Bilinear low-rank coding framework and extension for robust image recovery and feature representation

Zhao Zhang, Shuicheng Yan, Mingbo Zhao, Fan-Zhang Li

1. Introduction

Vision data (e.g., images) and non-vision data in the real-world emerging applications, such as face recognition [13,21,27,39], robust alignment of images [26], and document retrieval, can usually be characterized by using high-dimensional attributes or features. Also, plenty of real-world multimedia data, including images, videos and documents, can also be characterized by low-rank subspaces, so recent decade has witnessed lots of efforts and increasing attention on the research of recovering low-dimensional or low-rank structures from high-dimensional data with important information in data preserved by feature learning or low-rank coding. Representative works dedicated to these topics include [1–11,15,26,33–38,41–46,50,56]. In this paper, we mainly focus on the study on the bilinear low-rank coding for image recovery, error correction and image representation.

One most representative low-rank recovery criterion is named Robust Principal Component Analysis (RPCA) [3,8,9,16]. For a given observed data matrix $X = [x_1, x_2, \ldots, x_N] \in \mathbb{R}^{n \times N}$ corrupted by certain sparse errors $E_0$, RPCA recovers $X_0(X = X_0 + E_0)$ by solving the following nuclear norm minimization problem:

$$\min Y \|Y\|_n + \gamma \|E\|_n, \text{ Subj } X = Y + E. \quad (1)$$

where $\|\cdot\|_n$ is the nuclear norm of a matrix, i.e., the sum of singular values of the matrix, $\|\cdot\|_l$ is $l^1$-norm ($\|\cdot\|_1$) or $l^2$-norm ($\|\cdot\|_2$) to characterize the sparse errors, and $\gamma$ is a positive weighting parameter. The minimizer $Y$ corresponds to the principal components of $X$ and is also the low-rank recovery to $X_0$. Note that RPCA can well address gross corruptions with large magnitude if only a fraction of entries are corrupted [3,6], But RPCA is a transductive model, so it cannot handle new data [6]. Besides, RPCA implicitly assumes that the underlying data structures lie in or lie near a single low-rank subspace, but most real data are described by using a union of multiple subspaces [1,2], so the recovery of RPCA may be inaccurate in reality. To enable RPCA for including outside data, Inductive Robust

---

Corresponding author at: School of Computer Science and Technology, Soochow University, Suzhou 215006, China.
E-mail address: cszzhang@gmail.com (Z. Zhang).

http://dx.doi.org/10.1016/j.knosys.2015.06.001
0950-7051/© 2015 Elsevier B.V. All rights reserved.
Principal Component Analysis (IRPCA) [6] was recently proposed by seeking a low-rank projection \( U = [u_1, u_2, \ldots, u_n] \in \mathbb{R}^{n \times n} \) to deal with outside data. IRPCA solves the projection \( U \) and the principal components \( Y = [y_1, y_2, \ldots, y_n] \) from the following convex nuclear norm based problem:

\[
\min_{U,E} \|U\|_* + \gamma \|E\|_*, \quad \text{Subj } X = Y + E, \ Y = UX. \tag{2}
\]

The original data can be recovered as \( U^T X \) (or \( X - E \)) by IRPCA. Based on the learnt \( U \), given data can be mapped onto the underlying subspaces and the possible corruptions can be efficiently removed [6]. Note that IRPCA performs recovery along column direction of \( X \), so row information of data is lost by IRPCA.

To well address mixed data with (grossly) corrupted observations, another low-rank criterion called Low-Rank Representation (LRR) [1,2] was also recently proposed for subspace recovery, clustering and segmentation. For subspace segmentation, LRR aims at computing a low-rank representation \( Y = [y_1, y_2, \ldots, y_n] \in \mathbb{R}^{n \times N} \) among all candidates that represent all data vectors as the linear combination of bases in a given dictionary \( D \). By setting \( X \) itself as the dictionary (i.e., \( D - X \)), the convex optimization criterion of LRR is defined as

\[
\min_{U,E} \|V\|_* + \gamma \|E\|_*, \quad \text{Subj } X = DV + E, \ D = X. \tag{3}
\]

After obtaining the optimal solution \( (V^*, E^*) \), the original data is recovered as \( X - E^* \) (or \( X^* \)). Different from IRPCA, LRR recovers or segments given data along row direction, but column information of given matrix is similarly lost by LRR. Note that LRR is also a transductive criterion as RPCA, so it cannot handle new points efficiently. As a result, both LRR and RPCA are inappropriate for the practical applications requiring fast online computation [6]. For image recovery and subspace segmentation, LRR applies the matrix \( X \) itself as dictionary, so it requires that sufficient noiseless data is available in dictionary (i.e., only a part of \( D \) is corrupted). But most real data are contaminated by various errors, e.g., corruptions and noise, so directly setting \( X \) itself as dictionary may be invalid and may depress the robustness performance for subspace recovery and segmentation [11,23].

To overcome the shortcomings of LRR and IRPCA for image recovery, we incorporate the concept of tensor representation [12,28,30] into the low-rank recovery and present a bilinear coding criterion, Tensor Low-Rank Representation (TLRR), for enhancing the robustness of image recovery to noise, corruptions or missing values in data. Compared with the existing studies, the contributions of this paper are summarized as follows. First, to enhance the robustness of image recovery to noise, corruptions or missing values can be greatly improved by our proposed bilinear TLRR model theoretically. Second, we present an out-of-sample extension of TLRR for dealing with the outside images, since TLRR is essentially a transductive criterion as LRR and RPCA. To enable such capability, we add a Least Square (LS)-style [49] regularization term into the objective function of TLRR to compute a projection for correlating features with the low-rank recovery of images so that the bilinear low-rank recovery of new test images can be directly obtained by embedding them onto the projection. Third, we propose two similarity preserving and global structure preserving low-rank subspace learning methods by using the outputted bilinear low-rank codes of TLRR as inputs for image feature extraction and classification.

The paper is summarized as follows. Section 2 briefly reviews the related work. Section 3 proposes the TLRR algorithm mathematically. Subsequently, in Section 4 we present the out-of-sample extension of bilinear TLRR for including outside images. We in Section 5 discuss bilinear low-rank coding for subspace learning. Section 6 shows the settings and evaluates our methods. Finally, the paper is concluded in Section 7. For easy to follow the work, we first present the important notations and abbreviations of algorithms in Table 1.

### 2. Related work

Most real-world data includes noise or corruptions, so recent years have witnessed lots of efforts and increasing interests on low-rank data representation and subspace recovery in the literature. In general, existing works can be roughly divided into two categories. The first category mainly focuses on seeking the low-rank representation for subspace segmentation, recovery and clustering, for instance \([1–3,6,7,11,19,36–38,42–44,50]\). The other category is mainly for low-rank representation by designing overcomplete dictionaries, such as \([23,45]\).

In the first category, the representative criteria are RPCA [4], IRPCA [6], LRR [1,2], Latent LRR (LatLRR) [11], and Fixed LRR (FLRR) [43], etc. In addition, several researchers have suggested effective extensions and enhanced modifications to the original formulations. For example, P. Favaro et al. [45] proposed to enforce the symmetric positive semidefinite constraint explicitly during

---

**Table 1**

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
<th>Abbreviation</th>
<th>Full name of algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Original data matrix</td>
<td>RPCA [3]</td>
<td>Robust Principal Component Analysis</td>
</tr>
<tr>
<td>(x_i)</td>
<td>The i-th sample of X</td>
<td>IRPCA [6]</td>
<td>Inductive Robust Principal Component Analysis</td>
</tr>
<tr>
<td>D</td>
<td>Dictionary matrix</td>
<td>LRR [1,2]</td>
<td>Low-Rank Representation</td>
</tr>
<tr>
<td>U</td>
<td>Low-rank codes matrix</td>
<td>LRR-PSD [45]</td>
<td>LRR with Positive Semi-Definite constraint</td>
</tr>
<tr>
<td>V</td>
<td>Embedding matrix</td>
<td>FLRR [43]</td>
<td>Fixed Low-Rank Representation</td>
</tr>
<tr>
<td>(y_i)</td>
<td>Low-rank recovery</td>
<td>TLRR</td>
<td>Tensor Low-Rank Representation</td>
</tr>
<tr>
<td>d</td>
<td>Reduced dimension</td>
<td>iTLRR</td>
<td>Inductive Tensor Low-Rank Representation</td>
</tr>
<tr>
<td>(W^{at})</td>
<td>Similarity matrix</td>
<td>L-LSRL</td>
<td>Local Low-Rank Subspace Learning</td>
</tr>
<tr>
<td>(|\cdot|_F)</td>
<td>Frobenius norm</td>
<td>G-LSRL</td>
<td>Global Low-Rank Subspace Learning</td>
</tr>
</tbody>
</table>
the process of low-rank coding and presented a new criterion, \textit{LRR with Positive Semi-Definite constraint} (LRR-PSD). The problem of LRR-PSD is defined as

\[
\min_{V,E} \|V\|_l + \gamma \|E\|_l, \quad \text{Subj } X = DV + E, \ D = X, V \geq 0, \tag{4}
\]

where \(V \geq 0\) is a positive semidefinite constraint. A similar work is [42] that aims at calculating a non-negative low-rank and sparse representation \(V\) for recovering low-rank subspaces. The problem is defined as

\[
\min_{V,E} \|V\|_l + \beta \|V\|_l + \gamma \|E\|_l, \quad \text{Subj } X = DV + E, \ D = X, V > 0. \tag{5}
\]

The authors of [36–38] have also suggested matrix bi-factorization and tri-factorization methods for low-rank image recovery. But it is worth noting that virtually all criteria in this category use matrix \(X\) itself as a dictionary to learn the low-rank representation for subspace recovery and segmentation. But if given dataset is corrupted by dense noise, setting \(X\) itself as the dictionary may not be a good choice, since the included noise or corruptions in most real data may directly degrade the subspace recovery or segmentation performance.

To enhance the recovery and error correction performance, typical methods, such as [23,45] belonging to the second category, were recently proposed. The objective is to decompose given data matrix \(X\) as the sum of a clean self-expressive low-rank dictionary plus an error matrix. The non-convex problem that involves three main variables (i.e., dictionary \(D\), low-rank representation \(V\) and sparse errors \(E\)) is formulated as

\[
\min_{D,V,E} \|V\|_l + \gamma \|E\|_l, \quad \text{Subj } X = D + E, \ D = DV. \tag{6}
\]

Note that the above problem is similar to the non-convex formulations of [31,48] which involve three similar variables, namely dictionary, sparse representation [13,30,47] and sparse errors, to estimate from a non-convex problem. According to the Theorem 3.1 of [23], it is easy to check that the above criterion can be equivalent to \(\min_{D,E} \|D\|_l + \gamma \|E\|_l\), Subj \(X = D + E\) and \(V = \text{SIM}(D)\) is a shape interaction matrix [23,40]. It is worth noting that the above low-rank coding formulation can also do simultaneous dictionary learning and low-rank representation, but our approach is different from theirs. The model in Eq. (6) is directly built on LRR by involving the dictionary and low-rank representation to estimate for subspace recovery, while our bilinear low-rank coding model aims at recovering given data from two directions (i.e. row and column) by calculating a pair of low-rank matrix factors to well handle cases corrupted by noise and corruptions in addition to recovering low-rank subspaces. Besides, the dictionary in Eq. (6) is updated directly at each iteration, but our model seeks a low-rank projection \(U\) to update the dictionary instead by projecting given data into their underlying subspaces.

In this paper, we propose a unified framework that can seamlessly integrate the low-rank representation and dictionary learning. The objective is to recover or reconstruct given data along both row and column directions for well handling the observations that may be corrupted by noise and missing values. No extra constraint is included during the low-rank coding process. So, the most related low-rank recovery criteria to ours are IRPCA, LRR and LatLRR. The principles of IRPCA and LRR are illustrated in the top left and top right of Fig. 1, respectively. Obviously, IRPCA and LRR perform image recovery and error correction along either column or row direction of given data respectively, so row or column information of images is lost by them. To well handle the issue of insufficient sampling suffered in LRR and improve the robustness to noise, LatLRR constructs the dictionary by using both observed and unobserved hidden data and solves the following problem:

\[
\min_{V,E} \|V\|_l, \quad \text{Subj } X = DV, \ D = [X,X_0], \tag{7}
\]

where \(X_0\) is the hidden data, and the concatenation (along column) of \(X\) and \(X_0\) is used as the dictionary \(D\). Finally, the optimization problem of LatLRR is formulated as

\[
\min_{U,V,E} \|U\|_l + \|V\|_l + \gamma \|E\|_l, \quad \text{Subj } X = UX + XV + E \tag{8}
\]

when corrupted data is included. Clearly, LatLRR reconstructs given data matrix \(X\) using both row and column information to some extent. Previous studies showed that LatLRR could solve the issue of insufficient sampling to some extent and was proved to be more robust than LRR [11] for recovering low-rank matrices and correcting errors. But it is worth noting that both observed and hidden data in the formulation of LatLRR are sampled from the same collection of low-rank subspaces [11]. Thus, LatLRR may suffer from the same issue as LRR, as one still cannot ensure there are sufficient noiseless data available in dictionary \(D = [X,X_0]\). The principle of LatLRR is given in the bottom of Fig. 1. After the optimal solution \((U^*, V^*, E^*)\) is obtained, LatLRR decomposes given data matrix \(X\) into a

![Fig. 1. The principles of IRPCA (top left), LRR (top right), TLRR (top) and LatLRR (bottom).](image-url)
low-rank part $X^*$, a low-rank part $U^*$, and a sparse part $E^*$ fitting errors.

From Fig. 1, one can intuitively find that LatLRR is a combination of LRR and IRPCA. More specifically, although column and row information of given data matrix $X$ are reflected in the final reconstructive procedure of LatLRR, i.e., $U^*X + X^*$, note that the low-rank matrices $U$ and $V$ of LatLRR can be alternately calculated from the following two equivalent convex problems at each iteration:

\[
\text{Min}_{V E} \| V \|_* + \lambda \| E \|_1, \quad \text{Subj} \ X = DV + E_0, \quad D = X, \quad E_0 = UX + E, \quad (9a) \\
\text{Min}_{U E} \| U \|_* + \lambda \| E \|_1, \quad \text{Subj} \ X = UX + E_0, \quad E_0 = XV + E. \quad (9b)
\]

When solving $V$ for low-rank representation at each iteration, $E_0$ is fixed and the problem in Eq. (9a) is equivalent to LRR by setting the matrix $X$ as dictionary as $E_0$ is considered as the error matrix. Similarly, when solving the low-rank projection $U$, $E_0$ is fixed and the problem in Eq. (9b) is equivalent to IRPCA for encoding the principal components of $X$ if $E_0$ is regarded as the error matrix. It is also worth noting that the so-called hidden effects brought to the problem by $X^*$ still remain unclear. To boost the robustness to noise and well handle data with missing values, we present a new image recovery mechanism from two directions at the same time, as shown in the top of Fig. 1, from which we can see clearly that our approach is different from LatLRR, although row and column information of given data are all considered by them. Based on such strategy, our presented TLRR exhibits certain properties over existing criteria, such as the enhancement of robustness against noise and corruptions, which will be evaluated by simulations.

3. Bilinear low-rank coding for image recovery

We present the bilinear low-rank coding framework named Tensor Low-Rank Representation (TLRR). TLRR for image recovery is driven by learning two low-rank matrix factors $U$ and $V$ at the same time to recover given matrix $X$ from row and column directions in the form of tensor representation [12,28,30]. In tensor scenarios, the matrix $X$ can be regarded as the second-order tensor in the tensor space $\mathbb{R}^n \otimes \mathbb{R}^n$ [12,28,30]. Denote by $U = (u_1, u_2, \ldots, u_d) \in \mathbb{R}^{d_1 \times n}$ and $V = (v_1, v_2, \ldots, v_d) \in \mathbb{R}^{d_2 \times n}$ to represent given matrix $X$, the tensor product $U \otimes V$ is a subspace of $\mathbb{R}^{d_1 \times d_2}$ and the projection of $X$ onto the subspace $U \otimes V$ is $XV \in \mathbb{R}^{d_1 \times d_2}$. In this paper, we mainly consider $d_1 = n$ and $d_2 = N$. We will show the proposed bilinear recovery criterion can perform simultaneous dictionary learning and low-rank representation based on a simple model.

3.1. The objective function

For a data matrix $X \in \mathbb{R}^{n \times N}$ corrupted by certain sparse errors or missing values $E_0$, we propose to recover the original data $X_0(X = X_0 + E_0)$ along both row and column directions simultaneously. More specifically, we aim at calculating a low-rank representation matrix $V \in \mathbb{R}^{N \times N}$ and a low-rank projection matrix $U \in \mathbb{R}^{n \times n}$ such that $X \approx DV$, $D = UX$ from the following rank minimization problem:

\[
\text{Min}_{U V} \text{rank}(U) + \text{rank}(V) + \gamma \| X - \hat{Y} \|_F, \quad \text{Subj} \ \hat{Y} = DV, \quad D = UX. \quad (10)
\]

where $\| \cdot \|_F$ is the nuclear norm of a matrix, $X - \hat{Y}$ identifies the errors. $\| \cdot \|_*$ (either $l^1$-norm $\| \cdot \|_1$ or $l^2$-norm $\| \cdot \|_2$) is for characterizing the sparse errors $E$, and $\gamma$ is a positive control parameter. An informative dictionary $D = UX$ is constructed by calculating a low-rank projection matrix $U$ to project given data onto the underlying subspaces and the entries will be updated at each iteration. Note that $\| E \|_1 = \sum_{ij} |E_{ij}|$ is designed for handling random corruptions, and $\| E \|_{2,1} = \sum_{i=1}^n \sqrt{\sum_{j=1}^n E_{ij}^2}$ can well model sample-specific corruptions and outliers [1,2]. If $l^1$-norm is imposed on the error term $E$, we can rewrite the above problem as

\[
\text{Min}_{U V E} \text{rank}(U) + \text{rank}(V) + \gamma \| E \|_{2,1}, \quad \text{Subj} \ X = \hat{Y} + E, \quad \hat{Y} = DV, \quad D = UX. \quad (11)
\]

Following the common practice in rank minimization problems [1–4], we can replace the rank function with the nuclear norm $\| \cdot \|_*$. Then the above optimization problem further becomes

\[
\text{Min}_{U V E} \| U \|_* + \| V \|_* + \gamma \| E \|_{2,1}, \quad \text{Subj} \ X = DV + E, \quad D = UX, \quad (12)
\]

from which the optimal solution $(U^*, V^*, E^*)$ can be achieved. Thus, the original data can be reconstructed or recovered as $U^*V^*$ (or $X - E^*$). Note that both $U^*$ and $V^*$ are required to recover $X_0$ from two directions, but it is difficult to compute them simultaneously. In this work, we solve $U$ and $V$ alternately, i.e., other variables are fixed when optimizing $U$ or $V$ at each iteration. Specifically, we can compute $U$ and $V$ alternately from the following two equivalent convex surrogates to Eq. (12) during the iterative optimization:

\[
\text{Min}_{U E} \| U \|_* + \| V \|_* + \gamma \| E \|_{2,1}, \quad \text{Subj} \ X = U \bar{A} + E, \quad \bar{A} = XV, \quad (13a) \\
\text{Min}_{V E} \| V \|_* + \gamma \| E \|_{2,1}, \quad \text{Subj} \ X = D \bar{V} + E, \quad \bar{D} = UX, \quad (13b)
\]

where $\bar{A} = XV$ denotes the errors corrected and noise removed data matrix using the low-rank representation $V$, and $\bar{D} = UX$ is a clean informative dictionary obtained by projecting given data onto the underlying subspaces by using the low-rank projection $U$ as IRPCA. That is, the proposed bilinear low-rank coding framework can perform simultaneous subspace recovery, error correction and dictionary learning.

Note that the above two convex problems in Eq. (13) can be solved by using various methods, e.g., Augmented Lagrange Multiplier (ALM) [8,9]. When seeking the low-rank factor $U$ to column-reconstruct $\bar{A}$, we firstly set $U$ to be an identity matrix, namely the dictionary is initialized with the given matrix $X$. After obtaining $V$ from Eq. (13b), we can update the low-rank projection $U$ from Eq. (13a) to column-reconstruct $\bar{A}$. With both $U$ and $V$ obtained alternately, the sparse errors $E$ can be computed from Eq. (12). Note that, in the alternating optimizations, the convex problem in Eq. (13a) can be considered as enhanced IRPCA using errors corrected and noise removed matrix $\bar{A}$ by $V$. Recall that IRPCA also computes a low-rank projection to remove the possible corruptions in given data by projecting the data onto the underlying subspaces [6], but it is worth noting that the data structures represented by $\bar{A} = XV$ will be easier to be projected onto the underlying subspaces than $X$, since the process of encoding $XV$ has already corrected the possible errors in data to some extent and segmented points into their respective subspaces by low-rank representation. Similarly, the problem in Eq. (13b) can be treated as enhanced LRR with a clean informative dictionary $\bar{D}$ learnt from Eq. (13a). Thus, TLRR has the potential to improve the robustness to noise and missing values, compared with LRR and IRPCA. Based on the relationships of LRR, IRPCA and LatLRR shown in Section 2, TLRR can also be considered as enhanced LatLRR. An early version of this paper was presented in [32]. This version also discusses the out-of-sample extension of TLRR for including the outside images, provides the discussion of bilinear
low-rank coding for subspace learning, and conducts a thorough simulation evaluation on image representation and classification.

3.2. Optimization

We employ the inexact Augmented Lagrange Multiplier (ALM) method [9] to solve our bilinear low-rank coding problem in Eq. (12) for efficiency. We first convert the problem to the following equivalent one:

\[
\begin{align*}
\min_{J \in U \times V} \|J\|_F + \|F\|_F + \gamma\|E\|_2,1, \quad \text{Subj } X = UXV + E, \quad U = J, \quad V = F.
\end{align*}
\]

(14)

By constructing the augmented Lagrangian function

\[
H(J, F, U, V, E, Y_1, Y_2, Y_3, \mu) \in \mathbb{R}^{n \times N}, \quad Y_2 \in \mathbb{R}^{n \times N}, \quad Y_3 \in \mathbb{R}^{n \times N}
\]

are Lagrange multipliers, \( \mu \) is a positive parameter and \( \|\cdot\|_F \) is matrix Frobenius norm. We first propose to optimize \( J \). Note that \( J \) are independent of the optimization of \( F \) and the rest problem dependent on \( J \) is also convex. When solving \( J_{k+1}, \) both \( Y_2 \) and \( U \) are assumed to be constants and are set to \( Y_2^{k} \) and \( U_k \) respectively. Then the solution of \( J_{k+1} \) is inferred as

\[
J_{k+1} = \arg \min_J \|J\|_F + \left\langle Y_2^k, U_k - J \right\rangle + \left\langle \mu_k/2 \right\|U_k - J\|_F^2. \]

(16)

Based on simple computations, it is easy to check that the above formulation can be rewritten as

\[
J_{k+1} = \arg \min_J \left( 1/\mu_k \right) \|J\|_F + \left(1/2\right) \left\|J - \Phi_k^\top \right\|_F^2. \]

(17)

where \( \Phi_k^\top = U_k + (1/\mu_k) Y_2^k \). Let \( \Omega, [x] = \text{sgn}(x) \max(|x| - 0) \) denote the shrinkage operator [3.8.9.25], the solution \( J_{k+1} \) can be calculated by using the singular value thresholding algorithm as

\[
J_{k+1} = M_k \Omega_{(1/\mu_k)} \left[ \Sigma_k \right]^T, \quad \text{where } M_k \Sigma_k Q_k^T = \text{svd}(\Phi_k^\top) \text{ is the SVD of } \Phi_k^\top. \]

Note that the optimization of \( F_{k+1} \) is very similar to solve \( J_{k+1} \). Specifically, when optimizing \( F_{k+1} \), the terms independent of \( F \) are constants, and the rest dependent on \( F \) is also convex. To solve \( F_{k+1} \), constants \( Y_2 \) and \( V \) are assumed to be \( Y_2^k \) and \( V_k \) respectively. Then the solution of \( F_{k+1} \) can be computed from the following problem:

\[
F_{k+1} = \arg \min_F \left( 1/\mu_k \right) \|F\|_F + \left(1/2\right) \left\|F - \Phi_k^\top \right\|_F^2. \]

(18)

where \( \Phi_k^\top = V_k + (1/\mu_k) Y_3^k \). Thus \( F_{k+1} \) can be similarly calculated by thresholding singular values [8.9.25] as

\[
F_{k+1} = M_k \Omega_{(1/\mu_k)} \left[ \Sigma_k \right]^T, \quad \text{where } M_k \Sigma_k Q_k^T = \text{svd}(\Phi_k^\top) \text{ is the SVD of } \Phi_k^\top. \]

Note that \( \Phi_k^\top \) is the i-th column of matrix \( \Phi_k \), the i-th column \( E_i^{k+1} \) of solution \( E_{k+1} \) can be obtained as

\[
E_i^{k+1} = \begin{cases} \left\| \Phi_i^\top \right\|_2, & \text{if } \left\| \Phi_i^\top \right\|_2 < \mu_k \left\| \Phi_i^\top \right\|_2, \\ 0, & \text{otherwise}. \end{cases} \]

(25)

For completeness of the method, we summarize the procedure of solving TLR in Algorithm 1, which is based on the inexact ALM. Note that if \( \ell^1 \)-norm is imposed on the sparse error term \( E \), that is to solve \( E \) from

\[
\min_{E \in \mathbb{R}^{n \times N}} \|U\|_F + \|V\|_F + \gamma\|E\|_1
\]

with respect to \( X = DV + E \), \( D = UX \), the sparse errors \( E_{k+1} \) can be calculated from

\[
E_{k+1} = \arg \min_{E \in \mathbb{R}^{n \times N}} \left( E \in \mathbb{R}^{n \times N} \right) \|E\|_F + \|E - \left( \Pi_k^E + (1/\mu_k) Y_1^k \right) \|_F^2
\]

(24)

where \( \Pi_k^E \) is the projection of \( E \) onto \( \mathbb{R}^{n \times N} \).

Let \( \tilde{\Phi}_k = \Pi_k^E + (1/\mu_k) Y_1^k \), according to [1–3], denote by \( \tilde{\Phi}_k \) the i-th column of matrix \( \tilde{\Phi}_k \), the i-th column \( E_i^{k+1} \) of solution \( E_{k+1} \) can be obtained as

\[
E_i^{k+1} = \begin{cases} \left\| \tilde{\Phi}_i \right\|_2, & \text{if } \left\| \tilde{\Phi}_i \right\|_2 < \mu_k \left\| \tilde{\Phi}_i \right\|_2, \\ 0, & \text{otherwise}. \end{cases} \]

(26)

For efficient procedure of solving TLR in Algorithm 1 is Step 1 and Step 2. For efficiency, motivated by [1.6.11], we instead compute \( U \) and \( V \) by converting Eq. (12) to a simpler problem. Based on similar argument, the optimal \( U \) to Eq. (12) can be factorized into

\[
U = \hat{U} (R^{T})^T, \quad \text{with } R \text{ obtained by orthogonalizing the columns of } X. \]

Similarly, the solution \( V \) to Eq. (12) can be factorized into

\[
V = Q V^T, \quad \text{where } Q \text{ is calculated through orthogonalizing the columns of } X^T U. \]

As a result, the objective function in Eq. (12) can be equivalent to the following formulation by replacing \( V \) and \( U \) with \( V^T Q \) and \( U (R^{T})^T \) respectively:

\[
\min_{J \in U \times V} \|J\|_F + \|F\|_F + \gamma\|E\|_2,1
\]

(26)

where \( B = (R^{T})^T X Q^T \). Note that this problem can be similarly solved as Algorithm 1. Based on such strategy, the computational burden can be reduced, especially for the low-dimensional large-scale
problems, i.e., $N$ is larger and $n (n \leq N)$ is relatively smaller, since the above problem is solved with a complexity of $O(n^2N + n^3)$. When orthogonalizing $R'$ and $Q'$, $U$ and $V$ are defined as identity matrices. Note that the conversion property of inexact ALM was well studied when the number of blocks is at most two and it could generally perform well in practice [1,24]. But since there are more than two blocks in our TLRR, which is similar to IRPCA, LRR and LatLRR, and the objective function of Eq. (14) is non-convex, it would be hard to prove that the solution of our TLRR converges to the global optimum theoretically, which is a common situation when handling non-convex problems [19,23,45]. In Algorithm 1, we approach TLRR by alternately solving the five blocks, which are easily solvable at each iteration. This paper finds that $\eta = 1.2$ is a good choice for TLRR. Under this setting, we experimentally observe that TLRR often converges with the iteration number $k$ within the range of 30–250.  

Algorithm 1. Bilinear Low-Rank Coding Framework (TLRR)

**Inputs:** Data matrix $X = [x_1, x_2, \ldots, x_n] \in \mathbb{R}^{n \times N}$, and parameter $\gamma$.

1. Initialize: $k = 0$, $J_k = 0$, $U_k = I$, $F_k = V_0 = 0$, $E_k = 0$, $Y_k^1 = 0$, $Z_k^1 = 0$, $\max_t = 10^{10}$, $\bar{\mu} = 10^{-6}$, $\bar{\gamma} = 10^{-7}$;
2. **While not converged** do

(a) fix the other variables and update $J_k$ with (17);
(b) fix the other variables and update $F_{k+1}$ with (18);
(c) fix the other variables and update $V_{k+1}$ with (19);
(d) fix the other variables and update $U_{k+1}$ with (20);
(e) fix the other variables and update $E_{k+1}$ with (21);
3. update the multipliers $V_1$, $V_2$ and $Y_3$ with
4. update the parameter $\mu$ with $\mu_{k+1} = \min (\eta \mu_k, \max_t)$;
5. check the convergence condition: if

   \[ \max_t (\|Z_{k+1} - J_{k+1}\|_2, \|L_{k+1} - S_{k+1}\|_\infty, \|X - U_{k+1}XV_{k+1}\|_2 - E_{k+1}^{[t]} \|_\infty) < \varepsilon, \text{ stop; else } k = k + 1; \]

   **End while**

3. **Outputs:** $U^* = U_{k+1}$, $V^* = V_{k+1}$, $E^* = E_{k+1}$.

4. Out-of-sample extension for bilinear low-rank coding

Reassembling RPCA [3] and LRR [1], our proposed bilinear low-rank coding algorithm (TLRR) is also essentially a transductive method, namely it is unable to represent unseen data vectors and requires recalculating all points when a new sample vector is inputted, which is computationally expensive and makes the approaches of the kind inappropriate for the real-world applications needing fast online computation. To address this issue, we propose an inductive extension of our bilinear low-rank coding, termed Inductive TLRR (iTTLRR), by incorporating a least square (LS)-style [49] regularization term to bridge features of images by a projection $P \in \mathbb{R}^{n \times n}$ with the bilinear low-rank recovery ($\bar{Y}$) so that the extracted features ($P^T$X) by an informative projection $P$ can be applied to characterize the low-rank recovery of images. Thus, the objective function of iTTLRR is formulated as

\[
\begin{array}{ll}
\min_{U^*,V_*} & \|U\|_F + \|V\|_F + \beta \|P^TX - \bar{Y}\|_F^2 + \gamma \|E\|_F \\
\text{subject to} & \bar{X} = \bar{Y} + E, \bar{Y} = DTV, D = UX, P \in \mathbb{R}^{n \times n},
\end{array}
\]

where $\beta$ and $\gamma$ are two positive weighting parameters for trading-off terms in the problem. It should be noted that the presented objective function of iTTLRR can be similarly optimized as TLRR does. First, we can similarly convert the above problem of iTTLRR to the following equivalent one:

\[
\begin{array}{ll}
\min_{J_k, U_k, F_k, V_k, E_k} & \|U\|_F + \|V\|_F + \beta \|P^TX - \bar{Y}\|_F^2 + \gamma \|E\|_F \\
\text{subject to} & \bar{X} = \bar{Y} + E, \bar{Y} = DTV, D = UX, P \in \mathbb{R}^{n \times n},
\end{array}
\]

The corresponding augmented Lagrangian function $\nabla (f, g, U, V, P, X, Y)$ and $U_k$ can be similarly addressed as

\[
\begin{array}{ll}
\nabla (f, g, U, V, P, X, Y) & = \|U\|_F + \|V\|_F + \beta \|P^TX - \bar{Y}\|_F^2 + \gamma \|E\|_F + (X - UX - V) \\
& + (Y_{k+1} - J_k) + (Y_k, V - F) \\
& + \frac{\mu_{k+1}}{2} \left( \|X - UX - V\|_F^2 + \|U - J_k\|_F^2 + \|V - F\|_F^2 \right).
\end{array}
\]

It is worth noting that the optimization of $f, g, U$ and $F$ are the same as those of TLRR. Next, we will show how to optimize $U, V$ and $P$. We first solve $V$. For the optimization of $V$, both $U$ and $P$ are constants. By dropping terms independent of $V$ from the Lagrangian function $\nabla (f, g, U, V, P, X, Y)$, we have

\[
\begin{array}{ll}
\nabla (V, Y_1, Y_3, \mu) & = \beta \|P^TX - \bar{Y}\|_F^2 + \gamma (X - UX - V - F) \\
& + \frac{\mu_{k+1}}{2} \left( \|X - UX - V\|_F^2 + \|U - J_k\|_F^2 + \|V - F\|_F^2 \right).
\end{array}
\]

\[
\nabla (V, Y_1, Y_3, \mu) = \beta \|P^TX - \bar{Y}\|_F^2 + \gamma (X - UX - V - F) \\
+ \frac{\mu_{k+1}}{2} \left( \|X - UX - V\|_F^2 + \|U - J_k\|_F^2 + \|V - F\|_F^2 \right).
\]

Since $\|P^TX - \bar{Y}\|_F^2 = tr (P^TX - \bar{Y}) (P^TX - \bar{Y})^T$, by taking the derivative w.r.t. $V$ and zeroing it, we can achieve the solution of $V$ at the $(k + 1)$-th iteration as

\[
\begin{array}{l}
V_{k+1} = \left( (\beta + 1)X^T U_k^T U_k X + I_N \right)^{-1} \times \\
\left[ F_k + \frac{\beta}{\mu} (X^T U_k^T P^T X)^T + \frac{1}{\mu} (X^T U_k^T Y_k - Y_3) + X^T U_k^T X - X^T U_k^T E_k \right].
\end{array}
\]

where $tr(\cdot)$ is the trace operator. Similarly, both $V$ and $P$ are constants for the optimization of $U$. By dropping terms independent of $U$, and by taking the derivative with respect to $U$ and zeroing it, we can obtain $U_{k+1}$ as

\[
\begin{array}{l}
U_{k+1} = \left[ I_k + \frac{\beta}{\mu} (P^T X V_k^T X)^T + \frac{1}{\mu} (X^T V_k^T X - Y_k) + X V_k^T X - E_k V_k^T X \right] \\
\times \left( (\beta + 1)X V_k^T X + I_k \right)^{-1}.
\end{array}
\]

\[
\begin{array}{l}
U_{k+1} = \left[ I_k + \frac{\beta}{\mu} (P^T X V_k^T X)^T + \frac{1}{\mu} (X^T V_k^T X - Y_k) + X V_k^T X - E_k V_k^T X \right] \\
\times \left( (\beta + 1)X V_k^T X + I_k \right)^{-1}.
\end{array}
\]

After both $U$ and $V$ are obtained, we are now ready to calculate the projection matrix $P$ to extract features from inside and outside images for characterizing their low-rank recoveries. Based on the least square formulation and optimization [49], the solution of $P_{k+1}$ can be delivered as

\[
\begin{array}{l}
P_{k+1} = (XX^T)^{-1} X \tilde{X}_{k+1} = (XX^T)^{-1} X V_k^T X U_k^T \tilde{X}_{k} = U_k X V_k.
\end{array}
\]

With computed $P$, the out-of-sample problem can be solved. More specifically, when inputting a new test data vector $x_{new} \in \mathbb{R}^n$, its low-rank recovery or reconstruction can be simply approximated as $P^T X_{new}$. Note that the method of iTTLRR can be
similarly implemented as Algorithm 1. For efficiency, we can similarly seek $U$ and $V$ by converting Eq. (28) to a simpler problem by replacing $V$ and $U$ with $Q^*V$ and $\hat{U}(R')^T$ respectively:

$$
\min_{J,F,U,V,P,E} \|U\|_2 + \|F\|_2 + \beta \|P^TX - \hat{U} \hat{B} V\|_F^2 + \gamma \|E\|_1,
$$

(34)

where $Q^*, R'$ and $B = (R')^T XQ^*$ are similarly defined as that of TLRR in Eq. (26), and the step of orthogonalizing both $R'$ and $Q^*$ are also similarly conducted.

5. Subspace learning by bilinear low-rank coding

In this section, we discuss the proposed bilinear low-rank coding framework for unsupervised subspace learning and feature extraction. Two methods, i.e. similarity preserving and global structure preserving, are presented.

5.1. Local Low-Rank Subspace Learning (L-LRSL)

We first aim at proposing a similarity preserving Local Low-Rank Subspace Learning (L-LRSL) approach for dimensionality reduction and feature extraction. L-LRSL seeks an orthogonal projection matrix $\hat{P} \in \mathbb{R}^{d \times n}$ onto which the similarity and locality between points in the constructed bilinear low-rank representation space can be preserved. Let $\|\cdot\|$ be the Euclidean norm, L-LRSL computes the projection $\hat{P}$ by solving the following similarity preserving objective function:

$$
\hat{P} = \arg \min_{P \in \mathbb{R}^{d \times n}} \frac{1}{2} \sum_{i=1}^{N} \| \hat{P}^T(UV^j) - \hat{P}^T(UV^l) \|^2_{W_{ij}^R}, \text{ Subj } \hat{P}^T \hat{P} = I_d,
$$

(35)

where $UV^j$ is the bilinear low-rank recovery to the original data, $d$ is the reduced dimension, $\hat{P}^T \hat{P} = I_d$ is an orthogonal constraint, and $W_{ij}^R$ denotes a symmetric weight matrix for measuring pairwise similarities between images. Based on the low-rank recovery to the original data with corruptions and errors corrected, we construct the entries $W_{ij}^R$ of $W_{ij}^R$ by using the cosine similarity-style measure as

$$
W_{ij}^R = W_{ij}^R = \left( \left\langle (UXV_j^T)^2, (UXV_l)^2 \right\rangle \right) / \left( \| (UXV_j)^2 \| \cdot \| (UXV_l)^2 \| \right).
$$

(36)

Let $D_{ij}^R$ denote a diagonal matrix with the entries being $D_{ij}^R = \sum W_{ij}^R$, by using the matrix expressions, the objective function of L-LRSL can be converted into

$$
\hat{P} = \arg \max_{P \in \mathbb{R}^{d \times n}} \left( \hat{P}^T UXV^j \left( W_{ij}^R - D_{ij}^R \right)^T V^T X^j U \right) \text{ Subj } \hat{P}^T \hat{P} = I_d.
$$

(37)

Thus, the projection axes in $\hat{P}^T$ can be obtained as orthogonal eigenvectors corresponding to the d largest eigenvalues of the following eigen-decomposition problem:

$$
UXV^j \left( W_{ij}^R - D_{ij}^R \right) V^T X^j U \psi_j = \lambda_j \psi_j.
$$

(38)

Clearly, L-LRSL can perform feature reduction and extraction. More specifically, after seeking an orthogonal projection $\hat{P}$ from training set, L-LRSL can embed new coming data for classification. For a given new data vector $x_{new} \in \mathbb{R}^d$, feature extraction can be performed in the form of $\hat{P}^T x_{new} \in \mathbb{R}^d$. We summarize the proposed bilinear low-rank coding based L-LRSL method in Algorithm 2.

Algorithm 2. Local Low-Rank Subspace Learning

Inputs: Data matrix $X \in \mathbb{R}^{n \times n}$, reduced dimension ($d$).
1. Calculate low-rank matrices $U$ and $V$ by TLRR;
2. Obtain the bilinear low-rank recovery $\hat{Y} = U X V^T$;
3. Construct the cosine similarities based on $\hat{Y}$;
4. Output the optimal projection matrix $\hat{P}$.

5.2. Global Low-Rank Subspace Learning (G-LRSL)

We then propose a structure preserving Global Low-Rank Subspace Learning (G-LRSL) algorithm for feature learning. Different from L-LRSL, the formulation of G-LRSL learns an embedding matrix $\hat{P} \in \mathbb{R}^{d \times n}$ that can best preserve the bilinear low-rank representations. The objective function of G-LRSL is defined as

$$
\hat{P} = \arg \min_{P \in \mathbb{R}^{d \times n}} \sum_{i=1}^{N} \| \hat{P}^T(UV^j) - \hat{P}^T(UV^l) \|^2_{W_{ij}^G}, \text{ Subj } \hat{P}^T \hat{P} = I_d.
$$

(39)

Since

$$
\| \hat{P}^T(UV^j) - \hat{P}^T(UV^l) \|^2 = tr\left( \left( \hat{P}^T(UV^j) - \hat{P}^T(UV^l) \right)^T \left( \hat{P}^T(UV^j) - \hat{P}^T(UV^l) \right) \right),
$$

based on setting $\hat{P}^T XX^T \hat{P} = I_d$ and applying the matrix expressions, the above problem can be transformed into

$$
\hat{P} = \arg \max_{P \in \mathbb{R}^{d \times n}} \left( \| \hat{P}^T \left( X X^T U^T + UXV^j UXV^l U^T \right) \hat{P} \| - \lambda \| \hat{P}^T XX^T \hat{P} - I_d \| \right).
$$

(40)

The Lagrangian function of the above problem can be formulated as

$$
J(\hat{P}) = tr\left( \hat{P}^T \left( X X^T U^T + UXV^j UXV^l U^T \right) \hat{P} \right) - \lambda \left( \| \hat{P}^T XX^T \hat{P} - I_d \| \right).
$$

(41)

By taking the derivative with respect to $\hat{P}$ and zeroing it, we can obtain the projection axes in $\hat{P}$ as the orthogonal eigenvectors corresponding to $d$ leading eigenvalues of $\left( X X^T U^T + UXV^j UXV^l U^T \right) \hat{P} = \lambda \hat{P}^T XX^T \hat{P}$. Note that G-LRSL does feature reduction similarly as L-LRSL, and can be similarly implemented.

6. Simulation results and analysis

In this section, we evaluate the validity of the proposed bilinear low-rank coding criteria (TLRR and iTLRR) and low-rank subspace learning algorithms (G-LRSL and L-LRSL) for image recovery against different corruptions, and image feature extraction in classification respectively.

6.1. Baselines and settings

For image recovery and error correction, the performance of our TLRR is mainly compared with the most related criteria, i.e., LRR, LatLRR, RPCA, IRPCA, and Sparse Representation (SR) [13]. For

1. The MATLAB code of LRR is available at https://sites.google.com/site/guangcanliu/LRR.
2. The MATLAB code of LatLRR is available at https://sites.google.com/site/guangcanliu/LatLRR.
3. The MATLAB code of RPCA is available at http://perception.csl.illinois.edu/matrix-rank/sample_code.html.
unsupervised subspace learning, we compare the proposed G-LRSL and L-LRSL with five most popular unsupervised dimensionality reduction methods, i.e., Principal Component Analysis (PCA) [22], Locality Preserving Projections (LPP) [27], Neighborhood Preserving Embedding (NPE) [20], Sparsity Preserving Projections (SPP) [21], and the Sparse Locality Preserving Projections (SLPP). For feature extraction by projection (without dimensionality reduction), the performance of our iTLRR is mainly compared with IRPCA and LatLRR that can also output a projection to extract features from images by embedding images onto the projections [6,11].

(a) SR has similar appearance and applications as LRR, such as subspace recovery and error correction. SR seeks a sparse representation $S$ from the following $l^1$-norm or $l^2$-norm based minimization problem [13]:

$$\begin{align*}
\min_{S} \|S\|_1 + \lambda\|E\|_1, \\
\text{Subj} \quad X = DS + E, \quad D = X, \quad \text{Diag}(S) = 0,
\end{align*}$$

(42)

where SR enforces $\text{Diag}(S) = 0$ to avoid the trivial solution $S = I$, and given matrix $X$ is set as the dictionary for learning the sparsest representations. After the minimizer $S^* = [S_1, S_2, \ldots, S_k] \in \mathbb{R}^{N \times N}$ (with respect to $S$) to the problem is achieved, the original data can be similarly reconstructed as $X^*$ (or $X - E$), which is analogous to the recovery of LRR, where each column vector $s_j$ represents the coefficients for reconstructing $x_i$ and each entry $s^*_{ij}$ represents the contribution of $x_i$ for reconstructing each point $x_i$. Note that the sparse coding process aim at preserving as much information as possible, i.e., the reconstruction error or compression loss over each sample is minimized. Based on this point, one representative SR based subspace learning criterion is SPP that is achieved, the original data can be similarly reconstructed.

(b) Parameter settings. For RPCA, IRPCA, LRR, LatLRR, SR and our TLRR, there is a common parameter $\lambda$ that depends on the noise level of datasets [1]. According to [1], a relatively large $\lambda$ should be used when the included errors are slight and otherwise one should tune $\lambda$ to be relatively small. Besides, LPP and NPE need to estimate the neighborhood size $k$. The kernel width of the Gaussian function used in LPP is similarly estimated as in [18]. In this study, the regularization parameter $\lambda$ in each sparse or low-rank coding problem and the trade-off parameter $\beta$ in our iTLRR are chosen from $[10^{-8}, 10^{-6}, 10^{-4}]$ for fair comparison and better results over tuned parameters are reported for comparing the performance.

(c) Evaluation metrics. For recovery and error correction, the result of each criterion is evaluated by using the reconstruction error $\rho = \|\hat{X}_G - \bar{X}_G\|_F / \|\hat{X}_G\|_F$, where $\hat{X}_G$ is the recovery to given data matrix which is not corrupted and $\bar{X}_G$ is the recovered result over different percentages of pixel corruptions. For feature extraction and image recognition, the one-nearest-neighbor (1NN) classifier with Euclidian distance is used due to its simplicity. In the simulations, each dataset is randomly split into training and test sets. Prior to feature extraction, PCA is used to eliminate the null space of the training set for efficiency. After seeking the projections from the training set, test data are projected onto the projective subspace for evaluating the accuracies. We perform all simulations on a PC with Intel (R) Core (TM) i5 CPU 650 @ 3.20 GHz 3.19 GHz 4G.

Seven real image datasets and one synthetic dataset are involved in this study. The image datasets include Yale face database [14], ORL face database [17], Georgia Tech face database [54], MIT-CBCL face database [29], 3D Object database [52], Phos object database [53], and the CASIA-HWDB1.1 handwriting image database [55]. The synthetic set is a spiral dataset that follows a nonlinear spiral distribution. Following the common practice, all the images of YALE, ORL, Georgia Tech, 3D Object, Phos and MIT-CBCL databases are downsampled to 32×32 pixels due to the consideration of computational efficiency. Hence if each pixel of images is considered an input variable, each real image will correspond to a data point in a 1024-dimensional space.

6.2. Visual image analysis and data representation

We mainly investigate the low-rank recovery and error correction results via visual observation. We evaluate our TLRR and existing algorithms for reconstructing various errors corrupted real object images, real handwriting images, real face images and correcting errors in synthetic data, respectively.

6.2.1. Object image recovery

We first examine the performance of our TLRR for object image recovery. The 3D Object [52] and Phos [53] object databases are employed for simulations. The 3D Object database features modeling shots of 8 objects and 51 cluttered test shots containing multiple objects [52]. The Phos database is a color image database of 15 scenes captured under different illumination conditions. Every scene contains 15 different images that contain objects of different shapes, colors and textures. Typical object images of 3D Object and PHOS are respectively shown in the column (a) of each row Fig. 2. For each database, we choose 8 object images from 8 classes to form a matrix $X$ of size 256×256 for the study. Note that we add different noise to the two databases for observing the results. For 3D Object, we add Gaussian white noise of zero mean and variance 0.05 to the images, and add “salt and pepper” noise to the images with the noise density equaling to 0.05 for Phos in the simulations.

For each data matrix $X$, our TLRR decomposes it to a low-rank part $U^*V^*$ encoding principal features of images and a sparse error part $E^*$ that fits noise. In the simulations, we manually add “salt and pepper” noise to the data matrix, where the noise density is set as 0.05. The $l^2$-norm is regularized on $E$ of TLRR to detect noise if without special remarks. The original, corrupted, recovered and error images are given in Fig. 2, from which we see clearly that our TLRR can effectively identify the noise and recover the errors in object images.

6.2.2. Handwriting image recovery

In this study, we evaluate the proposed TLRR method for reconstructing the handwriting images based on the popular CASIA-HWDB1.1 database [55]. This database has 3755 Chinese characters and 171 alphanumeric and symbols, which are collected from 300 writers. We choose the handwriting digit and handwriting letter images for this simulation. But the original sizes of the handwritings in CASIA-HWDB1.1 are inconsistent, so we resize the
the handwriting digit and letter images to 14 × 14 pixels and 16 × 16 pixels respectively. We randomly select 10 handwriting digits from ‘0’–‘9’ to form a matrix $X$ of size 140 × 140, and choose 100 letter ‘A’ from the database for experiments. For handwriting recovery, we corrupt a percentage of randomly selected pixels in handwriting digits or letters by replacing the gray values with inverted values, that is, each gray value $g$ of corrupted pixel is replaced by using $255 - g$, similarly as [51]. The original, corrupted, recovered and error handwritings are shown in Fig. 3. We can see that our TLRR delivers promising results for detecting and recovering errors in images.

6.2.3. Face image recovery

In this simulation, we discuss the face image recovery against corruptions by low-rank coding. The MIT-CBCL face database [29] that consists of face images of 10 persons is involved in this study. The dataset provides two training sets: (1) High resolution pictures, including frontal, half-profile and profile view, (2) Synthetic images (324 images per person, that is, 3240 face images totally) rendered from 3D head models. The second face image set is used. Note that we have elaborated in Section 3.1 that both IRPCA and LRR are treated as special cases of our bilinear low-rank coding framework (TLRR), and they respectively lose row or column information of the images during the process of recovery. Thus in this study we compare the bilinear face recovery result of our TLRR with those of IRPCA and LRR in Fig. 4. For face image recovery, we corrupt a percentage of randomly selected pixels in face images by similarly replacing the gray values with inverted values. In Fig. 4, we show the original face images (first columns), corrupted face images, recovered face images and errors, respectively. The three rows respectively illustrate the results of IRPCA, LRR and our presented TLRR. From the results, we find our TLRR outperform both IRPCA and LRR for recovering the corruptions in images by considering both row and column information of the face images at the same time during bilinear low-rank coding.

6.2.4. Error correction on synthetic data

In addition to evaluating the low-rank coding methods on real-world databases, we also prepare an experiment for handling the random corruptions by using a synthetic spiral dataset $X$ in noisy case. The sampled set has 2000 vectors (i.e., samples) totally and each sample corresponds to a data point in a three-dimensional (3D) space. The recovery result of our TLRR is

![Fig. 2. Reconstruction of TLRR for handling the objects in 3D Object (First row) and PHOS (Second row) corrupted by Gaussian noise and “salt and pepper” noise, where column (a): original objects, column (b): noised objects, column (c): recovered objects ($U^XV^*$), and column (d): sparse errors.](image)

![Fig. 3. Reconstruction of TLRR to correct errors in handwritings, where column (a): original handwritings, column (b): noised handwritings, column (c): recovered handwritings, and column (d): sparse errors.](image)
mainly compared with those of IRPCA, LRR, SR and LatLRR. We illustrate the correction result of each model in Fig. 5, where Gaussian white noise is randomly added to the $x$-coordinates of 10 percent of data vectors (denoted by square symbol) and Gaussian noise of the same density is also added to the $y$-coordinates of another 10 percent of data vectors (denoted by circle symbol). The noised spiral dataset is given in Fig. 5(a). We mainly evaluate the error correction performance of each model by visually perceiving the number of far away points that are included in the reconstructed spiral manifold or distribution.

From the results, we see clearly that our presented TLRR outperforms other criteria for error correction, since less number of far away points are included in the recovered toy data, which is due to its capability of embedding given data into a low-rank tensor

![Comparison of face recovery](image)

**Fig. 4.** Comparison of face recovery by IRPCA (First row), LRR (Second row) and our TLRR (Third row), where the four columns in each row denote the original faces ($X$), corrupted faces, recovered faces ($U'X$ for IRPCA, $X'V$ for IRPCA, and $U'X'V$ for our TLRR) and estimated sparse errors ($E$).

![Error correction result](image)

**Fig. 5.** Error correction result of each algorithm on the spiral dataset: (a) noised dataset; (b) recovery ($U'X$) of IRPCA; (c) recovery ($X'V$) of LRR; (d) recovery ($XS$) of SR; (e) recovery ($U'X + X'V$) of LatLRR; (f) recovery ($U'X'V$) of our TLRR.
subspace onto which the corruptions can be automatically and effectively recovered from both row and column directions simultaneously. Note that LatLRR also works well on this toy by delivering the highly competitive results to our TLRR, but the performance difference between them is not very clear. Thus, in addition to visually evaluating the performance of LatLRR and our TLRR for correcting random errors, we also provide some numerical results for quantitative comparison between them. The numerical result is obtained by computing the standard deviation of the corresponding reconstructed spiral data by LatLRR and our TLRR. Note that a smaller standard deviation can indicate the distribution of the reconstructed toy data is more close to the main spiral manifold to some extent, i.e., less far away data points from the spiral distribution are included. The computed standard deviations for LatLRR and our TLRR are 3.25 and 3.16 respectively, that is, the recovered result by our proposed TLRR is also better than that of LatLRR on this toy set.

6.3. Quantitative evaluation of image de-noising

In this subsection, we mainly present the quantitative evaluation for the low-rank image de-noising capability by recovery of each method. The quantitative evaluation results are computed by handling the real face images under different levels of pixel corruptions. The quantitative evaluation result of our TLRR is mainly compared with those of RPCA, IRPCA, LRR and LatLRR. In this study, the face images are selected from the real-world ORL database that includes variation in facial expression (smiling/non smiling), facial details (glasses/no glasses) and poses. In total, the database has 40 persons and consists of 10 images per person. Three faces are selected for the experiments and a data matrix of size 32×96 is created. The gray values of the face images are normalized to [0, 1] for this study. To investigate the robustness of various low-rank coding methods, two configurations (that is, faces are corrupted by random corruptions and sample-specific corruptions) are tested respectively.

6.3.1. Recovery against random pixel corruptions

We first test the case that face images are corrupted by random pixel corruptions. In this simulation, we corrupt a percentage of randomly selected pixels from the face images by replacing the gray values with inverted values, i.e., each gray value \( g \) is similarly replaced by \( 1-g \). The corrupted pixels are randomly selected from the faces and the locations are unknown to each method. In this study, we vary the percentage of corrupted pixels from 0 percent to 90 percent, and accordingly increase the values of \( \lambda \) for each method. We use \( \exp(-\rho) \) as a quantitative evaluation of the recovery performance, i.e., the closer \( \hat{X}_{cl} \) and \( \hat{X}_{co} \) are, the bigger the value of \( \exp(-\rho) \) will be. Fig. 6 shows the results of TLRR and its four competitors as a function of the level of pixel corruptions. Fig. 6(b)–(e) mainly evaluate TLRR for recovering faces with 10 (or 20) percentage of corrupted pixels. Fig. 6(f) and (g) quantitatively evaluate each \( l^1 \)-norm or \( l^2 \)-norm based method for error correction. The results are averaged over 15 random pixel selections. From the results, we can find that our TLRR works better than other criteria for correcting corruptions, and the performance of TLRR degrades slower than the others with the increasing corruption percents. It is also observed that \( l^1 \)-norm based criteria are more appropriate choice for recovering the random corruptions than \( l^2 \)-norm based criteria.

6.3.2. Recovery against sample-specific corruptions

We then corrupt a percentage of randomly chosen sample-specific corruptions, i.e., columns of the data matrix. In this study, we add Gaussian noise with zero mean and 0.02 variance to the columns. The corrupted columns are randomly chosen and the locations are also unknown to users. We similarly vary the percentage of corrupted columns from 0 percent to 90 percent for each algorithm. Fig. 7 illustrates the result of each technique as a function of the level of column corruptions. Fig. 7(b)–(e) examine our TLRR for recovering the face images with 20 (or 50) percentage of column corruptions. Illustrations show our TLRR is able to effectively detect the corruptions and correct them then. Fig. 7(f) and (g) illustrate the quantitative evaluation of error correction for \( l^1 \)-norm or \( l^2 \)-norm based criterion. We average the results over 15 random column selections. Similar findings can be found, i.e., TLRR outperforms other methods in identifying and correcting corruptions in most cases. It is also observed that the \( l^1 \)-norm based regularization are able to well model the sample-specific corruptions than \( l^2 \)-norm based regularization.

6.4. Handwritten digits recognition

In this simulation, we mainly evaluate the proposed unsupervised subspace learning methods (i.e., G-LRSL and L-LRSL) for handwriting recognition based on the popular CASIA-HWDB1.1 database. This database consists of 3755 Chinese characters and 171 alphanumeric and symbols, which are collected from 300 writers. Because the original sizes of the handwriting images in the original CASIA-HWDB1.1 database are inconsistent, we resize all handwriting images to 14×14 pixels and then normalize the gray

![Image](image-url)
values of the pixels to [0, 1] for the simulations. In this study, a subset called HWDB1.1-D [10], including 2381 handwritten digits (‘0’–‘9’), from CASIA-HWDB1.1 is used for simulations, and some handwriting sample images of digit ‘0’ to ‘9’ are illustrated in Fig. 8. Three simulation settings over different numbers (T) of training images from each handwriting digit of the sampled set are evaluated. More specifically, \( T = 15, 25, \) and 35 handwriting images per digit are selected for training the projection matrix respectively, and the rest is applied as the test set.

The recognition performance of our G-LRSL and L-LRSL is mainly compared with those of PCA, LPP, NPE, SPP and SLPP. For fair comparison, the same \( l_1 \)-norm or \( l_2;1 \)-norm will be regularized on the sparse error term for each approach in the simulations. The 1NN accuracy using samples without dimensionality reduction is also included as the baseline. The results of our iTLRR for inductive handwriting classification by applying the projection matrix for direct feature embedding are also exhibited. The neighborhood size is set to seven for LPP and NPE if without extra remarks.

The highest mean accuracies and best records for each setting are highlighted in bold.

### Table 2

<table>
<thead>
<tr>
<th>Method</th>
<th>Result</th>
<th>HWDB1.1-D (T = 15 train)</th>
<th>HWDB1.1-D (T = 25 train)</th>
<th>HWDB1.1-D (T = 35 train)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Mean</td>
<td>Best</td>
<td>Dim</td>
</tr>
<tr>
<td>1NN</td>
<td></td>
<td>0.8111</td>
<td>0.8111</td>
<td>/</td>
</tr>
<tr>
<td>PCA</td>
<td></td>
<td>0.7563</td>
<td>0.8111</td>
<td>20</td>
</tr>
<tr>
<td>LPP</td>
<td></td>
<td>0.6118</td>
<td>0.7458</td>
<td>40</td>
</tr>
<tr>
<td>LPP</td>
<td></td>
<td>0.7598</td>
<td>0.8014</td>
<td>36</td>
</tr>
<tr>
<td>NPE</td>
<td></td>
<td>0.7254</td>
<td>0.7507</td>
<td>40</td>
</tr>
<tr>
<td>SPP ((E_k))</td>
<td></td>
<td>0.7351</td>
<td>0.7925</td>
<td>32</td>
</tr>
<tr>
<td>SLPP ((E_k))</td>
<td></td>
<td>0.7598</td>
<td>0.8014</td>
<td>40</td>
</tr>
<tr>
<td>IRPCA ((E_k))</td>
<td></td>
<td>0.8111</td>
<td>0.8111</td>
<td>/</td>
</tr>
<tr>
<td>LatLRR ((E_k))</td>
<td></td>
<td>0.8027</td>
<td>0.8027</td>
<td>/</td>
</tr>
<tr>
<td>iTLRR ((E_k))</td>
<td></td>
<td>0.8420</td>
<td>0.8420</td>
<td>/</td>
</tr>
<tr>
<td>G-LRSL ((E_k))</td>
<td></td>
<td>0.8097</td>
<td>0.8397</td>
<td>20</td>
</tr>
<tr>
<td>L-LRSL ((E_k))</td>
<td></td>
<td>0.8105</td>
<td>0.8396</td>
<td>20</td>
</tr>
<tr>
<td>SPP ((E_k))</td>
<td></td>
<td>0.7075</td>
<td>0.7597</td>
<td>56</td>
</tr>
<tr>
<td>LPP</td>
<td></td>
<td>0.7212</td>
<td>0.7793</td>
<td>44</td>
</tr>
<tr>
<td>NPE</td>
<td></td>
<td>0.8123</td>
<td>0.8123</td>
<td>/</td>
</tr>
<tr>
<td>SPP ((E_k))</td>
<td></td>
<td>0.8282</td>
<td>0.8282</td>
<td>/</td>
</tr>
<tr>
<td>SLPP ((E_k))</td>
<td></td>
<td>0.8312</td>
<td>0.8312</td>
<td>/</td>
</tr>
<tr>
<td>IRPCA ((E_k))</td>
<td></td>
<td>0.8062</td>
<td>0.8350</td>
<td>16</td>
</tr>
<tr>
<td>LatLRR ((E_k))</td>
<td></td>
<td>0.8087</td>
<td>0.8340</td>
<td>16</td>
</tr>
</tbody>
</table>

The highest mean accuracies and best records for each setting are highlighted in bold.
illustrate the averaged result of each method over 10 random splits of training/testing images. In this simulation, we fix the number of training data \((T)\) and vary the number of reduced dimension \(d\) from 4 to 60 with interval 4. Table 2 describes the mean results over different reduced dimensions. The highest mean accuracies and best records for each setting are highlighted in bold. The highest accuracies and optimal handwriting image subspace \((\text{Dim})\) according to the highest accuracy of each model are also illustrated. Note we divide the results into two groups according to the use of \(l^1\)-norm or \(l_1^{\frac{1}{2}}\)-norm in SPP, SLPP, IRPCA, LatLRR, iTLRR, G-LRSL and L-LRSL. We also highlight the best records in each group for comparison. We obtain the following observations. First, we see clearly that the recognition performance can be greatly boosted by the increasing number of training data. Second, for handwriting digit recognition by dimensionality reduction (i.e., comparing our G-LRSL and L-LRSL with PCA, LPP, NPE, SPP and SLPP), we can find that our G-LRSL and L-LRSL that are comparable outperform other existing methods by about 4–5% improvement in accuracy in most cases. In addition, the proposed G-LRSL and L-LRSL achieve the highest accuracy than other criteria by applying small number of reduced dimensions \(d\), which is desired for image feature extraction in classification. It is also noted that SPP and SLPP deliver better results than LPP and NPE in most cases. PCA also works well on this dataset. For handwriting digital image feature extraction by projection without dimensionality reduction (i.e., comparing our iTLRR with 1NN, IRPCA and LatLRR), we find our iTLRR gains comparable and even better results than 1NN, IRPCA and LatLRR in most cases. Note that LatLRR delivers the higher records than 1NN, IRPCA and our iTLRR for the case of \(T=25, 35\) and \(l^1\)-norm regularization.

6.5. Face recognition

In this study, we address a face recognition task to examine the face representation power of our unsupervised G-LRSL and L-LRSL by subspace learning and face image feature extraction. For this study, we prepare a new face set called Georgia-Yale face database by merging the face images of Georgia Tech face database [54] and Yale face database [14] into a single set. The Georgia Tech face database [54] contains face images of 50 people with 15 images taken at the Center for Signal and Image Processing at Georgia Institute of Technology. The average size of face images is \(150 \times 150\) pixels. The pictures show frontal and/or tilted faces with different facial expressions, lighting conditions and scale. The Yale face database [14] contains 11 face images of each of 15 persons and the face images demonstrate variations in lighting conditions, facial expressions and with/without glasses. For face recognition on Georgia-Yale, we normalize the gray values in each row of the training set into \([0, 1]\), fix the numbers of training face images from each person \((i.e., T=6, 8, and 10)\) and vary the number \(d\) of reduced dimensions from 5 to 90 with interval 5. Note that we show some face images of 15 individuals in the Georgia Tech and Yale face databases are shown in Fig. 9 for visual observation.

We illustrate the averaged recognition result of each method over 10 random splits of training and test images in Table 3, where we highlight the best records in each group \((i.e., l^1\)-norm or \(l_1^{\frac{1}{2}}\)-norm regularization on SPP, SLPP, IRPCA, LatLRR, iTLRR, G-LRSL and L-LRSL\) similarly for comparison. We find from the results that: (1) The increasing number of training images can significantly enhance the face recognition performance; (2) For dimension reduction for face recognition \((i.e., comparing our G-LRSL and L-LRSL with PCA, LPP, NPE, SPP and SLPP)\), we can observe that the presented G-LRSL and L-LRSL can deliver 2–3% improvement over other unsupervised feature learning criteria. More importantly, our formulations deliver the promising results by using small number of \(d\), compared with the other approaches. For face feature extraction by projection without dimensionality reduction \((i.e., comparing our iTLRR with 1NN, IRPCA and LatLRR)\), we can find that IRPCA LatLRR, and our iTLRR outperforms 1NN by delivering higher accuracies in the cases of \(T=25, 35\). We also notice that our iTLRR is able to deliver the better results than 1NN, IRPCA and LatLRR in most cases, and the exception is that IRPCA outperforms LatLRR and our iTLRR for recognition in the case of \(T=35\) and \(l_1^{\frac{1}{2}}\)-norm regularization. SLPP obtains slight higher accuracy than SPP in most cases, and both are superior to the baseline method, PCA and NPE. Note that the best records of LPP are very close to our algorithms, but the gained mean results by LPP are relatively worse, which may be caused by the unusual trend in performance.

![Georgia Tech face samples](image-1)

![Yale face samples](image-2)

**Fig. 9.** Some representative face images in the Georgia Tech and Yale face database.
We proposed a bilinear low-rank coding framework named Tensor Low-Rank Representation for image recovery and data representation. Our methodology seamlessly integrates low-rank coding and dictionary learning into a unified framework, providing a new mechanism for recovering low-rank images and learning a clean informative low-rank dictionary from a nuclear norm minimization problem simultaneously. Our model proceeds low-rank image recovery and representation by enhancing the robustness to noise and missing values by considering both column and row information of given data at the same time. We mathematically elaborate that our formulation can be regarded as the enhanced information of given data at the same time. We mathematically prove the convergence of inexact ALM based optimization algorithms for exact recovery of a corrupted low-rank matrix, Technical report, UILU-ENG-2010-2215, 2010.

Visual and quantitative evaluations of image recovery and error correction verified the effectiveness of our approach for boosting the robustness against corruptions. Classification on real handwriting and face databases demonstrated the superior performance of our methods for representing and recognizing images. But there are certain future directions to explore. First, to date it is still challenging to strictly prove the convergence of inexact ALM based recovery methods that include more than two blocks [1], including other existing criteria and our method. Second, extending the proposed bilinear low-rank coding model for handling images using matrix form directly, which is very efficient due to the small size, is also an important future work.

Acknowledgements

This work is partially supported by the National Natural Science Foundation of China (Grant Nos. 61402310 and 61373093), the Major Program of National Natural Science Foundation of China (Grant No. 61033012), and the Natural Science Foundation of Jiangsu Province of China (Grant No. BK20140008).