# Supervised Dimensionality Reduction on Streaming Data

Mao Ye
School of Computer Science and Engineering
University of Electronic Science
and Technology of China
Chengdu 610054, P.R.China
yem_mei29@hotmail.com

Xue Li, Maria E. Orlowska
School of Information Technology
and Electronic Engineering
The University of Queensland
Brisbane 4072, Australia
{xueli,maria}@itee.uq.edu.au

## Abstract

*We propose a sliding-window approach for the dimensionality reduction for linear discriminant analysis(LDA) on streaming data. Streaming data are time variant and can be in high dimensions. When a sliding window is moving along data stream, the data that have passed out of the window will be forgotten (i.e., deleted).We propose a LDA dimensionality reduction algorithm based on different sliding windows. The experiments on UCI data sets have been conducted and results are compared with the batch IDR/QR LDA method. It is shown that our algorithm present an efficient solution to the problem of dimensionality reduction on streaming data yet still have a good performance on computational cost and the classification accuracy.*

## 1. Introduction

The problem of dimensionality reduction is old but still alive today. Many dimensionality reduction methods have been proposed to transform high dimensional data to low dimensional data such that the information of high dimensional data is kept as much as possible (see [2] for a review). The approach of dimensionality reduction can be supervised or unsupervised. Supervised approach means we know the class label of data beforehand. Linear discriminant analysis (LDA) is a supervised dimensionality reduction algorithm [2]. It has been widely used in many areas, such as text retrieval [5], face recognition [3], and image processing [4]. The theory and applications of LDA have been studied thoroughly [2]. LDA seeks directions of efficient discrimination, i.e. the linear transformation matrix which maximizes the ratio of between-class distance to within-class distance. Traditional LDA is a generalized eigenvalue problem with within-class and between-class scatter matrices. These two matrices are computed in batch mode, which implies that we have collected all data in advance.

However, when we apply this approach to real-world applications, a complete training set might not be given beforehand. Traditional LDA algorithm may be useless because it requires all high dimensional data and all covariance matrices be kept in the memory and updated altogether. For this reason, A few one-pass incremental LDA algorithms have been proposed. In [6], a one-pass incremental dimensionality reduction algorithm namely ILDA is proposed. Although ILDA can perform incremental dimensionality reduction, this algorithm needs to solve a high dimensional generalized eigenvalue problem. If the dimension of the data is very high, ILDA will confront difficulty because of the general limitation of a machine's memory. By using QR decomposition, another one-pass incremental dimensionality reduction algorithm, namely incremental IDR/QR has been proposed in [2]. By carefully using QR decomposition, the whole data matrix needs not to be saved in main memory. By using some approximated matrices, computing the data covariance matrix can be avoided. However, the approximate error may be accumulated as very many new data are appended sequentially.

Recently, mining on data streams has brought an increasing interest [1]. Streaming data arrive continuously with high speed and large volume. The dimensionality of a data stream can be very high and data may be seen only once for the purpose of realtime processing. In this paper, we consider the dimensionality reduction on streaming data based on a sliding window approach. Along with the new data come in, the old data will be deleted explicitly one by one or in chunks. Thus, the previous incremental LDA learning algorithms are not directly applicable to the problem that we are discussing here. Our proposed algorithm can compute the discriminant directions exactly. This algorithm is named as Streaming LDA (SLDA). The rest of this paper is organized as follows. Section 2 introduces the traditional LDA and the batch IDR/QR algorithms. Streaming data models

are then defined. Algorithm called SLDA are proposed in section 3. Comprehensive experiments and discussions are presented in section 4. Conclusions are given in section 5.

## 2 Preliminary and Problem Statement

Given a data set $D = \{(x(t), c(t)), t = 1, \cdots, N\}$, where $x(t) = [x_1(t), x_2(t), \cdots, x_n(t)]^T \in R^n$, $c(t)$ is the class label of $x(t)$ which is already known. Assume data set is partitioned into $M$ classes as $I_1, I_2, \cdots, I_M$ with cardinality $n_1, n_2, \cdots, n_M$, where $\sum_{i=1}^{M} n_i = N$. For a given number $l$, the classical LDA method is to find a linear transformation $W \in R^{n \times l}$, which should preserve the class structure of original high $n$-dimensional space in the lower $l$-dimensional space as much as possible by maximizing the Fisher criterion

$$J(W) = \frac{\text{Trace}(W^T A W)}{\text{Trace}(W^T B W)}, \tag{1}$$

in which

$$\begin{aligned} B &= \sum_{i=1}^{M} \sum_{x \in I_i} (x - m_i)(x - m_i)^T, \\ A &= \sum_{i=1}^{M} n_i (m_i - m)(m_i - m)^T, \end{aligned}$$

where $m_i$ is the centroid of the $i$th class and $m$ is the centroid of the whole data set. $A$ and $B$ are referred to as within-class and between-class scatter matrices. Mathematically the solution $W$, to the optimization problem (1), is the matrix which consists of the first $l$ generalized eigenvectors as follows

$$AW = \Lambda B W \tag{2}$$

where $\Lambda$ is the diagonal matrix, which are the first $l$ generalized eigenvalues.

Next, let we simply introduce the batch IDR/QR algorithm, which is derived by using the following theorem in [2].

**Theorem 1** *Let $C = [m_1, \cdots, m_M]$ and $C = QR$ be the $QR$ decomposition of $C$, where $Q \in R^{n \times M}$ has orthonormal columns and $R \in R^{M \times M}$ is upper triangular. Then, $W^* = QM$ for any orthogonal matrix $M$ maximizes the separation between different classes, i.e. the criterion $J_1(W) = Trace(W^T A W)$.*

Since $W^{*T} A W^* = M^T (Q^T A Q) M$ and $W^{*T} B W^* = M^T (Q^T B Q) M$, by Theorem 1, the optimization problem $J(W)$ is equivalent to find matrix $M$ which maximizes $J_2(M) = \text{Trace}(M^T A_1 M)/\text{Trace}(M^T B_1 M)$, with $A_1 = Q^T A Q$ and $B_1 = Q^T B Q$. Note that $A_1, B_1 \in R^{M \times M}$, which has smaller size than the original scatter matrix $A$ and $B$. Compared with the traditional LDA method, the good performance of the batch IDR/QR method was shown in [2]. The procedure of the batch IDR/QR in [2] is follows : First, compute the centroid matrix $C$; Second, compute the $QR$ decomposition of $C = QR$; Then, compute the generalized eigenvector matrix $M$ of matrix pencil $(A_1, B_1)$; Finally, let $W = QM$, which is the wanted linear transformation.

Dimensionality reduction on streaming data is performed in a sliding window. Without loss of generality, at the beginning, we assume the data set in the sliding window of streaming data is $X(0) = \{(x(t), c(t)), t = 1, \cdots, N\}$, where $x(t) = [x_1(t), x_2(t), \cdots, x_n(t)]^T \in R^n$, $c(t)$ is the class label of $x(t)$ which is already known. The data set in the sliding window is changing when the time is increasing. Not only the new data will come in, but also the old data will be deleted explicitly one by one or in chunk way. Assume the data set in the sliding window was changed to $X(k)$ at time $k$. The problem in our paper is to find a linear transformation matrix $W(k)$ effectively, which maximizes the criterion function $J_1(W(k))$ for the data set $X(k)$.

In the sliding window of streaming data, at the same time the new data is coming, the old data will be deleted. So dimensionality reduction on streaming data is composed of two parts. The first part is to handle the new instances, previous incremental LDA learning algorithms can be used. The second part is to handle the cases that the old data are deleted. Algorithm SLDA is proposed to handle the second part.

## 3 Streaming LDA

We adopt the following convention: For any variable $X$, its updated version in the next time is denoted by $\widetilde{X}$. the streaming LDA algorithm consists of three steps: Firstly we compute the centroid matrix $\widetilde{C}$. Then we update the reduced within-class scatter matrix $B_1$. Finally, the reduced between-class scatter matrix $A_1$ is updated. Then, the linear transformation matrix $W$ can be obtained by solving generalized eigenvalue problem $(A_1, B_1)$. Before the data are deleted, assume the data set in the sliding window is partitioned into $k$ classes as $I_1, I_2, \cdots, I_k$ with cardinality $n_1, n_2, \cdots, n_k$, where $\sum_{i=1}^{k} n_i = N$. And the dimension of data is $n$.

### 3.1 Sequential Deletion of Old Data

Assume $y$ is the instance to be deleted which belongs to $i$th class. Two situations exist here when we delete old data $y$ from the data set of the sliding window. The first is that the $i$th class remains. The other is that the $i$th class will also be deleted because no data belongs to this class.

Case 1: When the instance $y$ is deleted, the number of $i$th class $n_i \neq 0$. There are three steps to compute the linear transformation matrix $W$.

Step 1: QR-updating of centroid matrix $C$. Since the data $y$ belongs to $i$th class, so the updated centroid matrix is

$$\widetilde{C} = [m_1, \cdots, \widetilde{m}_i, \cdots, m_k], \tag{3}$$

where $\widetilde{m}_i = m_i + f$ and $f = (m_i - y)/(n_i - 1)$. Thus, $\widetilde{C} = C + f \cdot g$ for $g = (0, \cdots, 1, \cdots, 0)^T$. By using the same method in [2], we can compute the $\widetilde{Q}$ and $\widetilde{R}$ of $QR$-updating as $\widetilde{C} = \widetilde{Q}\widetilde{R}$. The time complexity is $O(nk)$. And $O(nk)$ memory is needed to save the centroid matrix and the number of all classes.

Step 2: Updating the reduced within-class cluster matrix $B_1$ to $\widetilde{B}_1$. Since

$$\begin{aligned}\sum_{x \in I_i}(x - m_i)(m_i - \widetilde{m}_i)^T &= (\sum_{x \in I_i} x - n m_i)(m_i - \widetilde{m}_i)^T \\ &= 0,\end{aligned}$$

it follows that

$$\begin{aligned}\widetilde{B} &= B - (y - \widetilde{m}_i)(y - \widetilde{m}_i)^T \\ &\quad + n_i(m_i - \widetilde{m}_i)(m_i - \widetilde{m}_i)^T.\end{aligned}$$

Let $u = \widetilde{Q}^T(y - \widetilde{m}_i)$ and $v = \widetilde{Q}^T(m_i - \widetilde{m}_i)$, we have

$$\widetilde{B_1} = \widetilde{Q}^T B \widetilde{Q} - uu^T + n_i vv^T.$$

Computing $\widetilde{B}$ needs $O(n^2)$ time. Since the dimension of $u$ and $v$ is $d \times n$, $O(kn)$ time is used to compute the $\widetilde{B_1}$. To update the within-class cluster matrix $\widetilde{B_1}$, we need store the $n \times n$ matrix $B$ in the memory. Thus, step 2 costs most of time and memory in our algorithm.

Step 3: updating the reduced between-class scatter matrix $A_1$ to $\widetilde{A}_1$. Only $n_i$, $N$ and $m$ are updated to $\widetilde{n}_i = n_i - 1$, $\widetilde{N} = N - 1$ and $\widetilde{m} = \widetilde{C}r/\widetilde{N}$ where $r = [n_1, \cdots, \widetilde{n}_i, \cdots, n_k]^T$. Define

$$\begin{aligned}\widetilde{H}_a &= [\sqrt{n_1}(m_1 - \tilde{m}), \cdots, \sqrt{n_k}(m_k - \tilde{m})] \\ &= [m_1, \cdots, \widetilde{m}_i, \cdots, m_k, \tilde{m}]F,\end{aligned}$$

where the matrix

$$F = \begin{pmatrix} D \\ -h^T \end{pmatrix}$$

and $D = \text{diag}(\sqrt{n_1}, \cdots, \sqrt{\widetilde{n}_i}, \cdots, \sqrt{n_k})$, and $h = [\sqrt{n_1}, \cdots, \sqrt{\widetilde{n}_i}, \cdots, \sqrt{n_k}]^T$. Since $\widetilde{A}_1 = \widetilde{Q}^T \widetilde{A} \widetilde{Q} = \widetilde{Q}^T \widetilde{H}_a \widetilde{H}_a^T \widetilde{Q}$, by using the same method in [2], it follows that

$$\begin{aligned}\widetilde{A}_1 &= \left(\widetilde{R}D - \left(\frac{1}{\widetilde{N}}\widetilde{R} \cdot r\right) \cdot h^T\right) \\ &\quad \cdot \left(\widetilde{R}D - \left(\frac{1}{\widetilde{N}}\widetilde{R} \cdot r\right) \cdot h^T\right)^T.\end{aligned}$$

Since the dimension of matrices $\widetilde{R}$ and $D$ is $k \times k$ and the dimension of vectors $h$ and $r$ is $k$, the time complexity to compute the between-class scatter matrix is $O(k^3)$.

As described before, the matrices $A_1$ and $B_1$ are used to compute the linear transformation matrix $W$ instead of the matrices $A$ and $B$. The reason is that the dimensionality of generalized eigenvalue problem is reduced sharply from $n \times n$ to $k \times k$.

Case 2: The $i$th class is deleted. In this case, after $y$ is deleted, $n_i = 0$ which means that $i$th class is also deleted.

Step 1: QR-updating of centroid matrix $C$. The updated centroid matrix $\widetilde{C} = [m_1, \cdots, m_{i-1}, m_{i+1}, \cdots, m_k]$. Compute the $QR$ decomposition as $\widetilde{C} = \widetilde{Q}\widetilde{R}$, where $\widetilde{Q} \in R^{n \times (k-1)}$ and $\widetilde{R} \in R^{(k-1) \times (k-1)}$. The time complexity is $O(n(k-1))$. And $O(nk)$ memory is needed to save the centroid matrix and the number of all classes.

Step 2: Updating of $B_1$. Since

$$\begin{aligned}\widetilde{B} &= \sum_{j=1, j \neq i}^{k} \sum_{x \in I_j} (x - m_j)(x - m_j)^T \\ &\quad + (y - m_i)(y - m_i)^T \\ &= B.\end{aligned}$$

Thus, $\widetilde{B_1} = \widetilde{Q}^T B \widetilde{Q}$. Since the dimension of matrix $B$ is $n \times n$, so it takes $O((k-1)n)$ time to compute $\widetilde{B_1}$. And $O(n^2)$ memory is needed to save the within scatter matrix $B$.

Step 3: Updating of $A_1$. $N$ and $m$ are updated to $\widetilde{N} = N - 1$ and $\widetilde{m} = \widetilde{C}r/\widetilde{N}$ where $r = [n_1, \cdots, n_{i-1}, n_{i+1}, \cdots, n_k]^T$. Since $\widetilde{A} = \widetilde{H}_a \widetilde{H}_a^T$ where

$$\begin{aligned}\widetilde{H}_a &= [\sqrt{n_1}(m_1 - \tilde{m}), \cdots, \sqrt{n_k}(m_k - \tilde{m})] \\ &= [m_1, \cdots, m_{i-1}, m_{i+1}, \cdots, m_k, \tilde{m}]F,\end{aligned}$$

and the matrix

$$F = \begin{pmatrix} D \\ -h^T \end{pmatrix},$$

$D = \text{diag}(\sqrt{n_1}, \cdots, \sqrt{n_{i-1}}, \sqrt{n_{i+1}}, \cdots, \sqrt{n_k})$ and $h = [\sqrt{n_1}, \cdots, \sqrt{n_{i-1}}, \sqrt{n_{i+1}}, \cdots, \sqrt{n_k}]$, the algorithm to update $A_1$ is the same as that of case 1. As denoted before, the time complexity to compute the between-class scatter matrix $A_1$ is $O(k^3)$.

Time complexity of SLDA algorithm in this case is $O(n^2 + k^3)$ and $O(n^2)$ memory is needed. The main cost part is updating the within scatter matrix $B_1$ which needs $O(n^2)$ time memory. The other parts only cost $O(nk)$ time and memory.

### 3.2 Deletion of Chunk Old Data

The chunk old data $Y = \{y_1, \cdots, y_L\}$ will be deleted from the sliding window. Corresponding to the data classes

of previous sliding window, the data set $Y$ is partitioned into $d$ classes as $J_i(i \in \varphi)$ where $\varphi$ is the subset of $\{1, \cdots, k\}$ with number $d$. There is $l_i$ instances to be deleted belong to class $J_i$ and $\sum_{i\in\varphi} l_i = L$. Assume $Y_1 = \{i|l_i = n_i, i \in \varphi\}$ and $Y_2 = \{i|l_i \neq n_i, i \in \varphi\}$ with $|Y_1| = d_1$.

Step 1: QR-updating of centroid matrix $C$. For any $i \in Y_1$, $n_i$ is updated to $\widetilde{n}_i = 0$ which means the $i$th class is deleted. We should delete the vectors $m_i$ from the centroid matrix $C$. For any $i \in Y_2$, $n_i$ is updated to $\widetilde{n}_i = n_i - l_i$, and $\widetilde{m}_i = (n_i m_i - \sum_{y\in J_i} y)/\widetilde{n}_i$. The column vectors of updated centroid matrix $\widetilde{C}$ is composed of $\widetilde{m}_i$ for $i \in Y_2$ and $m_i$ for $i \in \{1, \cdots, k\} \setminus \varphi$. And the order of these column vectors is the same as that of previous centroid matrix $C$. Compute the $QR$ decomposition as $\widetilde{C} = \widetilde{Q}\widetilde{R}$. The time complexity is $O(n(k-d_1))$. And $O(nk)$ memory is needed to save the centroid matrix and the number of all classes.

Step 2: Updating the reduced within-class cluster matrix $B_1$ to $\widetilde{B}_1$. Since for $i \in Y_2$,$\sum_{x\in I_i}(x-m_i)(m_i-\widetilde{m}_i)^T = 0$, it follows that

$$\begin{aligned}\widetilde{B} &= B - \sum_{i\in Y_1}\sum_{x\in J_i}(x-m_i)(x-m_i)^T \\ &\quad - \sum_{i\in Y_2}\sum_{x\in J_i}(x-\widetilde{m}_i)(x-\widetilde{m}_i)^T \\ &\quad + \sum_{i\in Y_2} n_i(m_i-\widetilde{m}_i)(m_i-\widetilde{m}_i)^T.\end{aligned}$$

Let $u_i = \widetilde{Q}^T(x-\widetilde{m}_i)$, $z_i = \widetilde{Q}^T(x-m_i)$ and $v_i = \widetilde{Q}^T(m_i-\widetilde{m}_i)$, we have

$$\begin{aligned}\widetilde{B_1} &= \widetilde{Q}^T B\widetilde{Q} - \sum_{i\in Y_1}\sum_{x\in J_i} z_i z_i^T - \sum_{i\in Y_2}\sum_{x\in J_i} u_i u_i^T \\ &\quad + \sum_{i\in Y_2} n_i v_i v_i^T.\end{aligned}$$

Computing $\widetilde{B}$ needs $O(n^2)$ time. Since the dimension of $u$ and $v$ is $k \times n$, $O(kn)$ time is used to compute the $\widetilde{B_1}$. To update the within-class cluster matrix $\widetilde{B_1}$, we need store the $n \times n$ matrix $B$ in the memory. Thus, step 2 costs most of time and memory in our algorithm.

Step 3: Updating the reduced between-class scatter matrix $A_1$ to $\widetilde{A}_1$. The $n_i$ for $i \in Y_2$, $N$ and $m$ are updated to $\widetilde{n}_i = n_i - l_i$, $\widetilde{N} = N - L$ and $\widetilde{m} = \widetilde{C}r/\widetilde{N}$ where the components of column vector $r$ are the values $n_i$ for $i \in \{1, \cdots, k\} \setminus \varphi$ and $\widetilde{n}_i$ for $i \in Y_2$. The order of the components of vector $r$ is corresponding to that of column vectors of the centroid matrix $\widetilde{C}$. By using the same method as previous section, we have that

$$\begin{aligned}\widetilde{A}_1 &= (\widetilde{R}D - \widetilde{R}\cdot r\cdot h^T/\widetilde{N}) \\ &\quad \cdot(\widetilde{R}D - \widetilde{R}\cdot r\cdot h^T/\widetilde{N})^T,\end{aligned}$$

where $D$ is a diagonal matrix and $h$ is a column vector, whose components are the values $\sqrt{n_i}$ for $i \in \{1, \cdots, k\}\setminus\varphi$ and $\sqrt{\widetilde{n}_i}$ for $i \in Y_2$ and the order of these components is corresponding to that of column vectors of the centroid matrix $\widetilde{C}$. As denoted before, the time complexity to compute the between-class scatter matrix $A_1$ is $O((k-d_1)^3)$.

The time complexity of SLDA algorithm in this case is $O(n^2 + (k-d_1)^3)$ and $O(n^2)$ memory is needed. The main cost part is updating the within scatter matrix $B_1$ which needs $O(n^2)$ time and memory respectively. The other parts only cost $O(n(k-d_1))$ time and $O(nk)$ memory.

## 4 Simulations and Discussions

The UCI benchmark ORL face database is used to simulate our problem. ORL is a well-known face image data set, which includes $40$ face individuals, i.e. $40$ classes. Every individual has ten pictures. The image size of ORL is reformulated as $46 \times 56 = 2576$ pixels. This will be our dimension of data set.

To simulate the streaming data by using ORL face database, first, we number the classes as $1, 2, \cdots, 40$, then randomly choose $8$ samples of every class as training samples and the remaining $2$ samples as test samples. At the beginning, assume the data set $X(0)$ is composed of the training samples of the first $30$ classes. At time k, the remaining training samples of $30 - 40$ classes are randomly chosen to insert in the data set $X(k)$ in one by one or chunk way. We delete the training samples of classes $1 - 10$ randomly one by one or in chunk way at the same time.

In the experiments, we use the test samples of classes $20 - 40$ and some noise pictures as the test set. At time $k$, the data set $X(k)$ and the test set are projected into the lower $l$-dimensional space by using our algorithms and the batch IDR/QR method, respectively. The dimension $l$ is set to the class number in the sliding window roughly. Then, the classification accuracy of our algorithm and the batch IDR/QR method is obtained by using K-Nearest Neighbor method with $K = 5$. And threshold value $10$ will be used in KNN to handle the noise pictures. When the distances to the training samples of a test sample are bigger than the threshold value $10$, this test sample will be considered as a new class sample or noise sample which does not belong to the existing classes.

Since SLDA algorithm computes the discriminant directions exactly, the accuracy of SLDA algorithm will be the same as that of the batch IDR/QR method. Our interest will focus on the efficiency of SLDA with different chunk size. Here, the chunk size is set as a constant value when the instances are inserted or deleted in all time. The execution time of SLDA algorithm with different constant chunk sizes which equal to $1, 2, 4, 8, 16, 40$ and the batch IDR/QR method is compared in Fig. 1.

From Fig.1, we could observe that the execution time of SLDA algorithm with small chunk size is smaller than that

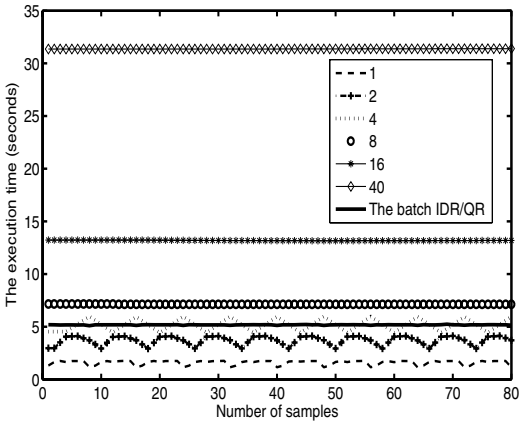


**Figure 1. The execution time of SLDA algorithm with different chunk sizes which equal to $1, 2, 4, 8, 16, 40$ and the batch IDR/QR method.**

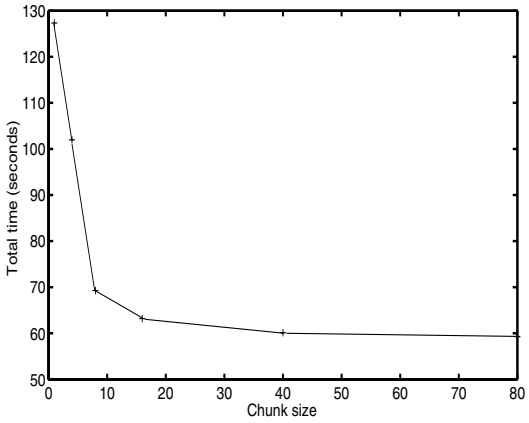


**Figure 2. The total time (seconds) of SLDA algorithm to process $80$ samples with different chunk sizes which equal to $1 \sim 80$.**

of batch IDR/QR method. And the bigger the chunk size of SLDA algorithm is, the more execution time is needed. This is because the computation complexity of algorithm SLDA is increased when the size of chunk data is increased. However, since the batch IDR/QR method need save all the data of sliding window in the memory, if the data size in the sliding window is very big, we have to save the data in the database and read them again. Thus, the execution time of the batch IDR/QR method will increased very much. Although more execution time of SLDA is needed when large chunk size is used, the total time by using larger chunk size will decrease. As it is shown in Fig.2, the total time to process $80$ samples is reduced from $127.3240$ to $59.2550$ when the chunk size is changed from $1$ to $80$.

## 5 Conclusions

In this paper, we have proposed an algorithm that can perform supervised dimensionality reduction in the sliding window on streaming data. Experiments have illustrated that the performance of this algorithm is competitive with that of the batch IDR/QR method. As denoted before, the costs of algorithm are the time and memory spent on the computations of the within-class scatter matrix. The further study lies in how to minimize these computations and maintaining the within-class scatter matrix in the memory.

## Acknowledgments

This work was supported by Application Research Foundation of Science and Technology Bureau of Sichuan Province of China, Grant No. 2006J13-065.

## References

[1] B.Babcock. Models and issues in data stream systems. *Proc. 21th ACM SIGMOD-SIGACT-SIGART Sysmposium on Principles of Database Systems*, pages 1–16, 2002.

[2] J.P. Ye, Q.Li, H.Xiong, H.Park, R.Janardan, and V.Kumar. IDR/QR: An incremental dimension reduction algorithm via qr decomposition. *IEEE Trans. Knowledge and Data Engineering*, 17(9):1208–1221, September 2005.

[3] A. Martinez and R. A.Kak. PCA versus LDA. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23:228–233, 2001.

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

[5] S. Chakrabarti, S.Roy, and M. Soundalgekar. Fast and accurate text classification via multiple linear discriminant projections. *Proc. Very Large Data Base Conf.*, pages 658–669, 2002.

[6] S.Pang, S.Ozawa, and N. Kasabov. Incremental linear discriminant analysis for classification of data streams. *IEEE Trans.System,Man,Cybernetics Part B*, 35(5):905–914, October 2005.