IDR/QR: An Incremental Dimension Reduction Algorithm via QR Decomposition

Jieping Ye* Qi Li† Hui Xiong* Haesun Park* Ravi Janardan* Vipin Kumar*

ABSTRACT

Dimension reduction is critical for many database and data mining applications, such as efficient storage and retrieval of high-dimensional data. In the literature, a well-known dimension reduction scheme is Linear Discriminant Analysis (LDA). The common aspect of previously proposed LDA based algorithms is the use of Singular Value Decomposition (SVD). Due to the difficulty of designing an incremental solution for the eigenvalue problem on the product of scatter matrices in LDA, there is little work on designing incremental LDA algorithms. In this paper, we propose an LDA based incremental dimension reduction algorithm, called IDR/QR, which applies QR Decomposition rather than SVD. Unlike other LDA based algorithms, this algorithm does not require the whole data matrix in main memory. This is desirable for large data sets. More importantly, with the insertion of new data items, the IDR/QR algorithm can constrain the computational cost by applying efficient QR-updating techniques. Finally, we evaluate the effectiveness of the IDR/QR algorithm in terms of classification accuracy on the reduced dimensional space. Our experiments on several real-world data sets reveal that the accuracy achieved by the IDR/QR algorithm is very close to the best possible accuracy achieved by other LDA based algorithms. However, the IDR/QR algorithm has much less computational cost, especially when new data items are dynamically inserted.

Categories and Subject Descriptors: H.2.8 [Database Management]: Database Applications - Data Mining

General Terms: Algorithms

Keywords: Dimension reduction, Linear Discriminant Analysis, incremental learning, QR Decomposition

*Department of Computer Science & Engineering, University of Minnesota, Minneapolis, MN 55455, U.S.A. {jieping,huix,hpark,janardan,kumar}@cs.umn.edu
†Department of Computer Science, University of Delaware, Newark, DE, U.S.A. qili@cis.udel.edu

1. INTRODUCTION

Efficient storage and retrieval of high-dimensional data is one of the central issues in database and data mining research. In the literature, many efforts have been made to design multi-dimensional index structures [2], such as R-trees, $R^\star$-trees, X-trees, SR-tree, etc, for speeding up query processing. However, the effectiveness of queries using any indexing schemes deteriorates rapidly as the dimension increases, which is the so-called *curse of dimensionality.* A standard approach to overcome this problem is dimension reduction, which transforms the original high-dimensional data into a lower-dimensional space with limited loss of information. Once the high-dimensional data is transfered into a low-dimensional space, indexing techniques can be applied effectively to organize this low-dimensional space and facilitate efficient retrieval of data [14].

A well-known dimension reduction scheme is Linear Discriminant Analysis (LDA) [7, 9], which computes a linear transformation by maximizing the ratio of the between-class distance to the within-class distance, thereby achieving maximal discrimination. LDA has been applied to many applications such as text retrieval [3] and face recognition [1, 17, 21]. In the past, many LDA extensions have been developed to deal with the singularity problem encountered by classical LDA. There are three major extensions: regularized LDA, PCA+LDA, and LDA/GSVD. The common point of these algorithms is the use of Singular Value Decomposition (SVD) or Generalized Singular Value Decomposition (GSVD). The difference among these LDA members is as follows: Regularized LDA increases the magnitude of the diagonal elements of the scatter matrix by adding a scaled identity matrix; PCA+LDA first applies PCA on the raw data to get a more compact representation so that the singularity of the scatter matrix is decreased; LDA/GSVD solves a trace optimization problem using GSVD.

The above LDA algorithms have certain limitations. First, SVD or GSVD requires that the whole data matrix is stored in main memory. This requirement imposes difficulties on making the LDA algorithms scalable to large data sets. Also, the expensive computation of SVD or GSVD can significantly degrade the computational performance of the LDA algorithms when dealing with large data sets. Finally, since it is difficult to design an incremental solution for the eigenvalue problem on the product of scatter matrices, little effort has been made to design incremental LDA algorithms. However, in many practical applications, acquisition of representative training data is expensive and time-consuming. It is thus common to have a small chunk of data available over a period of time. In such settings, it is necessary to develop an algorithm that can run in an incremental fashion to accommodate the new data.

The goal of this paper is to design an efficient and incremental dimension reduction algorithm while preserving competitive performance. More precisely, when queries are conducted on the reduced dimension data from the proposed algorithm, the query accuracy should be comparable with the best possible query accuracy achieved by other LDA based algorithms.

To resolve these issues, we design an LDA based incremental dimension reduction algorithm, called IDR/QR, which applies QR Decomposition rather than SVD or GSVD. This algorithm has two stages. The first stage is to maximize the separability between different classes. This is fulfilled by QR Decomposition. The distinct property of this stage is its low time and space complexity. The second stage incorporates both between-class and within-class information by applying LDA on the "reduced" scatter matrices resulting from the first stage. Unlike other LDA based algorithms, IDR/QR does not require the whole data matrix in main memory. This is desirable for large data sets. Also, our theoretical analysis indicates that the computational complexity of IDR/QR is linear in the number of the data items in the training set as well as the number of dimensions. More importantly, the IDR/QR algorithm can work incrementally. When new data items are dynamically inserted, the computational cost of the IDR/QR algorithm can be constrained by applying efficient QR-updating techniques.

Finally, we have conducted extensive experiments on several well-known real-world datasets. The experimental results show that the IDR/QR algorithm can be an order of magnitude faster than SVD or GSVD based LDA algorithms and the accuracy achieved by IDR/QR is very close to the best possible accuracy achieved by other LDA based algorithms. Also, when dealing with dynamic updating, the computational advantage of IDR/QR over SVD or GSVD based LDA algorithms becomes more substantial while still achieving comparable accuracy.

Overview: The rest of this paper is organized as follows. Section 2 introduces related work. In Section 3, we give a review of LDA. The batch implementation of the IDR/QR algorithm is presented in Section 4. Section 5 describes the incremental implementation of the IDR/QR algorithm. A comprehensive empirical study of the performance of the proposed algorithms is presented in Section 6. We conclude in Section 7 with a discussion of future work.

2. RELATED WORK

Principal Component Analysis (PCA) is one of the standard and well-known methods for dimension reduction [13]. Because of its simplicity and ability to extract the global structure of a given data set, PCA is used widely in computer vision [22].

Most previous work on PCA and LDA requires that all the training data be available before the dimension reduction step. This is known as the *batch method*. There is some recent work in vision and numerical linear algebra literature for computing PCA incrementally [4, 11]. Despite the popularity of LDA in the vision community, there is little work for computing it incrementally. The main difficulty is the involvement of the eigenvalue problem on the product of scatter matrices, which is hard to maintain incrementally. Although iterative algorithms have been proposed for neural network based LDA [5, 16], they require $O(d^2)$ time for each update, where d is the dimension of the data. This is still expensive, especially when the data has high dimension.

Notations	Descriptions
n	number of training data points
n_i	number of data points in i-th class
d	dimension of the training data
c	number of classes
G	transformation matrix
A	data matrix
A_i	data matrix of the i-th class
S_b	between-class scatter matrix
S_w	within-class scatter matrix
C	centroid matrix
m	global centroid of the training set
m_i	centroid of the i-th class

Table 1: Notations

3. LINEAR DISCRIMINANT ANALYSIS

For convenience, we present in Table 1 the important notations used in the paper.

This section gives a brief review of classical LDA, as well as its three extensions: regularized LDA, PCA+LDA, and LDA/GSVD.

Given a data matrix $A \in \mathbb{R}^{d\times n}$, we consider finding a linear transformation $G \in \mathbb{R}^{d\times \ell}$ that maps each column a_j of A, for $1 \leq j \leq n$, in the d-dimensional space to a vector $y_j = G^T a_j$ in the ℓ-dimensional space.

Classical LDA aims to find the transformation G such that class structure of original high-dimensional space is preserved in the reduced space. Let the data matrix A be partitioned into c classes as $A = [A_1, A_2, \cdots, A_c]$, where $A_i \in \mathbb{R}^{d\times n_i}$, and $\sum_{i=1}^{c} n_i = n$.

Let I_i be the set of column indices that belong to the ith class, i.e., a_j, for $j \in I_i$, belongs to the ith class.

In general, if each class is tightly grouped, but well separated from the other classes, the quality of the cluster is considered to be high. In discriminant analysis, two scatter matrices, within-class and between-class scatter matrices are defined to quantify the quality of the cluster, as follows [9]:

$$\begin{aligned} S_w &= \sum_{i=1}^{c}\sum_{j\in I_i}(a_j - m_i)(a_j - m_i)^T, \\ S_b &= \sum_{i=1}^{c}\sum_{j\in I_i}(m_i - m)(m_i - m)^T \\ &= \sum_{i=1}^{c} n_i(m_i - m)(m_i - m)^T, \end{aligned}$$

where m_i is the centroid of the ith class and m is the global centroid. Define the matrices

$$\begin{aligned} H_w &= [A_1 - m_1 \cdot e_1^T, \cdots, A_c - m_c \cdot e_c^T] \in \mathbb{R}^{d\times n}, \quad (1) \\ H_b &= [\sqrt{n_1}(m_1 - m), \cdots, \sqrt{n_c}(m_c - m)] \in \mathbb{R}^{d\times c}, (2) \end{aligned}$$

where $e_i = (1, \cdots, 1)^T \in \mathbb{R}^{n_i}$.

Then the scatter matrices S_w and S_b can be expressed as $S_w = H_w H_w^T$, $S_b = H_b H_b^T$. The traces of the two scatter matrices can be computed as follows,

$$\begin{aligned} \text{trace } (S_w) &= \sum_{i=1}^{c} \sum_{j \in I_i} ||a_j - m_i||^2 \\ \text{trace } (S_b) &= \sum_{i=1}^{c} n_i ||m_i - m||^2. \end{aligned}$$

Hence, trace (S_w) measures the closeness of the vectors within classes, while trace (S_b) measures the separation between classes.

In the lower-dimensional space resulting from the linear transformation G, the within-class and between-class scatter matrices become

$$\begin{aligned} S_w^L &= (G^T H_w)(G^T H_w)^T = G^T S_w G, \\ S_b^L &= (G^T H_b)(G^T H_b)^T = G^T S_b G. \end{aligned}$$

An optimal transformation G would maximize trace (S_b^L) and minimize trace (S_w^L). A common optimization in classical LDA [9] is

$$G = \arg \max_{g_i^T S_w g_j = 0, \forall i \neq j} \text{trace } \left((G^T S_w G)^{-1} (G^T S_b G) \right), \quad (3)$$

where g_i is the ith column of G.

The solution to the optimization in Eq. (3) can be obtained by solving the eigenvalue problem on $S_w^{-1} S_b$, if S_w is non-singular, or on $S_b^{-1} S_w$, if S_b is non-singular. There are at most $c - 1$ eigenvectors corresponding to nonzero eigenvalues, since the rank of the matrix S_b is bounded above by $c - 1$. Therefore, the reduced dimension by classical LDA is at most $c - 1$. A stable way to solve this eigenvalue problem is to apply SVD on the scatter matrices. Details on this can be found in [21].

Classical LDA requires that one of the scatter matrices be non-singular. For many applications involving undersampled data, such as text and image retrieval, all scatter matrices are singular. Classical LDA is thus not applicable. This is the so-called *singularity or undersampled problem*. To cope with this challenge, several methods, including two-stage PCA+LDA, regularized LDA, and LDA/GSVD have been proposed in the past.

A common way to deal with the singularity problem is to apply an intermediate dimension reduction stage, such as PCA, to reduce the dimension of the original data before classical LDA is applied. The algorithm is known as PCA+LDA, or subspace LDA. In this two-stage PCA+LDA algorithm, the discriminant stage is preceded by a dimension reduction stage using PCA. A limitation of this approach is that the optimal value of the reduced dimension for PCA is difficult to determine.

Another common way to deal with the singularity problem is to add some constant value to the diagonal elements of S_w, as $S_w + \mu I_d$, for some $\mu > 0$, where I_d is an identity matrix [8]. It is easy to check that $S_w + \mu I_d$ is positive definite, hence non-singular. This approach is called regularized LDA (RLDA). A limitation of RLDA is that the optimal value of the parameter μ is difficult to determine. Cross-validation is commonly applied for estimating the optimal μ [15].

The LDA/GSVD algorithm in [12, 23] is a recent work along the same line. A new criterion for generalized LDA was presented in [23]. The inversion of the matrix S_w is avoided by applying the Generalized Singular Value Decomposition (GSVD). LDA/GSVD computes the solution exactly without losing any information. However, one limitation of this method is the high computational cost of GSVD, which limits its applicability for large datasets, such as image and text data.

4. BATCH IDR/QR

In this section, we present the batch implementation of the IDR/QR algorithm. This algorithm has two stages. The first stage maximizes the separation between different classes via QR Decomposition [10]. Without the concern of minimizing within-class distance, this stage can be used independently as a dimension reduction algorithm. The second stage addresses the concern on within-class distance, while keeping low time/space complexity.

The first stage of IDR/QR aims to solve the following optimization problem,

$$G = \arg \max_{G^T G = I} \text{trace}(G^T S_b G). \quad (4)$$

Note that this optimization only addresses the concern of maximizing between-class distance. The solution can be obtained by solving the eigenvalue problem on S_b. The solution can also be obtained through QR Decomposition on the centroid matrix C, which is the so-called Orthogonal Centroid Method (OCM) [19], where

$$C = [m_1, m_2, \cdots, m_c] \quad (5)$$

consists of the c centroids. More specifically, let $C = QR$ be the QR Decomposition of C, where $Q \in \mathbb{R}^{n \times c}$ has orthonormal columns and $R \in \mathbb{R}^{c \times c}$ is upper triangular. Then

$$G = QM \quad (6)$$

solves the optimization problem in Eq. (4), for any orthogonal matrix M. Note the choice of orthogonal matrix M is arbitrary, since $\text{trace}(G^T S_b G) = \text{trace}(M^T G^T S_b G M)$, for any orthogonal matrix M. In OCM [19], M is set to be the identity matrix for simplicity.

The second stage of IDR/QR refines the first stage by addressing the concern on within-class distance. It incorporates the within-class scatter information by applying a relaxation scheme on M in Eq. (6) (relaxing M from an orthogonal matrix to an arbitrary matrix). Note that the trace value in Eq. (3) is the same for an arbitrary non-singular M, however the constraints in Eq. (3) will not be satisfied for arbitrary M. In the second stage of IDR/QR, we look for a transformation G such that $G = QM$, for some M. Note that M is not required to be orthogonal. The original problem on computing G is equivalent to computing M. Since

$$\begin{aligned} G^T S_b G = M^T (Q^T S_b Q) M, \\ G^T S_w G = M^T (Q^T S_w Q) M, \end{aligned}$$

the original optimization on finding optimal G is equivalent to finding M, with $B = Q^T S_b Q$ and $W = Q^T S_w Q$ as the reduced between-class and within-class scatter matrices, respectively. Note that B has much smaller size than the original scatter matrix S_b (similarly for W).

The optimal M can be computed efficiently using many existing LDA based methods, since we are dealing with matrices B and W of much smaller size c by c. A key observation is that the singularity problem of W will not be as

Algorithm 1: Batch IDR/QR

```
Input:    data matrix $A$;
Output: optimal transformation matrix $G$;
/* Stage I: */
1. Construct centroid matrix $C$;
2. Compute QR Decomposition of $C$ as $C = QR$;
   where $Q \in \mathbb{R}^{d\times c}$, $R \in \mathbb{R}^{c\times c}$;
/* Stage II: */
3. $Z \leftarrow H_w^T Q$;
4. $Y \leftarrow H_b^T Q$;
5. $W \leftarrow Z^T Z$; /*Reduced within-class scatter matrix*/
6. $B \leftarrow Y^T Y$; /*Reduced between-class scatter matrix*/
7. Compute the $c$ eigenvectors $\phi_i$ of $(W + \mu I_c)^{-1}B$,
   with decreasing eigenvalues;
8. $G \leftarrow QM$, where $M = [\phi_1, \cdots, \phi_c]$.
```

Methods	Time Complexity	Space Complexity
IDR/QR	$O(ndc)$	$O(dc)$
PCA+LDA	$O(n^2 d)$	$O(nd)$
LDA/GSVD	$O((n+c)^2 d)$	$O(nd)$
OCM	$O(nd + c^2 d)$	$O(dc)$
PCA	$O(n^2 d)$	$O(nd)$

Table 2: Complexity comparison: n is the number of training data points, d is the dimension, and c is the number of classes.

severe as the original S_w, since W has much smaller size than S_w. We can compute optimal M by simply applying regularized LDA, that is, we compute M, by solving a small eigenvalue problem on $(W + \mu I_c)^{-1}B$, for some positive constant μ. The pseudo-code for this algorithm is given in **Algorithm 1**.

4.1 Time and space complexity

We close this section by analyzing the time and space complexity of the IDR/QR algorithm.

It takes $O(dn)$ for the formation of the centroid matrix C in Line 1. The complexity for QR Decomposition in Line 2 is $O(c^2 d)$ [10]. Lines 3 and 4 take $O(ndc)$ and $O(dc^2)$ respectively for matrix multiplications. It then takes $O(c^2 n)$ and $O(c^3)$ for matrix multiplications in Lines 5 and 6, respectively. Line 7 computes the eigen-decomposition of a c by c matrix, hence takes $O(c^3)$ [10]. The matrix multiplication in Line 8 takes $O(dc^2)$.

Note that the dimension, d, and the number, n, of points are usually much larger, compared with the number, c, of classes. Thus, the most expensive step in **Algorithm 1** is in Line 3, which takes $O(ndc)$. Therefore, the time complexity of IDR/QR is linear in the number of points, linear in the number of classes, and linear in the dimension of the dataset.

It is clear that only the c centroids are required to reside in the main memory, hence the space complexity of IDR/QR is $O(dc)$. Table 2 lists the time and space complexity of several dimension reduction algorithms discussed in this paper. It is clear from the table that IDR/QR and OCM are more efficient than other methods.

5. INCREMENTAL IDR/QR

The incremental implementation of the IDR/QR algorithm is discussed in details in this section. We will adopt the following convention: For any variable X, its updated version after the insertion of a new instance is denoted by $\tilde{X}$. For example, the number, n_i, of elements in the ith class is changed to $\tilde{n}_i$, while centroid m_i is changed to $\tilde{m}_i$.

With the insertion of a new instance, the centroid matrix C, H_w and H_b will change accordingly, as well as W and B. The incremental updating in IDR/QR proceeds in three steps: (1) QR-updating of the centroid matrix $C = [m_1, \cdots, m_k]$ in Line 2 of **Algorithm 1**; (2) Updating of the reduced within-class scatter matrix W in Line 5; and (3) Updating of the reduced between-class scatter matrix B in Line 6.

Let x be a new instance inserted, which belongs to the ith class. Without loss of generality, let us assume we have data from the 1st to the kth class, just before x is inserted. In general, this can be done by switching the class labels between different classes. In the rest of this section, we consider the incremental updating in IDR/QR in two distinct cases: (1) x belongs to an existing class, i.e., $i \le k$; (2) x belongs to a new class, i.e., $i > k$. As will be seen later, the techniques for these two cases are quite different.

5.1 Insertion of a new instance from an existing class ($i \le k$)

Recall that we have data from the 1st to kth classes, when a new instance x is being inserted. Since x belongs to the ith class, with $1 \le i \le k$, the insertion of x will not create a new class. In this section, we show how to do the incremental updating in three steps.

5.1.1 Step 1: QR-updating of the centroid matrix C

Since the new instance x belongs to the ith class, $\tilde{C} = [m_1, \cdots, m_i + f, \cdots, m_k]$, where $f = \frac{x - m_i}{\tilde{n}_i}$, and $\tilde{n}_i = n_i + 1$. Hence, $\tilde{C}$ can be rewritten as $\tilde{C} = C + f \cdot g^T$, for $g = (0, \cdots, 1, \cdots, 0)^T$, where the 1 appears at the ith position.

The problem of QR-updating of the centroid matrix C can be formulated as follows: Given the QR Decomposition of the centroid matrix $C = QR$, for $Q \in \mathbb{R}^{d\times k}$ and $R \in \mathbb{R}^{k\times k}$, compute the QR Decomposition of $\tilde{C}$.

Since $\tilde{C} = C + f \cdot g^T$, the QR-updating of the centroid matrix C can be formulated as a rank-one QR-updating. However, the algorithm in [10] cannot be directly applied, since it requires the complete QR Decomposition, i.e., the matrix Q is square. While in our case, we use the skinny QR Decomposition, i.e. Q is rectangular. Instead, we apply a slight variation of the algorithm in [6] via the following two-stage QR-updating: (1) A complete rank-one updating as in [10] on a small matrix; (2) A QR-updating by an insertion of a new row. Details are given below.

Partition f into two parts: the projection onto the orthogonal basis Q, and its orthogonal complement. Mathematically, f can be partitioned into $f = QQ^T f + (I - QQ^T)f$. It is easy to check that $Q^T(I - QQ^T)f = 0$, i.e. $(I - QQ^T)f$ is orthogonal to, or lies in the orthogonal complement of, the subspace spanned by the columns of Q. It follows that

$$\begin{aligned}\tilde{C} &= C + f \cdot g^T \\ &= QR + QQ^T f \cdot g^T + (I - QQ^T)f \cdot g^T \\ &= Q(R + f_1 \cdot g^T) + f_2 \cdot g^T,\end{aligned}$$

where $f_1 = Q^T f$, $f_2 = (I - QQ^T)f$. Next, we show how to compute the QR Decomposition of $\tilde{C}$ in two stages. The first stage updates the QR Decomposition of $Q(R+f_1 \cdot g^T)$. It corresponds to a rank-one updating and can be done at $O(kd)$ [10]. This results in the updated QR Decomposition as $Q(R+f_1 \cdot g^T) = Q_1 R_1$, where $Q_1 = QP_1$, and $P_1 \in \mathbb{R}^{k\times k}$ is orthogonal.

Assume $||f_2|| \neq 0$. Denote $q = \frac{f_2}{||f_2||}$. Since q is orthogonal to the subspace spanned by the columns of Q, it is also orthogonal to the subspace spanned by the columns of $Q_1 = QP_1$, i.e. $[Q_1, q]$ has orthonormal columns.

The second stage computes QR-updating of

$$\tilde{C} = [Q_1, q] \begin{pmatrix} R_1 \\ ||f_2|| g^T \end{pmatrix},$$

which corresponds to the case that $||f_2|| g^T$ is inserted as a new row. This stage can be done at $O(dk)$ [10]. The updated QR Decomposition is

$$[Q_1, q] \begin{pmatrix} R_1 \\ ||f_2|| g^T \end{pmatrix} = [\tilde{Q}, \tilde{q}] \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix} = \tilde{Q}\tilde{R},$$

where $[\tilde{Q}, \tilde{q}] = [Q_1, q]P_2$, for some orthogonal matrix P_2.

Combining both stages, we have

$$\tilde{C} = Q_1 R_1 + ||f_2|| q \cdot g^T = [Q_1, q] \begin{pmatrix} R_1 \\ ||f_2|| g^T \end{pmatrix} = \tilde{Q}\tilde{R}$$

as the updated QR Decomposition of $\tilde{C}$, assuming $||f_2|| \neq 0$. If $||f_2|| = 0$, then $\tilde{C} = Q_1 R_1$ is the updated QR Decomposition of $\tilde{C}$. Note that f_2 can be computed efficiently as $f_2 = f - (Q(Q^T f))$, by doing matrix-vector multiplication twice. Hence, the total time complexity for the QR-updating of the centroid matrix C is $O(dk)$.

5.1.2 Step 2: Updating of W

Next we consider the updating of the reduced within-class scatter matrix $W = Q^T H_w H_w^T Q$ (Line 5 of **Algorithm 1**). Let $\tilde{W} = \tilde{Q}^T \tilde{H}_w \tilde{H}_w^T \tilde{Q}$ be its updated version.

Note that $H_w = [A_1 - m_1 \cdot e_1^T, \cdots, A_k - m_k \cdot e_k^T] \in \mathbb{R}^{d\times n}$. Its updated version $\tilde{H}_w$ differs from H_w on the ith block. Let the ith block of H_w be $H_i = A_i - m_i \cdot e_i^T$. Then the ith block of its updated version $\tilde{H}_w$ is

$$\begin{aligned} \tilde{H}_i &= \tilde{A}_i - \tilde{m}_i \cdot \tilde{e}_i^T = [A_i, x] - \tilde{m}_i \cdot \tilde{e}_i^T \\ &= [A_i - m_i \cdot e_i^T, x - m_i] - (\tilde{m}_i - m_i) \cdot \tilde{e}_i^T \\ &= [H_i, u] - v \cdot \tilde{e}_i^T, \end{aligned} \quad (7)$$

where $u = x - m_i$, $v = \tilde{m}_i - m_i$ and $\tilde{e}_i = \begin{pmatrix} e_i \\ 1 \end{pmatrix} \in \mathbb{R}^{n_i+1}$.

The product $\tilde{H}_i \tilde{H}_i^T$ can be computed as

$$\begin{aligned} \tilde{H}_i \tilde{H}_i^T &= ([H_i, u] - v \cdot \tilde{e}_i^T)([H_i, u] - v \cdot \tilde{e}_i^T)^T \\ &= [H_i, u] \begin{pmatrix} H_i^T \\ u^T \end{pmatrix} - v \cdot \tilde{e}_i^T \begin{pmatrix} H_i^T \\ u^T \end{pmatrix} \\ &- [H_i, u]\tilde{e}_i \cdot v^T + (v \cdot \tilde{e}_i^T)(\tilde{e}_i \cdot v^T) \\ &= H_i H_i^T + u \cdot u^T - v \cdot u^T - u \cdot v^T + (n_i + 1) v \cdot v^T \\ &= H_i H_i^T + (u - v) \cdot (u - v)^T + n_i v \cdot v^T, \end{aligned} \quad (8)$$

where the third equality follows, since $(H_i, u)\tilde{e}_i = \sum_{j\in I_i}(a_j - m_i) + u = u$, and $(v \cdot \tilde{e}_i^T)(\tilde{e}_i \cdot v^T) = vv^T(\tilde{e}_i^T \cdot \tilde{e}_i) = (n_i+1)vv^T$.

Since $H_w H_w^T = \sum_{j=1}^k H_j H_j^T$, we have

$$\begin{aligned} \tilde{H}_w \tilde{H}_w^T &= \sum_{j=1}^k \tilde{H}_j \tilde{H}_j^T \\ &= \sum_{1\le j\le k, j\neq i} \tilde{H}_j \tilde{H}_j^T + \tilde{H}_i \tilde{H}_i^T \\ &= \sum_{j=1}^k H_j H_j^T + (u - v) \cdot (u - v)^T + n_i v \cdot v^T. \end{aligned}$$

It follows that

$$\begin{aligned} \tilde{W} &= \tilde{Q}^T \tilde{H}_w \tilde{H}_w^T \tilde{Q} \\ &= \tilde{Q}^T H_w H_w^T \tilde{Q} + \tilde{Q}^T (u - v) \cdot (u - v)^T \tilde{Q} + n_i \tilde{Q}^T v \cdot v^T \tilde{Q} \\ &= \tilde{Q}^T H_w H_w^T \tilde{Q} + (\tilde{u} - \tilde{v}) \cdot (\tilde{u} - \tilde{v})^T + n_i \tilde{v} \cdot \tilde{v}^T \\ &\approx Q H_w H_w^T Q + (\tilde{u} - \tilde{v}) \cdot (\tilde{u} - \tilde{v})^T + n_i \tilde{v} \cdot \tilde{v}^T \\ &= W + (\tilde{u} - \tilde{v}) \cdot (\tilde{u} - \tilde{v})^T + n_i \tilde{v} \cdot \tilde{v}^T, \end{aligned} \quad (9)$$

where $\tilde{u} = \tilde{Q}^T u$, and $\tilde{v} = \tilde{Q}^T v$. The assumption of the approximation in Eq. (9) is that the updated $\tilde{Q}$ with the insertion of a new instance is close to Q.

The computation of $\tilde{u}$ and $\tilde{v}$ takes $O(dk)$. Thus, the computation for updating W takes $O(dk)$.

5.1.3 Step 3: Updating of B

Finally, let us consider the updating of the reduced between-class scatter matrix $B = Q^T H_b H_b^T Q$ (Line 6 of **Algorithm 1**). Its updated version is $B = \tilde{Q}^T \tilde{H}_b \tilde{H}_b^T \tilde{Q}$.

The key observation for efficient updating of B is that

$$\tilde{H}_b = [\sqrt{\tilde{n}_1}(\tilde{m}_1 - \tilde{m}), \cdots, \sqrt{\tilde{n}_k}(\tilde{m}_k - \tilde{m})]$$

can be rewritten as

$$\tilde{H}_b = [\tilde{m}_1, \tilde{m}_2, \cdots, \tilde{m}_k, \tilde{m}]F = [\tilde{C}, \tilde{m}]F,$$

where $F = \begin{pmatrix} D \\ -h^T \end{pmatrix}$, $D = \text{diag}(\sqrt{\tilde{n}_1}, \cdots, \sqrt{\tilde{n}_k})$, and $h = [\sqrt{\tilde{n}_1}, \cdots, \sqrt{\tilde{n}_k}]^T$.

By the updated QR Decomposition $\tilde{C} = \tilde{Q}\tilde{R}$, we have

$$\tilde{Q}^T \tilde{H}_b = [\tilde{Q}^T \tilde{C}, \tilde{Q}^T \tilde{m}]F = [\tilde{R}, \tilde{Q}^T \tilde{m}]F = \tilde{R}D - \tilde{Q}^T \tilde{m} \cdot h^T.$$

It is easy to check that $\tilde{m} = \frac{1}{\tilde{n}}\tilde{C} \cdot r$, where $r = (\tilde{n}_1, \cdots, \tilde{n}_k)^T$. Hence, $\tilde{Q}^T \tilde{m} = \tilde{Q}^T \frac{1}{\tilde{n}} \tilde{C} \cdot r = \frac{1}{\tilde{n}} \tilde{R} \cdot r$. It follows that

$$\begin{aligned} \tilde{B} &= \tilde{Q}^T \tilde{H}_b \tilde{H}_b^T \tilde{Q} = (\tilde{R}D - \tilde{Q}^T \tilde{m} \cdot h^T) \cdot (\tilde{R}D - \tilde{Q}^T \tilde{m} \cdot h^T)^T \\ &= \left(\tilde{R}D - \left(\frac{1}{\tilde{n}}\tilde{R} \cdot r\right) \cdot h^T\right) \left(\tilde{R}D - \left(\frac{1}{\tilde{n}}\tilde{R} \cdot r\right) \cdot h^T\right)^T. \end{aligned}$$

Therefore, it takes $O(k^3)$ time for updating B.

Overall, the total time for QR-updating of C and updating of W and B with the insertion of a new instance from an existing class is $O(dk + k^3)$. The pseudo-code is given in **Algorithm 2**.

5.2 Insertion of a new instance from a new class ($i > k$)

Recall that we have data from the 1st to kth classes, upon the insertion of x. Since x belongs to ith class, with $i > k$, the insertion of x will result in a new class. Without loss of generality, let us assume $i = k + 1$. Hence the $(k + 1)$th centroid $\tilde{m}_{k+1} = x$. Then the updated centroid matrix $\tilde{C} = [m_1, m_2, \cdots, m_k, x] = [C, x]$. In the following, we focus on

```
Algorithm 2: Updating Existing Class
Input:   centroid matrix $C = [m_1, m_2, \cdots, m_k]$, its
         QR Decomposition $C = QR$, the matrix $W$,
         the size $n_j$ of the $j$-th class for each $j$, and a
         new point $x$ from the $i$-th class, $i \leq k$
Output:  updated matrix $\tilde{W}$, updated centroid matrix
         $\tilde{C}$, its QR Decomposition $\tilde{C} = \tilde{Q}\tilde{R}$, and
         updated matrix $\tilde{B}$;
1. $\tilde{n}_j \leftarrow n_j$, for $j \neq i$; $\tilde{n}_i \leftarrow n_i + 1$; $f \leftarrow \frac{x - m_i}{\tilde{n}_i}$;
2. $\tilde{m}_i \leftarrow m_i + f$; $\tilde{m}_j \leftarrow m_j$, for each $j \neq i$;
3. $\tilde{C} \leftarrow [\tilde{m}_1, \cdots, \tilde{m}_i, \cdots, \tilde{m}_k]$;
4. $f_1 \leftarrow Q^T f$; $f_2 \leftarrow (I - QQ^T)f$;
5. do rank-one QR-updating of $Q(R + f_1 \cdot g^T)$
   as $Q(R + f_1 \cdot g^T) = Q_1 R_1$;
6. if $||f_2|| = 0$
7.     $\tilde{Q} \leftarrow Q_1$; $\tilde{R} \leftarrow R_1$;
8. else
9.     $q \leftarrow \frac{(I - QQ^T)f}{||(I - QQ^T)f||}$; $g \leftarrow (0, \cdots, 1, \cdots, 0)^T$;
10.    do QR-updating of $[Q_1, q]\begin{pmatrix} R_1 \\ ||f_2||g^T \end{pmatrix}$ as
       $[Q_1, q]\begin{pmatrix} R_1 \\ ||f_2||g^T \end{pmatrix} = Q_2 R_2$;
11.    $\tilde{Q} \leftarrow Q_2$; $\tilde{R} \leftarrow R_2$;
12. endif
13. $u \leftarrow x - m_i$; $v \leftarrow \tilde{m}_i - m_i$;
14. $\tilde{u} \leftarrow \tilde{Q}^T u$; $\tilde{v} \leftarrow \tilde{Q}^T v$;
15. $\tilde{W} \leftarrow W + (\tilde{u} - \tilde{v})(\tilde{u} - \tilde{v})^T + n_i \tilde{v}\tilde{v}^T$;
16. $D \leftarrow \text{diag}(\sqrt{\tilde{n}_1}, \cdots, \sqrt{\tilde{n}_k})$; $h = [\sqrt{\tilde{n}_1}, \cdots, \sqrt{\tilde{n}_k}]^T$;
17. $r \leftarrow (\tilde{n}_1, \cdots, \tilde{n}_k)^T$; $\tilde{r} \leftarrow \frac{1}{\tilde{n}}\tilde{R} \cdot r$;
18. $\tilde{B} \leftarrow (\tilde{R}D - \tilde{r} \cdot h^T)(\tilde{R}D - \tilde{r} \cdot h^T)^T$;
```

the case when x does not lie in the space spanned by the k centroids $\{m_i\}_{i=1}^k$.

5.2.1 Step 1: QR-updating of the centroid matrix C

Given the QR Decomposition $C = QR$, it is straightforward to compute the QR Decomposition of $\tilde{C}$ as $\tilde{C} = \tilde{Q}\tilde{R}$ by the Gram-Schmidt procedure [10], where $\tilde{Q} = [Q, q]$, for some q. The time complexity for this step is $O(dk)$.

5.2.2 Step 2: Updating of W

With the insertion of x from a new class $(k+1)$, the $(k+1)$th block $\tilde{H}_{k+1}$ is created, while H_j, for $j = 1, \cdots, k$ keep unchanged. It is easy to check that $\tilde{H}_{k+1} = 0$. It follows that $\tilde{H}_w \tilde{H}_w^T = H_w H_w^T$. Hence

$$\begin{aligned} \tilde{W} &= \tilde{Q}^T \tilde{H}_w \tilde{H}_w^T \tilde{Q} = \tilde{Q}^T H_w H_w^T \tilde{Q} = [Q, q]^T H_w H_w^T [Q, q] \\ &\approx \begin{pmatrix} Q^T H_w H_w^T Q & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} W & 0 \\ 0 & 0 \end{pmatrix}. \end{aligned}$$

The assumption in the above approximation is that W is the dominant part in $\tilde{W}$.

5.2.3 Step 3: Updating of B

The updating of B follows the same idea as in the previous case. Note that

$$\tilde{H}_b = [\sqrt{\tilde{n}_1}(\tilde{m}_1 - \tilde{m}), \cdots, \sqrt{\tilde{n}_{k+1}}(\tilde{m}_{k+1} - \tilde{m})]$$

can be rewritten as

$$\tilde{H}_b = [\tilde{m}_1, \tilde{m}_2, \cdots, \tilde{m}_{k+1}, \tilde{m}]F,$$

```
Algorithm 3: Updating New Class
Input:   centroid matrix $C = [m_1, m_2, \cdots, m_k]$,
         its QR Decomposition $C = QR$, the size
         $n_j$ of the $j$-th class for each $j$, and a
         new point $x$ from the $(k+1)$-th class
Output:  updated matrix $\tilde{W}$, updated centroid matrix
         $\tilde{C}$, its QR Decomposition $\tilde{C} = \tilde{Q}\tilde{R}$, and
         updated matrix $\tilde{B}$;
1. $\tilde{n}_j \leftarrow n_j$, for $j = 1, \cdots, k$; $\tilde{n}_{k+1} \leftarrow 1$; $\tilde{n} \leftarrow n + 1$;
2. do QR-updating of $\tilde{C} = [C, x]$ as $\tilde{C} = \tilde{Q}\tilde{R}$;
3. $\tilde{W} \leftarrow \begin{pmatrix} W & 0 \\ 0 & 0 \end{pmatrix}$;
4. $D \leftarrow \text{diag}\left(\sqrt{\tilde{n}_1}, \cdots, \sqrt{\tilde{n}_{k+1}}\right)$; $h \leftarrow \left(\sqrt{\tilde{n}_1}, \cdots, \sqrt{\tilde{n}_{k+1}}\right)^T$;
5. $r \leftarrow (\tilde{n}_1, \cdots, \tilde{n}_{k+1})^T$; $\tilde{r} \leftarrow \frac{1}{\tilde{n}}\tilde{R}r$;
6. $\tilde{B} \leftarrow (\tilde{R}D - \tilde{r} \cdot h^T)(\tilde{R}D - \tilde{r} \cdot h^T)^T$;
```

where the matrix $F = \begin{pmatrix} D \\ -h^T \end{pmatrix}$, and D is an diagonal matrix $D = \text{diag}(\sqrt{n_1}, \cdots, \sqrt{n_{k+1}})$, and $h = [\sqrt{n_1}, \cdots, \sqrt{n_{k+1}}]^T$.

By the updated QR Decomposition $\tilde{C} = \tilde{Q}\tilde{R}$, we have

$$\begin{aligned} \tilde{Q}^T \tilde{H}_b &= \tilde{Q}^T [\tilde{C}, \tilde{m}]F = [\tilde{Q}^T \tilde{C}, \tilde{Q}^T \tilde{m}]F \\ &= [\tilde{R}, \tilde{Q}^T \tilde{m}]F = \tilde{R}D - \tilde{Q}^T \tilde{m} \cdot h^T. \end{aligned}$$

Since $\tilde{m} = \frac{1}{\tilde{n}}\tilde{C} \cdot r$, where $r = (\tilde{n}_1, \cdots, \tilde{n}_{k+1})^T$, we have $\tilde{Q}^T \tilde{m} = \tilde{Q}^T \frac{1}{\tilde{n}}\tilde{C} \cdot r = \frac{1}{\tilde{n}}\tilde{R} \cdot r$.

Then $\tilde{B}$ can be computed by similar arguments as in the previous case. Therefore, it takes $O(k^3)$ for updating B.

Thus, the time for QR-updating of C and updating of W and B with the insertion of a new instance from a new class is $O(dk + k^3)$. The pseudo-code is given in **Algorithm 3**.

5.3 Main algorithm

With the above two incremental updating schemes, the incremental IDR/QR works as follows: For a given new instance x, determine whether it is from an existing or new class; If it is from an existing class, update the QR Decomposition of the centroid matrix C and W and B by applying **Algorithm 2**; otherwise update the QR Decomposition of the centroid matrix C and W and B by applying **Algorithm 3**; The above procedure is repeated until all points are considered. With the final updated $\tilde{W}$ and $\tilde{B}$, we can compute the $\tilde{k}$ eigenvectors $\{\phi_i\}_{i=1}^{\tilde{k}}$ of $(\tilde{W} + \mu I_{\tilde{k}})^{-1}\tilde{B}$, and assign $[\phi_1, \cdots, \phi_{\tilde{k}}]$ to M, where $\tilde{k}$ is the (updated) number of classes ($\tilde{k}$ equals k, if x is from an existing, and $k+1$ otherwise). Then the transformation $G = \tilde{Q}M$, assuming $\tilde{C} = \tilde{Q}\tilde{R}$ is the updated QR Decomposition.

The incremental IDR/QR proposed obeys the following general criteria for an *incremental learning* algorithm [20]: (1) It is able to learn new information from new data; (2) It does not require access to the original data; (3) It preserves previously acquired knowledge; (4) It is able to accommodate new classes that may be introduced with new data.

6. EMPIRICAL EVALUATION

In this section, we evaluate both the batch version and the incremental version of the IDR/QR algorithm. The performance is mainly measured by the computational cost in terms of the classification accuracy and execution time. In

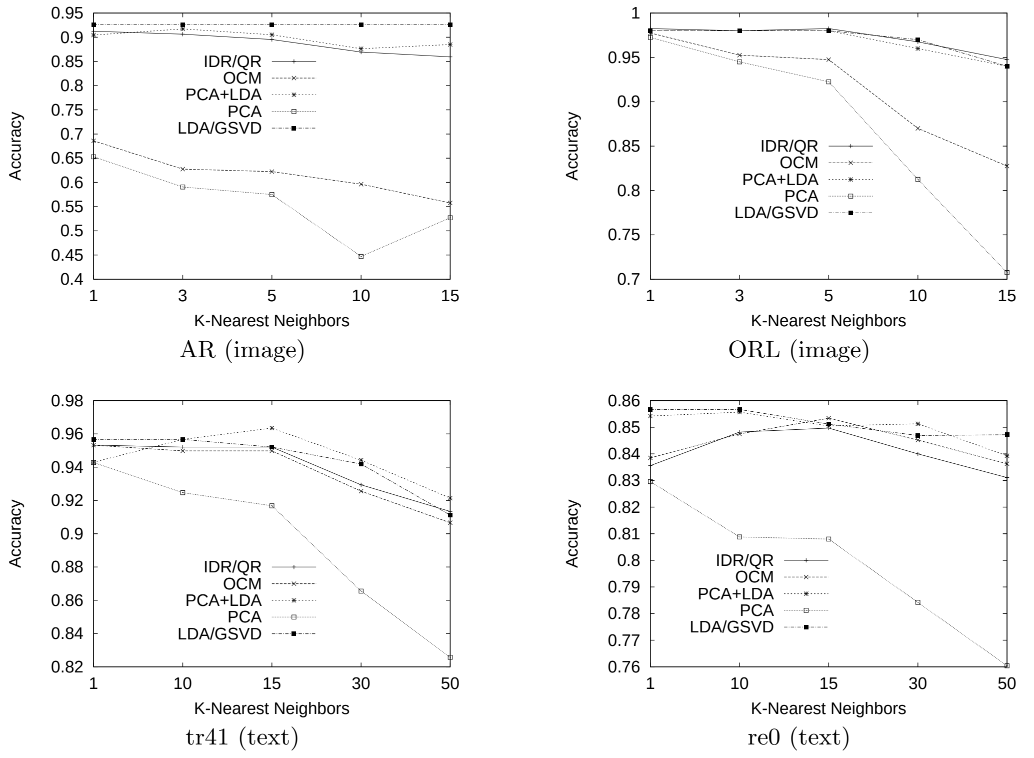

Figure 1: Comparison of classification accuracy on four test data sets

the experiment, we applied the K-Nearest Neighbor (K-NN) method [7] as the classification algorithm and classification accuracies are estimated by 10-fold cross validation.

Experimental Platform: All experiments were performed on a PC with a P4 1.8GHz CPU and 1GB main memory running a Linux operating system.

Experimental Data Sets: Our experiments were performed on the following four real-world data sets, which are from two different application domains, including face recognition and text retrieval. Some characteristics of these data sets are shown in Table 3.

1. AR[1] is a popular face image data set [18]. The face images in AR contain a large area of occlusion, due to the presence of sun glasses and scarves, which leads to a relatively large within-class variance in the data set. In our experiments, we use a subset of the AR data set. This subset contains 1638 face images of entire face identities (126). The image size of this subset is 768×576. We first crop the image from row 100 to 500, column 200 to 550, and then subsample the cropped images down to a size of $101 \times 88 = 8888$.

2. ORL[2] is another popular face image data set, which includes 40 face identities, i.e., 40 classes. The face images in ORL only contain pose variation, and are perfectly centralized/localized. The image size of ORL is $92 \times 112 = 10304$. All dimensions (10304) are used to test our dimension reduction algorithms.

[1]http://rvl1.ecn.purdue.edu/∼aleix/aleix_face_DB.html
[2]http://www.uk.research.att.com/facedatabase.html

Data set	Size	Dim	# of classes
AR	1638	8888	126
ORL	400	10304	40
tr41	878	7454	10
re0	1504	2886	13

Table 3: Statistics for our test data sets

3. tr41 document data set is derived from the TREC-5, TREC-6, and TREC-7 collections [3].

4. re1 document data set is derived from *Reuters-21578* text categorization test collection Distribution 1.0 [4].

6.1 The performance of batch IDR/QR

In this experiment, we compare the performance of the batch IDR/QR with several other dimension reduction algorithms including PCA+LDA, LDA/GSVD, OCM, and PCA. Note that IDR/QR applies regularization to the reduced within-class scatter, i.e., $W + \mu I_c$. We chose $\mu = 0.5$ in our experiments, while it produced good overall results.

6.1.1 Classification Accuracy

Figure 1 shows the classification accuracies on our four test data sets using five different dimension reduction algorithms. Main observations are as follows:

[3]http://trec.nist.gov
[4]http://www.research.att.com/∼lewis

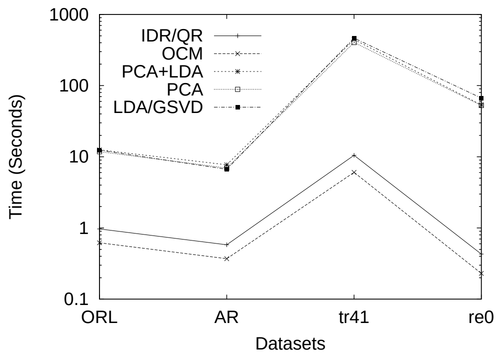

Figure 2: Comparing the efficiency of computing the transformation (measured in seconds in log-scale)

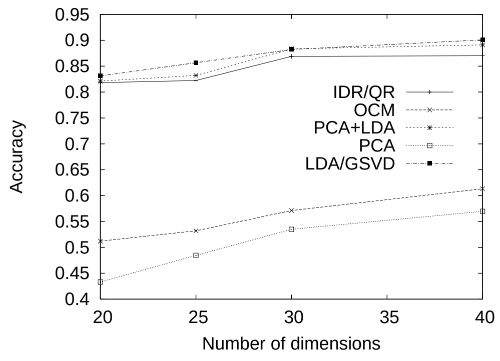

Figure 3: The effect of small reduced dimension on classification accuracy using AR

- The most interesting result is from the AR data set. We can observe that batch IDR/QR, PCA+LDA and LDA/GSVD significantly outperform other two dimension reduction algorithms, PCA and OCM, in terms of the classification accuracy. Recall that the face images in the AR data set contain a large area of occlusion, which results in the large within-class variance in each class. The effort of minimizing of the within-class variance achieves distinct success in this situation. However, neither PCA nor OCM has the effort in minimizing the within-class variance. This explains why they have a poor classification performance on AR.

- Another interesting observation is that OCM performs well on text data sets. This observation is likely due to the fact that text data sets tend to have relatively small within-class variances. This observation suggests that OCM is a good choice in practice if the data is known to have small within-class variances.

6.1.2 Efficiency in computing the transformation

Figure 2 shows the execution time (in log-scale) of different tested methods for computing the transformation. Even with log-scale presentation, we can still observe that the execution time for computing the transformation by IDR/QR or OCM is significantly smaller than that by PCA+LDA, LDA/GSVD, and PCA.

6.1.3 The Effect of Small Reduced Dimension

Here, we evaluate the effect of small reduced dimension on the classification accuracy using the AR data set. Recall that the reduced dimension by the IDR/QR algorithm is c, where c is the number of classes in the data set. If the value c is large (such as AR, which contains 126 classes), the reduced representation may not be suitable for efficient indexing and retrieval. Since the reduced dimensions from IDR/QR are ordered by their discriminant powers (see Line 7 of **Algorithm 1**), an intuitive solution is to choose the first few dimensions in the reduced subspace from IDR/QR. The experimental results are shown in Figure 3. As can be seen, the accuracy achieved by keeping the first 20 dimensions only is still sufficiently high.

6.2 The Performance of incremental IDR/QR

In this experiment, we compare the performance of incremental IDR/QR with that of batch IDR/QR in terms of classification accuracy and the computational cost. We randomly order the data items in the data set and insert them into the training set one by one incrementally with the given order. The remaining data is used as the test set. Initially, we select the first 30% data items as the training set. Incremental updating is then performed with the remaining data items inserted one at a time.

Figure 4 shows the achieved classification accuracies by batch IDR/QR and incremental IDR/QR on four data sets. In the figure, the horizontal axis shows the portion of training data items, and the vertical axis indicates the classification accuracy (as a percentage). We observe a trend that the accuracy increases when more and more training data items are involved. Another observation is that the accuracy by incremental IDR/QR is quite close to that by batch IDR/QR. Indeed, on four data sets, the maximal accuracy deviation between incremental IDR/QR and batch IDR/QR is within 4%. Recall that incremental IDR/QR is carried through QR Decomposition in three steps: (1) QR-updating of the centroid matrix C; (2) Updating of the reduced within-class scatter W; and (3) Updating of the reduced between-class scatter B. The first and third steps are based on the exact scheme, while the second step involves approximation. Note that the main rationale behind our approximation scheme in updating W is that the change of Q matrix is relatively small and can be neglected for each single updating, where $C = QR$ is the QR Decomposition of C.

To give a concrete idea of the benefit of using incremental IDR/QR from the perspective of efficiency, we give a comparison on the compuational cost between batch IDR/QR and incremental IDR/QR. The experimental results are given in Figure 5. As can be seen, the execution time of incremental IDR/QR is significantly smaller than that of batch IDR/QR. Indeed, for a single updating, incremental IDR/QR takes $O(dk + k^3)$, while batch IDR/QR takes $O(ndk)$, where k is the number of classes in the current training set and n is the size of the current training set. The time for a single updating in incremental IDR/QR is almost a constant $O(dc + c^3)$, when all classes appear in the current training set, and the speed-up of incremental IDR/QR over batch IDR/QR keeps increasing when more points are inserted into the training set. Note that we only count the time for Lines 1–6 in **Algorithm 1**, since each updating in incremental IDR/QR only involves the updating of the QR Decomposition (Line 2), W (Line 5) and B (Line 6).

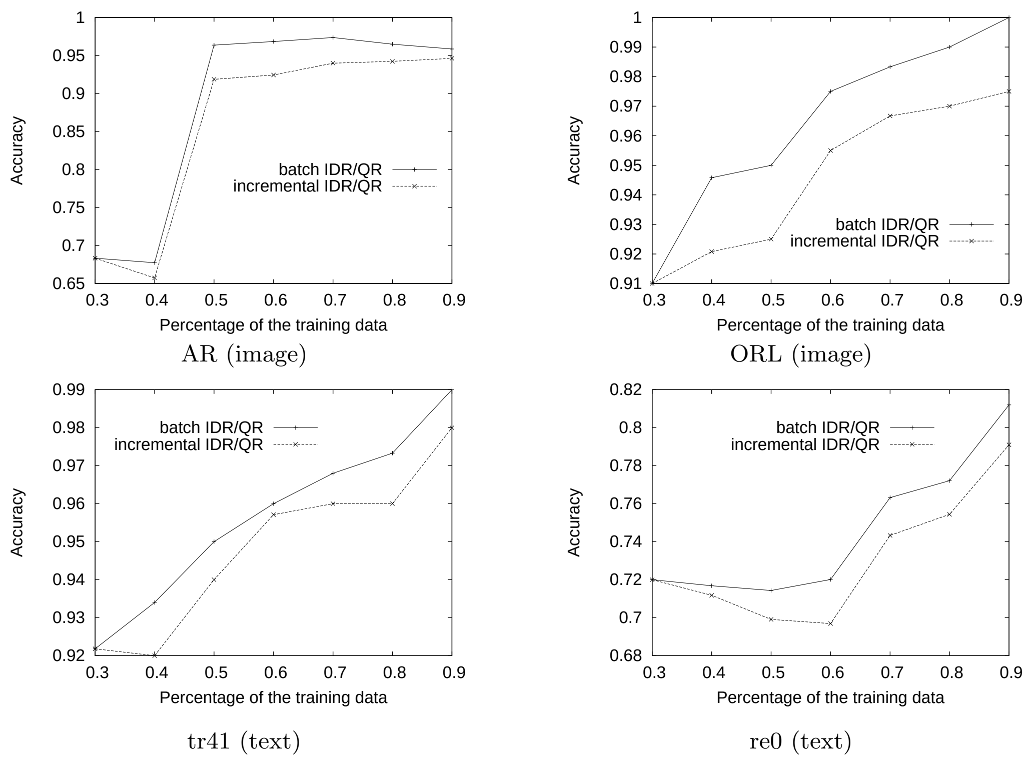

Figure 4: Comparison of classification accuracy between incremental IDR/QR and batch IDR/QR

7. CONCLUSIONS

In this paper, we have proposed an LDA based incremental dimension reduction algorithm, called IDR/QR, which applies QR Decomposition rather than SVD. The IDR/QR algorithm does not require whole data matrix in main memory. This is desirable for large data sets. More importantly, the IDR/QR algorithm can work incrementally. In other words, when new data items are dynamically inserted, the computational cost of the IDR/QR algorithm can be constrained by applying efficient QR-updating techniques. In addition, our theoretical analysis indicates that the computational complexity of the IDR/QR algorithm is linear in the number of the data items in the training data set as well as the number of classes and the number of dimensions. Finally, our experimental results show that the accuracy achieved by the IDR/QR algorithm is very close to the best possible accuracy achieved by other LDA based algorithms. However, the IDR/QR algorithm can be an order of magnitude faster. When dealing with dynamic updating, the computational advantage of IDR/QR over SVD or GSVD based LDA algorithms becomes more dramatic while still achieving the comparable accuracy.

As for future research, we plan to investigate the applications of the IDR/QR algorithm on searching extremely high-dimenional multimedia data, such as video.

Acknowledgement

This research was sponsored, in part, by the Army High Performance Computing Research Center under the auspices of the Department of the Army, Army Research Laboratory cooperative agreement number DAAD19-01-2-0014, and the National Science Foundation Grants CCR-0204109, ACI-0305543, IIS-0308264, and DOE/ LLNL W-7045-ENG-48. The content of this work does not necessarily reflect the position or policy of the government and the National Science Foundation, and no official endorsement should be inferred. Access to computing facilities was provided by the AHPCRC and the Minnesota Supercomputing Institute.

8. REFERENCES

[1] P.N. Belhumeour, J.P. Hespanha, and D.J. Kriegman. Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

[2] C. Böhm, S. Berchtold, and D. A. Keim. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Computing Surveys*, 33(3):322–373, 2001.

[3] S. Chakrabarti, S. Roy, and M. Soundalgekar. Fast and accurate text classification via multiple linear discriminant projections. In *VLDB*, pages 658–669, Hong Kong, 2002.

[4] S. Chandrasekaran, B. S. Manjunath, Y. F. Wang, J. Winkeler, and H. Zhang. An eigenspace update algorithm for image analysis. *Graphical Models and Image Processing: GMIP*, 59(5):321–332, 1997.

[5] C. Chatterjee and V. P. Roychowdhury. On self-organizing algorithms and networks for class-separability features. *IEEE Trans. Neural Networks*, 8(3):663–678, 1997.

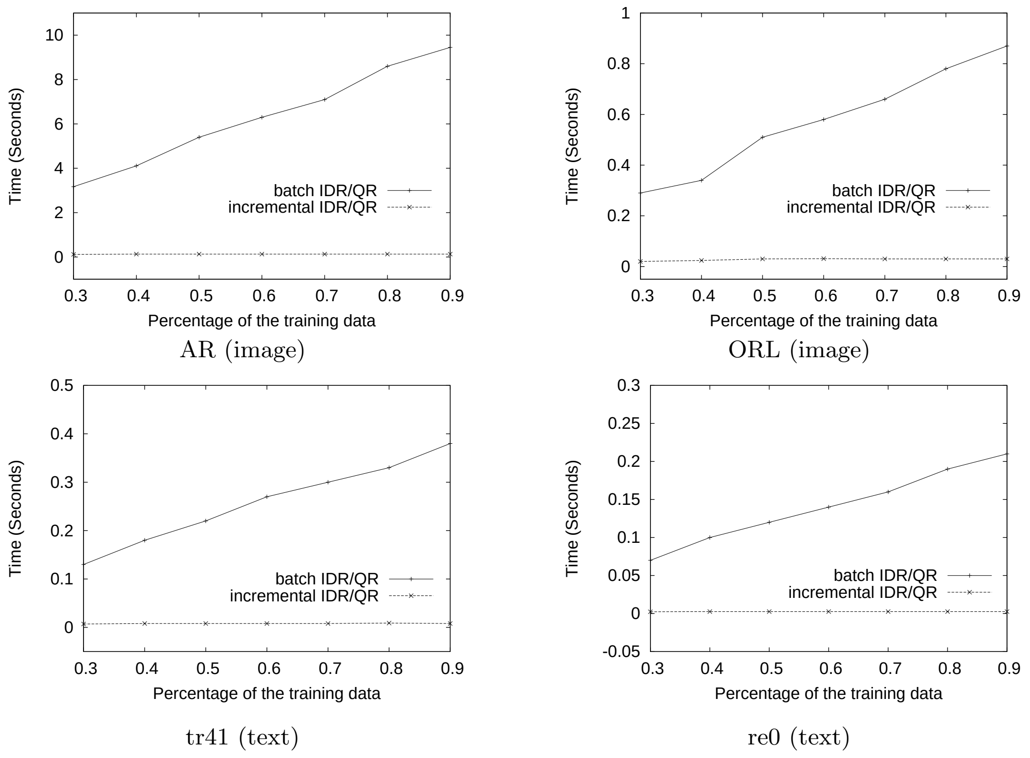

Figure 5: Comparison of computional cost between incremental IDR/QR and batch IDR/QR.

[6] J.W. Daniel, W. B. Gragg, L. Kaufman, and G. W. Stewart. Reorthogonalization and stable algorithms for updating the gram-schmidt QR factorization. *Mathematics of Computation*, 30:772–795, 1976.

[7] R.O. Duda, P.E. Hart, and D. Stork. *Pattern Classification*. Wiley, 2000.

[8] J. H. Friedman. Regularized discriminant analysis. *Journal of the American Statistical Association*, 84(405):165–175, 1989.

[9] K. Fukunaga. *Introduction to Statistical Pattern Classification*. Academic Press, USA, 1990.

[10] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, MD, USA, third edition, 1996.

[11] P. Hall, D. Marshall, and R. Martin. Merging and splitting eigenspace models. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(9):1042–1049, 2000.

[12] P. Howland, M. Jeon, and H. Park. Structure preserving dimension reduction for clustered text data based on the generalized singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 25(1):165–179, 2003.

[13] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

[14] K. V. Ravi Kanth, D.t Agrawal, A. E. Abbadi, and A. Singh. Dimensionality reduction for similarity searching in dynamic databases. *Computer Vision and Image Understanding: CVIU*, 75(1–2):59–72, 1999.

[15] W.J. Krzanowski, P. Jonathan, W.V McCarthy, and M.R. Thomas. Discriminant analysis with singular covariance matrices: methods and applications to spectroscopic data. *Applied Statistics*, 44:101–115, 1995.

[16] J. Mao and K. Jain. Artificial neural networks for feature extraction and multivariate data projection. *IEEE Trans. Neural Networks*, 6(2):296–317, 1995.

[17] A. Martinez and A. Kak. PCA versus LDA. In *IEEE Trans. Pattern Analysis and Machine Intelligence*, volume 23, pages 228–233, 2001.

[18] A.M. Martinez and R. Benavente. The AR face database. Technical Report No. 24, 1998.

[19] H. Park, M. Jeon, and J.B. Rosen. Lower dimensional representation of text data based on centroids and least squares. *BIT*, 43(2):1–22, 2003.

[20] R. Polikar, L. Udpa, S. Udpa, and V. Honavar. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE Trans. Systems, Man, and Cybernetics*, 31:497–508, 2001.

[21] D. L. Swets and J.Y. Weng. Using discriminant eigenfeatures for image retrieval. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(8):831–836, 1996.

[22] F.D.L. Torre and M. Black. Robust principal component analysis for computer vision. In *ICCV*, volume I, pages 362–369, 2001.

[23] J. Ye, R. Janardan, C.H. Park, and H. Park. An optimization criterion for generalized discriminant analysis on undersampled problems. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(8):982–994, 2004.