Unsupervised Feature Extraction Inspired by Latent Low-Rank Representation

Yaming Wang, Vlad I. Morariu, Larry S. Davis
University of Maryland, College Park, MD, 20742, USA
{wym, morariu, lsd}@umiacs.umd.edu

Abstract

Latent Low-Rank Representation (LatLRR) has the empirical capability of identifying "salient" features. However, the reason behind this feature extraction effect is still not understood. Its optimization leads to non-unique solutions and has high computational complexity, limiting its potential in practice. We show that LatLRR learns a transformation matrix which suppresses the most significant principal components corresponding to the largest singular values while preserving the details captured by the components with relatively smaller singular values. Based on this, we propose a novel feature extraction method which directly designs the transformation matrix and has similar behavior to LatLRR. Our method has a simple analytical solution and can achieve better performance with little computational cost. The effectiveness and efficiency of our method are validated on two face recognition datasets.

1. Introduction

Recently, Low-Rank Representation (LRR) [3, 12, 11, 4, 5] has attracted attention in the field of unsupervised subspace segmentation [16], since it can effectively cluster high-dimensional data into low-dimensional subspaces by learning the lowest rank representation of the data matrix. Among various versions of LRR, Latent Low-Rank Representation (LatLRR) [13] has the novel ability to extract "salient features" from visual data, which was interpreted as the deviation of each sample from the "principal features".

LatLRR solves the following optimization problem

$$\begin{aligned} \text{minimize} \quad & \|Z\|_* + \|L\|_* + \lambda\|E\|_1, \\ \text{s.t.} \quad & X = XZ + LX + E, \end{aligned} \tag{1}$$

where X is the data matrix with each column being one data sample and $\|\cdot\|_*$ denotes the nuclear norm. In [13], the authors empirically found out that using the second term, LX, for classification can significantly improve accuracy, thanks to its "salient features". Although LatLRR originally aimed to improve subspace clustering and the salient feature extraction effect was just a by-product, it outperformed many mainstream dimensionality reduction techniques, such as Principal Component Analysis (PCA) [15], Neighborhood Preserving Embedding (NPE) [7], Locality Preserving Projection (LPP) [8] and Nonnegative Matrix Factorization (NMF) [6].

While LatLRR has shown promissing results, it suffers from the following problems.

- An explanation for its observed capability to identify "salient" features is unknown, which constrains its potential.

- It has been observed that the solution to (1) is not unique, which potentially reduces its reliability. Zhang *et al.* [18, 19] have derived a closed form solution to the noiseless version of (1), but the reliability problem for "salient" feature extraction remains.

- The complexity of LatLRR depends on the dimensionality of the feature vectors [13]. Previous improvements to LatLRR have actually increased the complexity of the optimization, such as the introduction of a more complex objective function [22, 21, 20] or more complex constraints [17], which further increases the computational burden.

We provide an explanation of LatLRR's feature extraction effect. Based on this, we propose a new and computationally simpler feature extraction method. The contributions of this paper are twofold.

- We show that the singular values of L learnt by LatLRR have a reweighting effect, which suppresses the most significant principal components of data matrix X, while highlighting the detailed information carried by the components corresponding to X's relatively smaller singular values.

- Using our characterization of the solutions produced by LatLRR, we design a new feature extraction method that produces solutions similar to LatLRR, but with a simple unique analytical solution. Our method outperforms LatLRR by computing a single SVD decomposition, and thus can be applied to higher dimensional data.

2. Analysis of LatLRR's Feature Extraction

In this section, we interpret the feature extraction effect of LatLRR, mainly from the perspective of SVD decomposition.

Suppose X and L have a skinny SVD

$$X = U_X \Sigma_X V_X^{\mathrm{T}} \quad \text{and} \quad L = U_L W_L V_L^{\mathrm{T}}. \tag{2}$$

The meaning of "skinny" is that Σ_X and W_L are square matrices of size $\mathrm{rank}(X)$ and $\mathrm{rank}(L)$ respectively, only containing non-zero singular values.

Let l_i be the i^{th} singular value of L, and v_{Li} (u_{Li}) the i^{th} column of V_L (U_L). Consider the effect of L operating on a single data sample x_0:

$$Lx_0 = U_L W_L V_L^{\mathrm{T}} x_0 = \sum_{i=1}^{r} l_i \langle v_{Li}, x_0 \rangle u_{Li}, \tag{3}$$

where $r = \mathrm{rank}(L)$. According to the definition of SVD, by multiplying by L, the data has been projected onto each column of V_L and weighted by the singular values. Then the weighted projections are used as the coefficients of $\{u_{Li}\}$, a subset of an orthonormal basis of L's column space. This interpretation of SVD decomposition lays the foundation of our analysis below.

2.1. Preliminary

Recently, Zhang *et al.* [18] found that the solution to LatLRR is not unique. Moreover, they derived a closed form solution for the noise free LatLRR

$$\text{minimize} \quad \|Z\|_* + \|L\|_*, \quad \text{s.t.} \quad X = XZ + LX. \tag{4}$$

We restate the main result of [18] on noise free LatLRR in the following theorem.

Theorem 1. *The complete solutions to problem (4) must be of the following form*

$$Z = V_X W_Z V_X^{\mathrm{T}} \quad \textit{and} \quad L = U_X (I - W_Z) U_X^{\mathrm{T}}, \tag{5}$$

where W_Z is any block diagonal matrix satisfying: 1. if $[\Sigma_X]_{ii} \neq [\Sigma_X]_{jj}$ then $[W_Z]_{ij} = 0$; 2. both W_Z and $I - W_Z$ are positive semi-definite [18].

Notice that in practice the singular values of X are usually distinct, therefore W_Z becomes a diagonal matrix $\mathrm{diag}\{z_1, z_2, ..., z_r\}$ with $0 \leq z_i \leq 1$ for all i. Then (5) becomes equivalent to (2), *i.e.*

$$U_L = V_L = U_X, \quad W_L = I - W_Z, \tag{6}$$
$$r = \mathrm{rank}(L) = \mathrm{rank}(X),$$

and the effects of Z and L on X become

$$\begin{aligned} XZ &= U_X \Sigma_X W_Z V_X^{\mathrm{T}} \\ &= U_X \begin{bmatrix} z_1 \sigma_{X1} & & \\ & \ddots & \\ & & z_r \sigma_{Xr} \end{bmatrix} V_X^{\mathrm{T}} \end{aligned} \tag{7}$$

$$\begin{aligned} LX &= U_X (I - W_Z) \Sigma_X V_X^{\mathrm{T}} \\ &= U_X \begin{bmatrix} l_1 \sigma_{X1} & & \\ & \ddots & \\ & & l_r \sigma_{Xr} \end{bmatrix} V_X^{\mathrm{T}}, \end{aligned} \tag{8}$$

where $l_i = 1 - z_i$, σ_{Xi} is the i^{th} largest singular value of X.

Moreover, (3), the effect of L on a single data x_0 becomes

$$Lx = \sum_{i=1}^{r} l_i \langle u_{Xi}, x_0 \rangle u_{Xi}, \tag{9}$$

i.e. the data is projected onto the components $\{u_{Xi}\}$ and weighted by $\{l_i\}$.

From (8) and (9), as will be shown in Section 2.2 and Section 2.3, $W_L = I - W_Z$ suppresses the components corresponding to very large σ_{Xi}, and preserves the ones corresponding to relatively small σ_{Xi}. Since the orthogonal matrices in (5) are determined as U_X and V_X, the numerical optimization with respect to Z and L is essentially learning the singular values $\{z_i\}$ and $\{l_i\}$, resulting in a solution from the solution set defined by (5). This is exactly what LatLRR is accomplishing.

In the rest of this section, we gain insight from an empirical result in Section 2.2, and then theoretically explore the source of the empirical results in Section 2.3.

2.2. Empirical Analysis

The following phenomenon can provide some insight into LatLRR's feature extraction capability. Our empirical results suggest that different optimization ordering leads to different reweightings of principal components, which strongly affects the performance.

In the iterative algorithm described in [13] (the noise free version summarized in Algorithm 1), within each iteration, Z is updated before L. However, if we update L before Z in each iteration (exchange Step 3 with Step 4 in Algorithm 1), the final classification performance decreases significantly (*e.g.* from approximately 89% to 39% on Extended Yale Database B).

Let $\{l_{1i}\}$ be the singular values of L learnt by the original Algorithm 1, and $\{l_{2i}\}$ the singular values learnt with the exchanged steps. The singular values of L, X, and LX

Algorithm 1 Solving Problem (4) by Inexact ALM

```
Initialize: $Z_0 = J_0 = 0$, $L_0 = S_0 = 0$, $Y_{10} = Y_{20} = Y_{30} = 0$, $\mu_0 > 0$. Set parameters ($\rho$ and $\epsilon$).
while not converged do
  1. Fix others and update $J$ by setting $J_{k+1} = \mathrm{argmin}_J \frac{1}{\mu_k}\|J\|_* + \frac{1}{2}\|J - (Z_k + Y_{2k}/\mu_k)\|_{\mathrm{F}}^2$.
  2. Fix others and update $S$ by setting $S_{k+1} = \mathrm{argmin}_S \frac{1}{\mu_k}\|S\|_* + \frac{1}{2}\|S - (L_k + Y_{3k}/\mu_k)\|_{\mathrm{F}}^2$.
  3. Fix others and update $Z$ by setting
     $Z_{k+1} = (I + X^{\mathrm{T}}X)^{-1}(X^{\mathrm{T}}(X - L_kX) + J_{k+1} + (X^{\mathrm{T}}Y_{1k} - Y_{2k})/\mu_k)$.
  4. Fix others and update $L$ by setting
     $L_{k+1} = ((X - XZ_{k+1})X^{\mathrm{T}} + S_{k+1} + (Y_{1k}X^{\mathrm{T}} - Y_{3k})/\mu_k)(I + XX^{\mathrm{T}})^{-1}$
  5. Update the mulipliers by
     $Y_{1(k+1)} = Y_{1k} + \mu_k(X - XZ_{k+1} - L_{k+1}X)$,
     $Y_{2(k+1)} = Y_{2k} + \mu_k(Z_{k+1} - J_{k+1})$,
     $Y_{3(k+1)} = Y_{3k} + \mu_k(L_{k+1} - S_{k+1})$.
  6. Update $\mu$ by $\mu_{k+1} = \rho\mu_k$.
  7. Check the convergence conditions:
     $\|X - XZ_{k+1} - L_{k+1}X\|_\infty < \epsilon$,
     $\|Z_{k+1} - J_{k+1}\|_\infty < \epsilon$,
     and $\|L_{k+1} - S_{k+1}\|_\infty < \epsilon$.
end while
```

are displayed in Figure 1 for both cases. For the successful optimization ordering, the u_{Li} with the largest σ_{Xi} and largest $l_{1i}\sigma_{Xi}$ are displayed in Figure 2.

As can be seen from Figure 1 and (9), $\{l_{1i}\}$ plays the role of reweighting, which suppresses the components corresponding to the first several largest σ_{Xi}'s, and thus highlights those containing detailed information. On the other hand, however, l_{2i} put relative small weights on the basis corresponding to small σ_{Xi} and mainly preserves the information carried by components with very large σ_{Xi}, which significantly reduces the performance. Therefore, the ability of LatLRR to extract features is related to the weighting effect of $\{l_i\}$.

Our analysis is consistent with the observation made by [1], that dropping the first three principal components in PCA can effectively improve the classification accuracy of faces. [1] claimed that the first a few principal components might mainly capture the variations caused by photometric factors such as illumination and shadow, therefore removing those irrelevant variations might account for the effectiveness of such practical technique. LatLRR, which reweights the components, can be regarded as a "soft version" of dropping the first few components.

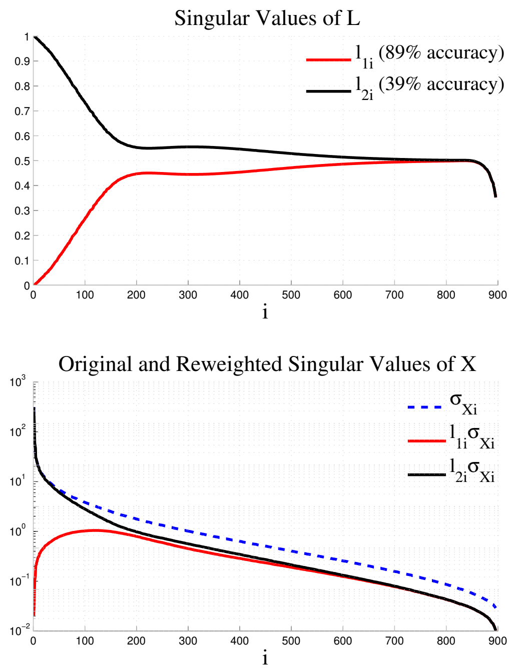

Figure 1. The optimization ordering that yields the best performance (red) down-weights the most significant principal components (low i), while the other ordering (black) gives them more weight. Top: the singular values of transformation matrix L. Bottom: the singular values of data matrix X and transformed data LX. The latter is displayed in a semi-log graph due to the large variations in scale.

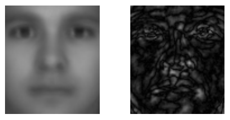

Figure 2. An example principal component (left) and the component corresponding to relatively small σ_{Xi} (right). On the left is the u_{Xi} corresponding to the largest σ_{Xi}, and on the right is the one corresponding to the largest $l_{1i}\sigma_{Xi}$ after reweighting.

2.3. Theoretical Analysis

In this subsection, we analyze the inexact Augmented Lagrange Multiplier (ALM) method [10] adopted by [13], for the noise free LatLRR (4), and derive a closed-form approximate solution that has the same effect as ALM on component weighting. Our analysis explains our empirical observations above, where the singluar values of L suppress the most pincipal u_{Xi}'s corresponding to the largest σ_{Xi}'s, while preserving the components corresponding to relatively smaller σ_{Xi}'s. In the rest of this paper, we will keep using the letter i to denote the index of singular values, while we use letter k to indicate the number of iterations.

2.3.1 LatLRR's Algorithm Overview

The inexact ALM method is outlined in Algorithm 1. Step 1 and 2 are solved by singular value thresholding operator [2], *i.e.*

$$\begin{aligned}\mathcal{D}_{1/\mu}(Y) &= \arg\min_X \left\{ \frac{1}{\mu}\|X\|_* + \frac{1}{2}\|X-Y\| \right\}, \\ &= U_Y \mathrm{diag}\left\{ \max\left(0, \sigma_{Yi} - \frac{1}{\mu}\right) \right\} V_Y^{\mathrm{T}}, \quad (10)\end{aligned}$$

where σ_{Yi} is the i^{th} singular value of Y, whose SVD is

$$Y = U_Y \mathrm{diag}\{\sigma_{Yi}\} V_Y^{\mathrm{T}}.$$

The only assumption made to simplify the analysis is that ρ is relatively large. The form of the solution is first given by Proposition 1 and then the specific solution is derived from Proposition 2 and 3. Proofs of Propositions 1-3 are provided in the supplementary material.

2.3.2 Simplification of the Analysis

Proposition 1. *During the iteration procedure described in Algorithm 1, Z_k and L_k always keep the form*

$$Z_k = V_X W_{Zk} V_X^{\mathrm{T}}, \quad L_k = U_X W_{Lk} U_X^{\mathrm{T}}, \quad (11)$$

$\forall$ positive integer k. Where W_{Zk} and W_{Lk} are $r \times r$ diagonal matrices containing the singular values.

Furthermore, the orthogonal matrices involved in the SVD decomposition of any matrix in Algorithm 1 must be U_X or V_X, depending on its shape.

Proposition 1 can be proved easily by induction. It suggests that we can simplify the analysis by only focusing on the singular values of the matrices. Furthermore, in Algorithm 1, singular values with the same index are modified together, independent from those with different indices, therefore we omit the index i and only focus on the iteration number k. Let the lower case letters denote the singular values of their corresponding matrices in upper case letters. For instance, $z_k = \{z_{ki}, i = 1,2,...,r\}$ and $l_k = \{l_{ki}, i = 1,2,...,r\}$ denote the singular values of Z_k and L_k, respectively, and $z_k + l_k = \{z_{ki} + l_{ki}, i = 1,2,...,r\}$. Hence, the iteration procedure in Algorithm 1 is equivalent to Table 1.

Under the assumption that ρ is relatively large, we can simplify the analysis by omitting the last term on the right side of Step 3 and 4 in Table 1, since it can be proved by induction that

$$y_{1k} = \sigma_X \sum_{t=1}^{k} \mu_{t-1} e_{1t}, \quad y_{2k} = \sum_{t=1}^{k} \mu_{t-1} e_{2t},$$
$$y_{3k} = \sum_{t=1}^{k} \mu_{t-1} e_{3t}, \quad \mu_k = \rho^k \mu_0, \quad (12)$$

```
Algorithm 1 (Simplified)
Initialize: $z_0 = j_0 = l_0 = s_0 = y_{10} = y_{20} = y_{30} = 0, \mu_0 > 0$.
while not converged do
  1. $j_{k+1} = \max\{0, z_k + y_{2k}/\mu_k - 1/\mu_k\}$
  2. $s_{k+1} = \max\{0, l_k + y_{3k}/\mu_k - 1/\mu_k\}$
  3. $z_{k+1} = \frac{1}{1+\sigma_X^2}(\sigma_X^2(1-l_k) + j_{k+1}) + \frac{\sigma_X y_{1k} - y_{2k}}{(1+\sigma_X^2)\mu_k}$
  4. $l_{k+1} = \frac{1}{1+\sigma_X^2}(\sigma_X^2(1-z_{k+1}) + s_{k+1}) + \frac{\sigma_X y_{1k} - y_{3k}}{(1+\sigma_X^2)\mu_k}$
  5. $y_{1(k+1)} = y_{1k} + \mu_k \sigma_X (1 - z_{k+1} - l_{k+1})$
     $y_{2(k+1)} = y_{2k} + \mu_k (z_{k+1} - j_{k+1})$
     $y_{3(k+1)} = y_{3k} + \mu_k (l_{k+1} - s_{k+1})$
  6. $\mu_{k+1} = \rho\mu_k$
  7. Check the convergence conditions.
end while
```

Table 1. The simplification of Algorithm 1 by focusing on the operations of the singular values.

where $e_{1t} = 1 - z_t - l_t,\ e_{2t} = z_t - j_t,\ e_{3t} = l_t - s_t$.

Then the last term on the right side of Step 3 becomes

$$\begin{aligned}&\frac{\sigma_X y_{1k} - y_{2k}}{(1+\sigma_X^2)\mu_k} \\ =& \sum_{t=1}^{k} \frac{\mu_{t-1}}{\mu_k}\left(\frac{\sigma_X^2}{1+\sigma_X^2} e_{1t} - \frac{1}{1+\sigma_X^2} e_{2t} \right) \\ =& \sum_{t=1}^{k} \left(\frac{1}{\rho}\right)^{k+1-t} \left(\frac{\sigma_X^2}{1+\sigma_X^2} e_{1t} - \frac{1}{1+\sigma_X^2} e_{2t} \right) \\ <& e_{\max} \sum_{t=1}^{+\infty} \left(\frac{1}{\rho}\right)^t = \frac{e_{\max}}{\rho - 1} \quad (13)\end{aligned}$$

In practice $e_{\max}$ is very small. Therefore (13) approaches zero when ρ is relatively large, and omitting it can provide a simple and good approximation. Through similar analysis by replacing y_{2k} with y_{3k}, we can obtain the same conclusion for the last term of Step 4.

Since Step 1 and 2 in Table 1 perform thresholding according to the value of $1/\mu_k$, we divide the analysis into 2 stages: large $1/\mu_k$ when μ_k is very small at the beginning, and small $1/\mu_k$ when μ_k becomes very large by the end of the iteration.

2.3.3 The Closed-Form Solution

When μ_k is very small, $1/\mu_k$ is so large that $j_{k+1} = s_{k+1} = 0$. Combine the approximation upon Step 3 and 4 discussed above, and thus the iteration procedure is equivalent to

$$z_{k+1} = \alpha(1 - l_k), \quad l_{k+1} = \alpha(1 - z_{k+1}), \quad (14)$$

where

$$\alpha = \frac{\sigma_X^2}{1+\sigma_X^2} = 1 - \frac{1}{1+\sigma_X^2}, \quad (15)$$

Solving the linear recursive sequence (14) gives the following proposition.

Proposition 2. *Assuming ρ is relatively large, when μ_k is small, the iteration procedure in Table 1 is approximately equivalent to (14), and the solution after the k^{th} iteration is as follows.*

$$z_k = \frac{\alpha + \alpha^{2k}}{1+\alpha}, \quad l_k = \alpha(1 - z_k) = \frac{\alpha^3 + \alpha^{2k+1}}{1+\alpha}, \quad (16)$$

where α is defined by (15).

When μ_k is very large, however, $1/\mu_k \approx 0$ so that

$$\begin{aligned} j_{k+1} &= \max\{0, z_k + y_{2k}/\mu_k - 1/\mu_k\} \\ &= z_k + y_{2k}/\mu_k - 1/\mu_k \approx z_k \end{aligned} \quad (17)$$

Similarly, it follows that

$$s_{k+1} \approx l_k. \quad (18)$$

Plugging in (17) and (18) into Step 3 and 4, and adopting the same approximation as that of Proposition 2, we obtain an equivalent procedure for the large-μ_k case.

$$\begin{aligned} z_{k+1} &= \alpha(1 - l_k) + (1-\alpha) z_k, \\ l_{k+1} &= \alpha(1 - z_{k+1}) + (1-\alpha) l_k, \end{aligned} \quad (19)$$

Proposition 3. *Assuming ρ is relatively large, when μ_k is large, the iteration procedure in Table 1 is approximately equivalent to (19). When the iteration terminates, the final results z and l satisfy*

$$z \to \frac{(1-\alpha) z_{k_0} - l_{k_0} + 1}{2-\alpha}, \quad l \to 1 - z. \quad (20)$$

where α is defined by (15), and k_0 is some starting point from which the large-μ_k condition holds.

When ρ is relatively large, the transient state between the two stages only lasts a very short time. Therefore, omitting it can provide a good and simple approximation to the solution. We pick up a dividing point k_0 of the two stages, use (14) to approximate the iterations when $k \le k_0$, and use (19) to approximate those when $k > k_0$. Plugging (16) into z_{k_0} and l_{k_0} of (20), we conclude that, when the iteration terminates,

$$z \to 1 - \frac{1 - \alpha^{2k_0}}{(2-\alpha)(1+\alpha)}, \quad l \to \frac{1 - \alpha^{2k_0}}{(2-\alpha)(1+\alpha)}, \quad (21)$$

where α is defined by (15) and k_0 is the dividing point of the two stages.

According to (15), α is monotonically increasing as σ_X increases from 0 to infinity. Therefore, when σ_X approaches 0, α also approaches 0 and l will approach $1/2$; when σ_X approaches infinity, α approaches 1 and l will approach 0. l as a function of α is displayed in Figure 3. As can be seen, for relatively small α, which corresponds to small σ_X, l decreases very slowly from $1/2$; for α close to 1, which corresponds to very large σ_X, l drops very quickly to a small value.

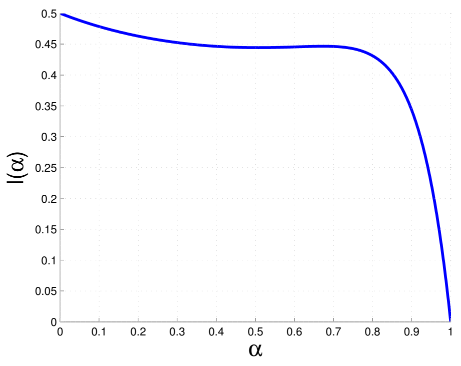

Figure 3. l as a function of α given by (21). The definition of α is given in (15). For small σ_X, α approaches 0 and l decreases very slowly from $1/2$; for very large σ_X, α approaches 1 and l drops very quickly. Parameter settings: $k_0 = 6$.

2.3.4 Result Evaluation

According to (21), our theoretical result for the i^{th} singular value of L becomes

$$l_i = \frac{1 - \alpha_i^{2k_0}}{(2-\alpha_i)(1+\alpha_i)}, \quad (22)$$

where

$$\alpha_i = \frac{\sigma_{Xi}^2}{1+\sigma_{Xi}^2} = 1 - \frac{1}{1+\sigma_{Xi}^2}. \quad (23)$$

The comparison between the reweighting behavior of the L matrix according to theoretical approximation and actual result of ALM is displayed in Figure 4. We plot the singular values of L obtained in practice by Algorithm 1 (displayed in blue) as well as that calculated theoretically by (22) (displayed in red). As can be seen, the theoretical approximation convincingly explains the behavior of Algorithm 1, and the minor error occuring at the tail is mainly due to the fact that the stopping criterion of the theoretical analysis is the limit condition of ALM.

From (8) and (21), we conclude that L can suppress principal components corresponding to very large σ_{Xi}'s, by putting near-zero weights on them through its singular values l_i's. On the other hand, for those components corresponding to relatively small σ_{Xi}'s, multiplying by L will not affect them too much since the corresponding l_i's are almost constant.

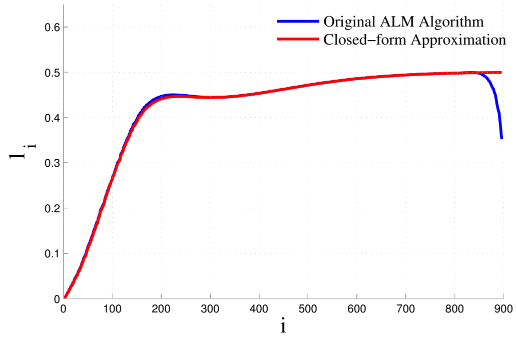

Figure 4. The singular values of our theoretical approximation (red) compared to those achieved by ALM in practice (blue), where l_i denotes the i^{th} singular value of L. Our theoretical approximation very closely models the behavior of ALM. Parameter settings: $\mu_0 = 10^{-6}, \rho = 5, \epsilon = 10^{-4}$ (practice); $k_0 = 6$ (theory).

3. Unsupervised Feature Extraction Inspired by Latent Low-rank Representation

We can now directly design a transformation matrix W_Z that behaves similarly to the original LatLRR algorithm. After W_Z has been obtained, with Z and L constructed according to (5), X can be decomposed into a principal part XZ and a detailed part LX. Then following the approach of LatLRR [13], LX can be used for classification.

Concretely, the problem boils down to designing the objective function f for the following problem

$$\begin{aligned} \text{minimize} \quad & f \\ \text{s.t.} \quad & X = XZ + LX \\ & Z = V_X W_Z V_X^{\mathrm{T}},\ L = U_X(I - W_Z)U_X^{\mathrm{T}}, \\ & W_Z = \mathrm{diag}(z_1, z_2, ..., z_r), 0 \le z_i \le 1\ \forall i, \end{aligned} \tag{24}$$

such that $(I - W_Z)$ down-weights the most significant principal components while preserving the others. To achieve this goal, a suitable objective function should at least satisfy the following two properties

(i) XZ must contain most information of X, *i.e.* the error $X - XZ$ cannot be too large.

(ii) XZ is only allowed to contain the most principal features, which means that the columns of XZ must be similar to each other. Thus $\|Z\|_*$ cannot be large, since the nuclear norm reflects low-rank self-expressiveness, which is a very good similarity measure for multiple samples [16].

Attempting to balance (i) and (ii), a natural objective function is

$$f = \|Z\|_* + \lambda\|X - XZ\|_{\mathrm{F}}^2, \tag{25}$$

where λ is a trade-off parameter, which is expected to be small, considering the scale of the error.

Plugging (25) into (24), the formulation becomes

$$\begin{aligned} \text{minimize} \quad & \|Z\|_* + \lambda\|X - XZ\|_{\mathrm{F}}^2, \\ \text{s.t.} \quad & X = XZ + LX \\ & Z = V_X W_Z V_X^{\mathrm{T}},\ L = U_X(I - W_Z)U_X^{\mathrm{T}}, \\ & W_Z = \mathrm{diag}(z_1, z_2, ..., z_r), 0 \le z_i \le 1\ \forall i. \end{aligned} \tag{26}$$

Since the z_i's are the only independent variables of problem (26), we can eliminate other variables by using singular values to express the norms, which results in the following equivalent problem.

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{r} z_i + \lambda \sum_{i=1}^{r} (1 - z_i)^2 \sigma_{Xi}^2 \\ \text{s.t.} \quad & 0 \le z_i \le 1 \quad \forall i. \end{aligned} \tag{27}$$

Problem (27) can be solved analytically as

$$w_i^* = \begin{cases} 1 - \frac{1}{2\lambda\sigma_{Xi}^2}, & \sigma_{Xi} \ge \sqrt{\frac{1}{2\lambda}} \\ 0, & \sigma_{Xi} < \sqrt{\frac{1}{2\lambda}} \end{cases} \tag{28}$$

The result can be interpreted as follows. For a small σ_{Xi}, since the information it adds to XZ is too detailed, its negative contribution to the error is smaller than its positive contribution to the nuclear norm, therefore it is filtered out by $z_i = 0$. On the other hand, the larger σ_{Xi} is, the larger z_i will be. This means that the most significant principal components have been preserved.

In contrast,

$$l_i^* = 1 - w_i^* = \min\left\{\frac{1}{2\lambda\sigma_{Xi}^2},\ 1\right\} \tag{29}$$

extracts features corresponding to small σ_{Xi}'s. Our feature extraction procedure is summarized in Table 2.

Given training data X_{train}
$[U_X, \mathrm{diag}\{\sigma_{Xi}\}, V_X] = \mathrm{svd}(X_{\text{train}})$ $l_i^* = \min\left\{\frac{1}{2\lambda\sigma_{Xi}^2},\ 1\right\}$ $L^* = U_X \mathrm{diag}\{l_i^*\} U_X^{\mathrm{T}}$
Use $L^* X_{\text{train}}$ and $L^* X_{\text{test}}$ for classification.

Table 2. Our feature extraction procedure.

4. Experiments

In this section we evaluate both the performance and efficiency of our method. We mainly compared our approach with LatLRR, since it has already been reported in [13] that LatLRR outperforms dimensionality deduction methods such as Locality Preserving Projection (LPP) [8], Neighborhood Preserving Embedding (NPE) [7] and Nonnegative Matrix Factorization (NMF) [6] with a large margin (see Table 2 of [13]).

Datasets We tested our feature extraction using both the Extended Yale Database B [9] and CMU PIE face databases [14], two common datasets for face recognition. Extended Yale B consists of 2414 frontal face images of 38 individuals, and each individual has approximately 64 images. For CMU PIE face databases, the subset of frontal faces (referred to as C27) with different illumination and facial expressions was used, which contains 3329 images of 68 individuals.

Experimental Settings For fair comparison, we adopted the same settings as [13] when conducting performance test on Extended Yale Database B. Each image was resized to 32×28 and reshaped into a data vector of dimension 896, whose entries were normalized to $[0, 1]$. 47% of the randomly split data was used for training and the rest for testing. After Z^* and L^* were learnt, only $L^* X_{\text{train}}$ and $L^* X_{\text{test}}$ were fed into the K-nearest neighbor classifier (K-NN) based on Euclidean distance. We implemented and measured our own method with the trade-off parameter $\lambda = 0.02$, while copying the results of LatLRR from [13].

When conducting the performance test on PIE, the settings remained the same except for the following: we resized each image to 32×32 and used 33% of the data for training; we ran the implementation from the author of [13] to test LatLRR.

To test the efficiency of our method, we recorded the running time of our method followed by 1-NN classification and compared it with that of LatLRR. This experiment was performed on Extended Yale Database B, using a machine with two Intel(R) Xeon(R) E5603 @ 1.6GHz.

Results and Analysis The results of performance tests are displayed in Table 3 and Table 4. As can be seen, our method has similar behavior to LatLRR: it largely outperforms the baseline of "Raw Data", and the product $L^* X$ is a suitable input for dimensionality reduction techniques such as PCA. In Figure 5, we plot the singular values $\{l_i^*\}$ of L^* calculated by (29), and compare with those learnt by LatLRR. From Figure 5, it becomes clear that our method reaps the benefit of weighting effect discussed in Section 2. Moreover, since our method is specifically designed for feature extraction, it further outperforms LatLRR, which was originally designed for subspace clustering. Some examples of using our method to extract detailed features are displayed in Figure 6.

The results of the efficiency test are displayed in Table 5. Since our method only requires a single SVD decomposition, it has an overwhelming advantage over LatLRR when the dimensions of the feature vectors are the same.

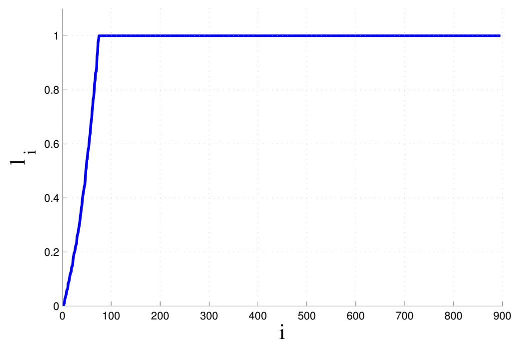

Figure 5. The singular values of L learnt by our method, where l_i denotes the i^{th} singular value. By comparing with Figure 4, we can see that our method has the reweighting effect similar to LatLRR.

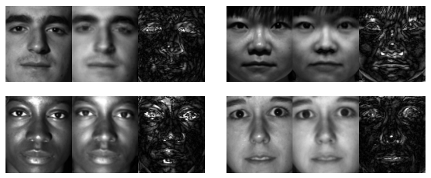

Figure 6. The visualization of the decomposition by our method. In each group of the same individual, the original data X (left) is decomposed into a principal part XZ^* (middle) and a detailed part $L^* X$ (right).

With such efficiency, our method can be applied to higher dimensional data.

Brief Summary Specifically designed for feature extraction, our method can achieve better performance than LatLRR with little computational cost, and can be applied to higher dimensional data effectively.

5. Conclusion

In this paper, we have shown that the weighting effect of the singular values of L accounts for the feature extraction ability of LatLRR. From this insight, we proposed a novel unsupervised feature extraction method by directly designing the transformation matrix, which has a simple analytical solution and achieves better classification performance than LatLRR on two face recognition datasets. Experiment results suggest that our method is efficient enough to be scalable.

References

[1] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(7):711–720, 1997.

	Raw Data	Raw Data+PCA (317D)	LatLRR	LatLRR+PCA (400D)	Ours	Ours+PCA (400D)
1-NN	61.07	61.54	88.76	87.28	93.72	93.09
3-NN	59.81	60.03	87.76	85.95	94.27	93.49
5-NN	58.16	58.54	86.03	85.87	93.88	93.56

Table 3. Classification accuracies (%, averaged over 20 runs) on Extended Yale Database B. For fair comparison, the results related to raw data and LatLRR are cited from [13], who chose the dimension 317D to obtain the best result within the range of 400D.

	Raw Data	Raw Data+PCA (317D)	LatLRR	LatLRR+PCA (400D)	Ours	Ours+PCA (400D)
1-NN	77.52	77.48	94.83	94.83	96.25	96.25
3-NN	71.32	71.32	93.71	93.71	96.12	96.16
5-NN	63.25	63.16	91.21	90.82	96.08	96.03

Table 4. Classification accuracies (%, averaged over 20 runs) on CMU PIE face databases.

Image Size	Running Time(s) LatLRR	Accuracy(%) LatLRR	Running Time(s) Ours	Accuracy(%) Ours
32×28(896D)	160.26	88.78	4.21	93.72
48×42(2016D)	458.28	88.50	10.69	93.38
96×84(8064D)	1097.48	85.61	44.93	94.04

Table 5. Results of efficiency test (averaged over 20 runs) on Extended Yale Database B.

[2] J. Cai, E. Candes, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 2010.

[3] E. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 2011.

[4] J. Costeira and T. Kanade. A multibody factorization method for independently moving objects. *IJCV*, 1998.

[5] M. Fazel. *Matrix rank minimization with applications*. PhD thesis, Stanford University, 2002.

[6] D. Guillamet and J. Vitrià. Non-negative matrix factorization for face recognition. In *Proc. CCIA*. 2002.

[7] X. He, D. Cai, S. Yan, and H. Zhang. Neighborhood preserving embedding. In *Proc. ICCV*, 2005.

[8] X. He and P. Niyogi. Locality preserving projections. In *Proc. NIPS*, 2003.

[9] K. Lee, J. Ho, and D. Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *IEEE TPAMI*, 2005.

[10] Z. Lin, M. Chen, L. Wu, and Y. Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *Technical report, UILU-ENG-09-2215*, 2009.

[11] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma. Robust recovery of subspace structures by low-rank representation. *IEEE TPAMI*, 2013.

[12] G. Liu, Z. Lin, and Y. Yu. Robust subspace segmentation by low-rank representation. In *Proc. ICML*, 2010.

[13] G. Liu and S. Yan. Latent low-rank representation for subspace segmentation and feature extraction. In *Proc. ICCV*, 2011.

[14] T. Sim, S. Baker, and M. Bsatn. The cmu pose, illumination, and expression (pie) database. In *Proc. FG*, 2002.

[15] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 1991.

[16] R. Vidal. A tutorial on subspace clustering. *IEEE Signal Processing Magazine*, 2010.

[17] M. Yin, S. Cai, and J. Gao. Robust face recognition via double low-rank matrix recovery for feature extraction. In *Proc. ICIP*, 2013.

[18] H. Zhang, Z. Lin, and C. Zhang. A counterexample for the validity of using nuclear norm as a convex surrogate of rank. In *Proc. ECML PKDD*. 2013.

[19] H. Zhang, Z. Lin, C. Zhang, and J. Gao. Robust latent low rank representation for subspace clustering. *To appear in Neurocomputing*, 2014.

[20] Z. Zhang, C. Liu, and M. Zhao. Handwriting representation and recognition through a sparse projection and low-rank recovery framework. In *Proc. IJCNN*, 2013.

[21] Z. Zhang, S. Yan, and M. Zhao. Robust image representation and decomposition by laplacian regularized latent low-rank representation. In *Proc. IJCNN*, 2013.

[22] Z. Zhang, S. Yan, and M. Zhao. Similarity preserving low-rank representation for enhanced data representation and effective subspace learning. *Neural Networks*, 2014.