.ia.ac.cn

Abstract

Linear discriminant analysis with nearest neighborhood classifier (LDA + NN) has been commonly used in face recognition, but it often confronts with two problems in real applications: (1) it cannot incrementally deal with the information of training instances; (2) it cannot achieve fast search against large scale gallery set. In this paper, we use incremental LDA (ILDA) and hashing based search method to deal with these two problems. Firstly two incremental LDA algorithms are proposed under spectral regression framework, namely exact incremental spectral regression discriminant analysis (EI-SRDA) and approximate incremental spectral regression discriminant analysis (AI-SRDA). Secondly we propose a similarity hashing algorithm of sub-linear complexity to achieve quick recognition from large gallery set. Experiments on FRGC and self-collected 100,000 faces database show the effective of our methods.

1. Introduction

As one of the most successful biometric technology, face recognition has been a hot topic in computer vision and widely used in many commercial situations. However, in many face recognition applications such as forward criminal search, social networking consolidation, training data may come in an incremental way and the gallery set grows to be of large scale. Conventional face recognition techniques, such as batch LDA + NN, may not be suitable for these applications.

Linear Discriminant Analysis (LDA) has been widely used in supervised dimension reduction [15]. In many face applications, labeled faces comes sequentially as time elapses. Therefore it is desirable to update the optimal projection matrix with the incoming data incrementally instead of the time consuming batch-mode recalculation. To achieve this goal, some incremental LDA methods have been proposed recently. In [7], a method was proposed to incrementally update within-class and between-class scatter matrix, but it can't incrementally update the most time consuming eigenvalue decomposition problem. Based on LDA/GSVD [13], a GSVD-ILDA algorithm [14] was proposed, in which minor components are removed to reduce the cost. In [5], sufficient spanning set is used to update the between-class and within-class matrices. However, it is difficult to determine the degree of trade-off between speed and accuracy in both [14] and [5]. In [6], an incremental version of least square LDA [12] was proposed, and a closed form solution with little complexity cost is given. However, the constraint $rank(S_b) + rank(S_w) = rank(S_t)$ may not hold in real applications.

Nearest Neighborhood (NN) is commonly used to do recognition after faces are projected from original feature space to discriminant feature space according to the learned LDA projection matrix. However, in many face recognition applications, gallery set tends to reach a scale of tens of thousands or even millions of images [8]. Under this circumstance, a query by NN classifier demands a huge time cost. There has been little work paying attention to the query speed. [11] uses component-based local feature voting to get a small relevant subset firstly, and then a global hamming signature was adopted to re-rank. Since local feature voting is linear complexity, it is still time consuming when gallery is large enough. Hashing based methods are perhaps the most popular ones, especially the locality-sensitive hashing (LSH) [3, 2]. LSH have been commonly used in image retrieval, however, [11] found that their performance degrades quickly in face recognition. The possible reason is that the binarization in LSH loses too much discriminative information for face recognition.

Different from widely used LDA+NN framework, our ILDA+Hashing framework can be updated incrementally and can achieve both accuracy and efficiency. The remainder of the paper is organized as follows. In Section 2, we give our two ISRDA algorithms. In Section 3 details of the similarity hashing algorithm are given. Experiments on

*Stan Z. Li is the corresponding author.

FRGC and a self-collected database are given in Section 4. Finally, we provide some concluding remarks in Section 5.

2. Incremental Spectral Regression Discriminant Analysis

Firstly, we give a brief review of spectral regression discriminant analysis. Suppose we have N training instances $X = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N]$, where $\mathbf{x}_j$ is a $m \times 1$ vector. To simplify exposition, we suppose X has been centered. SRDA is equivalent to traditional LDA by solving the following two equations [1].

$$\begin{aligned} W\mathbf{y} &= \alpha\mathbf{y} \\ X^T\mathbf{a} &= \mathbf{y} \end{aligned} \tag{1}$$

where $\mathbf{y}$ is the low dimension embedding for training samples and $\mathbf{a}$ is the projective vector from the original data space to the discriminant subspace. W is the similarity matrix that describes the relationship among samples. [1] shows that the solution of $\mathbf{y}$ can be constructed straightforwardly by

$$\mathbf{y}_k = [\underbrace{0, \cdots, 0}_{\sum_{i=1}^{k-1} m_i}, \underbrace{1, \cdots, 1}_{m_k}, \underbrace{0, \cdots, 0}_{\sum_{i=k+1}^{K} m_i}]^T \quad k = 1, \cdots, K \tag{2}$$

The remaining work is to obtain the solution of $\mathbf{a}$. In order to avoid the overfitting problem in training set, a regularized least squares regression method is adopted so that the solution of $\mathbf{a}$ can be obtained as

$$(XX^T + \lambda I)\mathbf{a}_k = X\mathbf{y}_k \tag{3}$$

where λ is a regularized parameter. The solution of (Eq.3) can be derived by three steps. (1)Calculate the lower triangular matrix L, which is the Cholesky decomposition of $(XX^T + \lambda I)$ that satisfies $(XX^T + \lambda I) = LL^T$.(2)Calculate $C = [\mathbf{c}_1, \cdots, \mathbf{c}_K]$ where $\mathbf{c}_k = X\mathbf{y}_k$. (3)Calculate $A = [\mathbf{a}_1, \cdots, \mathbf{a}_K]$, where $\mathbf{a}_k$ satisfies that $LL^T\mathbf{a}_k = \mathbf{c}_k$.

2.1. Exact Incremental SRDA

In some real applications, training instances may come in an incremental way. It is too expensive to integrate incremental training instances in batch SRDA for the calculation of Cholesky decomposition and C since the computation complexity is $O(n^3)$; what's more, the traditional incremental LDA algorithms fail to achieve the same high accuracy as batch SRDA. Now we will propose our EI-SRDA algorithm to achieve the same solution as LDA but in an incremental manner of low computation cost. Following the batch SRDA procedure, we incrementally update Cholesky decomposition, C and A.

2.1.1 Update the Cholesky Decomposition

Given an incremental training instance $\mathbf{v}$, the training set becomes $X' = [X, \mathbf{v}]$. Updating the Cholesky decomposition equals finding a lower triangular matrix that satisfies the following equation:

$$L'L'^T = [X, \mathbf{v}][X, \mathbf{v}]^T = LL^T + \mathbf{v}\mathbf{v}^T \tag{4}$$

This problem is called rank-one update and have been researched in [4]. By adding a zero vector $\mathbf{0}$, we can get

$$L'L'^T + \mathbf{0}\mathbf{0}^T = LL^T + \mathbf{v}\mathbf{v}^T \tag{5}$$

$$[\mathbf{0}, L'] \begin{bmatrix} \mathbf{0}^T \\ L'^T \end{bmatrix} = [\mathbf{v}, L] \begin{bmatrix} \mathbf{v}^T \\ L^T \end{bmatrix} \tag{6}$$

If we can find a set of orthogonal matrices $\{G_j\}$ which satisfy

$$\begin{bmatrix} \mathbf{0}^T \\ L'^{T*} \end{bmatrix} = G_m \cdots G_2 G_1 \begin{bmatrix} \mathbf{v}^T \\ L^T \end{bmatrix} \tag{7}$$

where L'^* is a lower triangular matrix, we know that:

$$[\mathbf{0}, L'^*] \begin{bmatrix} \mathbf{0}^T \\ L'^{*T} \end{bmatrix} = [\mathbf{v}, L] \begin{bmatrix} \mathbf{v}^T \\ L^T \end{bmatrix} \tag{8}$$

Because of the uniqueness of Cholesky decomposition, L'^* is actually equivalent to L'.

There are many $\{G_j\}$ combinations which satisfy (Eq.7). In this paper we use a set of Givens matrix. Denoting $s = sin(\theta)$ and $c = cos(\theta)$, an orthogonal Givens matrix can be constructed as:

$$G_\theta(i, i) = c \quad , \quad G_\theta(i, j) = s \tag{9}$$

$$G_\theta(j, i) = -s \quad , \quad G_\theta(j, j) = c \tag{10}$$

while other elements are the same with the $m \times m$ identity matrix. Givens rotation rotates the ith and the jth elements of a vector with θ angle while keeping other elements unchanged. In this work, we use G_j to rotate $[\widehat{v}_j, \widehat{L}_{jj}]$ to $[0, \widehat{L}'_{jj}]$, where $[\widehat{v}, \widehat{L}] = G_{j-1}G_{j-2}\cdots G_1[v, L]$. The parameters of G_j are

$$r_j = \sqrt{\widehat{v}_j^2 + \widehat{L}_{jj}^2} \quad c_j = \frac{\widehat{v}_j}{r} \quad s_j = \frac{\widehat{L}_{jj}}{r} \tag{11}$$

After m rotations, we can get the new L. The dimension of G is $(m+1) \times (m+1)$. Since G_j is a highly sparse matrix, the calculation of matrix multiplication is fast.

2.1.2 Update the C

We use C to denote $[\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_K]$, where $\mathbf{c}_k = X\mathbf{y}_k$. When a new instance is added, $\mathbf{c}'_k = [X, \mathbf{v}]\mathbf{y}'_k$. There are two possible cases when updating C.

Case1: When the added instance belongs to a new class, the total class number becomes $K+1$. For $k \in \{1, 2 \cdots, K\}$, we have $\mathbf{y}_k^T = [\mathbf{y}_k, 0]^T$, and then $\mathbf{c}'_k = [X, \mathbf{v}][\mathbf{y}, 0]^T = \mathbf{c}_k$. For $k = K+1$ we have $\mathbf{y}_k = [\underbrace{0, \cdots, 0}_{N}, 1]^T$ and then $\mathbf{c}_k = [X, \mathbf{v}][\underbrace{0, \cdots, 0}_{N}, 1]^T = \mathbf{v}$. $C' = [\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_K, \mathbf{c}_{K+1}] = [C, \mathbf{v}]$ in this case.

Case2: When the added instance belongs to a class which is already in the original training set, suppose the class label is $k1$. The total class number remains the same. For $k \neq k1$,$\mathbf{y}_k^T = [\mathbf{y}_k, 0]^T$, so $\mathbf{c}'_k = [X, \mathbf{v}][\mathbf{y}_k, 0]^T = \mathbf{c}_k$. For $k = k1$, we have $\mathbf{c}'_k = \mathbf{c}_k + \mathbf{v}$. $C' = [\mathbf{c}_1, \cdots, \mathbf{c}_{k1-1}, \mathbf{c}_{k1} + \mathbf{v}, \mathbf{c}_{k1+1}, \cdots, \mathbf{c}_K]$ in this case.

To sum up:

$$\begin{cases} C' = [C, \mathbf{v}], & \text{Case1;} \\ C' = [\mathbf{c}_1, \cdots, \mathbf{c}_{k1-1}, \mathbf{c}_{k1} + v, \mathbf{c}_{k1+1}, \cdots, \mathbf{c}_K], & \text{Case2.} \end{cases} \quad (12)$$

2.1.3 Update the A

When we have new L and C, we can calculate $\mathbf{a}_k$ by solving two triangular linear equations. Since L is a lower triangular matrix, $L\mathbf{b} = \mathbf{c}_k$ can be solved by forward substitution,and s $L^T\mathbf{a}_k = \mathbf{b}$ can be solved by back substitution. We solve $\mathbf{a}_k$ respectively and then combine $\mathbf{a}_k$ to A.

Algorithm 1 shows the procedure of our **Exact Incremental Spectral Regression Discriminant Analysis(EI-SRDA)** algorithm. Note that in the algorithm every $\mathbf{x}$ should be subtracted by the mean vector. Since the mean value may change at every update procedure, we need to update the total X according to its new mean vector. Fortunately we can use a simple trick to avoid this by adding a new element "1" to each $\mathbf{x}$ as discussed in [1]. Without loss of clearness, we still use $\mathbf{x}$ in the context.

Only one instance update procedure is shown in Algorithm. 1, when multiple new instances come at the same time, step2 to step5 are to be repeated multiple times accordingly and then go to step6 to step10.

2.2. Approximate Incremental SRDA

Although EI-SRDA can get a close form solution, it may be not suitable for real applications since that all the gallery features need to be updated once the project matrix is changed. It's a huge computation when gallery set is of the large scale for example, up to tens of thousand or million scale. To solve the problem, we give an **Approximate Incremental Spectral Regression Discriminant Analysis (AI-SRDA)** algorithm that can integrate information of incremental training items while only one column of the projection matrix is changed at one update procedure.

Every class corresponding to a projection vector $\mathbf{a}_k$ in SRDA and EI-SRDA. This inspires us to update the projection vector whose corresponding instances have been changed while keeping other projection vectors unchanged.

Similarly two situations are to be considered when updating the projection vector. In Case1 as described in Sec. 2.1.2 we need to add $\mathbf{a}_{K+1}$ to A. In Case2 in Sec. 2.1.2 we update $\mathbf{a}_{k1}$. The procedure of AI-SRDA is nearly the same as EI-SRDA except that the step 7-10 in Algorithm 1 simplified to the steps in Algorithm 2

Algorithm 1 EI-SRDA

1: **Input**
the lower triangular matrix of Cholesky decomposition L;
$C = [\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_K]$;
an incremental item $\mathbf{v}$ with class label $k1$.
2: **for** $j \in [1, m]$ **do**
3: Calculate $\{G_j\}$ according to (Eq.11).
4: $[\mathbf{v}, L]^T = G_j[\mathbf{v}, L]^T$
5: **end for**
6: Update C according to (Eq.12) and update class number K.
7: **for all** $\mathbf{c}_k$ in C **do**
8: solving $L\mathbf{b} = \mathbf{c}_k$ by forward substitution to get $\mathbf{b}$.
9: $L^T\mathbf{a}_k = \mathbf{b}$ by backward substitution to get $\mathbf{a}_k$.
10: **end for**
11: **output**
Updated $A = [\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_K]$, L and C.

Algorithm 2 AI-SRDA

1: Step1:Step6 in Algorithm 1.
2: **for all** $\mathbf{c}_k$ in C **do**
3: **if** $\mathbf{c}_k$ is changed or $\mathbf{c}_k$ is newly added.
4: solving $L\mathbf{b} = \mathbf{c}_k$ by forward substitution to get $\mathbf{b}$.
5: $L^T\mathbf{a}_k = \mathbf{b}$ by back substitution to get $\mathbf{a}_k$.
6: **endif**
7: **end for**
8: **output**
Updated $A = [\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_K]$, L and C.

2.3. Complexity Analysis

The corresponding computation complexity of SRDA, EI-SRDA and AI-SRDA are listed in Table. 1.

Table 1. Complexity Comparison

Method	Time Complexity
Batch SRDA	$\frac{1}{2}Nm^2 + KmN + \frac{1}{6}m^3 + 2Km^2$
EI-SRDA	$6m^2 + 9m + 2Km^2$
AI-SRDA	$6m^2 + 9m + 2m^2$

In SRDA, the calculation of XX^T takes $\frac{1}{2}Nm^2$ operations, the calculation of $C = [\mathbf{c}_1, \cdots, \mathbf{c}_K] =$

$[X\mathbf{y}_1, \cdots, X\mathbf{y}_k]$ takes Kmn operations, the calculation of Cholesky decomposition takes $\frac{1}{6}m^3$ operations and solving the $2K$ triangular linear equations takes $2Km^2$ operations.

Compared with batch SRDA, EI-SRDA can incrementally update Cholesky decomposition and C. It makes EI-SRDA much faster than batch SRDA in incremental situation. Calculating m Givens transform matrices takes $6m$ operations, while calculating m sparse matrix multiplication takes $6m^2 + 3m$ operations, which is faster than $\frac{1}{3}m^3$ operations in decomposing the total matrix. It takes m operations to update C compared with Kmn in batch algorithm. Since updating $\mathbf{a}_k$ takes $2m^2$ step, updating A takes $2Km^2$ operations.

AI-SRDA is an approximate to EI-SRDA. Since there's only one column changed during one update procedure, we need $2m^2$ calculation to update A, which is much faster than $2Km^2$ in EI-SRDA and SRDA. The main reason why we propose AI-SRDA is that it's more suitable for update gallery features, since only one dimension need to be updated at one update procedure.

3. Similarity Hashing

After all instances are projected into a discriminant space which is learned by SRDA or I-SRDA, NN is commonly used to do the recognition. However the similarity between probe and gallery set must be computed in a pairwise way in NN based face recognition which may be too time-consuming for large scale face recognition. In this part, we use hashing technology to approximate the similarity without pairwise comparison.

Hashing function is used to map data from $\mathcal{R}^M$ to $\mathcal{R}$. In typical LSH algorithms, hash function is set to be $f(\mathbf{x}) = sign(\mathbf{w}^T\mathbf{x} + b)$, which maps high dimensional data to a binary code, where $\mathbf{w}$ is a vector and b is an offset. In this work, we use $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ instead of sign function to preserve more discriminative information, where $\mathbf{w}$ is a random vector generated according to Gaussian distribution.

The similarity in high dimensional space can be preserved in low dimensional space to some extent. For example, when $d(f(\mathbf{x}), f(\mathbf{y}))$ is small, $d(\mathbf{x}, \mathbf{y})$ is more likely to be small. We use an adaptive step function, which is a coarse approximation to to convert the distance to similarity as follows.

$$sim_f(\mathbf{x}, \mathbf{y}) = \begin{cases} 1, & |f(\mathbf{x}) - f(\mathbf{y})| < q; \\ 0, & |f(\mathbf{x}) - f(\mathbf{y})| \geq q. \end{cases} \quad (13)$$

Since single hash function is too weak to deal with complex problems, we use a set of hash functions $\mathcal{F} = \{f_1, f_2, \cdots, f_T\}$ to give a more robust inference. The similarity coding function $sim_{\mathcal{F},\mathbf{y}}(\mathbf{x})$ is defined as,

$$sim_{\mathcal{F},\mathbf{y}}(\mathbf{x}) = sim_{\mathcal{F}}(\mathbf{x}, \mathbf{y}) = \frac{1}{T}\sum_{k=1}^{T} sim_{f_k}(\mathbf{x}, \mathbf{y}) \quad (14)$$

When $T \to \infty$, $sim_{\mathcal{F}}(\mathbf{x}, \mathbf{y})$ is a robust approximation to the similarity between $\mathbf{x}$ and $\mathbf{y}$.

In real applications, the gallery set can be pre-collected and preprocessed offline. Given a gallery set $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$, it can be preprocessed by the following three steps:

1. calculating the hashing values
$$G_k = \{f_k(\mathbf{x}_1), f_k(\mathbf{x}_2), \cdots, f_k(\mathbf{x}_N)\}, k = 1, 2, \cdots T$$
2. ascending sorting G_k as S_k
3. storing the index of S_k in G_k as I_k.

For example, if $G_k = \{0.1, 0.3, 0.2, 0.6, 0.4\}$, then $S_k = \{0.1, 0.2, 0.3, 0.4, 0.6\}$ and $I_k = \{1, 3, 2, 5, 4\}$. Below we will show that how face recognition can be speeded up by these preprocessing operations.

Instead of pairwisely calculating $sim_{f,\mathbf{y}}(\mathbf{x}_i)$, we can get $sim_{f,\mathbf{y}}(\mathcal{X}) = \{sim_f(\mathbf{x}_1, \mathbf{y}), \cdots, sim_f(\mathbf{x}_N, \mathbf{y})\}$ at the same time. As S_k is in ascending order, calculating $sim_{f,\mathbf{y}}(\mathbf{x}_i)$ equals two binary search and some index operations: (1) Find the nearest index in S_k whose hashing value is nearest with $\{f(\mathbf{y}) - q\}$ and $\{f(\mathbf{y}) + q\}$ by $2\log_2(N)$ comparisons, recorded as $[l, u]$; (2) Translate $[l, u]$ to the original index in G_k by using I_k, recorded as H; (3) Set the corresponding slots in $sim_{f,\mathbf{y}}(\mathcal{X})$ indexed by H to 1, otherwise set to 0. By setting neighborhood's range b, we can further reduce the amount of calculation. The detailed process of the proposed method is described in Algorithm 3,

Algorithm 3 Similarity Hashing

Input
hash functions set: $f_t, t = 1, \cdots, T$
sorted hash values of gallery: $S_t, t = 1, \cdots, T$
the index of S_t: $I_t, t = 1, \cdots, T$
threshold to control the size of candidate set: η
a probe: $\mathbf{y}$
N dimension vector C, with all elements initialized as 0.
for $t \in [1, T]$ **do**
  Calculate $f_i(y)$.
  Use binary search to find the nearest neighborhood of $f_i(y)$ in S_k, suppose the position is g_t.
  for $i \in [g_t - b, g_t + b]$ **do**
    $sim_{\mathcal{F},y}(I_t(i)) = sim_{\mathcal{F},y}(I_t[i]) + 1$.
  end for
end for
Get candidate set: those samples corresponding with the entries of $sim_{\mathcal{F},y}$ larger than η.
Recognition: Re-Rank the candidate set using traditional methods.

where $sim_{\mathcal{F},y}$ is used to store approximation similarity between novel probe face and all gallery faces.

Thereinto η is a threshold controlling the size of candidate set, which is usually chosen by considering the trade-off between accuracy and speed, *e.g.* when $\eta = 0$, all samples in gallery set are judged as candidates. Note that re-rank can be performed using any traditional face recognition algorithms, such as Sparse Representation, Cosine distance+NN and so on.

Taking cosine distance+NN as re-rank algorithm for example, the proposed method takes $T(2b+1)$ add operations, $\rho \times NM$ multiply operations, $T\log_2(N)$ comparison operations and $T(2b+1)$ index operations. When $N \to \infty$, the ratio of computation complexity between similarity and NN is

$$\lim_{N\to\infty} \frac{(2b+1)T + \rho \times NM + T\log_2(N) + (2b+1)T}{MN} = \rho \tag{15}$$

where M is the original data dimension and ρ is candidate set's proportion of gallery. By setting a proper η, we can make ρ small enough without much accuracy loss, *e.g.* $\rho = 0.001$. Since Cosine distance+NN takes NM operations, our algorithm is much faster.

4. Experiments on Face Recognition

Our experiments are conducted on FRGC v2.0 [9]. To simulate the large scale face recognition, we also collected 100,000 face images with various pose, expression and illumination conditions from the web. All the images are rotated, scaled and cropped to 142×120 according to the eye positions. The boosted Gabor features [10] are extracted to represent faces.

4.1. EI-SRDA, AI-SRDA v.s. SRDA

Firstly we compare our EI-SRDA, AI-SRDA with SRDA in terms of accuracy and efficiency. 100 classes from the 222 classes in FRGC training set are selected to train the initial recognition model. The left 122 classes are simulated as the incremental coming samples. The recognition performance following Exp. 1 and 4 protocols in FRGC. Accuracy was measured according to the verification rate (VR) when false accept rate (FAR) is 0.001.

From Fig. 1 we can find that batch SRDA and EI-SRDA can achieve nearly the same accuracy in both Exp. 1 and Exp. 4. As an approximation, EI-SRDA have little accuracy decrease.

Training time of batch SRDA increases as the number of total trained instances increases, while update time of our incremental algorithms remains almost unchanged, which is much lower than SRDA. 36 to 64 instances was added at one time in this experiment, and the fluctuation in the added number causes corresponding fluctuation in update time. The lower the number of new coming instances at one update procedure, the more acceleration our algorithms are expected to achieve.

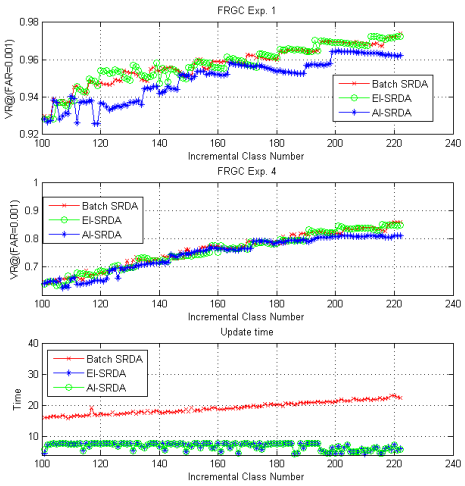

Figure 1. Batch SRDA vs. Incremental SRDA in accuracy and time.

4.2. Similarity Hashing v.s. NN

By projecting all features to the discriminant space learned by incremental SRDA, we further compare similarity hashing with NN. The comparison is conducted according to the FRGC protocol. $CosineDistance$ is used to do Re-Ranking. We give results when $SR = 1, 0.5, 0.2, 0.1, 0.05, 0.005$ on Exp. 1 and $SR = 1, 0.5, 0.2, 0.1, 0.05$ on Exp. 4. It's NN when $SR = 1$. In Fig. 2 we can find that our algorithm can achieve VR=0.9703(@FAR=0.001) which is nearly the same with NN by searching 0.5% of the total gallery set on Exp. 1. As Exp. 4 is more challenging, our algorithm searches 5% of total gallery set to get VR=0.8526(@FAR=0.001) which is close to VR=0.8603(@FAR=0.001) of NN. From this experiment we can find that our algorithm can achieve comparable results with NN while gallery samples being searched are much less.

FRGC Exp1		FRGC Exp4	
Search Rate	VR	Search Rate	VR
1(NN)	0.9739	1(NN)	0.8603
0.5	0.9739	0.5	0.8586
0.2	0.9739	0.2	0.8584
0.1	0.9739	0.1	0.8572
0.05	0.9739	**0.05**	**0.8526**
0.005	**0.9703**	0.01	0.8235

Figure 2. VR with different SR On FRGC(@FAR=0.001)

To evaluate the performance of our method on large scale database, a 100,000 people database is added to the target set of FRGC as the total gallery set. The range of the neighborhood may influence the performance: on one hand, if the range of neighborhood is too narrow, a lot of positive samples will be ignored; on the other hand, if the range is too broad, step similarity function may lose discriminative ability. In Fig. 3, the number of hashing function is empirically set to be 1000 and $Accuracy$(@SR=0.001) is used to compare different parameters. In Exp. 1 we can find that when neighborhood range is set to be 2000, the $Accuracy$ corresponding to the $SR = 0.001$ is more than 99%. Exp. 4 is more interesting: when the neighborhood range is 1000, the $Accuracy$ is the highest which is 0.98 $@(SR = 0.001)$; when the neighborhood range enlarges to 2000, performance drops, for more negative samples with similar illumination and blur kick out positive ones. As most similarity samples are considered, when range reaches 5000, Accuracy returns to 97% but with more calculation.

FRGC Exp1		FRGC Exp4	
Range	Accuracy	Range	Accuracy
300	0.9356	300	0.6405
400	0.9583	400	0.6406
500	0.9688	500	0.7541
600	0.9753	600	0.7935
1000	0.9890	**1000**	**0.9852**
2000	0.9951	2000	0.8989
5000	**0.9997**	5000	0.9771

Figure 3. HR on FRGC&100,000 people database(@SR=0.001)

When the neighborhood range is set 1000, number of hashing function is 1000 and SR is 0.001, a C++ implementation on a PC with single Intel Core2 CPU and 16GB memory shows that recognizing a probe from 116,028 gallery faces only takes 32ms while NN takes 970ms.

5. Conclusion

In this paper, incremental SRDA and similarity hashing are proposed to improve the LDA+NN framework. There are several advantages of our methods: (1)Our EI-SRDA is an exact solution as SRDA and only need very little calculation to update projection matrix. (2) In AI-SRDA, only one column of the projection matrix changed at one update procedure , so that it costs little to update gallery features. (3) Compared with NN, our similarity hashing can achieve much speed-up and keep the accuracy. The proposed EI-SRDA, AI-SRDA and Hashing algorithm provide a novel insight to incremental and large scale face recognition problem, and it is reasonably believed they can be applied to a wide range of applications.

Acknowledgements

The authors would like to acknowledge the following funding sources: the Chinese National Natural Science Foundation Project #61070146, the National Science and Technology Support Program Project #2009BAK43B26, the AuthenMetric R&D Funds (2004-2011), and the TABULA RASA project (http://www.tabularasa-euproject.org) under the Seventh Framework Programme for research and technological development (FP7) of the European Union (EU), grant agreement #257289.

References

[1] D. Cai, X. He, and J. Han. Srda: An efficient algorithm for large-scale discriminant analysis. *TKDE*, 20(1):1–12, 2008. 2, 3

[2] M. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, pages 380–388, 2002. 1

[3] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Symposium on Computational Geometry*, pages 253–262, 2004. 1

[4] P. Gill, G. Golub, W. Murray, and M. Saunders. Methods for modifying matrix factorizations. *Mathematics of Computation*, 28(126):505–535, 1974. 2

[5] T. Kim, B. Stenger, J. Kittler, and R. Cipolla. Incremental linear discriminant analysis using sufficient spanning sets and its applications. *IJCV*, pages 1–17, 2011. 1

[6] L. Liu, Y. Jiang, and Z. Zhou. Least square incremental linear discriminant analysis. In *ICDM'09*, pages 298–306. IEEE, 2009. 1

[7] S. Pang, S. Ozawa, and N. Kasabov. Incremental linear discriminant analysis for classification of data streams. *TSMCB*, 35(5):905–914, 2005. 1

[8] G. W. Q. Patrick J. Grother and P. J. Phillips. Report on the evaluation of 2d still-image face recognition algorithms. Technical report, NISF, 2010. 1

[9] P. Phillips, P. Flynn, T. Scruggs, K. Bowyer, J. Chang, K. Hoffman, J. Marques, J. Min, and W. Worek. Overview of the face recognition grand challenge. 2005. 5

[10] S. Shan, P. Yang, X. Chen, and W. Gao. Adaboost gabor fisher classifier for face recognition. In *AMFG*, pages 279–292, 2005. 5

[11] Z. Wu, Q. Ke, J. Sun, and H.-Y. Shum. Scalable face image retrieval with identity-based quantization and multi-reference re-ranking. In *CVPR*, pages 3469–3476, 2010. 1

[12] J. Ye. Least squares linear discriminant analysis. In *ICDM2007*, pages 1087–1093. ACM, 2007. 1

[13] J. Ye, R. Janardan, C. Park, and H. Park. An optimization criterion for generalized discriminant analysis on undersampled problems. *TPAMI*, 26(8):982–994, 2004. 1

[14] H. Zhao and P. Yuen. Incremental linear discriminant analysis for face recognition. *TSMCB*, 38(1):210–221, 2008. 1

[15] W. Zhao, R. Chellappa, P. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *Acm Computing Surveys (CSUR)*, 35(4):399–458, 2003. 1