INCREMENTAL REGULARIZED LEAST SQUARES FOR DIMENSIONALITY REDUCTION OF LARGE-SCALE DATA

XIAOWEI ZHANG[†], LI CHENG[‡], DELIN CHU[§], LI-ZHI LIAO[¶], MICHAEL K. NG[¶], AND ROGER C. E. TAN[§]

Abstract. Over the past a few decades, much attention has been drawn to large-scale incremental data analysis, where researchers are faced with huge amount of high-dimensional data acquired incrementally. In such a case, conventional algorithms that compute the result from scratch whenever a new sample comes are highly inefficient. To conquer this problem, we propose a new incremental algorithm IRLS that incrementally computes the solution to the regularized least squares (RLS) problem with multiple columns on the right-hand side. More specifically, for a RLS problem with c ($c > 1$) columns on the right-hand side, we *update* its unique solution by solving a RLS problem with single column on the right-hand side whenever a new sample arrives, instead of solving a RLS problem with c columns on the right-hand side from scratch. As a direct application of IRLS, we consider the supervised dimensionality reduction of large-scale data and focus on linear discriminant analysis (LDA). We first propose a new batch LDA model that is closely related to RLS problem, and then apply IRLS to develop a new incremental LDA algorithm. Experimental results on real-world datasets demonstrate the effectiveness and efficiency of our algorithms.

Key words. Incremental regularized least squares, linear discriminant analysis, supervised dimensionality reduction, LSQR.

AMS subject classifications. 65F10, 65F22, 65F50, 68T05

1. Introduction.

Regularized least squares (RLS) with multiple columns on the right-hand side, which is also known as Tikhonov regularization [32] or multivariate Ridge Regression (RR) [1], is one of the most widely used methods for statistical estimation and regularization of ill-posed problems. Given matrices $A \in \mathbb{R}^{n\times d}$ and $B \in \mathbb{R}^{n\times c}$ with $c > 1$, RLS solves the problem

$$\min_{X\in\mathbb{R}^{d\times c}} \|AX - B\|_F^2 + \lambda^2\|X\|_F^2, \tag{1.1}$$

where $\lambda > 0$ is a regularization parameter. Many problems in practical applications [9, 23, 31] can be formulated as (1.1). In these applications, A usually denotes the data matrix with n samples from d-dimensional space and B denotes the corresponding label information in classification or responses in regression. RLS problem (1.1) has been extensively studied in fields as diverse as numerical linear algebra, statistics, machine learning and data mining. It has an unique explicit solution X^* given by

$$X^* = (A^TA + \lambda^2 I_d)^{-1}A^TB = A^T(AA^T + \lambda^2 I_n)^{-1}B. \tag{1.2}$$

However, computing (1.2) costs $O(cnd\min\{n, d\})$ flops [18], which can be very slow for large-scale high-dimensional data where both n and d are huge. To keep pace with the growing scale of data in the modern age of "Big Data", numerous iterative approaches have been proposed, including both deterministic methods like LSQR [26] and LSMR [16] and randomized methods like Blendenpik [2], randomized extended Kaczmarz [41] and LSRN [25]. Despite their effectiveness, all these iterative approaches assume that data are given at one time.

Over the past a few decades, due to the advances in data collection and storage capabilities, there has been an explosion in the amount of high-dimensional data. Moreover, in many applications data are acquired incrementally. On the one hand, when a new data is added, the statistical model used to analyse the data may not change much and the solution should be very close to the old one. On the other hand, even for efficient learning algorithms, computing the solution from scratch whenever a new sample is acquired is computationally expensive and requires huge amount of memory as more data are collected, which makes them infeasible for large-scale high-dimensional data. Consequently, it is imperative to develop algorithms that can run in an incremental fashion to accommodate the new data. More specifically, we need to design

[†]Bioinformatics Institute, A*STAR, Singapore (zhangxw@bii.a-star.edu.sg).

[‡]Bioinformatics Institute, A*STAR, Singapore and School of Computing, National University of Singapore, Singapore (chengli@bii.a-star.edu.sg).

[§]Department of Mathematics, National University of Singapore, Singapore 119076 (matchudl@nus.edu.sg, scitance@nus.edu.sg).

[¶]Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong (liliao@hkbu.edu.hk, mng@math.hkbu.edu.hk).

an algorithm satisfying the following criteria [29]: It is able to learn additional information from new data without forgetting previously acquired knowledge, and it can accommodate new data from new sources (e.g. new classes). Compared with batch learning, incremental learning has the advantages of being adaptive and responsive, and, different from batch learning, incremental learning deals with theory revision at each time step instead of theory creation from scratch. Another challenge in dealing with large-scale high-dimensional data is the "curse of dimensionality", due to which the effectiveness of conventional learning methods deteriorates exponentially as the dimensionality increases. A common way to overcome this problem is to reduce the dimension of the data, see [6] for a review of various dimensionality reduction methods.

In this paper, we attempt to tackle both problems by proposing a new incremental algorithm for dimensionality reduction of large-scale data. Specifically, we are interested in designing incremental algorithms for RLS with multiple columns on the right-hand side so that, when a new sample is added, we can compute the solution of (1.1) with $n + 1$ data points by updating the solution of (1.1) with n data points, instead of computing it from scratch. As an application, we then apply this new incremental RLS method to linear discriminant analysis (LDA) [15, 19], which is one of the most powerful tools for supervised dimensionality reduction, resulting a novel incremental LDA algorithm for dimensionality reduction of large-scale data.

1.1. Our contributions. The contributions of this paper are three folds: (1) We propose a novel incremental algorithm for RLS problem (1.1) and name it IRLS. To compute the new solution when a new sample arrives, IRLS updates the old one by solving a RLS problem with single column on the right-hand side, instead of solving the RLS problem with c columns on the right-hand side. IRLS has two properties: (i) The new solution resulted from updating the old one is the exact solution of RLS problem (1.1). (ii) It can handle large-scale data with $\min\{n, d\} \gg 1$ and both cases $n \geq d$ and $n \leq d$. The advantages of IRLS over conventional RLS are more prominent when c is large or when the data matrix A is sparse. (2) We propose a new batch LDA algorithm named LDADL. This algorithm is designed for a new batch LDA model formulated as a RLS problem (1.1) with A being the data matrix and B recording the class information. We prove that the solution to the new LDA model converges to the LDA solution as the regularization parameter λ converges to zero. Moreover, we provide theoretical results connecting our LDA model with regularized LDA (RLDA) in [39] and spectral regression discriminant analysis (SRDA) in [7]. Different from existing least squares LDA models [7, 28, 36], our model is derived from the perspective of *data lifting* and highly adoptable to incremental LDA. (3) We propose a new incremental LDA algorithm named ILDADL by applying the newly proposed IRLS method to the new batch LDA model. This new incremental LDA algorithm can handle new samples from both existing classes and new classes. In addition, the effectiveness and efficiency of the proposed methodologies are demonstrated on extensive real-world datasets.

1.2. Related work. Limited work has been done in the area of incremental least squares (LS). In [4], Bertsekas considered incremental methods for nonlinear LS problem. However, incremental methods discussed in [4] have quite different meaning from that of the current paper in that they solve nonlinear LS problems iteratively and each iterate is updated from the previous iterate by incrementally processing samples in A. In [3] and [8] matrix A is assumed to have full column rank. Baboulin et al. [3] proposed a parallel incremental solver for dense LS problem by updating the QR factorization of A. Cassioli et al. [8] proposed an incremental algorithm based on updating the inverse of A^TA. However, the computations of QR factorization of A and the inverse of A^TA are time-consuming and need huge memory when the data matrix A is very big. A closely related topic is the online ridge regression [34] that learns one sample at a time and makes prediction for the sample.

There is some recent work on incremental LDA algorithms [27, 37, 22, 40, 24, 21]. Ye et al. [37] proposed IDR/QR by applying regularized LDA in a projected subspace spanned by the centroid matrix. The main limitation of IDR/QR is that much information is lost in the first projection and some useful components are discarded in the updating process, resulting in inferior performance as pointed out in [7]. Pang et al. [27] introduced a way of updating the between-class and within-class scatter matrices. Unfortunately, the updating of the discriminative transformation is not mentioned, which is one of the most difficult steps in incremental LDA. In [22, 21], the authors presented a different incremental LDA algorithm, which we denote ILDA/SSS, by applying the concept of sufficient spanning set approximation to update the scatter matrices. The main drawback of ILDA/SSS is that three eigenvalue thresholds need to be given in advance. In [40], the authors proposed GSVD-ILDA which is an incremental version of LDA/GSVD [20]. As illustrated in [24], both ILDA/SSS and GSVD-ILDA suffer from a common problem, that is, it is difficult to strike a balance between the classification performance and the computational efficiency. As an improvement, [24] adopted the idea of LS-LDA [36] and proposed a new incremental algorithm named LS-ILDA via updating the pseudo-inverse of the centred data matrix. In [13], the authors developed ILDA/QR that computes the LDA solution by updating the economical QR factorization of the data matrix. However, [13] assumes that all data points are linearly independent, which does not hold when more data points are collected.

1.3. Notations. We use lowercase letters (e.g. c, d, n, etc.) or greek letters (e.g. α, λ, γ, etc.) to denote scalars, use bold letters (e.g. $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{e}$, etc.) to denote vectors, and use uppercase letters (e.g. A, B, X, etc.) to denote matrices. All vectors are assumed to be column vectors. I_n is the $n \times n$ identity matrix; $\mathbf{0}$ is the matrix of zeros whose size should be clear from the context. We use trace$(\cdot)$ and rank$(\cdot)$ to denote the trace and the rank of matrix, respectively. We also adopt the following conventions in the rest of this paper: *For any variable X, its updated version after the insertion of new samples is denoted by $\tilde{X}$.* For example, the data matrix A becomes $\tilde{A}$ after the insertion of a new sample $\boldsymbol{a}$. We use the "$*$" superscript (e.g. G^*, X^*, etc.) to denote optimal solutions to optimization problems, use the underline of letters (e.g. $\underline{A}$, $\underline{G}$, $\underline{\boldsymbol{a}}_j$, etc.) to denote matrices or vectors obtained by appending *one row at the bottom or one column at the right end*, and use the "$\hat{\ }$" accent (e.g. $\hat{X}$, $\hat{\boldsymbol{a}}$) to denote matrices or vectors obtained by appending *multiple rows* at the bottom.

The rest of the paper is organized as follows. In Section 2, we provide a detailed derivation of the IRLS algorithm. In Section 3, we give a brief review of the LDA problem. In Section 4, we propose a new fast batch LDA algorithm. In Section 5, we develop a new incremental LDA algorithm that is the exact incremental version of our new batch LDA algorithm. In Section 6, we provide extensive experimental results using real-world datasets. Finally, we provide some concluding remarks in Section 7.

2. Incremental regularized least squares (IRLS). In this section, we derive the incremental algorithm IRLS for RLS problem (1.1).

Let $A_\lambda = \begin{bmatrix} A & \lambda I_n \end{bmatrix}$ and $\hat{X} = \begin{bmatrix} X \\ W \end{bmatrix}$ where $X \in \mathbb{R}^{d\times c}$ and $W \in \mathbb{R}^{n\times c}$, then the unique RLS solution X^* given in (1.2) is closely related to the minimum Frobenius norm solution of the following LS problem:

$$\min\left\{ \|A_\lambda \hat{X} - B\|_F^2 \;\middle|\; \hat{X} \in \mathbb{R}^{(d+n)\times c} \right\}. \tag{2.1}$$

The detailed description is provided in Lemma 2.1.

Lemma 2.1. *Let $\hat{X}^* = \begin{bmatrix} X^* \\ W^* \end{bmatrix}$ be the minimum Frobenius norm solution to the LS problem (2.1), then X^* is the unique solution to the RLS problem (1.1).*

Proof. Since A_λ has full row rank and $\hat{X}^*$ is the minimum Frobenius norm solution to the LS problem (2.1), it follows that $\hat{X}^*$ is uniquely determined by

$$\hat{X}^* = A_\lambda^\dagger B = A_\lambda^T (A_\lambda A_\lambda^T)^{-1} B = A_\lambda^T (AA^T + \lambda^2 I_n)^{-1} B,$$

where $A_\lambda^\dagger$ denotes the Moore-Penrose pseudo-inverse [18] of A_λ. Therefore, we have $X^* = A^T(AA^T + \lambda^2 I_n)^{-1}B$, which is the unique solution of the RLS problem (1.1). ∎

Now, we consider the problem of computing $\hat{X}^*$ using the QR factorization of A_λ^T. Let

$$A_\lambda^T = QR = \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} R \tag{2.2}$$

be the economical QR factorization of A_λ^T, where $R \in \mathbb{R}^{n\times n}$ is an upper-triangular matrix, $Q_1 \in \mathbb{R}^{d\times n}$ and $Q_2 \in \mathbb{R}^{n\times n}$. Since A_λ^T has full column rank, R is nonsingular and it is easy to verify that

$$X^* = Q_1 R^{-T} B. \tag{2.3}$$

This formula will be frequently used as the theoretical base in IRLS to update X^* when new data are inserted.

Let $\boldsymbol{a} \in \mathbb{R}^d$ denote the newly added sample and define $\hat{\boldsymbol{a}} = \begin{bmatrix} \boldsymbol{a} \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{d+n}$, and $\boldsymbol{b} \in \mathbb{R}^c$ denote the corresponding right-hand side, we have

$$\tilde{A}_\lambda^T = \begin{bmatrix} A_\lambda^T & \hat{\boldsymbol{a}} \\ \mathbf{0} & \lambda \end{bmatrix} \quad \text{and} \quad \tilde{B} = \begin{bmatrix} B \\ \boldsymbol{b}^T \end{bmatrix}.$$

Suppose the economical QR factorization of A_λ^T is given by (2.2) and let

$$\boldsymbol{r} = Q^T \hat{\boldsymbol{a}} = Q_1^T \boldsymbol{a}, \quad \boldsymbol{q} = \begin{bmatrix} \boldsymbol{q}_1 \\ \boldsymbol{q}_2 \end{bmatrix} = \frac{(I - QQ^T)\hat{\boldsymbol{a}}}{\tau},$$

where $\boldsymbol{q}_1 \in \mathbb{R}^d$, $\boldsymbol{q}_2 \in \mathbb{R}^n$ and $\tau = \sqrt{\|(I - QQ^T)\hat{\boldsymbol{a}}\|^2 + \lambda^2} > 0$, we have

$$\tilde{A}_\lambda^T = \begin{bmatrix} QR & Q\boldsymbol{r} + (I - QQ^T)\hat{\boldsymbol{a}} \\ \boldsymbol{0} & \lambda \end{bmatrix} = \begin{bmatrix} Q & \boldsymbol{q} \\ \boldsymbol{0} & \frac{\lambda}{\tau} \end{bmatrix} \begin{bmatrix} R & \boldsymbol{r} \\ \boldsymbol{0} & \tau \end{bmatrix}, \tag{2.4}$$

which is the economical QR factorization of $\tilde{A}_\lambda^T$, and implies that $\tilde{Q}_1 = \begin{bmatrix} Q_1 & \boldsymbol{q}_1 \end{bmatrix} \in \mathbb{R}^{d\times(n+1)}$. From the formula of X^* in (2.3), we have

$$\begin{aligned}
\tilde{X}^* &= \begin{bmatrix} Q_1 & \boldsymbol{q}_1 \end{bmatrix} \begin{bmatrix} R & \boldsymbol{r} \\ \boldsymbol{0} & \tau \end{bmatrix}^{-T} \begin{bmatrix} B \\ \boldsymbol{b}^T \end{bmatrix} \\
&= \begin{bmatrix} Q_1 & \boldsymbol{q}_1 \end{bmatrix} \begin{bmatrix} R^{-T} & \boldsymbol{0} \\ -\frac{1}{\tau}\boldsymbol{r}^T R^{-T} & \frac{1}{\tau} \end{bmatrix} \begin{bmatrix} B \\ \boldsymbol{b}^T \end{bmatrix} \\
&= \begin{bmatrix} Q_1 & \boldsymbol{q}_1 \end{bmatrix} \begin{bmatrix} R^{-T}B \\ \frac{1}{\tau}(\boldsymbol{b}^T - \boldsymbol{r}^T R^{-T} B) \end{bmatrix} \\
&= Q_1 R^{-T} B + \frac{\boldsymbol{q}_1}{\tau}(\boldsymbol{b}^T - \boldsymbol{r}^T R^{-T} B) \\
&= X^* + \frac{\boldsymbol{q}_1}{\tau}\left(\boldsymbol{b}^T - \boldsymbol{a}^T X^*\right), \qquad (2.5)
\end{aligned}$$

where we used $\boldsymbol{r} = Q_1^T \boldsymbol{a}$ and $X^* = Q_1 R^{-T} B$ in the last equality.

To compute $\tilde{X}^*$ using the above equation, we need to know $\frac{\boldsymbol{q}_1}{\tau}$. However, computing $\frac{\boldsymbol{q}_1}{\tau}$ by explicitly performing QR factorization (2.2) costs $O(dn^2 + 2n^3/3)$ flops [18], which is prohibitively high for large-scale high-dimensional data. Thus, we have to find an alternative approach.

Let $\boldsymbol{w} = \boldsymbol{b} - X^{*T}\boldsymbol{a}$, $t = \arg\max_{1\le i\le c} |\boldsymbol{w}_i|$, $\tilde{\boldsymbol{x}}^*$ and $\boldsymbol{x}^*$ denote the t^{th} column of $\tilde{X}^*$ and X^*, respectively. If $\boldsymbol{w}_t = 0$, then $\tilde{X}^* = X^*$. So, we assume $\boldsymbol{w}_t > 0$. On the one hand, we have from the original RLS problem (1.1) that

$$\tilde{\boldsymbol{x}}^* = \arg\min_{\boldsymbol{x}} \|\tilde{A}\boldsymbol{x} - \tilde{B}_t\|_2^2 + \lambda^2 \|\boldsymbol{x}\|_2^2, \tag{2.6}$$

where $\tilde{A} = \begin{bmatrix} A \\ \boldsymbol{a}^T \end{bmatrix}$ and $\tilde{B}_t$ denotes the t^{th} column of $\tilde{B}$. On the other hand, based on formula (2.5) of updating X^*, we have

$$\tilde{\boldsymbol{x}}^* = \boldsymbol{x}^* + \frac{\boldsymbol{q}_1}{\tau}\boldsymbol{w}_t,$$

which implies that

$$\frac{\boldsymbol{q}_1}{\tau} = (\tilde{\boldsymbol{x}}^* - \boldsymbol{x}^*)/\boldsymbol{w}_t$$

and thus,

$$\tilde{X}^* = X^* + (\tilde{\boldsymbol{x}}^* - \boldsymbol{x}^*)\boldsymbol{w}^T/\boldsymbol{w}_t.$$

As we can see from the above derivation, the dominant work in updating X^* is to solve the RLS problem with single column on the right-hand side in (2.6). We adopt LSQR [26] to solve (2.6), which leads to an incremental RLS algorithm for one new sample as presented in Algorithm 1.

Now, we investigate the computational complexity of Algorithm 1 and compare it with that of RLS. In Algorithm 1 the dominant cost is that of applying LSQR. LSQR is an iterative algorithm which needs to compute two matrix-vector multiplications of the forms $A\boldsymbol{v}$ and $A^T\boldsymbol{u}$ in each iteration, which requires $4sn$ flops if the average number of non-zero features for one sample is s. The remaining computational cost of

Algorithm 1 (**IRLS**: Incremental RLS for one new sample)

```
Input: Data matrix $A$, response matrix $B$, unique solution $X^*$, a new sample $\boldsymbol{a}$ and the corresponding
    response $\boldsymbol{b}$.
Output: Updated $A$, $B$ and $X^*$
1: Compute $\boldsymbol{w} = \boldsymbol{b} - X^{*T}\boldsymbol{a}$ and $t = \arg\max_{1\le i\le c} |\boldsymbol{w}_i|$.
2: if $\boldsymbol{w}_t > 0$ then
3:     Apply LSQR to solve (2.6).
4:     Compute $\boldsymbol{q}_1 = (\tilde{\boldsymbol{x}}^* - \boldsymbol{x}^*)$ and $X^* \leftarrow X^* + \boldsymbol{q}_1(\boldsymbol{w}^T/\boldsymbol{w}_t)$.
5: end if
6: $A \leftarrow \begin{bmatrix} A \\ \boldsymbol{a}^T \end{bmatrix}$, $B \leftarrow \begin{bmatrix} B \\ \boldsymbol{b}^T \end{bmatrix}$.
7: return $A$, $B$, and $X^*$.
```

LSQR in each iteration is $9d + 5n$ flops [26]. Let N_{iter} denote the number of iterations used in LSQR, the computational cost of Algorithm 1 is $O(N_{iter}(4sn + 9d) + 4dc)$, where $O(4dc)$ is the cost of rank-1 update of X. For batch RLS, since we need to solve a RLS problem with c columns on the right-hand side, the computational cost is $O(cnd \min\{n, d\})$ when direct solver (1.2) is adopted, and is $O(cN_{iter}(4sn+9d))$ when iterative solver LSQR is adopted. In large-scale problems, the iterative solver usually outperforms the direct one, especially when the data matrix A is highly sparse. Therefore, for a RLS problem with c columns on the right hand side our incremental algorithm is approximately c times as efficient as the batch RLS in terms of flops. We should remark that RLS problem (1.1) can be decoupled into c independent RLS problems with single right hand side and share the same coefficient matrix A. Thus, these problems can be solved in a parallel or distributed way to reduce computational time. However, designing incremental algorithm for parallel or distributed computing architecture is more complicated than Algorithm 1 and is out of the scope of this paper. We left this topic for future research.

Remark 2.1. *Notice that, although we resort to the economical QR factorization of A_λ^T in the derivation of IRLS, we do not need it to update X^* in Algorithm 1. Moreover, LSQR is not the only choice for solving (2.6), any efficient RLS solver can be adopted in IRLS.*

In the sequel of this paper, we apply IRLS to reduce the dimension of large-scale data, and particularly, we are interested in applying IRLS to design incremental LDA algorithms. Before that, we need to briefly review LDA and establish the connection between LDA and RLS.

3. Linear discriminant analysis.

In this section, we briefly review LDA and its generalizations for undersampled problems, as well as the LS solutions to LDA.

3.1. A brief review.

Suppose we have a set of n data points $\boldsymbol{a}_1$, $\boldsymbol{a}_2$, ..., $\boldsymbol{a}_n \in \mathbb{R}^d$ from c classes, and all data points are given in the form of a data matrix

$$A = \begin{bmatrix} \boldsymbol{a}_1 & \cdots & \boldsymbol{a}_n \end{bmatrix}^T = \begin{bmatrix} A_1^T & A_2^T & \cdots & A_c^T \end{bmatrix}^T \in \mathbb{R}^{n\times d},$$

where each block matrix $A_k \in \mathbb{R}^{n_k\times d}$ $(1 \le k \le c)$ is a collection of data points from the k^{th} class, n_k is the number of data points in the k^{th} class and $n = \sum_{k=1}^c n_k$ is the total number of data points. The global centroid $\boldsymbol{c}$ of A and the local centroid $\boldsymbol{c}_k$ of each class A_k are given by $\boldsymbol{c} = \frac{1}{n}A^T\boldsymbol{e}$, $\boldsymbol{c}_k = \frac{1}{n_k}A_k^T\boldsymbol{e}_k$, respectively, where

$$\boldsymbol{e} = \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}^T \in \mathbb{R}^n, \quad \boldsymbol{e}_k = \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}^T \in \mathbb{R}^{n_k}, \quad k = 1, \cdots, c.$$

Let $\mathcal{N}_k$ denote the set of row indices that belong to the k^{th} class, and $S_b = \frac{1}{n}\sum_{k=1}^c n_k(\boldsymbol{c}_k - \boldsymbol{c})(\boldsymbol{c}_k - \boldsymbol{c})^T$, $S_w = \frac{1}{n}\sum_{k=1}^c \sum_{j\in\mathcal{N}_k}(\boldsymbol{a}_j - \boldsymbol{c}_k)(\boldsymbol{a}_j - \boldsymbol{c}_k)^T$ and $S_t = \frac{1}{n}\sum_{j=1}^n(\boldsymbol{a}_j - \boldsymbol{c})(\boldsymbol{a}_j - \boldsymbol{c})^T$ be the between-class scatter matrix, the within-class scatter matrix and the total scatter matrix, respectively. It is well known [20] that $S_t = S_b + S_w$, $\text{trace}(S_w) = \frac{1}{n}\sum_{k=1}^c \sum_{j\in\mathcal{N}_k} \|\boldsymbol{a}_j - \boldsymbol{c}_k\|_2^2$ and $\text{trace}(S_b) = \frac{1}{n}\sum_{k=1}^c n_k\|\boldsymbol{c}_k - \boldsymbol{c}\|_2^2$. Thus, trace$(S_b)$ measures the weighted distance between the local centroids and the global centroid, while trace(S_w) measures the distance between the data points and their corresponding local centroids. In the lower-dimensional space mapped upon using the linear transformation $G \in \mathbb{R}^{d\times l}$, the between-class, within-class and total scatter matrices are of the forms G^TS_bG, G^TS_wG, G^TS_tG, respectively. Ideally, the optimal transformation

G should maximize trace$(G^T S_b G)$ and minimize trace$(G^T S_w G)$ simultaneously, or equivalently, maximize trace$(G^T S_b G)$ and minimize trace$(G^T S_t G)$ simultaneously.

To make LDA applicable for the *undersampled problem* (that is, $d > n$), various extensions of the classical LDA can be found in the literature, see [7, 11, 12, 17, 20, 28, 35, 36, 38, 39] . Among those extensions, one popular extension is to replace the matrix inverse in classical LDA with pseudo-inverse, which leads to the following criterion

$$G^* = \arg\max_{G\in\mathbb{R}^{d\times l}} \operatorname{trace}\Big((G^T S_t G)^\dagger (G^T S_b G)\Big). \tag{3.1}$$

This extension of LDA has been applied successfully to many applications including machine learning, data mining and bioinformatics, which require to deal with high-dimensional data efficiently.

3.2. Least squares approach. It is well-known [15] that LDA for binary class ($c = 2$) case is equivalent to a LS problem. Recently, this connection has been extended to multi-class ($c > 2$) case by investigating the relationship between LDA and multivariate linear regression with a specific class indicator matrix [19, 28, 36]. In [36], the author proposed a new class indicator matrix $Y \in \mathbb{R}^{n\times c}$ defined as

$$Y_{jk} = \begin{cases} \sqrt{\frac{n}{n_k}} - \sqrt{\frac{n_k}{n}} & \text{if } j \in \mathcal{N}_k, \\ -\sqrt{\frac{n_k}{n}} & \text{otherwise,} \end{cases} \tag{3.2}$$

and showed that the multivariate linear regression solution $G^{MLR} \in \mathbb{R}^{d\times c}$ obtained from

$$G^{MLR} = \arg\min_{G\in\mathbb{R}^{d\times c}} \frac{1}{2}\|H_t G - Y\|_F^2, \tag{3.3}$$

where $H_t = \frac{1}{\sqrt{n}}(A - \boldsymbol{e}\boldsymbol{c}^T) \in \mathbb{R}^{n\times d}$ is the scaled centred data matrix, solves the corresponding LDA optimization problem (3.1). The same idea has been explored in [39], where the authors established an equivalence between RLDA and a ridge estimator for multivariate linear regression. In [28], a similar result has been obtained by investigating the relationship between LDA and *minimum squared error* (MSE). In [7], the authors cast LDA into a regression framework using spectral graph analysis and proposed a new algorithm named *spectral regression discriminant analysis* (SRDA) which is very efficient for large-scale data.

The equivalence relationship between LDA and LS problem not only facilitates efficient computation of the LDA solution, but also enables the use of regularization techniques (e.g., sparsity inducing regularization or Laplacian-based regularization). However, the use of centred data matrix H_t and class indicator matrix Y defined in (3.2) makes it difficult to incrementally update the LS solution to LDA, since one has to recalculate the whole centred data matrix and class indicator matrix when a new sample is inserted as [24] did. In the next section, we will show how to solve this problem with necessary modification to (3.3) and a different class indicator matrix.

4. A new batch LDA algorithm .

Notice from the LS problem (3.3) that, to compute G^{MLR}, one needs to calculate the centred data matrix H_t. In many applications (e.g., text-document processing) the data matrix A may be sparse, which can be exploited to facilitate computation and save memory. However, after centring, the centred data matrix H_t is not sparse any more. To avoid centring the data matrix, we propose a new LS model for LDA which is highly adoptable to incremental setting.

4.1. A new least squares LDA model. The following trick is commonly used [7, 28, 39] to avoid centring the data matrix: append a new component $''1''$ to each data $\boldsymbol{a}_j$ as $\underline{\boldsymbol{a}}_j = [\boldsymbol{a}_j^T \ 1]^T$. The revised LS problem is formulated as

$$(G^*, \boldsymbol{g}^*) = \arg\min \left\{ \frac{1}{2} \left\| [A \quad \boldsymbol{e}] \begin{bmatrix} G \\ \boldsymbol{g}^T \end{bmatrix} - Y \right\|_F^2 \;\middle|\; G \in \mathbb{R}^{d\times c},\ \boldsymbol{g} \in \mathbb{R}^c \right\}. \tag{4.1}$$

It can be proven that G^* solves the LS problem (3.3), hence a solution to LDA problem (3.1).

To make the above LS solution of LDA more adoptable to incremental setting, we propose to use the following class indicator matrix:

$$E := \begin{bmatrix} \boldsymbol{e}_1 & & \\ & \ddots & \\ & & \boldsymbol{e}_c \end{bmatrix} \in \mathbb{R}^{n \times c}. \tag{4.2}$$

Compared with the class indicator matrix Y in (3.2), the new indicator matrix E contains only 1 or 0 in each column, and thus, has the advantage that when a new sample is inserted we only need to append the corresponding class indicator vector to E, instead of recalculating the whole indicator matrix. More importantly, we have the following result relating LDA problem (3.1) and LS problem (4.1) with Y being replaced by E.

Lemma 4.1. *Let G^* be obtained from any solution to the following LS problem*

$$(G^*, \boldsymbol{g}^*) = \arg\min \left\{ \frac{1}{2} \left\| \begin{bmatrix} A & \boldsymbol{e} \end{bmatrix} \begin{bmatrix} G \\ \boldsymbol{g}^T \end{bmatrix} - E \right\|_F^2 \;\middle|\; G \in \mathbb{R}^{d \times c},\ \boldsymbol{g} \in \mathbb{R}^c \right\}, \tag{4.3}$$

then

$$G^* = \arg \max_{G \in \mathbb{R}^{d \times c}} \operatorname{trace}\Big((G^T S_t G)^\dagger (G^T S_b G) \Big).$$

Thus, the solution to the LS problem (4.3) also solves the LDA problem (3.1).

Proof. Since $(G^*, \boldsymbol{g}^*)$ is a solution to the LS problem (4.3), it must satisfy the associated normal equation

$$\left(\begin{bmatrix} A^T \\ \boldsymbol{e}^T \end{bmatrix} \begin{bmatrix} A & \boldsymbol{e} \end{bmatrix} \right) \begin{bmatrix} G^* \\ \boldsymbol{g}^{*T} \end{bmatrix} = \begin{bmatrix} A^T \\ \boldsymbol{e}^T \end{bmatrix} E,$$

which is equivalent to

$$\begin{bmatrix} A^T A & n\boldsymbol{c} \\ n\boldsymbol{c}^T & n \end{bmatrix} \begin{bmatrix} G^* \\ \boldsymbol{g}^{*T} \end{bmatrix} = \begin{bmatrix} A^T E \\ \boldsymbol{n}^T \end{bmatrix}, \tag{4.4}$$

where $\boldsymbol{n} = \begin{bmatrix} n_1 \cdots n_c \end{bmatrix}^T$. The last equation in the above linear system implies that

$$\boldsymbol{g}^* = \boldsymbol{n}/n - G^{*T}\boldsymbol{c}.$$

Substituting the above equality into the first row of linear system (4.4), we get

$$(A^T A - n\boldsymbol{c}\boldsymbol{c}^T) G^* = A^T E - \boldsymbol{c}\boldsymbol{n}^T.$$

It is easy to verify that

$$A^T A - n\boldsymbol{c}\boldsymbol{c}^T = n H_t^T H_t \text{ and } A^T E - \boldsymbol{c}\boldsymbol{n}^T = \sqrt{n} H_b^T D,$$

where $H_b = \begin{bmatrix} \sqrt{n_1}(\boldsymbol{c}_1 - \boldsymbol{c}) & \cdots & \sqrt{n_c}(\boldsymbol{c}_c - \boldsymbol{c}) \end{bmatrix}^T / \sqrt{n}$ and $D = \operatorname{diag}(\sqrt{n_1}, \cdots, \sqrt{n_c})$ is a diagonal matrix, which leads to

$$\sqrt{n} H_t^T H_t G^* = H_b^T D. \tag{4.5}$$

Let the reduced singular value decomposition (SVD) of H_t be $H_t = U_1 \Sigma_t V_1^T$, where $U_1 \in \mathbb{R}^{n \times \gamma}$ and $V_1 \in \mathbb{R}^{d \times \gamma}$ are column orthogonal, $\Sigma_t \in \mathbb{R}^{\gamma \times \gamma}$ is a diagonal matrix with positive diagonal entries and $\gamma = \operatorname{rank}(H_t)$. In addition, let the reduced SVD of $\Sigma_t^{-1} V_1^T H_b^T$ be $\Sigma_t^{-1} V_1^T H_b^T = P_1 \Sigma_b Q_1^T$, where $P_1 \in \mathbb{R}^{\gamma \times q}$, $Q_1 \in \mathbb{R}^{c \times q}$ are column orthogonal, $\Sigma_b \in \mathbb{R}^{q \times q}$ is diagonal and nonsingular. Then equation (4.5) becomes

$$V_1^T G^* = \Sigma_t^{-1} P_1 \Sigma_b Q_1^T D / \sqrt{n},$$

which further implies that

$$(G^*)^T S_t G^* = DQ_1\Sigma_b^2 Q_1^T D/n \text{ and } (G^*)^T S_b G^* = (G^*)^T H_b^T H_b G^* = DQ_1\Sigma_b^4 Q_1^T D/n.$$

Thus,

$$((G^*)^T S_t G^*)^\dagger (G^*)^T S_b G^* = D^{-1}Q_1\Sigma_b^2 Q_1^T D,$$

and

$$\text{trace}\Big(((G^*)^T S_t G^*)^\dagger (G^*)^T S_b G^*\Big) = \text{trace}(\Sigma_b^2).$$

On the other hand, it has been shown in [36] that

$$\max_G \text{trace}\Big((G^T S_t G)^\dagger G^T S_b G\Big) = \text{trace}(\Sigma_b^2).$$

Combining the above two equations completes the proof. ◻

Remark 4.1. *As can be seen from the proof of Lemma 4.1, for any class indicator matrix Y of the form*

$$Y = EM - \alpha \boldsymbol{e}\boldsymbol{y}^T,$$

where $M \in \mathbb{R}^{c\times c}$ is nonsingular, α is a scalar and $\boldsymbol{y} \in \mathbb{R}^c$ is a column vector, the solution to the LS problem (4.1) solves the LDA problem (3.1). The indicator matrix in [36] corresponds to the case when $M = \text{diag}(\sqrt{n/n_1}\ \cdots\ \sqrt{n/n_c})$, $\alpha = 1$ and $\boldsymbol{y} = [\sqrt{n_1/n}\ \cdots\ \sqrt{n_c/n}]^T$, the indicator matrix in [24] corresponds to the case when $M = \text{diag}(\sqrt{1/n_1}\ \cdots\ \sqrt{1/n_c})$ and $\alpha = 0$. The indicator matrix E used in (4.3) corresponds to $M = I_c$ and $\alpha = 0$.

For the sake of convenience and simplicity, *we use $\underline{A}$ and $\underline{G}$ to denote the augmented data matrix $\begin{bmatrix} A & \boldsymbol{e}\end{bmatrix}$ and the corresponding transformation $\begin{bmatrix} G \\ \boldsymbol{g}^T\end{bmatrix}$, respectively, and use d' to denote the dimension of the augmented data (i.e. $d' = d+1$).* In this way, the LS problem (4.3) becomes

$$\underline{G}^* = \arg\min_{\underline{G}\in\mathbb{R}^{d'\times c}} \frac{1}{2}\|\underline{A}\underline{G} - E\|_F^2. \tag{4.6}$$

Before proceeding to the proposal of new batch LDA model, let us consider a very special case when $\underline{A}$ has full row rank (i.e., all training samples are linearly independent). In such a case, by comparing LS problems (4.6) and (2.1), we can compute the solution $\underline{G}^*$ incrementally by applying the IRLS algrithm. However, in the incremental setting, the linear independence assumption among training data fails to hold as more and more new data are acquired. To overcome this problem and make the IRLS algorithm applicable, we introduce a new trick named *data lifting* to deal with the rank deficiency problem and propose a new batch LDA algorithm.

The idea behind data lifting is very simple. In order to make training samples linearly independent, we first *implicitly* append λI_n to the data matrix resulting a higher-dimensional data matrix

$$A_\lambda := \begin{bmatrix}\underline{A} & \lambda I_n\end{bmatrix} \in \mathbb{R}^{n\times(d'+n)},$$

then use A_λ as the data matrix in the LS problem (4.6), that is, solve the following LS problem

$$G_\lambda^* = \arg\min\left\{\frac{1}{2}\|A_\lambda G_\lambda - E\|_F^2 \;\middle|\; G_\lambda \in \mathbb{R}^{(d'+n)\times c}\right\}. \tag{4.7}$$

We then partition G_λ^* as follows:

$$G_\lambda^* = \begin{bmatrix}\underline{G}_{\lambda 1}^* \\ G_{\lambda 2}^*\end{bmatrix} = \begin{bmatrix} G_{\lambda 1}^* \\ \boldsymbol{g}_\lambda^{*T} \\ G_{\lambda 2}^*\end{bmatrix} \in \mathbb{R}^{(d'+n)\times c}, \tag{4.8}$$

where $G^*_{\lambda 1} \in \mathbb{R}^{d\times c}$, $\boldsymbol{g}^*_\lambda \in \mathbb{R}^c$ and $G^*_{\lambda 2} \in \mathbb{R}^{n\times c}$. Finally, we use $G^*_{\lambda 1}$ as the solution of the LDA problem (3.1).[1]

At the first sight, our trick looks contradictory to the goal of LDA to reduce the dimension. However, we should clarify that, since we are interested in $G^*_{\lambda 1}$, we only implicitly append λI_n to the data matrix. The augmented matrix A_λ is never formed explicitly, as can be seen from Theorem 4.2 and Algorithm 2. The idea behind this trick is somehow similar to that of the kernel trick [30] where all training samples are mapped *implicitly* to some high-dimensional or even infinite-dimensional feature space. Moreover, our method is supported by theoretical results as described in the following theorem.

THEOREM 4.2. *Let $\gamma = \operatorname{rank}(\underline{A})$ and σ_γ be the smallest nonzero singular value of $\underline{A}$, $\underline{G}^*$ be the minimum Frobenius norm solution to the LS problem (4.6), and G^*_λ be defined as in (4.8) be the minimum Frobenius norm solution to the LS problem (4.7). Then,*

(a) $\underline{G}^*_{\lambda 1}$ is the unique solution of the following RLS problem

$$\min_{\underline{G}\in\mathbb{R}^{d'\times c}} \frac{1}{2}\|\underline{A}\underline{G} - E\|_F^2 + \frac{\lambda^2}{2}\|\underline{G}\|_F^2. \tag{4.9}$$

(b) $\frac{\|\underline{G}^*_{\lambda 1}-\underline{G}^*\|_F}{\|\underline{G}^*\|_F} \le \frac{\lambda^2}{\lambda^2+\sigma_\gamma^2}$, and $\lim_{\lambda\to 0} \underline{G}^*_{\lambda 1} = \underline{G}^*$. Moreover, if $\sigma_\gamma \gg \lambda$, we have $\frac{\|\underline{G}^*_{\lambda 1}-\underline{G}^*\|_F}{\|\underline{G}^*\|_F} = O(\lambda^2)$, which means that $\underline{G}^*_{\lambda 1}$ converges to $\underline{G}^*$ quadratically with respect to λ.

Proof. (a) From the definition of $\underline{G}^*_{\lambda 1}$, we have from the proof of Lemma (2.1) that

$$\underline{G}^*_{\lambda 1} = \underline{A}^T(\underline{A}\underline{A}^T + \lambda^2 I_n)^{-1}E. \tag{4.10}$$

The rest of the proof is analogous to that of Lemma 2.1 and so is omitted.

(b) Since $\underline{G}^*$ is the minimum Frobenius norm solution to the LS problem (4.6), it follows that

$$\underline{G}^* = \underline{A}^\dagger E. \tag{4.11}$$

Let the reduced singular value decomposition (SVD) of $\underline{A}$ be $\underline{A} = U_1\Sigma V_1^T$, where $U_1 \in \mathbb{R}^{n\times\gamma}$ and $V_1 \in \mathbb{R}^{d'\times\gamma}$ are column orthogonal matrices, and $\Sigma = \operatorname{diag}(\sigma_1 \ \cdots \ \sigma_\gamma)$ with $\sigma_1 \ge \cdots \ge \sigma_\gamma > 0$, then equalities (4.10) and (4.11) can be reformulated as

$$\underline{G}^*_{\lambda 1} = V_1\Sigma(\Sigma^2 + \lambda^2 I_\gamma)^{-1}U_1^T E, \tag{4.12}$$

and

$$\underline{G}^* = V_1\Sigma^{-1}U_1^T E, \tag{4.13}$$

respectively. Combining equations (4.12) and (4.13) , we have

$$\underline{G}^*_{\lambda 1} - \underline{G}^* = -\lambda^2 V_1\Sigma^{-1}(\Sigma^2 + \lambda^2 I_\gamma)^{-1}U_1^T E,$$

which leads to

$$\|\underline{G}^*_{\lambda 1} - \underline{G}^*\|_F \le \lambda^2\|(\Sigma^2 + \lambda^2 I_\gamma)^{-1}\|_2\|V_1\Sigma^{-1}U_1^T E\|_F \le \frac{\lambda^2}{\lambda^2+\sigma_\gamma^2}\|\underline{G}^*\|_F.$$

Hence, $\frac{\|\underline{G}^*_{\lambda 1}-\underline{G}^*\|_F}{\|\underline{G}^*\|_F} \le \frac{\lambda^2}{\lambda^2+\sigma_\gamma^2}$. The rest part is straightforward by applying the obtained inequality. □

REMARK 4.2. *Theorem 4.2 provides the rationale behind our trick of data lifting. Theorem 4.2(a) establishes an equivalence between our method and ridge regression, which further implies that our method is capable of avoiding data over-fitting problem. Theorem 4.2(b) shows that the solution $\underline{G}^*_{\lambda 1}$ computed by our method converges to the minimum Frobenius norm solution of LS problem (4.6). Therefore, according*

[1]The lower-dimensional representation of a data point $\boldsymbol{x} \in \mathbb{R}^d$ is given by $\boldsymbol{x}^L := G^{*T}_{\lambda 1}\boldsymbol{x}$. In some literatures (e.g. [7]), $\boldsymbol{x}^L$ is also given by $\boldsymbol{x}^L := \underline{G}^{*T}_{\lambda 1}\begin{bmatrix}\boldsymbol{x}\\1\end{bmatrix} = G^{*T}_{\lambda 1}\boldsymbol{x}+\boldsymbol{g}^*_\lambda$. It is easy to see that the latter representation is just a translation of the former one. Thus, both representations will give the same classification results when classifiers depending on the mutual distance between data points (e.g. KNN) are used.

to Lemma 4.1, $G^*_{\lambda 1}$ converges to a solution of LDA problem (3.1), and the convergence rate is $O(\lambda^2)$ if λ is small enough.

Our model in (4.7) is related to some prior work. In [39], the authors consider the following RLS problem[2]

$$\min_{\underline{G}\in\mathbb{R}^{d'\times c}} \frac{1}{2}\|\underline{AG}-Y\|_F^2+\frac{\lambda^2}{2}\|G\|_F^2, \tag{4.14}$$

and establish an equivalence relationship between (4.14) and regularized linear discriminant analysis. The major difference between (4.14) and our model (4.9) lies in the regularization term where the norm of $\boldsymbol{g}$ is also considered in our model. An algorithm named RFDA/RR is also proposed in [39] that requires to compute the incomplete Cholesky decomposition (ICD) of $H_t^TH_t$ if $d\le n$ or of $H_tH_t^T$ if $d>n$. The computational cost of RFDA/RR with ℓ columns retained in the ICD is given in Table 1.

The SRDA proposed in [7] considers a similar RLS problem as (4.9), and has become one of the most efficient algorithms for large-scale discriminant analysis. Specifically, SRDA solves the following LS problem via LSQR

$$\underline{G}^*_{\lambda s}=\arg\min\left\{\frac{1}{2}\|\underline{AG}-\overline{Y}\|_F^2+\frac{\lambda^2}{2}\|\underline{G}\|_F^2\ \middle|\ \underline{G}\in\mathbb{R}^{d'\times(c-1)}\right\}, \tag{4.15}$$

where $\overline{Y}\in\mathbb{R}^{n\times(c-1)}$ is a response matrix obtained from the Gram-Schmidt orthogonalization of $\{\boldsymbol{e},E_1,\cdots,E_c\}$; see [7] for more details of the response matrix generation. We can establish the following connection between the solutions of (4.9) and (4.15) (i.e., $\underline{G}^*_{\lambda 1}$ and $\underline{G}^*_{\lambda s}$) in Lemma 4.3.

Lemma 4.3. *Let $D=\operatorname{diag}(\sqrt{n_1}\ \cdots\ \sqrt{n_c})$, there exists a column orthogonal matrix $Z\in\mathbb{R}^{c\times(c-1)}$ satisfying $\begin{bmatrix}\sqrt{n_1} & \cdots & \sqrt{n_c}\end{bmatrix}^T Z=\mathbf{0}$ such that*

$$\underline{G}^*_{\lambda s}=\underline{G}^*_{\lambda 1}D^{-1}Z. \tag{4.16}$$

Proof. Let $\{\overline{\boldsymbol{y}}_i\}_{i=0}^{c-1}$ be the set of orthonormal vectors obtained from the Gram-Schmidt orthogonalization of $\{\boldsymbol{e},E_1,\cdots,E_c\}$, then we have from [7] that

$$\overline{Y}=\begin{bmatrix}\overline{\boldsymbol{y}}_1 & \cdots & \overline{\boldsymbol{y}}_{c-1}\end{bmatrix} \quad\text{and}\quad \boldsymbol{e}^T\overline{Y}=\mathbf{0}.$$

Since $\boldsymbol{e}$ lies in the range space of E, it follows that $\overline{Y}$ lies in the range space of E and we can find some coefficient matrix $\mathcal{Y}\in\mathbb{R}^{c\times(c-1)}$ such that $\overline{Y}=E\mathcal{Y}$, which further implies that

$$I_{c-1}=\overline{Y}^T\overline{Y}=\mathcal{Y}^TE^TE\mathcal{Y}=\mathcal{Y}^TD^2\mathcal{Y}.$$

Define $Z:=D\mathcal{Y}$, then Z is column orthogonal and $\overline{Y}=ED^{-1}Z$. Moreover, $\mathbf{0}=\boldsymbol{e}^T\overline{Y}=\begin{bmatrix}\sqrt{n_1} & \cdots & \sqrt{n_c}\end{bmatrix}^T Z$. Therefore, the unique solution to problem (4.15) is given by

$$\underline{G}^*_{\lambda s}=\underline{A}^T(\underline{AA}^T+\lambda I_n)^{-1}\overline{Y}=\underline{A}^T(\underline{AA}^T+\lambda I_n)^{-1}ED^{-1}Z=\underline{G}^*_{\lambda 1}D^{-1}Z,$$

where in the last equality we used (4.10). □

Since we are only interested in $\underline{G}^*_{\lambda 1}$ which, due to the result presented in Theorem 4.2(a), is the unique solution to the RLS problem (4.9), we adopt LSQR to compute it in our model, which gives rise to a new bath LDA algorithm named LDADL as described in Algorithm 2. We also remark that, as explained in Remark 2.1, LSQR is not the only choice and any efficient LS solver can be adopted to solve (4.9).

From the analysis above and Lemma 4.3, we can expect that LDADL shall achieve similar performance as SRDA. Moreover, LDADL has two advantages over SRDA and RFDA/RR: (1) LDADL is theoretically supported in that we can establish a close relationship between LDADL and LDA problem (3.1) as presented in Theorem 4.2(b). (2) LDADL can be efficiently extended to the case that a new sample is added incrementally, as shown in Section 5 below. In contrast, the incremental versions of SRDA and RFDA/RR are difficult to develop. For SRDA, the difficulty lies in updating the response matrix $\bar{Y}$ obtained from the Gram-Schmidt orthogonalization, while for RFDA/RR the difficulty is incurred by the regularization term $\frac{\lambda^2}{2}\|G\|_F^2$. This is a significant difference between our LDADL and SRDA and RFDA/RR.

[2]For the sake of consistency, the optimization problem studied in [39] is presented here using our notations. The same situation applies to SRDA in (4.15).

Algorithm 2 (**LDADL**: Batch **LDA** with **D**ata **L**ifting)

```
Input: Training data $A \in \mathbb{R}^{n\times d}$, $n_k$ $(1 \le k \le c)$ and parameter $\lambda > 0$.
Output: Transformation matrices $\underline{G}^*_{\lambda 1} \in \mathbf{R}^{d'\times c}$.
1: Append $\boldsymbol{e}$ to data matrix $A$ to construct $\underline{A}$.
2: Compute $\underline{G}^*_{\lambda 1}$ via applying LSQR to solve (4.9).
3: return $\underline{G}^*_{\lambda 1}$.
```

Table 1

Comparison of computational complexity and memory usage: dimension (d), $d' = d+1$, the number of training data (n), $t = \min\{d, n\}$, the number of classes (c), the number of columns retained in ICD (ℓ), the number of iterations in LSQR (N_{iter}) and the average number of non-zero features for one sample (s).

Algorithms	Operation counts (flops)	Memory
RLDA [38]	$O(14dn^2 - 2n^3)$	$O(dn + 2dc)$
ROLDA [10]	$O(4dn^2 + 35n^3)$	$O(dn + 2n^2 + (d+n)c)$
ULDA/QR [11]	$O(4dn^2 + 4n^3)$	$O(d(n+c) + 3n^2)$
IDR/QR [35]	$O(2dnc)$	$O(dn + 2(dc + c^2))$
RFDA/RR [39]	$O(dn(3t+4c) + \ell t^2)$	$O(t^2 + d(n+c) + c^2)$
SRDA [7]	$O((c-1)N_{iter}(4sn + 9d'))$	$O(ns + (c+2)d')$
LDADL	$O(cN_{iter}(4sn + 9d'))$	$O(ns + cd')$

4.2. Time and space complexity. We close this section by providing a computational complexity analysis of the proposed algorithm. Since we need to solve a RLS problem with c columns on the right-hand side by applying LSQR, the time cost of LDADL is $O(cN_{iter}(4sn + 9d'))$. In our experiments, we use $N_{iter} = 20$, which is the default value in MATLAB. A comparison of computational complexity between LDADL and other LDA algorithms discussed in this paper is given in Table 1. From Table 1, we see that LDADL and SRDA have similar efficiency and both algorithms are faster than other competitors when the data matrix is large-scale and highly sparse. When the class number c is small or the data is dense, IDR/QR is faster than other competitors. In terms of memory usage, LDADL and SRDA use less memory and the advantage is more prominent when the data matrix is highly sparse.

5. Incremental version of LDADL. In this section, we study the incremental version of LDADL proposed in Section 4. To describe our method clearly, we divide this section into two parts, where in the first part we consider the case when the new sample belongs to an existing class and in the second part we consider the case when the new sample belongs to a new class.

Following the idea of IRLS in Section 2 and supposing the economical QR factorization of A_λ^T is given by (2.2), we know that the minimum Frobenius norm solution G^*_λ to the LS problem (4.7) is given by $G^*_\lambda = QR^{-T}E$, which further leads to $\underline{G}^*_{\lambda 1} = Q_1 R^{-T}E$. Let $\boldsymbol{a} \in \mathbb{R}^d$ denote the newly added sample, $\underline{\boldsymbol{a}} = \begin{bmatrix} \boldsymbol{a} \\ 1 \end{bmatrix} \in \mathbb{R}^{d'}$ and $\hat{\boldsymbol{a}} = \begin{bmatrix} \underline{\boldsymbol{a}} \\ \boldsymbol{0} \end{bmatrix} \in \mathbb{R}^{d'+n}$, the economical QR factorization of $\tilde{A}_\lambda^T$ is given by (2.4) with $\boldsymbol{r} = Q_1^T \underline{\boldsymbol{a}}$.

5.1. $\boldsymbol{a}$ belongs to an existing class k_0. In this case, the class indicator matrix becomes

$$\tilde{E} = \begin{bmatrix} E \\ \boldsymbol{z}^T \end{bmatrix},$$

where $\boldsymbol{z} \in \mathbb{R}^c$ is a vector with the k_0^{th} element set to 1 and all other elements set to 0.

Since the IRLS algorithm can be applied directly, we have by following the updating scheme (2.5) that

$$\underline{\tilde{G}}^*_{\lambda 1} = \underline{G}^*_{\lambda 1} + \frac{\boldsymbol{q}_1}{\tau}\left(\boldsymbol{z} - \underline{G}^{*T}_{\lambda 1}\underline{\boldsymbol{a}}\right)^T.$$

Let $\boldsymbol{w} = \boldsymbol{z} - \underline{G}^{*T}_{\lambda 1}\underline{\boldsymbol{a}}$, $t = \arg\max_{1\le i\le c} |\boldsymbol{w}_i|$, $\tilde{\boldsymbol{g}}^*$ and $\boldsymbol{g}^*$ denote the t^{th} column of $\underline{\tilde{G}}^*_{\lambda 1}$ and $\underline{G}^*_{\lambda 1}$, respectively. If $\boldsymbol{w}_t > 0$, we solve the following RLS problem

$$\tilde{\boldsymbol{g}}^* = \arg\min_{\boldsymbol{g}\in\mathbb{R}^{d'}} \left\|\tilde{\underline{A}}\boldsymbol{g} - \tilde{E}_t\right\|_2^2 + \lambda^2\|\boldsymbol{g}\|_2^2, \tag{5.1}$$

and update $\underline{G}^*_{\lambda 1}$ as follows:

$$\underline{\tilde{G}}^*_{\lambda 1} = \underline{G}^*_{\lambda 1} + (\tilde{\boldsymbol{g}}^* - \boldsymbol{g}^*)\boldsymbol{w}^T/\boldsymbol{w}_t.$$

Algorithm 3 (**ILDADL**: **I**ncremental **LDADL** for one new sample)

```
Input: Data matrix $\underline{A}$, label matrix $E$, optimal transformation $\underline{G}_{\lambda 1}^{*}$, a new sample $\boldsymbol{a}$ and the corresponding
    class label.
Output: Updated $\underline{A}$, $E$ and $\underline{G}_{\lambda 1}^{*}$
1: Compute $\boldsymbol{w} = -\underline{G}_{\lambda 1}^{*T}\underline{\boldsymbol{a}}$.
2: if $\boldsymbol{x}$ belongs to an existing class $k_0$ then
3:     Compute $\boldsymbol{w}_{k_0} \leftarrow \boldsymbol{w}_{k_0} + 1$, $t = \arg\max_{1\le i\le c} |\boldsymbol{w}_i|$, and apply LSQR to solve (5.1).
4:     Compute $\boldsymbol{q}_1 = (\tilde{\boldsymbol{g}}^* - \boldsymbol{g}^*)$ and $\underline{G}_{\lambda 1}^{*} \leftarrow \underline{G}_{\lambda 1}^{*} + \boldsymbol{q}_1(\boldsymbol{w}^T/\boldsymbol{w}_t)$.
5: else
6:     Apply LSQR to solve (5.2) and update $\underline{G}_{\lambda 1}^{*} \leftarrow \begin{bmatrix}\underline{G}_{\lambda 1}^{*} + \tilde{\boldsymbol{g}}^*\boldsymbol{w}^T & \tilde{\boldsymbol{g}}^*\end{bmatrix}$.
7: end if
8: return $A \leftarrow \begin{bmatrix} A \\ \boldsymbol{a}^T \end{bmatrix}$, $E$ and $\underline{G}_{\lambda 1}^{*}$.
```

5.2. *a* belongs to a new class. In this case, we have $\tilde{c} = c + 1$ and

$$\tilde{E} = \begin{bmatrix} E & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbb{R}^{(n+1)\times \tilde{c}}.$$

Although IRLS cannot be applied directly in this case since the number of classes has increased by one, the idea of updating the RLS solution is the same. In particular,

$$\begin{aligned} \underline{\tilde{G}}_{\lambda 1}^{*} &= \begin{bmatrix} Q_1 & \boldsymbol{q}_1 \end{bmatrix} \begin{bmatrix} R^{-T} & 0 \\ -\frac{1}{\tau}\boldsymbol{r}^T R^{-T} & \frac{1}{\tau} \end{bmatrix} \begin{bmatrix} E & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \\ &= \begin{bmatrix} Q_1 & \boldsymbol{q}_1 \end{bmatrix} \begin{bmatrix} R^{-T}E & \mathbf{0} \\ -\frac{1}{\tau}\boldsymbol{r}^T R^{-T}E & \frac{1}{\tau} \end{bmatrix} \\ &= \begin{bmatrix} Q_1 R^{-T}E - \frac{\boldsymbol{q}_1}{\tau}\boldsymbol{r}^T R^{-T}E & \frac{\boldsymbol{q}_1}{\tau} \end{bmatrix} \\ &= \begin{bmatrix} \underline{G}_{\lambda 1}^{*} - \frac{\boldsymbol{q}_1}{\tau}\left(\underline{\boldsymbol{a}}^T \underline{G}_{\lambda 1}^{*}\right) & \frac{\boldsymbol{q}_1}{\tau} \end{bmatrix}. \end{aligned}$$

Let $\boldsymbol{w} = -\underline{G}_{\lambda 1}^{*T}\underline{\boldsymbol{a}}$ and $\tilde{\boldsymbol{g}}^*$ denote the $\tilde{c}^{th}$ column of $\underline{\tilde{G}}_{\lambda 1}^{*}$. On the one hand, we have from equation (4.9) that

$$\tilde{\boldsymbol{g}}^* = \arg\min_{\boldsymbol{g}\in\mathbb{R}^{d'}} \left\| \underline{\tilde{A}}\boldsymbol{g} - \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} \right\|_2^2 + \lambda^2 \|\boldsymbol{g}\|_2^2. \tag{5.2}$$

On the other hand, the above formula of updating $\underline{G}_{\lambda 1}^{*}$ shows that $\tilde{\boldsymbol{g}}^* = \frac{\boldsymbol{q}_1}{\tau}$, which implies that

$$\underline{\tilde{G}}_{\lambda 1}^{*} = \begin{bmatrix} \underline{G}_{\lambda 1}^{*} + \tilde{\boldsymbol{g}}^*\boldsymbol{w}^T & \tilde{\boldsymbol{g}}^* \end{bmatrix}.$$

As can be seen from the above derivation, the dominant work in updating $\underline{G}_{\lambda 1}^{*}$ in both cases is to solve the RLS problems (5.1) and (5.2). Compared with the batch algorithm LDADL that solves a RLS problem with c columns on the right-hand side, the incremental algorithm solves a RLS problem with single column on the right-hand side, thus reduces the computational cost c times. The RLS problems (5.1) or (5.2) can be efficiently solved by LSQR, which lead to the incremental implementation of LDADL for one new sample as presented in Algorithm 3.

5.3. Time and space complexity. The dominant computational cost of Algorithm 3 consists of applying LSQR to solve one RLS problem. Let c denote the current number of classes and $\tilde{n}$ denote the number of training data after the single insertion. Then, the total computational cost of Algorithm 3 is $O(N_{iter}(4s\tilde{n} + 9d') + 4d'c)$ with $O(4d'c)$ being the cost of the rank-1 update of $\underline{G}_{\lambda 1}^{*}$, provided that LSQR terminates after N_{iter} iterations. Since in the process of updating $\underline{G}_{\lambda 1}^{*}$, we only need to store $\underline{A}$ and the latest $\underline{G}_{\lambda 1}^{*}$ as well as the class labels of data points in $\underline{A}$, the memory cost of Algorithm 3 is $O(\tilde{n}s + d'c)$.

We summarize the computational cost and memory usage of incremental LDA algorithms: IDR/QR [35], LS-ILDA [24], ILDA/SSS [21, 22] and ILDADL in Table 2. We observe from Table 2 that among the four

Table 2

Comparison of computational complexity for a single insertion: dimensionality (d), $d' = d+1$, the number of training data after the single insertion ($\tilde{n}$), the number of classes (c), the number of iterations in LSQR (N_{iter}) and the average number of non-zero features for one sample (s).

Algorithms	Operation counts (flops)	Memory
IDR/QR [35]	$O(2dc^2 + 91c^3/3)$	$O(2dc)$
LS-ILDA [24]	$O(14d\tilde{n} + 7dc)$	$O(2d\tilde{n} + dc)$
ILDA/SSS [22]	$O(2d\tilde{n}^2 + 12\tilde{n}^3)$	$O(d\tilde{n} + 2dc)$
ILDADL	$O(N_{iter}(4s\tilde{n} + 9d') + 4d'c)$	$O(\tilde{n}s + d'c)$

Table 3

Statistics of the datasets: dimensionality (d), the number of training data (n), the number of testing data (# test) and the number of classes (c). In the third column, the left value means the total number of training data and the right value means the number of incremental training data.

datasets	d	n	# test	c
Feret	6400	600 | 300	400	200
sports	126373	5151 | 1708	3429	7
Reuters	18933	4941 | 1182	3272	41
TDT2	36771	5648 | 1714	3746	30
20NEWS	26214	11314 | 6678	7532	20
new3	83487	5749 | 3066	3809	44
rcv1	47236	320504 | 81651	213631	53
amazon	262144	817268 | 91193	544841	7

incremental LDA algorithms, algorithms IDR/QR and ILDADL have the lowest computational complexities. Specifically, when the class number c is very small compared to the sample size $\tilde{n}$, IDR/QR should be faster than all other algorithms, while ILDADL shall be faster when c is large or when the data matrix $\underline{A}$ is highly sparse (i.e. s is small). LS-ILDA has much higher computational complexity than those of IDA/QR and ILDA/QR for large-scale sparse data. The computational complexities of ILDA/SSS is very high and increases cubically with respect to the accumulated number of training data. Among the four incremental LDA algorithms IDR/QR has the lowest memory usage when c is small and ILDADL has the lowest memory usage when c is large or when the data matrix $\underline{A}$ is highly sparse.

6. Experimental results. In this section, we investigate the performance of our batch and incremental algorithms, namely LDADL and ILDADL, and compare them with some state-of-the-art algorithms. This section consists of five parts. In Subsection 6.1, we describe the datasets used in the experiments and experimental settings. In Subsection 6.2, we compare the performance of our model (4.7) with different LS solvers for computing $\underline{G}^*_{\lambda 1}$. In Subsection 6.3, we investigate the effect of parameter λ in the data lifting trick (i.e. RLS problem (4.9)). In Subsection 6.4, we compare LDADL with some state-of-the-art batch LDA algorithms. In Subsection 6.5, we compare ILDADL with existing state-of-the-art incremental LDA algorithms.

6.1. Datasets and experiment settings. Eight datasets were used in our experimental study, most of which are high-dimensional sparse data. Important information of these datasets are summarized as follows:

- The *Feret* face dataset contains 1000 images of 200 individuals (each one has 5 images). The facial portion of each original image was automatically cropped based on the locations of eyes and mouths, and the cropped images were resized to 80×80 pixels. This dataset is downloaded from
  `http://www.itl.nist.gov/iad/humanid/feret/feret_master.html`.
- The *sports* dataset is derived from the San Jose Mercury newspaper articles that are distributed as part of the TREC collection [33]. The *new3* dataset is derived from TREC-5, TREC-6 and TREC-7 collections [33]. Each one of these datasets is constructed by selecting documents that are part of certain topics in which the various articles are categorized (based on the DESCRIPT tag). All these text-document data are downloaded from CLUTO at
  `http://glaros.dtc.umn.edu/gkhome/cluto/cluto/download`.
- *Reuters* corpus contains 21578 documents in 135 categories. We provide here the ModApte version. Those documents with multiple category labels are discarded. It left us with 8213 documents in

Table 4
Computational time of two LS solvers: Economical QR and LSQR.

datasets	QR	LSQR
Feret	**9.9e+0**	2.1e+1
sports	5.6e+4	**1.3e+0**
Reuters	1.6e+3	**2.7e+0**
TDT2	8.5e+3	**4.3e+0**
20NEWS	1.4e+3	**4.1e+0**
new3	2.2e+5	**1.2e+1**
rcv1	NA	**1.7e+2**
amazon	OOM	**1.1e+2**

65 categories. After preprocessing, this corpus contains 18933 distinct terms. We use the top 41 Categories (In these categories, each category contains no less than 10 documents).

- The *TDT2* dataset is a subset of the original TDT2 corpus (Nist Topic Detection and Tracking corpus) that consists of data collected during the first half of 1998 and taken from 6 sources, including 2 news wires (APW, NYT), 2 radio programs (VOA, PRI) and 2 television programs (CNN, ABC). It consists of 11201 on-topic documents which are classified into 96 semantic categories. In this subset, those documents appearing in two or more categories were removed, and only the largest 30 categories were kept, thus leaving us with 9394 documents in total.
- The *20NEWS* dataset is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. The data contains 18846 documents that are organized into 20 different newsgroups, each corresponding to a different topic. The *Reuters*, *TDT2* and *20NEWS* datasets are downloaded from
  `http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html`.
- The *rcv1* dataset is downloaded from
  `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.
- The *amazon* dataset [5, 14], downloaded from
  `http://www.mblondel.org/data/`,
  contains 1362109 reviews of Amazon products. Each review may belong to one of 7 categories (apparel, book, DVD, electronics, kitchen & housewares, music, video) and is represented as a 262144 dimensional vector.

For each dataset, we randomly split it into training and testing data in the following way: for each class with size n_k, we randomly select $\lceil 0.6n_k \rceil$ [3] data as training data and the rest as testing data. In experiments involving incremental LDA algorithms, we further partition the training data into two parts: in each of the first $\lfloor 0.5c \rfloor$ classes, we randomly select 80% of the data points as initial training data, and the rest of the training data as incremental training data that will be inserted into the initial training data *one by one*. Incremental learning is completed until all data points in the incremental training data are added into the initial training data. The classification accuracy of the final transformation matrix $\underline{G}^*_{\lambda 1}$ is evaluated on the testing data using the nearest neighbor (NN) classifier. The computational cost is the CPU time of updating $\underline{G}^*_{\lambda 1}$ for one single insertion. For each algorithm, to reduce the variability, this process is repeated 10 times, and the average results are reported. The important statistics of these datasets are summarized in Table 3, where the left value in the third column means the total number of training data and the right value means the number of incremental training data. All experiments were conducted on a workstation with 32 2.70 GHz Intel Xeon CPUs and 128 GB RAM in total. The number of iterations of LSQR is set to be 20 which is the default in MATLAB.

6.2. Comparison of different LS solvers. In this part, we compare the performance of our model (4.7) that uses two different LS solvers for computing $\underline{G}^*_{\lambda 1}$: economical QR factorization $\underline{G}^*_{\lambda 1} = Q_1 R^{-T} E$, where Q_1 and R are from the economical QR factorization of A_λ^T, and LSQR. The computational time is shown in Table 4. We omitted the results of the economical QR solver on datasets *rcv1* and *amazon*, since it takes a prohibitively long time to get the results on *rcv1*, and runs out of memory (OOM) on *amazon*. We

[3] $\lceil \cdot \rceil$ denotes the ceiling function and $\lfloor \cdot \rfloor$ denotes the floor function.

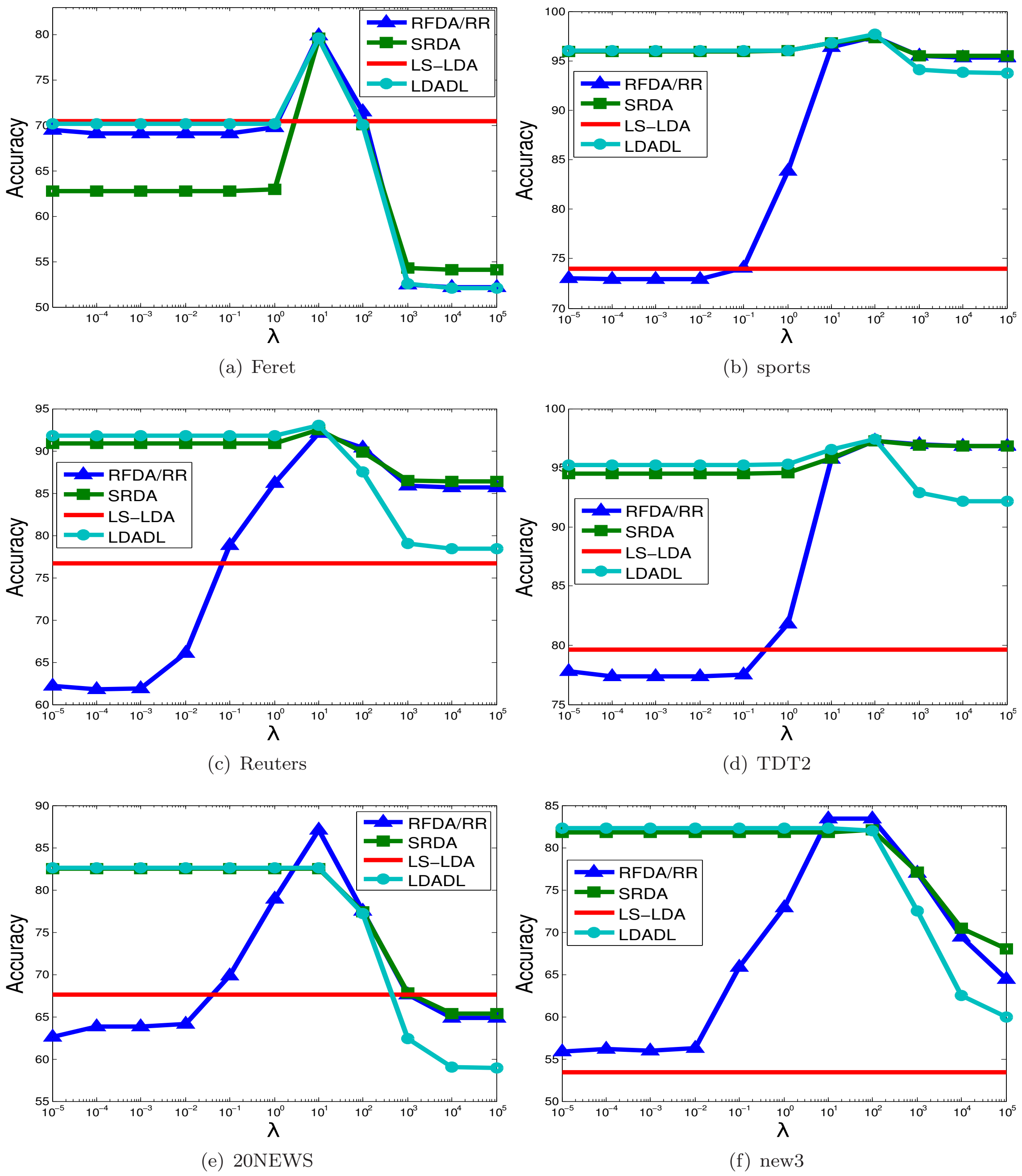


Fig. 1. *Model selection of LDADL: The curves show the classification accuracy of RFDA/RR, SRDA, LS-LDA and LDADL as a function of λ. It is clear that a proper regularization will mitigate data over-fitting. Moreover, SRDA and LDADL have similar performance regardless of the selection of λ and both outperform LS-LDA over a large range of λ.*

observe from Table 4 that LSQR is much more efficient. Therefore, we adopt LSQR as the LS solver in our model.

6.3. Effect of the parameter λ. To investigate the effect of $\lambda \geq 0$, we plot the classification accuracy of LDADL as a function of λ and compare it with LS-LDA that solves (4.6) (i.e. $\lambda = 0$) by computing $\underline{G}^* = \underline{A}^\dagger E$. Because of the similarity among LDADL, RFDA/RR and SRDA, we also plot the classification accuracy of RFDA/RR and SRDA with respective to different λ.

Table 5
Classification accuracy (mean ±standard deviation in percentage) of batch LDA algorithms.

datasets	RLDA	ROLDA	ULDA/QR	IDR/QR	RFDA/RR	SRDA	LS-LDA	LDADL
Feret	72.75±1.55	**85.98±1.85**	70.40±2.05	57.48±2.81	69.75±1.06	62.93±2.13	70.45±2.02	70.13±2.43
sports	**97.17±0.29**	74.44±1.26	74.42±1.29	94.59±0.36	83.90±0.12	95.98±0.29	73.94±1.48	96.06±0.30
Reuters	91.53±0.43	81.65±1.28	70.16±9.45	87.89±0.55	86.19±0.17	90.85±0.52	76.65±4.62	**91.79±0.31**
TDT2	**96.72±0.21**	78.34±1.92	77.47±2.64	95.92±0.32	81.85±0.11	94.54±0.29	79.65±3.35	95.29±0.24
20NEWS	**86.84±0.47**	70.21±0.98	67.70±3.23	67.99±0.66	78.92±0.72	82.47±0.36	67.56±3.35	82.58±0.35
new3	**83.21±0.98**	72.05±0.39	55.97±0.67	75.50±0.90	72.96±0.71	81.81±0.49	53.39±0.91	82.28±0.56
rcv1	OOM	OOM	OOM	89.14±0.07	OOM	89.83±0.04	OOM	**89.86±0.03**
amazon	OOM	OOM	OOM	86.27±0.01	OOM	92.43±0.14	OOM	**92.65±0.03**

Fig. 1 shows the performance of all four algorithms on different λ from

$$\{10^{-5}, 10^{-4}, \cdots, 10^{5}\}.$$

Since the last two datasets *rcv1* and *amazon* are too large and it takes a huge amount of time to compute solutions for all values of λ and all algorithms, we only show the results on the first 6 datasets. It is easy to see that both LDADL and SRDA have similar performance and achieve significantly higher accuracy than LS-LDA over a wide range of λ, which is consistent with the connection between LDADL and SRDA established in Lemma 4.3. Theoretically, we have shown that $\underline{G}^*_{\lambda 1}$ of LDADL converges to $\underline{G}^*$ of LS-LDA as $\lambda \to 0$. This is validated by the results on *Feret* data. However, for other datasets, there is a big difference between LDADL and LS-LDA when λ is close to zero. One reason for this observation is that we terminate LSQR after 20 iterations, thus the resulting solution is actually an approximation to $\underline{G}^*_{\lambda 1}$. According to Lemma 4.3, LDADL and SRDA are closely related, but we observe from Fig. 1 that LDADL is slightly worse than SRDA when λ is larger than a threshold. The underlying statistical mechanism behind this observation is interesting and needs to be studied further. In addition, we see that the accuracy of RFDA/RR, SRDA and LDADL increases first and then drops dramatically when we increase the value of λ, which verifies the claim that a proper regularization will mitigate data over-fitting. *To maintain a good overall performance, we fix $\lambda = 1$ for all three methods as well as ILDADL in the sequel.*

6.4. Comparison with batch LDA algorithms. We compare LDADL with the following seven batch LDA algorithms:

1. Regularized LDA (RLDA) [17, 38], which replaces S_t with $S_t + \alpha I$ for some $\alpha > 0$ to handle the singularity problem. We adopt the more efficient implementation of RLDA proposed in [38]. To select the optimal α, we perform 5-fold cross-validation on the candidate set $\{10^{-5}, 10^{-4.9}, \cdots, 10^{4.9}, 10^{5}\}$.
2. Regularized orthogonal LDA (ROLDA) [10], which is similar to RLDA but selects the regularization parameter α based on the user-specified deviation ϵ between ROLDA solution $G(\alpha)$ and OLDA solution G (i.e., $\|G(\alpha) - G\| \le \epsilon$). As suggested in [10], we select $\epsilon = 10^{-2}$.
3. Uncorrelated LDA based on QR factorization (ULDA/QR) [12, 35].
4. IDR/QR [37], an LDA variant which performs RLDA in the range space of the centroid matrix. The regularization parameter α is set to be 0.5, which produces good overall results in [37].
5. RFDA/RR [39], a regularized discriminant analysis algorithm related to the ridge regression problem (4.14).
6. SRDA [7], an efficient discriminant analysis algorithm based on spectral regression.
7. LS-LDA that solves (4.6) exactly by computing $\underline{G}^* = \underline{A}^\dagger E = Q(R^\dagger)^T E$, where Q and R are from the economical QR factorization of $\underline{A}^T$.

For each algorithm, the optimal LDA transformation is computed using all training data including both initial and incremental training data. The average results of classification performance and CPU time over 10 replications of all algorithms are shown in Table 5 and Table 6, respectively.

Main observations from Table 5 and 6 include the following:

- LDADL achieves the best overall performance in the sense that it strikes a good balance between high classification accuracy and efficiency. On the side of accuracy, RLDA achieves higher accuracy than the rest algorithms for most of the datasets due to the selection of optimal parameter by cross-validation, followed by LDADL. However, cross-validation takes a long time for large datasets, which makes RLDA inefficient. On the side of efficiency, IDR/QR, SRDA and LDADL are much faster

Table 6
Computational time (in seconds) of batch LDA algorithms.

datasets	RLDA	ROLDA	ULDA/QR	IDR/QR	RFDA/RR	SRDA	LS-LDA	LDADL
Feret	1.3e+2	2.1e+0	1.7e+0	**5.3e-1**	3.4e+0	2.2e+1	1.5e+0	2.1e+1
sports	2.9e+3	3.8e+4	3.5e+4	**9.2e-2**	1.1e+3	1.2e+0	3.6e+4	1.3e+0
Reuters	2.5e+3	1.3e+3	1.2e+3	**1.3e-1**	1.2e+3	3.0e+0	1.2e+3	2.7e+0
TDT2	1.9e+3	4.6e+3	4.4e+3	**1.4e-1**	1.7e+3	4.2e+0	4.4e+3	4.3e+0
20NEWS	9.1e+3	6.4e+3	5.0e+3	**1.2e-1**	5.9e+3	3.8e+0	4.9e+3	4.1e+0
new3	6.9e+3	2.2e+5	3.0e+5	**4.9e-1**	1.7e+3	1.2e+1	5.0e+5	1.2e+1
rcv1	OOM	OOM	OOM	**2.6e+0**	OOM	1.7e+2	OOM	1.7e+2
amazon	OOM	OOM	OOM	**3.0e+0**	OOM	9.7e+1	OOM	1.1e+2

than other algorithms, and among these three algorithms LDADL achieves the highest accuracy. For large datasets such as *rcv1* and *amazon*, only IDR/QR, SRDA and LDADL are applicable, other algorithms run out of memory.

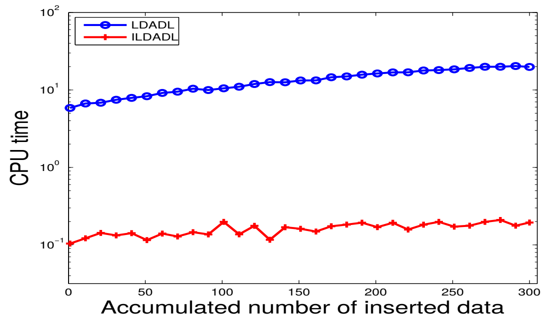
- Although IDR/QR is the most efficient approach, its classification performance is unstable as there is no theoretical relation between the optimization problem adopted by IDR/QR and that of LDA. This can be seen from its accuracy on *Feret*, *20NEWS* and *amazon* datasets.
- SRDA and LDADL have similar performance, which verifies the result in Lemma 4.3. Although SRDA is marginally faster than LDADL on some datasets, LDADL achieves higher accuracy on all datasets.
- The only difference between LS-LDA and LDADL is the Tikhonov regularization, but LDADL achieves much higher accuracy, which implies that over-fitting is a crucial problem in the LDA model and LDADL is capable of alleviating data over-fitting.

6.5. Comparison with incremental LDA algorithms. In this subsection, we compare our newly proposed incremental algorithm ILDADL with three stat-of-the-art incremental LDA algorithms: IDR/QR [37], LS-ILDA [24] and ILDA/SSS [21, 22]. ILDA/SSS has three parameters: two threshold for significant components of the total scatter matrix and the between-class scatter matrix, and one threshold for the discriminative components. Since [21, 22] did not provide a way to select these parameters, we set all three parameters to be 0.1 as [24] did, which enables the algorithm to achieve its best performance. For IDR/QR [37] and LS-ILDA [24] we implement their algorithms in MATLAB; for ILDA/SSS, we use the MATLAB codes provided by the authors [4].

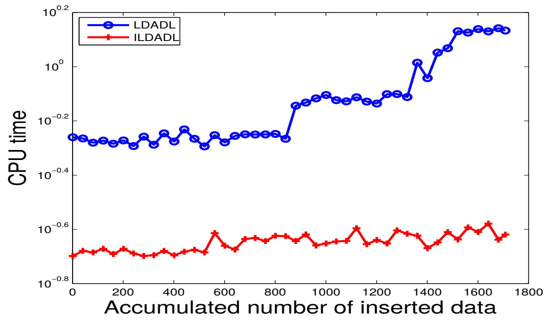
For all incremental algorithms, the optimal transformation is updated from the previously obtained information when a new sample is inserted. The classification accuracy is of the final transformation when all incremental training data have been inserted. The average results of classification performance evaluated on the testing data over 10 replications are shown in Table 7. The execution time for incremental algorithms is the CPU time of updating the transformation matrix when a new sample is inserted. Thus, every time a new sample is inserted there is a corresponding CPU time. The mean execution time of each updating for all algorithms over 10 replications is shown in Fig. 2 and Fig. 3, where the horizontal axis shows the accumulated number of inserted new samples from the incremental training data while the vertical axis indicates the execution time (in seconds).

In Fig. 2, we compare the computational time for no-incremental method–LDADL and incremental method–ILDADL. We see from Fig. 2 that ILDADL is much faster than LDADL and the speedup becomes more prominent as more new samples are inserted. This performance gain is attributed to the decrease of computational complexity of ILDADL, since it only needs to solve a RLS problem with single column on the right hand side while LDADL has to solve a RLS problem with c columns on the right hand side. We also observe that the computational time of LDADL is approximately an increasing step function of the class number c, which is consistent with the formula (4.9). Comparing the accuracy of ILDADL in Table 7 and that of LDADL in Table 5, we notice that they are more or less the same (the largest difference of accuracy is 1.9% on the Feret dataset, but for other dataset the difference is less than 0.7%), which validates the claim that the LDA solution computed by our incremental algorithm is in theory the same as the batch LDA solution.

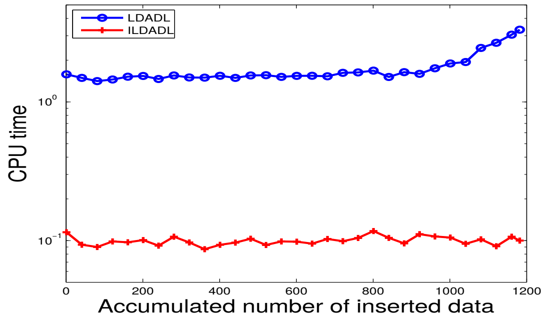
[4] `http://www.iis.ee.ic.ac.uk/~tkkim/code.htm`



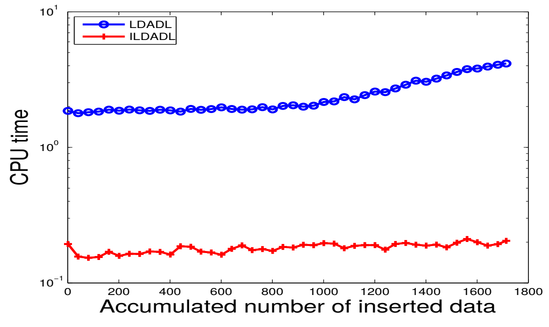
(a) Feret



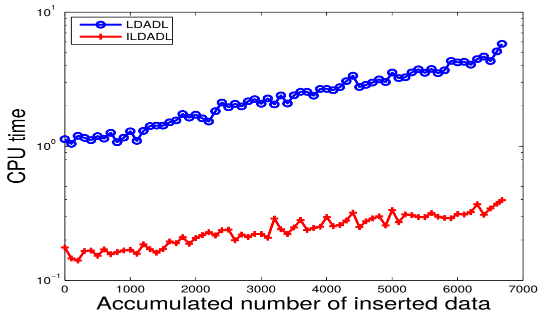
(b) sports



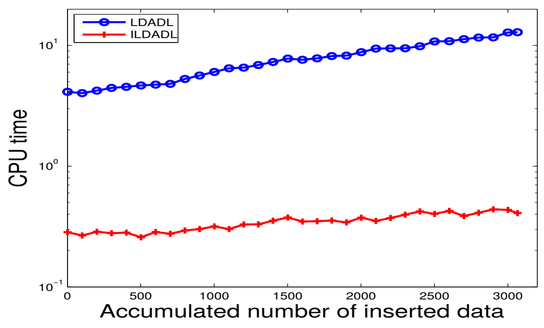
(c) Reuters



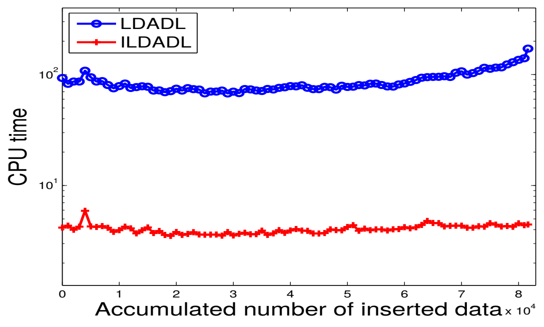
(d) TDT2



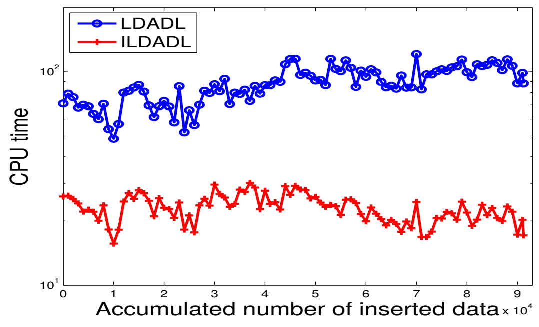
(e) 20NEWS



(f) new3



(g) rcv1



(h) amazon

Fig. 2. *Computational time comparison of LDADL and ILDADL: The curves show the CPU time as functions of the number of inserted samples so far. Since datasets rcv1 and amazon have large number of incremental training data, computing the LDA solution from scratch using LDADL for each new sample takes a huge amount of time. To make the comparison feasible for datasets rcv1 and amazon, we compute the LDA solution using LDADL for each batch of 1000 new samples, while for ILDADL we still record the updating time for each new sample.*

Table 7
Classification accuracy (mean ± standard deviation in percentage) of incremental LDA algorithms: LDA/SSS either takes too much time to get the result (NA) or runs out of memory (OOM) when applied to the last four datasets. LS-ILDA also runs out of memory on the last two datasets.

datasets	ILDADL	IDR/QR	LS-ILDA	LDA/SSS
Feret	68.23±2.30	51.98±1.93	**70.40±2.05**	**70.40±2.05**
sports	**96.03±0.27**	95.04±0.40	74.42±1.29	73.62±0.23
Reuters	**91.78±0.42**	86.79±0.35	73.36±7.31	82.25±1.00
TDT2	95.25±0.31	**95.83±0.55**	77.47±2.64	77.23±0.23
20NEWS	**81.88±0.46**	68.38±0.80	67.70±3.23	NA
new3	**81.44±0.82**	75.24±0.63	59.02±8.20	NA
rcv1	**89.69±0.06**	86.60±0.01	OOM	OOM
amazon	**92.68±0.12**	85.49±0.04	OOM	OOM

In Fig. 3, we plot the computational time of four incremental algorithms.*Since ILDA/SSS either takes too much time to get the result or runs out of memory on the last four datasets, there is no plot for ILDA/SSS on these datasets. Similarly, LS-ILDA also runs out of memory and thus there is no plot on the last two datasets.* We observe that IDR/QR and ILDADL significantly outperform the rest algorithms. Although IDR/QR is faster than ILDADL on most datasets, ILDADL is slightly faster than IDR/QR on datasets with large class number such as *Reuters* and *new3*. Regarding the classification performance, ILDADL is obviously better than other incremental algorithms, as can be seen from Table 7. In addition, we observe from Table 7 that the performance of IDR/QR is unstable and varies over different datasets.

In conclusion, ILDADL is the best among the four compared incremental algorithms in terms of classification accuracy, computational complexity and space complexity. It provides an efficient incremental dimensionality reduction for large-scale datasets.

7. Conclusions. In this paper, we have proposed an incremental algorithm IRLS that is capable of updating the solution to the RLS problem with multiple columns on the right-hand side when a new data is acquired. IRLS has several nice properties: (1) The resulted solution is the exact solution to the RLS problem. (2) It deals with both over-determined ($n \geq d$) and under-determined ($n \leq d$) RLS problems. (3) It is much faster than computing the RLS solution from scratch, and this advantage becomes more pronounced when c is large or when A is highly sparse. We have applied IRLS to supervised dimensionality reduction of large-scale data. In particular, we have considered incremental LDA. This has been accomplished by first proposing a new batch LDA model that relates LDA with RLS, and then applying IRLS to this newly proposed model. Extensive experiments on real-world datasets demonstrate that the new batch LDA algorithm is competitive with the state-of-the-art LDA algorithms and our incremental LDA algorithm is efficient and effective in processing large-scale data.

Acknowledgments. The authors would like to thank the editor and anonymous referees for their valuable comments and suggestions on earlier versions of this paper.

REFERENCES

[1] T. Anderson, *An Introduction to Multivariate Statistical Analysis*, Wiley, 3 ed., 2003.

[2] H. Avron, P. Maymounkov, and S. Toledo, *Blendenpik: Supercharging LAPACK's least-squares solver*, SIAM J. Sci. Comput., 32 (2010), pp. 1217–1236.

[3] M. Baboulin, L. Giraud, S. Gratton, and J. Langou, *Parallel tools for solving incremental dense least squares problems. application to space geodesy*, Journal of Algorithms and Computational Technology, 31 (2009), pp. 117–131.

[4] D. Bertsekas, *Incremental least squares methods and the extended kalman filter*, SIAM J. Opt., 6 (1996), pp. 807–822.

[5] M. Blondel, K. Seki, and K. Uehara, *Block coordinate descent algorithms for large-scale sparse multiclass classification*, Machine Learn., 93 (2013), pp. 31–52.

[6] C. Burges, *Dimension reduction: A guided tour*, Foundations and Trends in Machine Learning, 2 (2009), pp. 275–365.

[7] D. Cai, X. He, and J. Han, *SRDA: An efficient algorithm for large-scale discriminant analysis*, IEEE Trans. Knowledge Data Eng., 20 (2008), pp. 1–12.

[8] A. Cassioli, A. Chiavaioli, C. Manes, and M. Sciandrone, *An incremental least squares algorithm for large scale linear classification*, European J. Oper. Res., 224 (2013), pp. 560–565.

[9] N. Cesa-Bianchi, *Applications of regularized least squares to pattern classification.*, Theoret. Comput. Sci., 382 (2007), pp. 221–231.

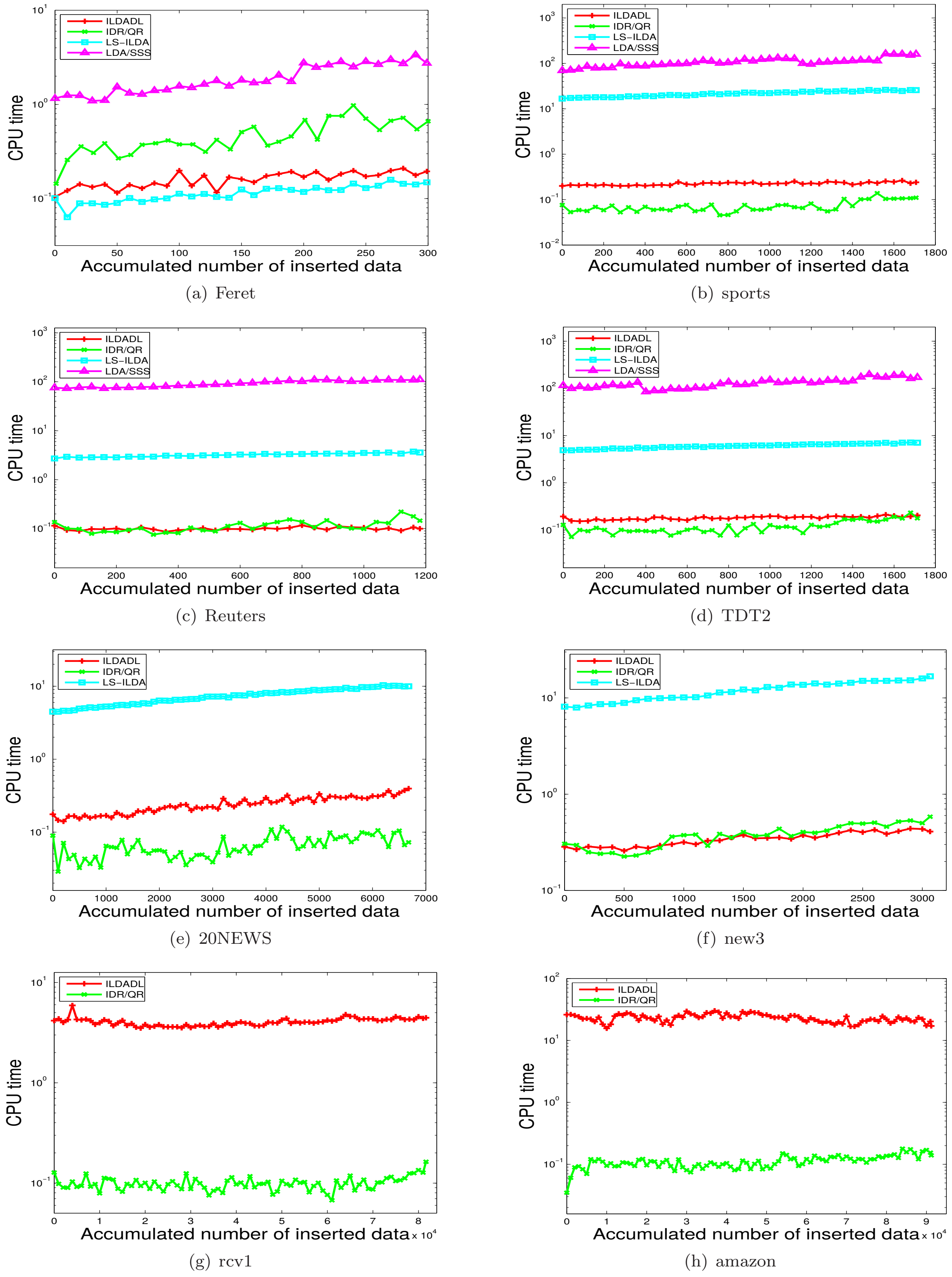


Fig. 3. *Computational time comparison of four incremental LDA algorithms: ILDADL, IDR/QR, LS-ILDA and ILDA/SSS. The curves show the CPU time as functions of the number of inserted samples so far.*

[10] W. Ching, D. Chu, L.-Z. Liao, and X. Wang, *Regularized orthogonal linear discriminant analysis*, Pattern Recognition, 45 (2012), pp. 2719–2732.

[11] D. Chu and S. Goh, *A new and fast orthogonal linear discriminant analysis on undersampled problems*, SIAM J. Sci. Comput., 32 (2010), pp. 2274–2297.

[12] D. Chu, S. Goh, and Y. Hung, *Characterization of all solutions for undersampled uncorrelated linear discriminant analysis problems*, SIAM J. Matrix Anal. Appl., 32 (2011), pp. 820–844.

[13] D. Chu, L.-Z. Liao, M. K. Ng, and X. Wang, *Incremental linear discriminant analysis: A new fast algorithm and comparisons*, IEEE Trans. Neural Netw. Learn. Syst., (2015).

[14] M. Dredze, K. Crammer, and F. Pereira, *Confidence-weighted linear classification*, in Proceedings of the 25th International Conference on Machine Learning, 2008, pp. 264–271.

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

[16] D. Fong and M. Saunders, *LSMR: An iterative algorithm for sparse least-squares problems*, SIAM J. Sci. Comput., 33 (2011), pp. 2950–2971.

[17] J. Friedman, *Regularized discriminant analysis*, J. Amer. Statist. Assoc., 84 (1989), pp. 165–175.

[18] G. Golub and C. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, 3rd ed., 1996.

[19] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, Springer, 2nd ed., 2009.

[20] P. Howland, M. Jeon, and H. Park, *Structure preserving dimension reduction for clustered text data based on the generalized singular value decomposition*, SIAM J. Matrix Anal. Appl., 25 (2003), pp. 165–179.

[21] T. Kim, B. Stenger, J. Kittler, and R. Cipolla, *Incremental linear discriminant analysis using sufficient spanning sets and its applications*, Int. J. Comput. Vis., 91 (2011), pp. 216–232.

[22] T. Kim, S. Wong, B. Stenger, J. Kittler, and R. Cipolla, *Incremental linear discriminant analysis using sufficient spanning set approximations*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2007.

[23] F. la Torre, *A least-squares framework for component analysis*, IEEE Trans. Pattern Anal. Machine Intell., 34 (2012), pp. 1041–1055.

[24] L. Liu, Y. Jiang, and Z. Zhou, *Least square incremental linear discriminant analysis*, in Proceedings of the 9th IEEE International Conference on Data Mining, 2009.

[25] X. Meng, M. Saunders, and M. Mahoney, *LSRN: A parallel iterative solver for strongly over- or under-determined systems*, SIAM J. Sci. Comput., 36 (2014), pp. C95–C118.

[26] C. Paige and M. Saunders, *LSQR: An algorithm for sparse linear equations and sparse least squares.*, ACM Trans. Math. Softw., 8 (1982), pp. 43–71.

[27] S. Pang, S. Ozawa, and N. Kasabov, *Incremental linear discriminant analysis for classification of data streams*, IEEE Trans. Syst., Man, Cybern. A, 35 (2005), pp. 905–914.

[28] C. Park and H. Park, *A relationship between linear discriminant analysis and the generalized minimum squared error solution*, SIAM J. Matrix Anal. Appl., 27 (2005), pp. 474–492.

[29] R. Polikar, L. Udpa, and V. Honavar, *Learn ++: An incremental learning algorithm for supervised neural networks*, IEEE Trans. Syst., Man, Cybern. C, Special Issue on Knowledge Management, 31 (2001), pp. 497–508.

[30] Bernhard S. and A. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, 2002.

[31] C. Saunders, A. Gammerman, and V. Vovk, *Ridge regression learning algorithm in dual variables*, in Proceedings of the 15th International Conference on Machine Learning, 1998, pp. 515–521.

[32] A. Tikhonov and V. Arsenin, *Solutions of ill-posed problems*, V. H. Winston & Sons, 1977.

[33] TREC, *Text retrieval conference.* `http://trec.nist.gov`, 1999.

[34] V. Vovk, *Competitive on-line linear regression*, in Advances in neural information processing systems 10, MIT Press, 1998, pp. 364–370.

[35] J. Ye, *Characterization of a family of algorithms for generalized discriminant analysis on undersampled problems*, J. Mach. Learn. Res., 6 (2005), pp. 483–502.

[36] ———, *Least squares linear discriminant analysis*, in Proceedings of the 24th International Conference on Machine Learning, 2007, pp. 1087–1094.

[37] J. 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 Data Eng., 17 (2005), pp. 1208–1222.

[38] J. Ye, T. Xiong, Q. Li, R. Janardan, J. Bi, V. Cherkassky, and C. Kambhamettu, *Efficient model selection for regularized linear discriminant analysis*, in Proceedings of the 15th ACM International Conference on Information and Knowledge Management, 2006, pp. 532–539.

[39] Z. Zhang, G. Dai, C. Xu, and M. Jordan, *Regularized discriminant analysis, ridge regression and beyond*, J. Mach. Learn. Res., 11 (2010), pp. 2199–2228.

[40] H. Zhao and P. Yuen, *Incremental linear discriminant analysis for face recognition*, IEEE Trans. Syst., Man, Cybern. B, 38 (2008), pp. 210–221.

[41] A. Zouzias and N. Freris, *Randomized extended kaczmarz for solving least squares*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 773–793.