

Regularized Locality Preserving Indexing via Spectral Regression*

Deng Cai[†], Xiaofei He[‡], Wei Vivian Zhang[‡], Jiawei Han[†]

[†]University of Illinois at Urbana-Champaign

[‡]Yahoo!

{dengcai2, han}@cs.uiuc.edu, {hex, zhangv}@yahoo-inc.com

ABSTRACT

We consider the problem of document indexing and representation. Recently, Locality Preserving Indexing (LPI) was proposed for learning a compact document subspace. Different from Latent Semantic Indexing (LSI) which is optimal in the sense of global Euclidean structure, LPI is optimal in the sense of local manifold structure. However, LPI is not efficient in time and memory which makes it difficult to be applied to very large data set. Specifically, the computation of LPI involves eigen-decompositions of two dense matrices which is expensive. In this paper, we propose a new algorithm called *Regularized Locality Preserving Indexing* (RLPI). Benefit from recent progresses on spectral graph analysis, we cast the original LPI algorithm into a regression framework which enable us to avoid eigen-decomposition of dense matrices. Also, with the regression based framework, different kinds of regularizers can be naturally incorporated into our algorithm which makes it more flexible. Extensive experimental results show that RLPI obtains similar or better results comparing to LPI and it is significantly faster, which makes it an efficient and effective data preprocessing method for large scale text clustering, classification and retrieval.

Categories and Subject Descriptors

H.3.1 [Information Storage and Retrieval]: Content Analysis and Indexing—*Indexing methods*; I.5.2 [Pattern Recognition]: Description Methodology—*Feature evaluation and selection*

General Terms

Algorithms, Performance, Theory

Keywords

Regularized Locality Preserving Indexing, Document Representation and Indexing, Dimensionality Reduction

*The work was supported in part by the U.S. National Science Foundation NSF IIS-05-13678, NSF BDI-05-15813 and MIAS (a DHS Institute of Discrete Science Center for Multimodal Information Access and Synthesis). Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the funding agencies.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CIKM'07, November 6–8, 2007, Lisboa, Portugal.

Copyright 2007 ACM 978-1-59593-803-9/07/0011 ...\$5.00.

1. INTRODUCTION

Document representation and indexing have been a key problem for document analysis and processing, such as clustering, classification and retrieval [8],[12],[23]. If we denote by *document space* the set of all the documents, different indexing algorithms see different structures of the document space. The Vector Space Model (VSM) might be one of the most popular model for document representation. Each document is represented as a *bag of words*. Correspondingly, the document space is associated with a Euclidean structure and the inner product (or, cosine similarity) is used as the standard similarity measure for documents. Unfortunately, VSM suffers from some problems such as *synonymy* and *polysemy*.

To deal with these problems, Latent Semantic Indexing (LSI) was proposed [8]. Given a term-document matrix X , LSI applies Singular Value Decomposition (SVD) to project the document vectors into a subspace so that cosine similarity can accurately represent semantic similarity. LSI aims to find the best subspace approximation to the original document space in the sense of minimizing the global reconstruction error. The basis functions obtained by LSI are the eigenvectors of the matrix XX^T . It would be important to note that LSI is different from Principal Component Analysis (PCA) in that XX^T is generally not the data covariance matrix. In fact, this occurs only when the documents has a zero mean. LSI received a lot of attentions during these years and many variants of LSI have been proposed [1], [13].

LSI is optimal in the sense of preserving the global geometric structure of the document space (inner product). However, it might not be optimal in the sense of discrimination. Specifically, LSI might not be optimal in separating documents with different topics. Recently, Locality Preserving Indexing (LPI) is proposed to discover the discriminant structure of the document space. It has shown that it can have more discriminative power than LSI. A reasonable assumption behind LPI is that, close inputs should have similar topics. The detailed discriminant analysis of LPI can be found in [12], [3]. Previous experiments on document clustering have demonstrated the effectiveness of LPI [4]. However, the computational complexity of LPI is very expensive because it involves eigen-decompositions of two dense matrices. It is almost infeasible to apply LPI on very large data set.

In this paper, we propose a new algorithm called **Regularized Locality Preserving Indexing** (RLPI). RLPI is fundamentally based on LPI. It shares the same locality preserving character as LPI, but can be efficiently computed. Specifically, RLPI decomposes the LPI problem as a graph embedding problem plus a regularized least squares problem. Such modification avoids eigen-decomposition of dense matrices and can significantly reduce both time and memory cost in computation. Moreover, with a specifically designed graph in supervised situation, the graph embedding problem in RLPI

becomes trivial and RLPI only needs to solve the regularized least squares problem which is a further saving of time and memory. Such properties make RLPI an efficient data processing method for large scale text clustering, classification, retrieval, *etc.*

The rest of the paper is organized as follows: in Section 2, we give a brief review of LSI and LPI. Section 3 introduces our algorithm and give a theoretical analysis of the algorithm. Extensive experimental results on document clustering and categorization are presented in Section 4. Finally, we provide some concluding remarks and suggestions for future work in Section 5.

2. A BRIEF REVIEW OF LSI AND LPI

In this Section, we provide a brief review of Latent Semantic Indexing (LSI) [8] and Locality Preserving Indexing (LPI) [12], followed by a computational complexity analysis of these two algorithms.

2.1 Latent Semantic Indexing

LSI is one of the most popular algorithms for document indexing [8]. It is fundamentally based on Singular Value Decomposition (SVD) and try to extract the most representative features (minimize the reconstruction error). Given a set of documents $\{\mathbf{x}_i\}_{i=1}^m \subset \mathbb{R}^n$, which can be represented as a term-document matrix $X = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m]$. Suppose the rank of X is r , LSI decompose the X by using SVD as follows:

$$X = U\Sigma V^T, \quad (1)$$

where $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_r)$ and $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r$ are the singular values of X , $U = [\mathbf{u}_1, \dots, \mathbf{u}_r]$ and \mathbf{u}_i 's are called left singular vectors, $V = [\mathbf{v}_1, \dots, \mathbf{v}_r]$ and \mathbf{v}_i 's are called right singular vectors. We have $U^T U = V^T V = I$. LSI uses the first d vectors in U as the transformation matrix to embed the original document into a d -dimensional subspace. It can be easily checked that the column vectors of U are the eigenvectors of XX^T . It would be important to note that XX^T becomes the data covariance matrix if the data points have a zero mean, i.e. $X\mathbf{e} = \mathbf{0}$ where $\mathbf{e} = [1, \dots, 1]^T$. In such a case, LSI is identical to Principal Component Analysis [9]. More details on theoretical interpretations of LSI using SVD can refer to [2], [19].

2.2 Locality Preserving Indexing

Different from LSI which aims to extract the most representative features, LPI aims to extract the most discriminative features. Given a similarity matrix W , LPI can be obtained by solving the following minimization problem:

$$\begin{aligned} \mathbf{a}^* &= \arg \min_{\mathbf{a}^T XDX^T \mathbf{a} = 1} \sum_{i=1}^m \sum_{j=1}^m (\mathbf{a}^T \mathbf{x}_i - \mathbf{a}^T \mathbf{x}_j)^2 W_{ij} \\ &= \arg \min_{\mathbf{a}^T XDX^T \mathbf{a} = 1} \mathbf{a}^T X L X^T \mathbf{a} \end{aligned} \quad (2)$$

where D is a diagonal matrix whose entries are column sums of W ($D_{ii} = \sum_j W_{ji}$) and $L = D - W$ is the *graph Laplacian* [7]. LPI constructs the similarity matrix W as:

$$W_{ij} = \begin{cases} \frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}, & \text{if } \mathbf{x}_i \in N_p(\mathbf{x}_j) \text{ or } \mathbf{x}_j \in N_p(\mathbf{x}_i) \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

where $N_p(\mathbf{x}_i)$ is the set of p nearest neighbors of \mathbf{x}_i . Thus, the objective function in LPI incurs a heavy penalty if neighboring points \mathbf{x}_i and \mathbf{x}_j are mapped far apart. Therefore, minimizing it is an attempt to ensure that if \mathbf{x}_i and \mathbf{x}_j are "close" then $y_i (= \mathbf{a}^T \mathbf{x}_i)$ and $y_j (= \mathbf{a}^T \mathbf{x}_j)$ are close as well [12]. Finally, the basis functions of LPI

are the eigenvectors associated with the smallest eigenvalues of the following generalized eigen-problem:

$$X L X^T \mathbf{a} = \lambda X D X^T \mathbf{a}.$$

Since we have $L = D - W$, it is easy to check that the **minimization** problem in Eqn. (2) is equivalent to the following **maximization** problem:

$$\mathbf{a}^* = \arg \max_{\mathbf{a}^T X D X^T \mathbf{a} = 1} \mathbf{a}^T X W X^T \mathbf{a} \quad (4)$$

and the optimal \mathbf{a} 's are also the **maximum** eigenvectors of eigen-problem:

$$X W X^T \mathbf{a} = \lambda X D X^T \mathbf{a}, \quad (5)$$

which in some cases can provide a more numerically stable solution [10]. In the following of our paper, we will discuss LPI based on this formulation.

To get a stable solution of the above eigen-problem, the matrix XDX^T is required to be non-singular [10]. However, in many text analysis tasks, the number of features (terms) is usually larger than the number of documents which indicates the singularity of XDX^T . In such a case, SVD can be used to solve this problem [4].

Suppose we have the SVD decomposition of X shown in Eqn. (1). Let $\tilde{X} = U^T X = \Sigma V^T$ and $\mathbf{b} = U^T \mathbf{a}$, we have

$$\mathbf{a}^T X D X^T \mathbf{a} = \mathbf{a}^T U \Sigma V^T D V \Sigma U^T \mathbf{a} = \mathbf{b}^T \tilde{X} D \tilde{X}^T \mathbf{b}$$

and

$$\mathbf{a}^T X L X^T \mathbf{a} = \mathbf{a}^T U \Sigma V^T L V \Sigma U^T \mathbf{a} = \mathbf{b}^T \tilde{X} L \tilde{X}^T \mathbf{b}.$$

Now, the objective function of LPI in (4) can be rewritten as:

$$\mathbf{b}^* = \arg \max_{\mathbf{b}^T \tilde{X} D \tilde{X}^T \mathbf{b} = 1} \mathbf{b}^T \tilde{X} W \tilde{X}^T \mathbf{b}.$$

and the optimal \mathbf{b} 's are the maximum eigenvectors of eigen-problem:

$$\tilde{X} W \tilde{X}^T \mathbf{b} = \lambda \tilde{X} D \tilde{X}^T \mathbf{b}. \quad (6)$$

It is easy to check that $\tilde{X} D \tilde{X}^T$ is nonsingular and the above eigen-problem can be stably solved. After we get \mathbf{b}^* , the \mathbf{a}^* can be obtained by solving a set of linear equations systems $U^T \mathbf{a} = \mathbf{b}^*$. By noticing that given U , \mathbf{b}^* , there will be infinitely many solutions of \mathbf{a} which satisfy this equations system¹. Among all these solutions,

$$\mathbf{a}^* = U \mathbf{b}^* \quad (7)$$

is obviously one of them. This approach has been used in ordinary LPI algorithm [12], [4] to solve the singularity problem.

2.3 Computational Complexity Analysis

Now let us analyze the computational complexity of LSI and LPI. We consider the case that the number of features (n) is larger than the number of documents (m), which is usually the case for text processing tasks. The term *flam* [21], a compound operation consisting of one addition and one multiplication, is used to present operation counts.

Since the term-document matrix X is usually sparse, we can use the Lanczos algorithm to efficiently compute the first d singular vectors of X . Let s denote the average number of non-zero features per document, the Lanczos algorithm requires $2q_1 d m (s + 8)$ flam, where q_1 is the number of iterations in Lanczos [22]. Thus,

¹Unless $n < m$ and $\text{rank}(X) = n$. In this case, U will be an orthogonal matrix and there is an unique solution of equation $U^T \mathbf{a} = \mathbf{b}^*$ which is exactly $U \mathbf{b}^*$.

computing the first d projective functions in LSI has linear-time complexity with respect to m .

The major computations in LPI include three steps:

1. p -nearest neighbor graph construction as in Eqn. (3). This step requires around $\frac{1}{2}m^2s + 2ms + m^2 \log m$ flam. $\frac{1}{2}m^2s + 2ms$ is used to calculate the pairwise cosine similarity and $m^2 \log m$ is used for p -nearest neighbor finding for all the m documents. We also need mp memory to store the p -nearest neighbor graph matrix.
2. A complete SVD decomposition of term document matrix as in Eqn. (1), which requires $\frac{3}{2}m^2s + \frac{9}{2}m^3$ flam [22]. Although the term-document matrix is usually sparse, both the left and right singular vector matrices are dense which requires $mn + m^2$ memory.
3. Computing the matrix $\tilde{X}W\tilde{X}^T$ ($\tilde{X}D\tilde{X}^T$) and computing the first d eigenvectors of the generalized eigen-problem in Eqn. (6). If X is of full rank, \tilde{X} will be a $m \times m$ dense matrix and computing $\tilde{X}W\tilde{X}^T$ ($\tilde{X}D\tilde{X}^T$) requires at least $2m^3$ flam. We can also use the Lanczos algorithm to compute the first d eigenvectors with $q_1dm^2 + \frac{1}{6}m^3$ flam [22]. This step at least requires additional nd memory to store the projective functions.

Thus, the overall time complexity of LPI measured by flam is at least

$$2m^2s + m^2 \log m + q_1dm^2 + \frac{20}{3}m^3$$

which is cubic-time complexity with respect to m . Besides storing the term-document matrix, the additional memory requirement of LPI is

$$mp + m^2 + mn + nd$$

Such a high computational cost (both on time and memory) makes it infeasible to apply LPI on document sets with large size.

3. REGULARIZED LOCALITY PRESERVING INDEXING

Although LPI can learn a compact document representation which is beneficial for many text analysis tasks such as clustering [4] and categorization, the high computational cost restricts it to be applied to large scale data sets. In this section, we describe our approach which can overcome this difficulty.

3.1 The Algorithm

In order to solve the eigen-problem in Eqn. (5) efficiently, we use the following theorem:

THEOREM 1. *Let \mathbf{y} be the eigenvector of eigen-problem*

$$W\mathbf{y} = \lambda D\mathbf{y} \quad (8)$$

with eigenvalue λ . If $X^T\mathbf{a} = \mathbf{y}$, then \mathbf{a} is the eigenvector of eigen-problem in Eqn. (5) with the same eigenvalue λ .

PROOF. We have $W\mathbf{y} = \lambda D\mathbf{y}$. At the left side of Eqn. (5), replace $X^T\mathbf{a}$ by \mathbf{y} , we have

$$XWX^T\mathbf{a} = XW\mathbf{y} = X\lambda D\mathbf{y} = \lambda XD\mathbf{y} = \lambda XDX^T\mathbf{a}$$

Thus, \mathbf{a} is the eigenvector of eigen-problem Eqn. (5) with the same eigenvalue λ . \square

Theorem 1 shows that instead of solving the eigen-problem in Eqn. (5), the LPI basis functions can be acquired through two steps:

1. Solve the eigen-problem in Eqn. (8) to get \mathbf{y} .
2. Find \mathbf{a} which satisfies $X^T\mathbf{a} = \mathbf{y}$. In reality, such \mathbf{a} might not exist. A possible way is to find \mathbf{a} which can best fit the equation in the least squares sense:

$$\mathbf{a} = \arg \min_{\mathbf{a}} \sum_{i=1}^m (\mathbf{a}^T \mathbf{x}_i - y_i)^2 \quad (9)$$

where y_i is the i -th element of \mathbf{y} .

The advantages of this two-step approach are as follows:

1. The matrix D is guaranteed to be positive definite and therefore the eigen-problem in Eqn. (8) can be stably solved. Moreover, both W and D are sparse matrices. The top d eigenvectors of eigen-problem in Eqn. (8) can be efficiently calculated.
2. There exist many efficient iterative algorithms (*e.g.*, LSQR [18]) that can handle very large scale least squares problems.

In text processing tasks, the number of documents is often smaller than the number of features (words). Thus, the minimization problem (9) is *ill posed*. We may have infinite many solutions for the linear equations system $X^T\mathbf{a} = \mathbf{y}$ (the system is underdetermined). The most popular way to solve this problem is to impose a penalty on the norm of \mathbf{a} :

$$\mathbf{a} = \arg \min_{\mathbf{a}} \left(\sum_{i=1}^m (\mathbf{a}^T \mathbf{x}_i - y_i)^2 + \alpha \|\mathbf{a}\|^2 \right) \quad (10)$$

This is so called regularization and is well studied in statistics [11]. The $\alpha \geq 0$ is a parameter to control the amounts of shrinkage. Now we can see the third advantage of the two-step approach:

3. Since the regression is used as a building block, the regularization techniques can be easily incorporated and produce more stable and meaningful solutions, especially when there exist a large number of features [11].

The above two-step approach essentially performs regression after the spectral analysis of the graph. Therefore, it can be named as *Spectral Regression* (SR) [5]. And we named our new algorithm as *Regularized Locality Preserving Indexing* (RLPI). Given a set of documents $\{\mathbf{x}_i\}_{i=1}^m \subset \mathbb{R}^n$, the algorithmic procedure of RLPI is stated below:

1. **Adjacency graph construction:** Let G denote a graph with m vertices, each vertex represents a document. Let W be a symmetric $m \times m$ matrix with W_{ij} having the weight of the edge joining vertices i and j .

$$W_{ij} = \begin{cases} \frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}, & \text{if } \mathbf{x}_i \in N_p(\mathbf{x}_j) \text{ or } \mathbf{x}_j \in N_p(\mathbf{x}_i) \\ 0, & \text{otherwise.} \end{cases} \quad (11)$$

where $N_p(\mathbf{x}_i)$ is the set of p nearest neighbors of \mathbf{x}_i .

2. **Eigen decomposition:** Solve the eigen-problem

$$W\mathbf{y} = \lambda D\mathbf{y}, \quad (12)$$

where D is a diagonal matrix whose entries are column (or row, since W is symmetric) sums of W , $D_{ii} = \sum_j W_{ji}$. Let $\{\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_d\}$ be the $d+1$ eigenvectors with respect to the $d+1$ maximum eigenvalues $\lambda_0 \geq \lambda_1 \geq \dots \geq \lambda_d$. It is easy to check that $\lambda_0 = 1$ and \mathbf{y}_0 is a vector of all 1 [7].

3. **Regularized least squares:** Find d vectors $\mathbf{a}_1, \dots, \mathbf{a}_d \in \mathbb{R}^n$, where \mathbf{a}_j ($j = 1, \dots, c-1$) is the solution of regularized least squares problem:

$$\mathbf{a}_j = \arg \min_{\mathbf{a}} \left(\sum_{i=1}^m (\mathbf{a}^T \mathbf{x}_i - y_i^j)^2 + \alpha \|\mathbf{a}\|^2 \right) \quad (13)$$

where y_i^j is the i -th element of \mathbf{y}_j .

4. **RLPI Embedding:** Let $A = [\mathbf{a}_1, \dots, \mathbf{a}_d]$, the embedding is as follows:

$$\mathbf{x} \rightarrow \mathbf{z} = A^T \mathbf{x}$$

where \mathbf{z} is a d -dimensional representation of the document \mathbf{x} and A is the transformation matrix.

3.2 Theoretical Analysis

The regularized least squares in Eqn. (10) can be rewritten in the matrix form as:

$$\mathbf{a} = \arg \min_{\mathbf{a}} \left((X^T \mathbf{a} - \mathbf{y})^T (X^T \mathbf{a} - \mathbf{y}) + \alpha \mathbf{a}^T \mathbf{a} \right). \quad (14)$$

Requiring the derivative of right side with respect to \mathbf{a} vanish, we get

$$\begin{aligned} (XX^T + \alpha I)\mathbf{a} &= X\mathbf{y} \\ \Rightarrow \mathbf{a} &= (XX^T + \alpha I)^{-1} X\mathbf{y} \end{aligned} \quad (15)$$

When $\alpha > 0$, this regularized solution will not satisfy the linear equations system $X^T \mathbf{a} = \mathbf{y}$ and \mathbf{a} will not be the eigenvector of eigen-problem in Eqn. (5). It is interesting and important to see when (15) gives the exact solutions of eigen-problem (5). Specifically, we have the following theorem:

THEOREM 2. *Suppose \mathbf{y} is the eigenvector of eigen-problem in Eqn. (5), if \mathbf{y} is in the space spanned by row vectors of X , the corresponding projective function \mathbf{a} calculated in Eqn. (15) will be the eigenvector of eigen-problem in Eqn. (5) as α decreases to zero.*

PROOF. See Appendix A. \square

In text processing tasks, the number of documents is often smaller than the number of features (words) and the document vectors are usually linearly independent, i.e., $\text{rank}(X) = m$. In this case, we will have a stronger conclusion which is shown in the following Corollary.

COROLLARY 3. *If the document vectors are linearly independent, i.e., $\text{rank}(X) = m$, all the projective functions calculated by Eqn. (15) are the eigenvectors of eigen-problem in Eqn. (5) as α decreases to zero. Thus, the solutions of RLPI are identical to the solutions of LPI.*

PROOF. See Appendix B. \square

3.3 RLPI in Supervised Situation

In supervised learning tasks, e.g., text categorization, the label information can be used to build the graph W . In this subsection, we will examine how to build the graph W with label information to further reduce the computational cost of RLPI.

Suppose the m documents $\{\mathbf{x}_i\}_{i=1}^m$ belong to c classes. Let m_j be the number of documents in the j -th class, $\sum_{j=1}^c m_j = m$. With label information available, a natural way to define W can be:

$$W_{ij} = \begin{cases} 1, & \text{if } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ share the same label;} \\ 0, & \text{otherwise.} \end{cases} \quad (16)$$

To simplify our exposition, we assume that the documents $\{\mathbf{x}_i\}_{i=1}^m$ are ordered according to their labels. It is easy to check that the matrix W defined in Eqn. (16) has a block-diagonal structure

$$W = \begin{bmatrix} W^{(1)} & 0 & \cdots & 0 \\ 0 & W^{(2)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & W^{(c)} \end{bmatrix} \quad (17)$$

where $W^{(j)}$ is a $m_j \times m_j$ matrix with all the elements equal to 1. We also have the D as the diagonal matrix. Thus, the eigenvalues and eigenvectors of $W\mathbf{y} = \lambda D\mathbf{y}$ are the union of the eigenvalues and eigenvectors of its blocks (the latter padded appropriately with zeros) [10]:

$$W^{(j)}\mathbf{y}^{(j)} = \lambda D^{(j)}\mathbf{y}^{(j)}.$$

It is straightforward to show that the above eigen-problem has the eigenvector $\mathbf{e}^{(j)} \in \mathbb{R}^{m_j}$ associated with the eigenvalue 1, where $\mathbf{e}^{(j)} = [1, 1, \dots, 1]^T$ [7]. Also $\text{Rank}(W^{(j)}) = 1$, there is only one non-zero eigenvalue of $W^{(j)}$. Thus there are exactly c eigenvectors of eigen-problem $W\mathbf{y} = \lambda D\mathbf{y}$. They are

$$\mathbf{y}_j = \left[\underbrace{0, \dots, 0}_{\sum_{i=1}^{j-1} m_i}, \underbrace{1, \dots, 1}_{m_j}, \underbrace{0, \dots, 0}_{\sum_{i=j+1}^c m_i} \right]^T. \quad (18)$$

These eigenvectors correspond to the same eigenvalue 1. Since 1 is a repeated eigenvalue, we could just pick any other c orthogonal vectors in the space spanned by $\{\mathbf{y}_j\}$ in Eqn. (18), and define them to be our c eigenvectors [10]. The vector of all ones is naturally in the spanned space. This vector is useless since the corresponding projective function will embed all the documents to the same point. In reality, we can pick the vector of all ones as our first eigenvector and use Gram-Schmidt process to get the remaining $c-1$ orthogonal eigenvectors. The vector of all ones can then be removed.

The above analysis shows that with the W defined as in Eqn. (16) in supervised case, the first two steps of RLPI become trivial. We can directly get the \mathbf{y} 's which is a significant saving of both time and memory for RLPI computation. It makes RLPI applicable for very large scale supervised learning tasks.

It is important to note that the values of the i -th and j -th entries of any vector \mathbf{y} in the space spanned by $\{\mathbf{y}_t\}$ in Eqn. (18) are the same as long as \mathbf{x}_i and \mathbf{x}_j belong to the same class. Thus the i -th and j -th rows of Y are the same, where $Y = [\mathbf{y}_1, \dots, \mathbf{y}_{c-1}]$. Corollary (3) shows that when the sample vectors are linearly independent, the $c-1$ projective functions of LPI are exactly the solutions of the $c-1$ linear equations systems $X^T \mathbf{a}_j = \mathbf{y}_j$. Let $A = [\mathbf{a}_1, \dots, \mathbf{a}_{c-1}]$ be the transformation matrix which embeds the data points into the LPI subspace as:

$$A^T X = Y^T.$$

The columns of matrix Y^T are the embedding results of samples in the LPP subspace. Thus, the documents with the same label are corresponding to the same point in the LPI subspace when the document vectors are linearly independent. This property will be very useful for designing efficient classifiers: We simply need to compare the distances of a test point to the centroids of each class in the subspace.

3.4 Computational Complexity Analysis

Now let us analyze the computational cost of RLPI. Let s denote the average number of non-zero features per document and we need calculate d projective functions in RLPI. The RLPI computation involves three steps:

Table 1: Computational cost of LPI and RLPI

		operation counts (<i>flam</i> [21])	memory (besides the term-document matrix)
Unsupervised	LPI	$m^2(2s + \log m + t_1d) + \frac{20}{3}m^3$	$m(p + m + n) + nd$
	RLPI	$m^2(\frac{1}{2}s + \log m) + md(pt_1 + 8t_1 + 2t_2s) + 5ndt_2$	$m(p + d + 1) + n(d + 2)$
Supervised	LPI	$m^2(\frac{3}{2}s + t_1c) + \frac{20}{3}m^3$	$m(m + n) + nc$
	RLPI	$mc(2t_2s + c) + 5nct_2$	$m(c + 1) + n(c + 2)$

m : the number of documents.

n : the number of features. We consider the case that $n > m$.

s : the average number of non-zero features per document.

d : the number of projective functions.

p : the number of nearest neighbors.

c : the number of classes.

t_1 : the number of iterations in Lanczos.

t_2 : the number of iterations in LSQR.

1. p -nearest neighbor graph construction in Eqn. (11) which requires around $\frac{1}{2}m^2s + 2ms + m^2 \log m$ flam and mp memory.
2. Computing the first d eigenvectors of the generalized eigenproblem in Eqn. (12). The W is sparse and has around mp non-zero entries. The first d eigenvectors can be calculated by the Lanczos algorithm within $dq_1m(p+8)$ flam, where q_1 is the number of iterations in Lanczos [22]. This step needs dm memory to store d eigenvectors.
3. Solving d regularized least squares problems in Eqn. (13). The iterative algorithm LSQR [18] can be used to efficiently solve these regularized least squares problems. In each iteration, LSQR needs to compute two matrix-vector products in the form of $X\mathbf{p}$ and $X^T\mathbf{q}$. The remaining work load of LSQR in each iteration is $3m + 5n$ flam [17]. Thus, the time cost of LSQR in each iteration is $2ms + 3m + 5n$. If LSQR stops after t iterations², the time cost is $t(2ms + 3m + 5n)$. Finally, the total time cost for d projective functions is $dt(2ms + 3m + 5n)$. Besides term-document matrix X , LSQR needs $m + 2n$ additional memory [17]. Finally, the additional memory cost in this step is $m + 2n + dn$, with dn to store the projective functions.

In supervised case, the first two steps are not necessary. We can directly get the responses through a Gram-Schmidt process on \mathbf{y}_j in Eqn. (18) which requires $(mc^2 - \frac{1}{3}c^3)$ flam [21].

We summarize our complexity analysis of RLPI in Table 1, together with LPI. The main conclusions include:

- LPI has cubic-time complexity with respect to m . While RLPI has quadratic-time complexity in unsupervised case and linear-time complexity in supervised case.
- The term-document matrix is usually sparse. However, LPI need the **complete** SVD decomposition, which can not get any benefit from the sparseness of the term-document matrix. Moreover, the left and right singular matrices are both dense which require $m(m+n)$ memory space to hold. They can not be fit into the memory when m is large. On the other hand, RLPI can fully explore the sparseness of the term-document matrix and gain significant computational saving on both time and memory. RLPI can successfully applied as long as the term-document matrix X can be fit into the memory.

²LSQR converges very fast [17]. In our experiments, 15 iterations are enough.

- Even the term-document matrix X is too large to be fit into the memory, RLPI can still be applied with some reasonable disk I/O. This is because in each iteration of LSQR, we only need to calculate two matrix-vector products in the form of $X\mathbf{p}$ and $X^T\mathbf{q}$, which can be easily implemented with X and X^T stored on the disk.

4. EXPERIMENTAL RESULTS

In this section, several experiments on TDT2 and 20Newsgroups data sets were performed to show the effectiveness of our proposed algorithm. All of our experiments have been performed on an Intel Pentium D 3.20GHz Linux machine with 2GB memory. For the purpose of reproducibility, we provide our algorithms and data sets used in these experiments at:

<http://www.cs.uiuc.edu/homes/dengcai2/Data/data.html>

4.1 Text Clustering on TDT2

Clustering is one of most crucial techniques to organize the documents in an unsupervised manner. The ordinary clustering algorithms (e.g. K-means) can be performed in the original document space or in the reduced document space (by using the dimensionality reduction algorithms, e.g., LSI, LPI). In this experiment, we investigate the use of dimensionality reduction algorithms for text clustering. The following six methods are compared in the experiment:

- K-means on original term-document matrix, which is treated as our baseline (denoted as **Baseline**)
- K-means after LSI (denoted as **LSI**)
- K-means after LPI (denoted as **LPI**)
- K-means after RLPI (denoted as **RLPI**)
- Clustering using Probabilistic Latent Semantic Indexing (denoted as **PLSI**). PLSI model is a probabilistic variant of LSI which was proposed in [13]. It provides a probabilistic approach for the discovery of latent variables which is more flexible and has a more solid statistical foundation than the standard LSI. Assuming that there exist a set of hidden factors underlying the co-occurrences among two sets of objects, PLSI uses Expectation-Maximization (EM) algorithm to estimate the probability values which measure the relationships between the hidden factors and the two sets of objects. For clustering tasks, the clusters can be treated as the hidden factors.
- Nonnegative Matrix Factorization-based clustering (denoted as **NMF-NCW** [23]). The weighted NMF-based cluster-

Table 2: Clustering performance on TDT2

c	Accuracy (mean±std-dev%)					
	Baseline	LSI	PLSI	LPI	RLPI	NMF-NCW
2	97.7±7.3	93.4±14.2	91.7±13.0	99.8±0.3	99.9±0.2	99.2±4.7
3	88.4±18.0	86.1±20.0	82.8±18.3	99.6±0.4	99.6±0.4	95.7±11.0
4	85.7±18.9	79.2±21.2	75.4±19.3	99.3±0.8	99.4±0.8	92.4±11.9
5	82.4±17.8	76.8±22.3	72.5±18.0	98.7±1.8	98.8±1.8	92.2±10.5
6	79.0±17.5	72.0±19.4	68.2±16.8	98.6±1.5	98.8±1.3	88.0±12.6
7	74.5±16.5	65.9±18.1	64.0±14.1	97.8±2.4	98.2±2.1	83.1±14.6
8	70.1±17.9	61.3±18.0	61.1±15.2	96.8±4.2	97.3±4.2	79.7±13.1
9	72.3±15.6	64.5±18.0	62.2±11.5	95.5±6.0	97.5±2.4	84.8±13.1
10	69.2±17.0	63.4±18.0	61.1±13.2	94.0±6.3	96.0±4.5	81.5±10.1
30	58.5	54.2	59.6	—*	86.7	61.0

c	Mutual Information (mean±std-dev%)					
	Baseline	LSI	PLSI	LPI	RLPI	NMF-NCW
2	91.3±23.1	82.4±34.4	72.5±36.7	97.6±3.4	97.9±2.8	96.6±12.9
3	81.5±27.1	77.4±30.3	67.1±29.3	96.6±3.0	97.0±2.9	90.7±18.3
4	82.0±22.8	73.9±25.9	64.8±24.6	96.1±4.5	96.4±4.5	87.7±17.3
5	79.0±19.9	71.9±24.1	64.6±21.4	94.6±5.3	95.1±5.2	86.6±13.3
6	78.1±17.0	70.2±19.6	64.1±18.0	94.7±3.9	95.3±3.8	84.2±11.9
7	74.5±16.5	65.7±17.8	60.7±15.7	93.3±4.5	94.1±4.4	79.9±13.7
8	71.5±16.9	61.9±17.3	59.7±16.1	92.1±4.7	93.1±3.4	76.2±13.4
9	75.1±13.7	67.2±16.4	63.4±12.3	91.7±6.9	93.4±4.6	81.8±12.3
10	73.1±15.0	66.1±16.1	63.4±13.6	89.6±7.5	91.4±5.5	78.3±11.1
30	70.5	65.1	67.1	—*	86.3	67.0

*LPI can not be applied due to the memory limit

Figure 1: Performance comparisons on clustering

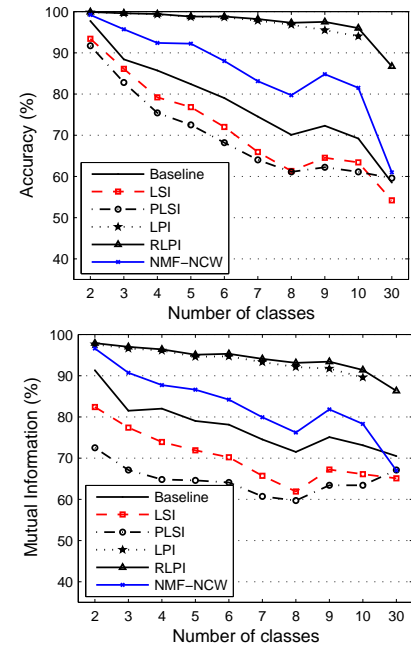


Table 3: Processing time on TDT2

c	Time on Dimension Reduction (s)				Time on Clustering (s)				Overall Time (s)					
	LSI	PLSI	LPI	RLPI	Baseline	LSI	LPI/RLPI	NMF-NCW	Baseline	LSI	PLSI	LPI	RLPI	NMF-NCW
2	0.36	3.22	11.45	1.02	6.25	0.08	0.06	6.0	6.25	0.43	3.22	11.50	1.08	6.0
3	0.51	7.49	26.68	1.82	18.23	0.16	0.13	23.1	18.23	0.67	7.49	26.81	1.95	23.1
4	0.66	11.23	33.34	2.40	29.74	0.30	0.22	63.3	29.74	0.96	11.23	33.56	2.62	63.3
5	0.90	18.67	75.98	4.11	61.82	0.61	0.39	113.7	61.82	1.50	18.67	76.37	4.50	113.7
6	0.94	20.70	64.80	3.86	66.51	0.84	0.50	238.4	66.51	1.78	20.70	65.30	4.37	238.4
7	1.32	31.57	143.05	6.84	117.63	1.65	0.81	389.5	117.63	2.97	31.57	143.86	7.65	389.5
8	1.63	40.06	177.73	7.97	171.76	2.57	1.30	766.6	171.76	4.20	40.06	179.03	9.27	766.6
9	1.82	45.57	226.18	8.99	193.85	3.19	1.94	869.7	193.85	5.00	45.57	228.12	10.93	869.7
10	2.38	56.79	264.12	10.67	261.05	4.10	2.42	1348.3	261.05	6.48	56.79	266.53	13.09	1348.3
30	18.77	511.53	—*	154.61	2720.21	113.35	70.10	15101	2720.21	132.12	511.53	—*	224.71	15101.0

*LPI can not be applied due to the memory limit

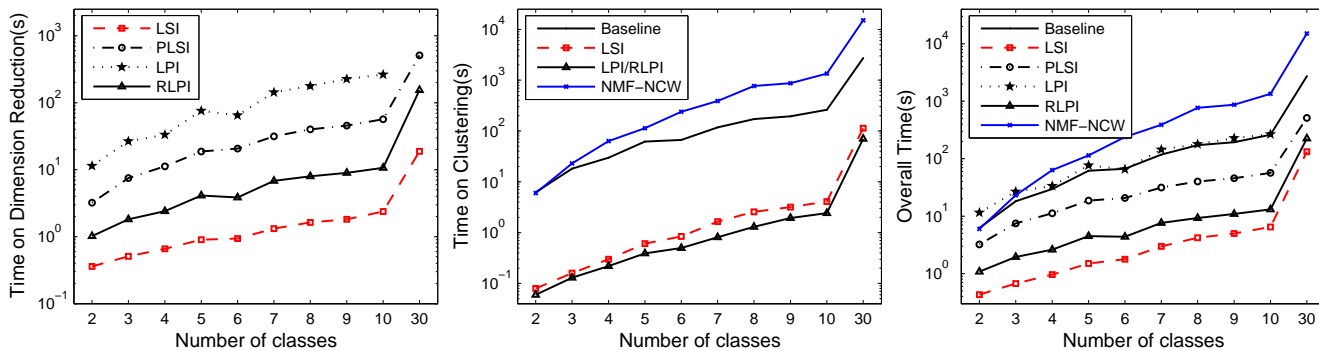


Figure 2: Processing time on TDT2

ing method is a recently proposed algorithm which has been shown to be very effective in document clustering [23].

Note that, the two methods LPI and RLPI need to construct a graph on the documents. In this experiment, we use the same graph for these two methods and the parameter p (number of nearest neighbors) was set to 7. The parameter α in RLPI was set to 0.1.

All these algorithms are tested on the TDT2 corpus. The TDT2 corpus³ consists of data collected during the first half of 1998 and taken from 6 sources, including 2 newswires (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 experiment, those documents appearing in two or more categories were removed, and only the largest 30 categories were kept, thus leaving us with 9,394 documents in total.

4.1.1 Evaluation Metric

The clustering result is evaluated by comparing the obtained label of each document with that provided by the document corpus. Two metrics, the accuracy (AC) and the normalized mutual information metric (\overline{MI}) are used to measure the clustering performance [4], [23]. Given a document \mathbf{x}_i , let r_i and s_i be the obtained cluster label and the label provided by the corpus, respectively. The AC is defined as follows:

$$AC = \frac{\sum_{i=1}^n \delta(s_i, \text{map}(r_i))}{n}$$

where n is the total number of documents and $\delta(x, y)$ is the delta function that equals one if $x = y$ and equals zero otherwise, and $\text{map}(r_i)$ is the permutation mapping function that maps each cluster label r_i to the equivalent label from the data corpus. The best mapping can be found by using the Kuhn-Munkres algorithm [16].

Let C denote the set of clusters obtained from the ground truth and C' obtained from our algorithm. Their mutual information metric $MI(C, C')$ is defined as follows:

$$MI(C, C') = \sum_{c_i \in C, c'_j \in C'} p(c_i, c'_j) \cdot \log_2 \frac{p(c_i, c'_j)}{p(c_i) \cdot p(c'_j)}$$

where $p(c_i)$ and $p(c'_j)$ are the probabilities that a document arbitrarily selected from the corpus belongs to the clusters c_i and c'_j , respectively, and $p(c_i, c'_j)$ is the joint probability that the arbitrarily selected document belongs to the clusters c_i as well as c'_j at the same time. In our experiments, we use the normalized mutual information \overline{MI} as follows:

$$\overline{MI}(C, C') = \frac{MI(C, C')}{\max(H(C), H(C'))}$$

where $H(C)$ and $H(C')$ are the entropies of C and C' , respectively. It is easy to check that $\overline{MI}(C, C')$ ranges from 0 to 1. $\overline{MI} = 1$ if the two sets of clusters are identical, and $\overline{MI} = 0$ if the two sets are independent.

4.1.2 Results

Besides clustering the whole data set into 30 clusters, the evaluations were also conducted with different number of clusters, ranging from 2 to 10. For each given cluster number c , 50 tests were conducted on different randomly chosen categories, and the average performance was computed over these 50 tests (except the 30 cluster case). For each test, K-means algorithm was applied 10

times with different start points and the best result in terms of the objective function of K-means was recorded. After LSI, LPI, or RLPI, how to determine the dimensions of the subspace is still an open problem. In this experiment, we keep c dimensions for all the three algorithms as suggested by previous study [4].

Table 2 and Figure 1 show the average accuracy and average mutual information of the six algorithms. LSI seems not promising in dimension reduction for clustering because the K-means on the LSI subspace is even worse than K-means on the original document space. One may iterate all the possible dimensions for better performance of LSI as suggested in [4]. However, it may not be possible to do so in a real case. Clustering using PLSI is even worse. NMF-NCW method achieves better performance than Baseline which is consistent with previous study [23], [4]. Both LPI and RLPI achieve significant improvements over other four algorithms. The reason is that LPI and RLPI try to reveal the local geometric structure of document space. More detailed analysis and experiments of document clustering using LPI are provided in [4].

Table 3 and Figure 2 show the processing time of the six algorithms. The processing time of LSI, LPI and RLPI include two parts: dimensionality reduction time and time of K-means on the reduced subspace. The processing time of Baseline and NMF-NCW methods are simply the time of clustering approaches (K-means and nonnegative matrix factorization). PLSI estimate the probability of each document belongs to each cluster, which can be directly used to infer the clustering result. Thus, the processing time of PLSI is only the dimensionality reduction (model estimation) time. After dimensionality reduction of LSI (LPI and RLPI), K-means is performed in a very low dimensional subspace thus is much more efficient than K-means in the original document space. The results here further show the advantage of dimensionality reduction for clustering. Clustering based on LSI is the most efficient approach. However, the low clustering accuracy makes LSI approach less attractive. Although the NMF-NCW method achieves better performance than Baseline method, the high computational cost (NMF-NCW spent more than 4 hours for clustering 9,394 documents into 30 classes!) makes it not applicable on large document set. The same shortcoming exists for LPI approach. It can not be applied with 9,394 documents due to the memory limit. Consider both accuracy and efficiency, RLPI is obviously the best among the six compared algorithms for document clustering.

4.2 Text Categorization on 20Newsgroups

Text categorization is an active research area in machine learning and information retrieval. Many statistical classification methods have been applied to this problem, including Naive Bayes, k Nearest Neighbor (kNN) and Support Vector Machine (SVM) [14], [24]. In this experiment, we investigate the use of dimensionality reduction algorithms for text categorization. The following three classifiers are used in the experiment:

- k Nearest Neighbor (kNN) [24]. kNN does not need to train a model, but it has to spent significant time on testing phase. The only parameter in kNN is the number of nearest neighbors k .
- Support Vector Machine (SVM) [14]. We used the LibSVM system [6] and tested it with the linear model, since previous researches [24] show that linear SVM is effective enough for text categorization. There is a parameter C to control the trade-off between large margin and the training error. The testing phase of linear SVM is very efficient. However, SVM needs a lot of time to train the model.

³Nist Topic Detection and Tracking corpus at <http://www.nist.gov/speech/tests/tdt/tdt98/index.htm>

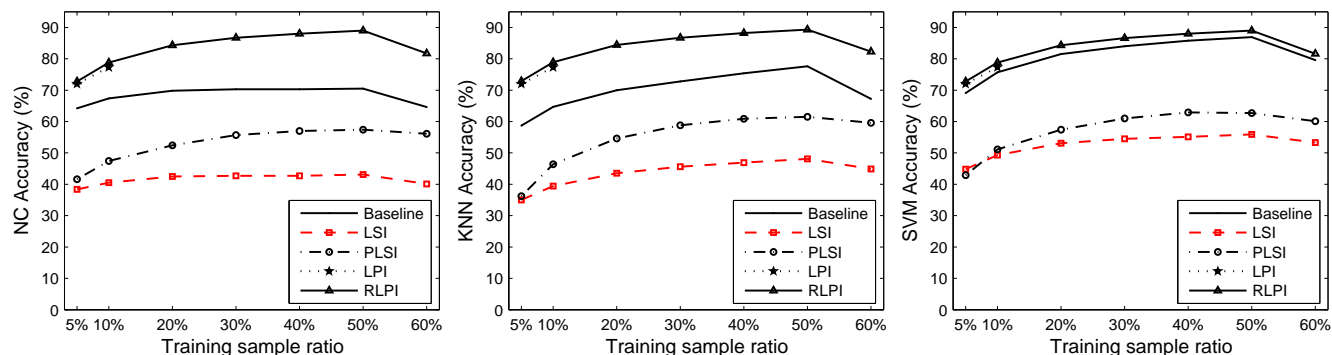
Table 4: Categorization performance on 20Newsgroup

	Nearest Centroid (NC)							
	Accuracy (mean±std-dev%)					Time on Training & Testing (s)		
	Baseline	LSI	PLSI	LPI	RLPI	Baseline	LSI/PLSI/LPI/RLPI	
5%	64.2±1.2	38.4±1.3	41.6±1.6	72.0±0.6	72.8±0.5	0.301 , 1.224	0 , 0.099	
10%	67.4±0.6	40.5±1.0	47.4±1.2	77.3±0.7	78.8±0.4	0.579 , 1.108	0.001 , 0.095	
20%	69.8±0.7	42.5±1.2	52.4±1.9	—*	84.3±0.2	1.218 , 0.923	0.004 , 0.085	
30%	70.3±0.6	42.7±0.8	55.7±2.6	—*	86.7±0.2	1.896 , 0.749	0.005 , 0.072	
40%	70.3±0.5	42.7±0.4	57.0±1.7	—*	88.0±0.2	2.599 , 0.590	0.006 , 0.061	
50%	70.5±0.4	43.1±0.5	57.4±1.9	—*	89.0±0.2	3.329 , 0.464	0.007 , 0.048	
60% (Orig Split)	64.6	40.1	56.1	—*	81.7	4.130 , 0.350	0.010 , 0.040	

	<i>k</i> Nearest Neighbor (kNN)							
	Accuracy (mean±std-dev%)					Time on Training & Test (s)		
	Baseline	LSI	PLSI	LPI	RLPI	Baseline	LSI/PLSI/LPI/RLPI	
5%	58.7±1.0	35.0±1.1	36.2±1.8	72.0±0.6	72.9±0.5	0 , 11.73	0 , 8.649	
10%	64.7±0.3	39.4±1.0	46.4±1.7	77.3±0.7	78.9±0.4	0 , 23.04	0 , 17.25	
20%	70.0±0.3	43.5±0.7	54.6±2.0	—*	84.4±0.2	0 , 43.98	0 , 33.34	
30%	72.8±0.6	45.6±0.5	58.8±3.1	—*	86.7±0.2	0 , 61.93	0 , 47.74	
40%	75.4±0.3	46.9±0.2	60.9±1.6	—*	88.2±0.2	0 , 76.58	0 , 59.59	
50%	77.6±0.4	48.1±0.5	61.5±2.0	—*	89.3±0.2	0 , 85.38	0 , 67.73	
60% (Orig Split)	67.2	44.9	59.6	—*	82.3	0 , 86.02	0 , 70.34	

	Support Vector Machine (SVM)							
	Accuracy (mean±std-dev%)					Time on Training & Test (s)		
	Baseline	LSI	PLSI	LPI	RLPI	Baseline	LSI/PLSI	LPI/RLPI
5%	69.1±0.7	44.8±1.0	42.9±1.3	72.0±0.6	72.8±0.5	2.100 , 2.265	0.183 , 0.159	0.036 , 0.159
10%	75.7±0.3	49.3±1.0	51.1±1.1	77.3±0.7	78.8±0.4	6.851 , 2.048	0.595 , 0.159	0.121 , 0.159
20%	81.5±0.4	53.1±0.7	57.4±1.8	—*	84.3±0.2	21.33 , 1.566	1.943 , 0.132	0.452 , 0.132
30%	84.0±0.2	54.5±0.6	61.0±2.5	—*	86.6±0.2	40.11 , 1.212	3.739 , 0.126	0.941 , 0.126
40%	85.8±0.3	55.1±0.4	62.9±1.7	—*	88.0±0.2	62.01 , 0.933	6.119 , 0.099	1.561 , 0.099
50%	86.9±0.2	55.9±0.5	62.7±1.8	—*	89.0±0.2	86.71 , 0.675	9.064 , 0.078	2.295 , 0.078
60% (Orig Split)	79.6	53.3	60.1	—*	81.6	109.7 , 0.420	12.11 , 0.056	3.090 , 0.056

*LPI can not be applied due to the memory limit


Figure 3: Performance comparisons on categorization

- Nearest Centroid (NC). This is a simple but efficient classifier which classifies a test example according to the label of its nearest centroid (centroid of each class in the training set). There is no parameter in this method. Both training and testing phases of this classifier are very efficient.

All the three classifiers are performed in original document space (Baseline) as well as LSI (PLSI, LPI and RLPI) subspace. The dimension of the LSI (PLSI and LPI) subspace is the number of categories $c (= 20)$ and the dimension of the RLPI subspace is $c - 1 (= 19)$. The value of parameter α in RLPI is also set to 0.1. The parameter k in kNN and C in SVM are tuned to achieve the best Baseline performance.

We use the popular 20 Newsgroups⁴ as our data set. The 20 Newsgroups is a data set collected and originally used for document classification by Lang [15]. We use the “bydate” version which contains 18,846 documents, evenly distributed across 20 classes. This corpus contains 26214 distinct terms after stemming and stop word removal. The original split is separated in time, with 11,314 (60%) training documents and 7,532 (40%) testing documents. In order to examine the effectiveness of different algorithms with different size of the training set, we further perform several tests that the training set contains 5%, 10%, 20%, 30%, 40% and 50% doc-

⁴<http://people.csail.mit.edu/jrennie/20Newsgroups/>

Table 5: Computational time of LSI, PLSI, LPI and RLPI (s)

Train Size	LSI	PLSI	LPI	RLPI
5%	3.84	62.33	12.97	9.560
10%	4.08	86.05	73.83	12.01
20%	5.43	131.7	—*	16.37
30%	6.35	173.4	—*	20.97
40%	7.51	214.5	—*	26.22
50%	9.02	254.3	—*	32.38
60% (Orig Split)	9.75	294.1	—*	39.52

*LPI can not be applied due to the memory limit

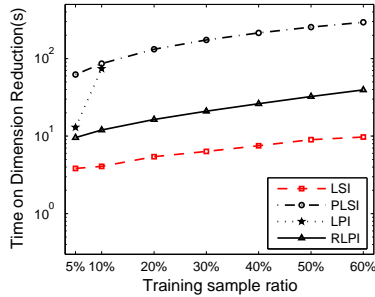


Figure 4: Computational time of LSI, PLSI, LPI and RLPI

uments. For these tests, we averaged the results over 10 random splits.

The classification results of the three classifiers on five document representation methods are listed in Table 4 and Figure 3. Table 5 and Figure 4 show the dimensionality reduction time of the four algorithms. The main observations from the performance comparisons include:

- LSI and PLSI are not promising in dimension reduction for document categorization. All the three classifiers get lower accuracy in the LSI (PLSI) subspace.
- LPI and RLPI use the label information to build the graph and the corresponding subspace well preserves the class structure. As a result, all the three classifiers achieve better performance in the LPI (RLPI) subspace than in original document space. We noticed that the performance improvements are especially significant for NC and kNN classifiers. For a more powerful classifier, *e.g.* SVM, the performance improvement is limited.
- The high computational cost of LPI (both in time and memory) makes it not practical for large data sets. As the size of the training set increases, LPI can not be applied due to the memory limit.
- All the three classifiers achieve almost the same performance in RLPI subspace. It is very interesting that the simplest and the most efficient Nearest Centroid classifier can achieve the similar performance with SVM in the RLPI subspace. Considering the efficiency of Nearest Centroid classifier in both training and testing phases, we believe performing Nearest Centroid rule in the RLPI subspace will become a very promising text categorization method.

4.3 Discussions

We summarize the experiments below:

1. The low dimensionality of the document subspace obtained in our experiments show that dimensionality reduction is indeed necessary as a preprocessing for document clustering, classification, retrieval, etc.
2. The effectiveness and efficiency are two essential factors in designing good dimensionality reduction algorithm. LSI, PLSI and ordinary LPI fail to meet these two requirements. As an alternative choice of LPI, RLPI avoids the expensive computation of LPI while remains its effectiveness. This property makes RLPI a good choice for dimensionality reduction in large scale text processing problems.

5. CONCLUSIONS AND FUTURE WORK

We have proposed a new algorithm for document indexing and representation, called Regularized Locality Preserving Indexing. RLPI shares the same locality preserving character as LPI while avoids the expensive computation. Specifically, RLPI can be computed by a sparse matrix eigen-decomposition followed with a regularized least squares. Moreover, with a specific graph design in supervised case, the eigen-decomposition becomes trivial and RLPI only needs to solve a set of regularized least squares problems. Such property makes RLPI can be efficiently computed even for a large scale data set. Extensive experiments on TDT2 and 20News-groups datasets demonstrated the effectiveness and efficiency of RLPI.

There is a parameter α which controls the smoothness of the basis functions of RLPI. Our theoretical analysis shows that RLPI gives the same solution of LPI when α decreases to zero. For supervised learning, when the training set is small, LPI tends to over-fit the training data. In such case, a smoother (with small norm) basis function is usually preferred and RLPI with $\alpha > 0$ can have better performance than LPI. However, it remains unclear that how to automatically estimate the best α .

6. REFERENCES

- [1] R. Ando. Latent semantic space: Iterative scaling improves precision of inter-document similarity measurement. In *Proc. 2000 Int. Conf. on Research and Development in Information Retrieval (SIGIR'00)*, Athens, Greece, July 2000.
- [2] B. T. Bartell, G. W. Cottrell, and R. K. Belew. Latent semantic indexing is an optimal special case of multidimensional scaling. In *Proc. 1992 Int. Conf. on Research and Development in Information Retrieval (SIGIR'92)*, pages 161–167, Copenhagen, Denmark, June 1992.
- [3] D. Cai and X. He. Orthogonal locality preserving indexing. In *Proc. International Conference on Research and Development in Information Retrieval (SIGIR'05)*, pages 3–10, Salvador, Brazil, 2005.
- [4] D. Cai, X. He, and J. Han. Document clustering using locality preserving indexing. *IEEE Transactions on Knowledge and Data Engineering*, 17(12):1624–1637, December 2005.
- [5] D. Cai, X. He, and J. Han. Spectral regression for dimensionality reduction. Technical report, Computer Science Department, UIUC, UIUCDCS-R-2007-2856, May 2007.
- [6] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [7] F. R. K. Chung. *Spectral Graph Theory*, volume 92 of *Regional Conference Series in Mathematics*. AMS, 1997.

- [8] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.
- [9] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, Hoboken, NJ, 2nd edition, 2000.
- [10] G. H. Golub and C. F. V. Loan. *Matrix computations*. Johns Hopkins University Press, 3rd edition, 1996.
- [11] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer-Verlag, 2001.
- [12] X. He, D. Cai, H. Liu, and W.-Y. Ma. Locality preserving indexing for document representation. In *Proc. 2004 Int. Conf. on Research and Development in Information Retrieval (SIGIR'04)*, pages 96–103, Sheffield, UK, July 2004.
- [13] T. Hofmann. Probabilistic latent semantic indexing. In *Proc. 1999 Int. Conf. on Research and Development in Information Retrieval (SIGIR'99)*, pages 50–57, Berkeley, CA, Aug. 1999.
- [14] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *ECML'98*, 1998.
- [15] K. Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339, 1995.
- [16] L. Lovasz and M. Plummer. *Matching Theory*. Akadémiai Kiadó, North Holland, Budapest, 1986.
- [17] C. C. Paige and M. A. Saunders. Algorithm 583 LSQR: Sparse linear equations and least squares problems. *ACM Transactions on Mathematical Software*, 8(2):195–209, June 1982.
- [18] C. C. Paige and M. A. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Transactions on Mathematical Software*, 8(1):43–71, March 1982.
- [19] C. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala. Latent semantic indexing: a probabilistic analysis. In *Proc. 17th ACM Symp. Principles of Database Systems*, Seattle, WA, June 1998.
- [20] R. Penrose. A generalized inverse for matrices. In *Proceedings of the Cambridge Philosophical Society*, volume 51, pages 406–413, 1955.
- [21] G. W. Stewart. *Matrix Algorithms Volume I: Basic Decompositions*. SIAM, 1998.
- [22] G. W. Stewart. *Matrix Algorithms Volume II: Eigensystems*. SIAM, 2001.
- [23] W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *Proc. 2003 Int. Conf. on Research and Development in Information Retrieval (SIGIR'03)*, pages 267–273, Toronto, Canada, Aug. 2003.
- [24] Y. Yang and X. Liu. A re-examination of text categorization methods. In *SIGIR'99*, 1999.

APPENDIX

A. PROOF OF THEOREM 2

PROOF. Suppose $\text{rank}(X) = r$, the SVD decomposition of X is

$$X = U\Sigma V^T$$

where $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_r)$, $U \in \mathbb{R}^{n \times r}$, $V \in \mathbb{R}^{m \times r}$ and we

have $U^T U = V^T V = I$. The \mathbf{y} is in the space spanned by row vectors of X , therefore, \mathbf{y} is in the space spanned by column vectors of V . Thus, \mathbf{y} can be represented as the linear combination of the column vectors of V . Moreover, the combination is unique because the column vectors of V are linear independent. Suppose the combination coefficients are b_1, \dots, b_r . Let $\mathbf{b} = [b_1, \dots, b_r]^T$, we have:

$$V\mathbf{b} = \mathbf{y} \Rightarrow V^T V\mathbf{b} = V^T \mathbf{y} \Rightarrow \mathbf{b} = V^T \mathbf{y} \Rightarrow V V^T \mathbf{y} = \mathbf{y} \quad (19)$$

To continue our proof, we need introduce the concept of pseudo inverse of a matrix [20], which we denote as $(\cdot)^+$. Specifically, pseudo inverse of the matrix X can be computed by the following two ways:

$$X^+ = V\Sigma^{-1}U^T$$

and

$$X^+ = \lim_{\alpha \rightarrow 0} (X^T X + \alpha I)^{-1} X^T$$

The above limit exists even if $X^T X$ is singular and $(X^T X)^{-1}$ does not exist [20]. Thus, the regularized least squares solution in Eqn. (15)

$$\mathbf{a} = (X X^T + \alpha I)^{-1} X \mathbf{y} \stackrel{\alpha \rightarrow 0}{=} (X^T)^+ \mathbf{y} = U \Sigma^{-1} V^T \bar{\mathbf{y}}$$

Combine with the equation in Eqn. (19), we have

$$X^T \mathbf{a} = V \Sigma U^T \mathbf{a} = V \Sigma U^T U \Sigma^{-1} V^T \mathbf{y} = V V^T \mathbf{y} = \mathbf{y}$$

By Theorem (1), \mathbf{a} is the eigenvector of eigen-problem in Eqn. (5). \square

B. PROOF OF COROLLARY 3

PROOF. The matrices W and D are of size $m \times m$ and there are m eigenvectors $\{\mathbf{y}_j\}_{j=1}^m$ of eigen-problem (8). Since $\text{rank}(X) = m$, all these m eigenvectors \mathbf{y}_j are in the space spanned by row vectors of X . By Theorem (2), all m corresponding \mathbf{a}_j of RLPI in Eqn (15) are eigenvectors of eigen-problem in Eqn. (5) as α decreases to zero. They are

$$\mathbf{a}_j^{RLPI} = U \Sigma^{-1} V^T \mathbf{y}_j.$$

Consider the eigen-problem in Eqn. (6), since the m eigenvectors \mathbf{y}_j are also in the space spanned by row vectors of $\tilde{X} = U^T X = \Sigma V^T$, eigenvector \mathbf{b}_j will be the solution of linear equations system $\tilde{X}^T \mathbf{b}_j = \mathbf{y}_j$. The row vectors of $\tilde{X} = \Sigma V^T$ are linearly independent, thus \mathbf{b}_j is unique and

$$\mathbf{b}_j = \Sigma^{-1} V^T \mathbf{y}_j.$$

Thus, the projective functions of LPI

$$\mathbf{a}_j^{LPI} = U \mathbf{b}_j = U \Sigma^{-1} V^T \mathbf{y}_j = \mathbf{a}_j^{RLPI}$$

\square