LEAST-SQUARES LDA VIA RANK-ONE UPDATES WITH CONCEPT DRIFT

Yi-Ren Yeh and Yu-Chiang Frank Wang

Research Center for Information Technology Innovation, Academia Sinica, Taipei, Taiwan, 11529
{yryeh, ycwang}@citi.sinica.edu.tw

ABSTRACT

Standard linear discriminant analysis (LDA) is known to be computationally expensive due to the need to perform eigen-analysis. Based on the recent success of least-squares LDA (LSLDA), we propose a novel rank-one update method for LSLDA, which not only alleviates the computation and memory requirements, and is also able to solve the adaptive learning task of concept drift. In other words, our proposed LSLDA can efficiently capture the information from recently received data with gradual or abrupt changes in distribution. Moreover, our LSLDA can be extended to recognize data with newly-added class labels during the learning process, and thus exhibits excellent scalability. Experimental results on both synthetic and real datasets confirm the effectiveness of our propose method.

Index Terms— Linear discriminant analysis, least squares solution, rank-one update, concept drift

1. INTRODUCTION

Linear discriminant analysis (LDA) [1] is a supervised dimension reduction technique, which projects the data into a low-dimensional linear subspace via eigen-analysis, and thus provides an improved separation between data from different classes. However, it is computationally expensive to perform eigen-analysis especially for large-scale problems. Various matrix factorization techniques (e.g., QR factorization, singular value decomposition (SVD) or generalized SVD) have been proposed to reduce the computational complexity or to alleviate the singularity problem in LDA [4, 5, 6], but these methods only work in *batch* mode, which makes the generalization to large-scale or streaming data problems very difficult. Therefore, how to learn the LDA model in *online* mode to recognize newly received data with changes in distribution is very challenging yet practical for real-world applications such as email spam filtering and malicious URL detection.

Recently, least-squares LDA (LSLDA) is proposed in [2, 3], which utilizes a pre-determined class indicator matrix and approaches the LDA problem as solving a multiple linear regression task. Since there is no need to perform eigen-analysis, LSLDA is preferable in solving incremental learning problems [7], or problems which require additional constraints (e.g. sparsity) in their LDA solutions [3]. In this paper, we propose an improved least-squares technique to update the LSLDA for recognizing newly received data instance with a known or a new class label. Our LSLDA does not require eigen-analysis, and there is no need to store the data covariance matrix during its learning process either.

In order to effectively extract the information from newly received data with gradual or abrupt changes in distribution, i.e. the data distribution might change over time in an unforeseen way, we approach the LSLDA problem as the learning of concept-drifting data with an online setting. We note that most prior LDA or LSLDA methods did not address this issue. For example, previous incremental LDA algorithms treat all data instances (existing and newly received ones) equally important (e.g., [9, 7]), so they cannot handle streaming data with any distribution changes. Although Kuncheva and Plumpton [8] proposed an incremental learning scheme to update their LDA, their approach needs to keep/update the data covariance matrix, and thus their LDA model is not preferable for high-dimensional problems. In this paper, we propose a rank-one update method for LSLDA. We aim to reduce computation complexity when updating the LSLDA. Moreover, we introduce the ability of recognizing concept-drifting data or those with new class labels to our LSLDA. Results from a variety of experiments will be presented to verify the effectiveness of our proposed method.

2. OUR PROPOSED LSLDA WITH CONCEPT DRIFT

2.1. Least-Squares Formulations for LDA

Given n data instances from K different classes in a p-dimensional space, the LDA solution $\mathbf{W}^{LDA}$ satisfies

$$\mathbf{\Sigma}_b\mathbf{W}^{LDA} = \mathbf{\Lambda}\mathbf{\Sigma}_t\mathbf{W}^{LDA}, \tag{1}$$

where $\mathbf{\Sigma}_t$ and $\mathbf{\Sigma}_b$ are total and between-class covariance matrices, respectively. In [2], the above formulation is converted into a multiple linear regression (MLR) problem, and the least-squares solution $\mathbf{W}^{LR}$ (will be denoted as $\mathbf{W}$ in the remaining of this paper for simplicity) can be calculated as

$$\mathbf{W}^{LR} = [\mathbf{W}^{LDA}\mathbf{D}, 0]\mathbf{Q}^\top = (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{Y}. \tag{2}$$

In (2), $\mathbf{D}$ is a diagonal matrix, $\mathbf{Q}$ is an orthogonal one, and $\mathbf{A} = [\mathbf{x}_1^\top; \mathbf{x}_2^\top; \cdots; \mathbf{x}_n^\top] \in \mathbb{R}^{n\times p}$ is a centered data matrix, where each row $\mathbf{x}_i$ indicates an instance with global mean removed. The matrix $\mathbf{Y} = [\mathbf{y}_1^\top; \mathbf{y}_2^\top; \cdots; \mathbf{y}_n^\top] \in \mathbb{R}^{n \times K}$ in (2) is a class indicator matrix; for each $\mathbf{y}_i$, only its j-th entry will be assigned a non-zero value if $\mathbf{x}_i$ belongs to the j-th class (see [2] for details). It is shown that $\mathbf{D}$ would be an identity matrix under a mild condition which typically occurs when $n \ll p$ [2]. Recently, Liu *et al.* [7] redefined $\mathbf{Y}$ and proposed an incremental LSLDA, which is more efficient than the one in [2] but cannot handle concept-drifting data.

In this paper, we focus on a rank-one update approach for LSLDA, which can be operated in either batch or online mode. To reduce the memory requirement and computational load for our LSLDA, we propose a simplified coding scheme to define the class indicator matrix $\mathbf{Y}$ in (2), and we have

$$\mathbf{Y}(i, c) = \begin{cases} 1 & \text{if instance } \mathbf{x}_i \in c \text{ (class label),} \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

We note that the above setting accelerates the updating processing for our LSLDA, since there is no need to update this matrix as shown in the next subsection. The following proposition shows that our LDA solution will be equivalent to those derived in [2] and [7], except for a constant normalization term which can be easily calculated.

Proposition 1 *Suppose that $\mathbf{y}_c$ is the c_{th} column of $\mathbf{Y}$, and $y_{ic} \in \{s, t\}$ means that the i_{th} entry in $\mathbf{y}_c$ will be s if its corresponding instance is from class c, or it equals t otherwise. Same remarks apply to $\mathbf{y}'_c$ and $y'_{ic} \in \{s', t'\}$. Let $\mathbf{A}$ be the centered data matrix, and $\mathbf{w}_c$ and $\mathbf{w}'_c$ be the least squares solutions with $\mathbf{y}_c$ and $\mathbf{y}'_c$, respectively. The two LDA solutions $\mathbf{w}_c$ and $\mathbf{w}'_c$ will satisfy*

$$\mathbf{w}_c = \frac{t - s}{t' - s'} \mathbf{w}'_c. \quad (4)$$

As a result, we observe the relationship between different LSLDA solutions:

$$\mathbf{w}_c = \frac{1}{\sqrt{n_c}} \mathbf{w}_{2c} = \frac{1}{\sqrt{n_c n}} \mathbf{w}_{1c} \text{ for } c = 1, \ldots, k, \quad (5)$$

where $\mathbf{w}_c$ is the solution using our simplified indicator matrix $\mathbf{Y}$, $\mathbf{w}_{2c}$ and $\mathbf{w}_{1c}$ are those derived in [2] and [7], respectively (n_c is the number of instances in class c). Due to space limit, we will provide detailed proofs in future longer versions. Nevertheless, this verifies the use of our indicator matrix for producing a valid LDA subspace. If n_c is very different between classes, our LDA solutions can be normalized by a factor $\sqrt{n_c}$ or $\sqrt{n_c n}$ accordingly if necessary.

2.2. Rank-One Update for LSLDA

From (2), it is clear that the calculation of $(\mathbf{A}^\top \mathbf{A})^{-1}$ is the key in LSLDA. Let $\mathbf{A}_i$ and $\mathbf{A}_{i+1}$ be current data matrix and the one receiving a new instance $\mathbf{x}_{i+1}$. The updated LSLDA solution is calculated as $\mathbf{W}_{i+1} = (\mathbf{A}_{i+1}^\top \mathbf{A}_{i+1})^{-1} \mathbf{A}_{i+1}^\top \mathbf{Y}_{i+1}$.

Algorithm 1: Our LSLDA with concept drift

```
Input: $\mathbf{x}_{i+1}, y_{i+1}$, the current mean $\mu_i$, $\tilde{\mathbf{W}}_i$, $\beta$,
       $\tilde{\mathbf{T}} = [\tilde{\mathbf{t}}_0 \cdots \tilde{\mathbf{t}}_{i-2}, \tilde{\mathbf{t}}_{i-1}]$, $\tilde{\mathbf{s}} = [\tilde{s}_0 \cdots \tilde{s}_{i-2}, \tilde{s}_{i-1}]$.
Output: The resulting LDA solution $\tilde{\mathbf{W}}_{i+1} \in \mathbb{R}^{p \times k}$.
begin
    $\mu_{i+1} \leftarrow \frac{i}{i+1}\mu_i + \frac{1}{i+1}\mathbf{x}_{i+1}$
    $\tilde{\mathbf{x}}_{i+1} \leftarrow \mathbf{x}_{i+1} - \mu_{i+1}$
    $\tilde{\mathbf{y}}_{i+1}^\top \leftarrow \mathbf{0} \in \mathbb{R}^{1 \times k}$
    if $y_{i+1} \in c$ then
        $\tilde{\mathbf{y}}_{i+1}(c) = 1$
    $\tilde{\mathbf{t}}_i \leftarrow \tilde{\mathbf{x}}_{i+1} - \frac{\tilde{\mathbf{t}}_0 \tilde{\mathbf{t}}_0^\top}{\tilde{s}_0}\tilde{\mathbf{x}}_{i+1} - \ldots - \frac{\tilde{\mathbf{t}}_{i-1}\tilde{\mathbf{t}}_{i-1}^\top}{\tilde{s}_{i-1}}\tilde{\mathbf{x}}_{i+1}$
    $\tilde{s}_i \leftarrow \beta^{i+1} + \mathbf{x}_{i+1}^\top \tilde{\mathbf{t}}_i$
    $\tilde{\mathbf{W}}_{i+1} \leftarrow \tilde{\mathbf{W}}_i + \frac{\tilde{\mathbf{t}}_i}{\tilde{s}_i}(\tilde{\mathbf{y}}_{i+1}^\top - \tilde{\mathbf{x}}_{i+1}^\top \tilde{\mathbf{W}}_i)$
```

To avoid computing $(\mathbf{A}_{i+1}^\top \mathbf{A}_{i+1})^{-1}$ whenever a new data point is received, and to utilize the information from the current $(\mathbf{A}_i^\top \mathbf{A}_i)^{-1}$, the Woodbury matrix identity [10] can be applied. The solution $\mathbf{W}_{i+1}$ can be updated as follows:

$$\begin{aligned} \mathbf{W}_{i+1} &= (\mathbf{A}_{i+1}^\top \mathbf{A}_{i+1})^{-1} \mathbf{A}_{i+1}^\top \mathbf{Y}_{i+1} \\ &= \mathbf{W}_i + \frac{\mathbf{M}_i^{-1}\mathbf{x}_{i+1}}{1 + \mathbf{x}_{i+1}^\top \mathbf{M}_i^{-1}\mathbf{x}_{i+1}}(\mathbf{y}_{i+1}^\top - \mathbf{x}_{i+1}^\top \mathbf{W}_i), (6) \end{aligned}$$

where $\mathbf{M}_i = \mathbf{A}_i^\top \mathbf{A}_i$. We note that this process needs to keep $\mathbf{A}_i^\top \mathbf{A}_i \in \mathbb{R}^{p \times p}$ in each iteration. While this will not cause problems for large-scale applications with low-dimensional data ($n \gg p$), this process prohibits LSLDA updates when the dimensionality p of data is large.

To address the above problem, we choose to update the vector $\mathbf{M}_i^{-1}\mathbf{x}_{i+1}$ directly, and our recursive rank-one update process is implemented in the following way:

$$\begin{aligned} \mathbf{M}_i^{-1}\mathbf{x}_{i+1} &= \mathbf{M}_{i-1}^{-1}\mathbf{x}_{i+1} - \frac{\mathbf{M}_{i-1}^{-1}\mathbf{x}_i\mathbf{x}_i^\top \mathbf{M}_{i-1}^{-1}}{1 + \mathbf{x}_i^\top \mathbf{M}_{i-1}^{-1}\mathbf{x}_i}\mathbf{x}_{i+1} \\ &\vdots \\ &= \mathbf{M}_0^{-1}\mathbf{x}_{i+1} - \sum_{k=0}^{i-1} \frac{\mathbf{M}_k^{-1}\mathbf{x}_{k+1}\mathbf{x}_{k+1}^\top \mathbf{M}_k^{-1}}{1 + \mathbf{x}_{k+1}^\top \mathbf{M}_k^{-1}\mathbf{x}_{k+1}}\mathbf{x}_{i+1} \\ &= \mathbf{x}_{i+1} - \sum_{k=0}^{i-1} \frac{\mathbf{t}_k \mathbf{t}_k^\top}{s_k}\mathbf{x}_{i+1} = \mathbf{t}_i, \quad (7) \end{aligned}$$

where $\mathbf{t}_i = \mathbf{M}_i^{-1}\mathbf{x}_{i+1}$, $s_i = 1+\mathbf{x}_{i+1}^\top \mathbf{M}_i^{-1}\mathbf{x}_{i+1}$, and $\mathbf{M}_0^{-1} = \mathbf{I}$. The update of $\mathbf{W}$ is thus reformulated as follows:

$$\mathbf{W}_{i+1} = \mathbf{W}_i + \frac{\mathbf{t}_i}{1 + \mathbf{x}_{i+1}^\top \mathbf{t}_i}(\mathbf{y}_{i+1}^\top - \mathbf{x}_{i+1}^\top \mathbf{W}_i). \quad (8)$$

From (7) and (8), we see that $\mathbf{t}_i$ can be calculated by $\{\mathbf{t}_0, \mathbf{t}_1, \ldots, \mathbf{t}_{i-1}]\}$ from previous iterations, and thus we only need to keep i of these p-dimensional $\mathbf{t}_i$ vectors when computing $\mathbf{W}_{i+1}$. This avoids the limitation in prior LSLDA methods which require to store a $p \times p$ data covariance matrix while the data dimensionality is very large (i.e. $p \gg n$).

2.3. LSLDA with Concept Drift

For the mining of concept-drifting data, the LSLDA needs to extract the information from newly received data with gradual or abrupt distribution changes, while suppressing the influence from outdated data. To add this ability to our rank-one update approach for LSLDA, we introduce a forgetting factor $\beta < 1$ into $\mathbf{A}_i^\top \mathbf{A}_i$, i.e. we have $\tilde{\mathbf{M}}_{i+1} = \tilde{\mathbf{A}}_{i+1}^\top \tilde{\mathbf{A}}_{i+1} = \beta \tilde{\mathbf{M}}_i + \mathbf{x}_{i+1}\mathbf{x}_{i+1}^\top$ and $\tilde{\mathbf{N}}_{i+1} = \tilde{\mathbf{A}}_{i+1}^\top \tilde{\mathbf{Y}}_{i+1} = \beta \tilde{\mathbf{N}}_i + \mathbf{x}_{i+1}\mathbf{y}_{i+1}^\top$. It can be seen that we suppress the contribution of outdated data by a factor of β in updating the outer product matrix, and this provides a new formulation of $\mathbf{W}$,

$$\tilde{\mathbf{W}}_{i+1} = \tilde{\mathbf{W}}_i + \frac{\tilde{\mathbf{M}}_i^{-1}\mathbf{x}_{i+1}}{\beta + \mathbf{x}_{i+1}^\top \tilde{\mathbf{M}}_i^{-1}\mathbf{x}_{i+1}}(\mathbf{y}_{i+1}^\top - \mathbf{x}_{i+1}^\top \tilde{\mathbf{W}}_i). \quad (9)$$

Following (7), we further simplify the numerator of the second term in the above equation by specifying

$$\begin{aligned}\tilde{\mathbf{M}}_i^{-1}\mathbf{x}_{i+1} &= \beta^{-i}(\mathbf{x}_{i+1} - \frac{\tilde{\mathbf{t}}_0\tilde{\mathbf{t}}_0^\top}{\tilde{s}_0}\mathbf{x}_{i+1} - \cdots - \frac{\tilde{\mathbf{t}}_{i-1}\tilde{\mathbf{t}}_{i-1}^\top}{\tilde{s}_{i-1}}\mathbf{x}_{i+1}) \\ &= \beta^{-i}\tilde{\mathbf{t}}_i, \quad (10)\end{aligned}$$

where $\tilde{s}_i = \beta^{i+1} + \beta^i \mathbf{x}_{i+1}^\top \tilde{\mathbf{M}}_i^{-1}\mathbf{x}_{i+1} = \beta^{i+1} + \mathbf{x}_{i+1}^\top \tilde{\mathbf{t}}_i$. Recall that $\tilde{\mathbf{W}}$, $\tilde{\mathbf{M}}$, and $\tilde{\mathbf{t}}$ denote the solutions with concept drift, and we have $\tilde{\mathbf{M}}_0 = \mathbf{I}$ and $\tilde{\mathbf{W}}_0 = \mathbf{0}$ for initialization. Therefore, the final solution of our LSLDA via rank-one update can be expressed as follows:

$$\tilde{\mathbf{W}}_{i+1} = \tilde{\mathbf{W}}_i + \frac{\tilde{\mathbf{t}}_i}{\tilde{s}_i}(\mathbf{y}_{i+1}^\top - \mathbf{x}_{i+1}^\top \tilde{\mathbf{W}}_i). \quad (11)$$

The pseudo code of our LSLDA is described in Alg. 1.

3. EXPERIMENTAL RESULTS

3.1. Recognizing Concept-Drifting Data

3.1.1. Synthetic dataset

We synthesize a 2D dataset with a total of 2000 instances for binary classification. The first 1000 instances are sampled from the following multivariate normal distributions (500 instances for each class): $\mathbf{x}_+ \sim N(\left[1, 1\right], 0.8 \times \mathbf{I})$ and $\mathbf{x}_- \sim N(\left[-1, -1\right], 0.8 \times \mathbf{I})$ (see Figure 1(a) or (c)), and the remaining 1000 instances are sampled from a different distribution setting: $\mathbf{x}'_+ \sim N(\left[1, -1\right], 0.8 \times \mathbf{I})$ and $\mathbf{x}'_- \sim N(\left[-1, 1\right], 0.8 \times \mathbf{I})$ (see in Figure 1(b) or (d)).

The black lines in Figures 1(a) and (b) are standard LSLDA solutions, and those in Figures 1(c) and (d) are produced by our LSLDA. Comparing Figures 1(a) and (c), there is negligible difference between the two solutions, since both LSLDA use the first half of the data to calculate their solutions. On the other hand, the two solutions in Figures 1(b) and (d) are very different. The solution in Figure 1(b) is produced by the standard LSLDA, which treats all 2000 (prior

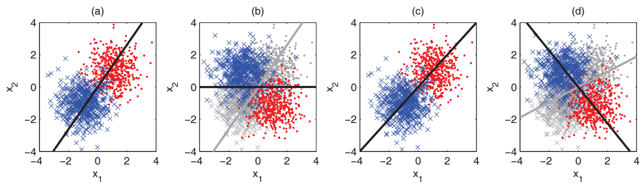

Fig. 1. Comparisons of LDA solutions (black lines): Standard LSLDA on (a) the first 1000 instances, (b) the entire dataset (with 2000 points); ours on (a) the first 1000 instances, and (d) the entire dataset.

and existing) instances equally important, and thus it fails to capture the behavior (distribution) of the newly received data. In Figure 1(d), it is clear that our LSLDA is able to recognize concept-drifting data, since it aims to recognize the newly received instances, while the separation between the previous 1000 points is not of major concern (we use $\beta = 0.99$).

3.1.2. Concept drift in real-world data

We now perform leaning tasks of concept drift using the `pendigits` dataset [1]. The concept drift scheme is achieved by changing the classes of interest during the learning process. We load the training data from digits 1 and 3 in a streaming fashion to update our initial LSLDA, and to recognize the test data from these two classes simultaneously. Once all training samples from these two classes are used, we change the classes of interest to digits 2 and 4 and use their training data instead (also in a streaming way), and our LSLDA will recognize the associated test data from these two classes. This is to see how quickly our LSLDA learns from concept-drifting data. To evaluate the performance, we calculate the recognition rate using test data from the corresponding classes (*not* in a steaming fashion) whenever a training sample is received during the above learning process.

Figure 2 shows recognition results with $\beta = 1, 0.99, 0.95$ and 0.9. When $\beta = 1$ (the dotted curve), our LSLDA turns into standard LSLDA without concept drift, thus it requires the longest time to recover the drop of recognition rate when concept drift occurs. While a smaller β value (e.g. the green curve $\beta = 0.9$) is used, our LSLDA focus more on recently received data and quickly updates its solution right after concept drift occurs. However, we note that such a solution will be less stable than those with larger β values as expected.

3.2. Recognizing Data with New Class Labels

One of the strengths of our LSLDA is the ability to recognize newly added data with unseen class labels. To achieve this, we extend our class indicator matrix in (3) by adding an additional column for the new class label of interest. Thus,

[1]The `pendigits` dataset is available at `http://archive.ics.uci.edu/ml/`

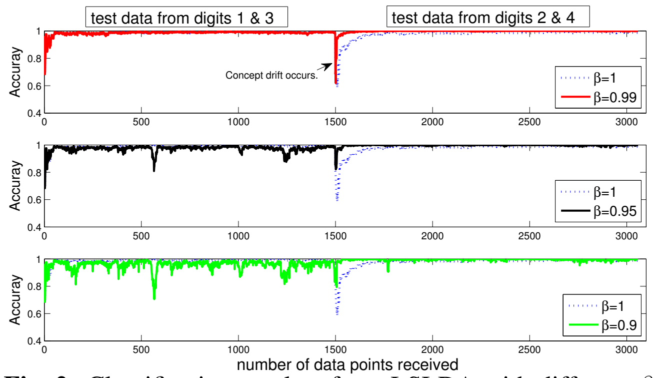

Fig. 2. Classification results of our LSLDA with different β values in the `pendigits` dataset. Note that $\beta = 1$ results in the LSLDA proposed in [7].

when receiving a new data instance with a new class label, our LSLDA solution will be calculated as

$$\tilde{\mathbf{W}}_{i+1} = \tilde{\mathbf{W}}_i + \frac{\tilde{\mathbf{t}}_i}{\beta^{i+1}\mathbf{x}_{i+1}^{\top}\tilde{\mathbf{t}}}(\tilde{\mathbf{y}}_{i+1}^{\top} - \mathbf{x}_{i+1}^{\top}\tilde{\mathbf{W}}_i), \tag{12}$$

where $\tilde{\mathbf{W}}_i = [\tilde{\mathbf{W}}_i, \mathbf{0}] \in \mathbb{R}^{p\times(K+1)}$, and $\tilde{\mathbf{y}}_{i+1}^{\top} \in \mathbb{R}^{1\times(K+1)}$.

To evaluate the performance of our LSLDA for such cases, we consider the `medline` dataset [2] and use the data from the first 3 classes (i.e. classes 1-3) to design the initial LSLDA model. Samples from classes 4-5 are considered as newly added data to be recognized, but they are not seen in the first half of the learning process. In other words, we extract 50% of the training data from classes 1-3, and use them to design the associated LDA model beforehand. When the experiment starts, we sequentially add the remaining training data from *all* classes 1-5 to update our LSLDA. Recognition performance on the test data from all classes is evaluated, as shown in Figure 3. Three curves in the figure represent accuracy of test data from classes 1-3 (blue), classes 4-5 (red), and the average one (black), respectively. The horizontal axis indicates the number of the training data received, and the vertical axis shows the classification rate for different cases. From these results, we see that the information of newly added data is successfully and gradually adopted into our LSLDA, and this verifies the effectiveness of our LSLDA in recognizing data with new class labels.

4. CONCLUSION

We proposed a rank-one update approach for LSLDA with concept drift, which efficiently updates the LDA subspace in an incremental fashion with the use of our class indicator matrix. This matrix is equivalent to more complex ones which were proposed in prior LSLDA models, while our LSLDA does not require the store the entire data covariance matrix.

[2]The `medline` dataset is available at `http://www.cc.gatech.edu/~hpark/data.html`.

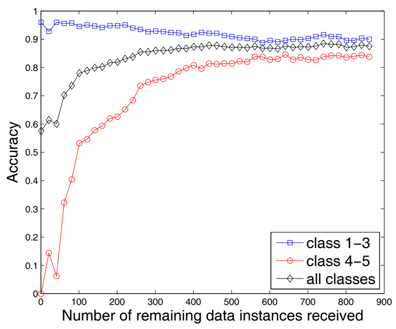

Fig. 3. Classification results of newly added classes.

Our experiments confirmed that our LSLDA is able to handle streaming data with gradual or abrupt distribution changes. Moreover, recognition of newly added data was achieved by our LSLDA, which confirms the generalization of our method to practical online learning applications.

5. ACKNOWLEDGEMENTS

This work is supported in part by the National Science Council of Taiwan under NSC99-2221-E-001-020. We also thank Prof. Yuh-Jye Lee from National Taiwan University Science and Technology for his valuable feedback on our LSLDA.

6. REFERENCES

[1] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, 2009.

[2] J. Ye, “Least squares linear discriminant analysis,” *Int’l Conf. Machine Learning*, 2007.

[3] L. Sun, S. Ji, and J. Ye, “A least squares formulation for a class of generalized eigenvalue problems in machine learning,” *Int’l Conf. Machine Learning*, 2009.

[4] P. Howland et al., “Structure preserving dimension reduction for clustered text data based on the generalized SVD,” *SIAM Journal on Matrix Analysis and Applications*, 2003.

[5] J. Ye, R. Janardan, C. Park, and H. Park, “An optimization criterion for generalized discriminant analysis on undersampled problems,” *IEEE Trans. PAMI*, 2004.

[6] J. Ye and Q. Li, “LDA/QR: an efficient and effective dimension reduction algorithm and its theoretical foundation,” *Pattern Recognition*, 2004.

[7] L.-P. Liu, Y. Jiang, and Z.-H. Zhou, “Least squares incremental LDA,” *IEEE Int’l Conf. Data Mining*, 2009.

[8] L. I. Kuncheva and C. O. Plumpton, “Adaptive learning rate for online linear discriminant classifiers,” *IAPR Workshop on SSPR & SPR*, 2008.

[9] H. Zhao and P.-C. Yuen, “Incremental LDA for face recognition,” *IEEE TSMC*, 2008.

[10] N. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, 2002.