A Sparse Projection and Low-Rank Recovery Framework for Handwriting Representation and Salient Stroke Feature Extraction

ZHAO ZHANG, Soochow University
CHENG-LIN LIU, Chinese Academy of Sciences
MING-BO ZHAO, City University of Hong Kong

In this paper, we consider the problem of simultaneous low-rank recovery and sparse projection. More specifically, a new Robust Principal Component Analysis (RPCA) based framework called Sparse Projection and Low-Rank Recovery (SPLRR) is proposed for handwriting representation and salient stroke feature extraction. In addition to achieving a low-rank component encoding principal features and identify errors or missing values from a given data matrix as RPCA, SPLRR also learns a similarity preserving sparse projection for extracting salient stroke features and embedding new inputs for classification. These properties make SPLRR applicable for handwriting recognition and stroke correction, and enable online computation. A Cosine-similarity-style regularization term is incorporated into the SPLRR formulation for encoding the similarities of local handwriting features. The sparse projection and low-rank recovery are calculated from a convex minimization problem that can be efficiently solved in polynomial time. Besides, the supervised extension of SPLRR is also elaborated. The effectiveness of our SPLRR is examined by extensive handwritten digital repairing, stroke correction and recognition based on benchmark problems. Compared with other related techniques, SPLRR delivers strong generalization capability and state-of-the-art performance for handwriting representation and recognition.

Categories and Subject Descriptors: I.2.6 [Artificial Intelligence]: Learning; I.4.10 [Image Processing and Computer Vision]: Image Representation; I.5.2 [Pattern Recognition]: Applications

General Terms: Algorithms, Design, Experimentation, Performance

Additional Key Words and Phrases: Sparse projection, low-rank recovery, similarity preservation, salient stroke feature extraction, handwriting representation and recognition

ACM Reference Format:

1. INTRODUCTION

The enormous and ever-increasing multimedia data (for instance documents, images, and videos) in the Internet and daily communication poses huge challenges of efficient high-dimensional data understanding. These data from the real world can often be characterized by using low-rank subspaces and in most cases contain certain errors or missing values that are caused by pixel corruptions, or even the variability...
of writing styles of different writers in handwriting recognition, which leads to considerable attention and interests on automatic recovery of low-rank structures and identification of the errors, e.g. [1-12], [29]. One most representative low-rank recovery criterion is called Robust Principal Component Analysis (RPCA) [3], [4], [8], which has been widely applied in various areas, for instance document data analysis [13], image tag refinement [14], texture transformation [15], and face recovery [7]. Specifically, for a given observed data matrix \( X = \{x_1, x_2, \ldots, x_n\} \in \mathbb{R}^{m \times n} \) corrupted by certain errors \( E \), RPCA can exactly recover the original data \( X_0 (X = X_0 + E) \) through solving a convex nuclear norm minimization problem:

\[
\{Y^*, E^*\} = \arg\min_{Y, E} \left\{ \|Y\|_* + \lambda \|E\|_1 \right\}, \quad \text{Subj } X = Y + E,
\]

where \( \|\cdot\|_* \) denotes the nuclear norm of a matrix [8], \( \|\cdot\|_1 \) denotes the \( \ell_1 \)-norm, \( \ell^2 \)-norm or squared Frobenius norm for characterizing the errors \( E \), and \( \lambda \) is a positive parameter. From the above problem, the minimizer \( Y^* \) (with respect to the variable \( Y \)) is called the principal components of \( X \) and is also the low-rank recovery to \( X_0 \). RPCA can well address the gross corruptions with large magnitude if only a fraction of entries are corrupted [8], but it is essentially a transductive method, that is, it is unable to represent unseen data and requires recalculating all points when new data is input [12]. To overcome this issue, a generalization to RPCA, termed Inductive Robust Principal Component Analysis (IRPCA) [12], was recently proposed. Unlike RPCA, IRPCA targets on the error correction by calculating a low-rank projection \( Q \in \mathbb{R}^{m \times r} \) to remove the possible corruptions from a given set of observations. From the data matrix \( X \), IRPCA learns the low-rank projection \( Q \) to produce the low-rank principal components \( Y = [y_1, y_2, \ldots, y_n] \) by solving the following convex nuclear norm minimization problem:

\[
\{Q^*, E^*\} = \arg\min_{Q, E} \left\{ \|Q\|_* + \lambda \|E\|_1 \right\}, \quad \text{Subj } X = Y + E, Y = QX.
\]

When obtaining the optimal solution \( Q^* \) (with respect to the variable \( Q \)), the sparse errors can be estimated as \( X - Q^*X \). Based on the learnt projection matrix \( Q^* \) from the training data, IRPCA can represent new data and is able to effectively remove the possible corruptions by projecting the datum onto the subspace spanned by \( Q^* \) [12]. But note that the similarities of local features to be encoded are not considered when seeking \( Q^* \) in IRPCA.

In this paper, we propose a unified nuclear norm minimization and sparse projection framework, termed Sparse Projection and Low-Rank Recovery (SPLRR), for recovering low-rank matrices, extracting visual salient features and removing the errors simultaneously. SPLRR is a generalization to the recent RPCA via incorporating a sparse feature extraction term and at the same time encoding the similarities of local salient features. SPLRR is mainly proposed for processing handwriting images, motivated by the following two facts: (1) The stroke directions and shapes of the handwritten digits or characters are usually irregular and even curved due to the variability of writing styles [26][28]. That is, handwriting images usually bear certain errors or missing stroke regions, as illustrated in Figures 2 and 4, to be repaired. Based on this point, we learn a low-rank principal component matrix to recover the irregular or curved strokes. (2) The irregularity of those stroke
directions and shapes may precisely reflect the individuality of writing styles and hence becoming useful in identifying the personality in handwriting recognition. To handle this problem, we learn a sparse projection to extract salient stroke features encoding irregular or curved strokes and simultaneously encoding the similarities of local features. As a result, the presented SPLRR will have the potential to facilitate personal handwriting identification, which will be discussed in detail in future.

Given a set of handwriting data matrix \( X \) corrupted by certain errors, the objective of SPLRR is to decompose it to a low-rank component encoding principal features to be recovered, a sparse component encoding salient stroke features of handwritings that can best distinguish different handwritings in classification, and a sparse part fitting errors. Formally, for a given set of handwriting data vectors containing certain errors, we in this paper consider the following recovery problem:

**Problem 1 (Handwriting recovery and salient stroke feature extraction):** For a given handwriting data vector \( x \) expressed as

\[
x = r + f + e, \quad \text{where } x, r, f, e \in \mathbb{R}^n,
\]

the goal is to automatically identify and recover the low-rank principal component \( r \) by regenerating the missing values, extract the visually salient stroke features \( f \) for handwriting classification, and correct the possible errors \( e \). The simulations of this work will illustrate how our SPLRR algorithm can effectively repair the missing pixel values and capture the saliency features of handwriting data. An important merit of our SPLRR framework is to learn a similarity preserving sparse projection to extract salient stroke features from a set of training samples, so Sparse Representation (SR) [17], [19] is another related criterion. In this work, we consider the following convex problem to calculate the sparse representation \( R \) of all data vectors jointly [6]:

\[
\begin{align*}
\left( R^*, E^* \right) &= \arg\min_{x, R, E} \left\{ \| R \|_1 + \lambda \| E \|_1 \right\}, \\
\text{Subj. } &\ X = DR + E, \ \text{Diag}(R) = 0,
\end{align*}
\]

where \( \text{Diag}(R) = 0 \) is to avoid the trivial solution \( R = I \), and \( X \) itself is usually set as the dictionary \( D \). SR has attracted considerable interests and demonstrated effective in the context of image representation and classification, for instance [16], [17], [18], [19], [20], [43]. Most importantly, the sparse representations are able to exhibit a natural discriminating power [17], [19]. However, estimating an optimal informative dictionary for sparse representation and coding is never easy and sparse coding is also a transductive criterion, i.e. it cannot represent new inputs for classification.

Compared with the previous studies, the major contributions of this present paper can be summarized as follows:

—Technically, we propose a simple yet effective framework, termed SPLRR, for dealing with handwritings, including handwritten recovery and salient stroke feature extraction. Note that SPLRR does not involve the step of learning dictionary and the criterion is convex, which can be solved in polynomial time.

—For salient stroke feature extraction, we derive a variant of SR to seek a sparse projection for embedding new inputs in handwriting recognition. To encode the
similarity of local handwriting features, a Cosine similarity based regularization term is incorporated into SPLRR for similarity preservation.

—The proposed SPLRR can be effectively extended to the supervised scenario. In addition, an effective unsupervised subspace learning framework based on SPLRR is also derived, which is able to deliver discriminant projections for embedding new handwriting patterns into their lower-dimensions for classification.

—Strong generalization capability and state-of-the-art performance are obtained by SPLRR for handwriting recovery, stroke correction and recognition, by comparing with other related criteria. Specifically, certain existing low-rank recovery and sparse coding formulations are considered as special cases of SPLRR.

The remainder of this paper is organized as follows. Section 2 briefly reviews the related studies. Section 3 proposes SPLRR. Section 4 discusses its connection with the other related criteria and introduces the supervised extensions of SPLRR. The applications of SPLRR to handwriting recovery, stroke correction, recognition are described in Section 5. Finally, we conclude the paper in Section 6.

2. BRIEF REVIEWS OF RELATED WORK

In recent years, the studies on low-rank recovery and sparse representation have attracted considerable attention and also have produced numerous new algorithms and extensions of previous work. Many algorithms, including RPCA, are based on nuclear-norm minimization, e.g., [1-12]. To well handle mixed data, a generalization to RPCA, termed Low-Rank Representation (LRR) [6], was recently proposed. LRR aims at calculating the low-rank representation \(Z = [z_1, z_2, \ldots, z_N] \in \mathbb{R}^{N \times K}\) among all the candidates that represent the vectors as the linear combination of the bases in a given dictionary \(D\). By setting the given data matrix \(X\) itself as the dictionary \(D\), the problem of LRR is formulated as

\[
\left(Z^*, E^*\right) = \arg\min_{Z,E} \|Z\|_1 + \lambda \|E\|_1, \quad \text{Subj.} \quad Z = DZ + E, \quad D = X.
\]  

(5)

Clearly, LRR reduces to RPCA when the dictionary \(D = I\). With the solution \((Z^*, E^*)\) to the above problem calculated, the original data \(X\) is recovered by \(X = DZ + E\). Different from RPCA, LRR is aimed for addressing the subspace clustering and segmentation problem by using \(Z^*\). Note that LRR shares a similar form as the \(\ell^1\)-norm minimization based SR problem, in the sense that both can perform recovery and subspace clustering [6]. Recently, Zhou et al. studied the random projections based approximated “low-rank plus sparse” decomposition of a matrix and developed a new method called Go Decomposition (GoDec) [5], which is formulated as

\[
\left(Y^*, A^*\right) = \arg\min_{Y,A} \|Y\|_* + \alpha \|A\|_0, \quad \text{Subj.} \quad \text{rank}(Y) \leq r, \quad \text{card}(A) \leq \alpha,
\]  

(6)

where \(r\) and \(\alpha\) are constants, and \(\text{card}(A)\) is the number of nonzero entries in matrix \(A\). Note that a common shortcoming of RPCA, SR, LRR and GoDec is that they are essentially transductive, i.e., these criteria need to recalculate over all data when new samples are input. This causes heavy computational burden and hinders fast computation in online application. Moreover, when given data is corrupted by dense noise, the performance of SR and LRR is largely influenced by the choice of
A Sparse Projection and Low-Rank Recovery Framework for Handwriting Representation and Salient Stroke Feature Extraction

3. PROBLEM FORMULATION

The SPLRR framework considers two major issues: low-rank recovery and similarity-preserving sparse projection for saliency feature extraction. The details are as follows.

3.1 Sparse Projection and Low-Rank Recovery (SPLRR) Framework

Given a set of handwritten vectors \( X = \{x_1, x_2, ..., x_n\} \in \mathbb{R}^{n \times N} \) corrupted by certain errors, SPLRR calculates a similarity preserving sparse projection \( S \in \mathbb{R}^{m \times N} \), and decomposes \( X \) into a low-rank principal component \( L \in \mathbb{R}^{m \times N} \), a sparse component \( SX \) encoding the salient stroke features and an error part \( E \) by solving the following problem:

\[
(Z^*, E^*) = \arg\min_{Z, E} \|X - Z\|_F \text{ Subj } Z = DX + E, D = [X, X_n].
\]
\[
(L', S', E') = \arg \min_{L, S, E} \left\{ (1 - \beta) \text{rank}(L) + \beta \|S\|_1 + \xi \left( \frac{1}{2} \|X - Y\|_2^2 + \lambda \|E\|_1 \right) \right\},
\]
Subject to \(Y = L + SX\)

(8)

where \(\|\cdot\|_0\) is the \(l^0\) norm, \(\|\cdot\|_1\) denotes the \(l^1\) norm, \(\beta \in [0, 1]\) is a parameter for trading off the low-rankness and the sparsity, \(\lambda > 0\) is a parameter relying on the level of errors, \(X - Y\) identifies the errors \(E\), and \(\tilde{f}_i(X)\) with a nonnegative parameter \(\xi\) denotes a regularization term that is integrated to encode the similarity between salient stroke features at each iteration. Note that the above problem is not directly tractable and is difficult to solve due to the discrete nature of the rank function, \(l^0\) norm and the \(l^1\) norm. As a common practice in rank minimization [6], [8], [9], we replace the rank function with the nuclear norm. Also, by relaxing the \(l^0\) norm and the \(l^1\) norm with \(l^2\) norm, and \(l^2\) norm, and \(l^2\) norm, respectively, we obtain the following convex surrogate:

\[
(L', S', E') = \arg \min_{L, S, E} \left\{ (1 - \beta) \|L\|_2 + \beta \|S\|_1 + \xi \left( \frac{1}{2} \|X - Y\|_2^2 + \lambda \|E\|_1 \right) \right\},
\]
Subject to \(Y = L + SX + E\)

(9)

where the matrix nuclear norm \(\|\cdot\|_\text{nu}\), matrix \(l^2\) norm \(\|\cdot\|_2\) and matrix \(l^1\) norm \(\|\cdot\|_1\) are respectively defined as

\[
\|L\|_\text{nu} = \sum \sigma_i(L), \quad \|S\|_1 = \sum \|S_i\|_1, \quad \|E\|_1 = \sum_{i,j} \|E_{i,j}\|_1,
\]

(10)

where \(\sum \sigma_i(L)\) denotes the sum of the singular values of matrix \(L\). For the extracted handwritten salient stroke features \(SX\) in the optimizations, we define \(\tilde{f}_i(X)\) as follows to preserve the similarities between them:

\[
\tilde{f}_i(X) = \frac{1}{2} \sum_{j=1}^N w_{ij}^0 d^2(\tilde{S}_j - S_j) = \frac{1}{2} \sum_{j=1}^N w_{ij}^0 \text{Tr} \left( [S_j - S_i] [S_j - S_i]^T \right),
\]

\[
= \text{Tr} \left( \sum_j S_j \left( \sum_i W_{ij}^0 x_i^T - \sum_j S_j W_{ij}^0 x_j^T \right) S_j^T \right)
\]

(11)

where \(d^2(\tilde{S}_j - S_i)\) represents the squared Euclidean distance between salient handwriting features \(S_i\) and \(S_j\), \(Q^0 = X (Q^0 - W^0) X^T = X F^0 X^T\) is a symmetric matrix, \(\|\cdot\|\) denotes the \(l^1\) norm, and \(Q^0 = \sum_{i,j} W_{ij}^0\). In this paper, we apply the Cosine similarity to define the weight matrix \(W^0\) for encoding the similarity between handwritten features, where the similarity is measured by the angle between them. It should be noted that the Cosine similarity measure was widely used for comparing documents in text mining [36], [50]. For given two handwritten vectors \(x_i\) and \(x_j\), the matrix \(W^0\) is weighted using the Cosine similarity as

\[
W_{ij}^0 = \exp(\cos(\theta)), \text{ where } \cos(\theta) = \frac{\langle x_i, x_j \rangle}{\|x_i\| \cdot \|x_j\|},
\]

(12)

where the inner product \(\langle x_i, x_j \rangle = x_i^T x_j\). Clearly, the resulting similarity values are within \([0, 1]\], that is, the higher of the values, the closer of the corresponding handwriting pairs. It should be noticed that the regularized technique has been
widely applied in the machine learning community, e.g., the Laplacian regularization in sparse coding for encoding the similarity and locality between features [18], [20]. It is worth noting that $k$-nearest-neighbor search [25] is involved in these studies, but estimating an optimal $k$ number for locality preservation still remains an open problem. From Eq.9, we refer to the minimizer $\mathcal{L}^*$ (with regard to variable $L$) as the lowest-rank recovery to the original data, and the minimizer $S^*$ (with regard to variable $S$) as the sparsest representation of $X$. Moreover, for a given new data $\Delta$, the saliency features can be extracted by using $S^*_\Delta$ and the errors can be estimated as $\Delta - \mathcal{L}^* - S^*_\Delta$. Thus, the sparse projection $S^*$ can be used for feature extraction in handwritten recognition. Since a sparse similarity preserving feature extraction term is integrated into the RPCA criterion, the derived framework is referred to as robust Sparse Projection and Low-Rank Recovery (SPLRR). In addition, SPLRR can avoid estimating an optimal dictionary matrix, which is challenging in reality.

3.2 Efficient Solutions by Convex Optimization

This section will describe how to calculate $L$, $S$ and $E$ for robust low-rank recovery and salient feature extraction. Because the proposed SPLRR formulation in Eq.9 is convex, it can be optimized by various approaches, e.g., the Argument Lagrange Multiplier (ALM) or the inexact ALM [9]. To facilitate the optimization, we first convert the convex problem in Eq.9 to the following equivalent one:

$$
\min_{L, W, S, E} \left( (1 - \beta) \|L\|^1 + \beta \|W\|^1 + \xi \text{Tr} \{ (SG^0)^T \} + \lambda \|E\|_1 \right),
$$

where $\text{Tr} \{ \cdot \}$ is the trace operator. The corresponding augmented Lagrangian function $\psi$ can be addressed as

$$
\psi(L, W, S, E, \gamma, \gamma, \mu) =
(1 - \beta) \|L\|^1 + \beta \|W\|^1 + \xi \text{Tr} \{ (SG^0)^T \} + \lambda \|E\|_1 + \gamma (S - W) + \frac{\mu}{2} \left( \|S - W\|^1 + \|X - L - S - E\|^1 \right),
$$

where $\gamma$, $\gamma$ are Lagrange multipliers, $\mu$ is a positive scalar, and $\| \cdot \|$ is the matrix Frobenius norm. Notation $A^T$ denotes the transpose of the matrix $A$. The ALM algorithm alternately updates the variables $L$, $W$, $S$ and $E$, through iteratively minimizing the augmented Lagrangian function $\psi$:

$$
\left( L_{i+1}, W_{i+1}, S_{i+1}, E_{i+1} \right) = \arg \min_{L, W, S, E} \psi(L, W, S, E, \mu)
Y^{i+1} = Y^i + \mu (S_{i+1} - W_{i+1})
Y^{i+1} = Y^i + \mu (X - L_{i+1} - S_{i+1} - E_{i+1})
$$

As elaborated in [21], [22], the iteration converges to the optimal solution of the problem in Eq.13 when $\{ \mu \}$ is a monotonically increasing. Since the variables $L$, $W$, $S$ and $E$ are dependent on each other, the above problem cannot be solved directly. This paper updates the variables alternately with others fixed through iteratively solving the following convex sub-problems:
Algorithm 1. Sparse Projection and Low-Rank Recovery Framework

Input: Original data matrix $X \in \mathbb{R}^{n \times r}$, and control parameters $\beta, \xi, \lambda$.
Output: The sparse projection matrix ($S' \leftarrow S_{\mu}$), low-rank recovery ($L \leftarrow L_{\mu}$), and sparse errors ($E \leftarrow E_{\mu}$).
Initialization: $k = 0$, $L_0 = 0$, $W_0 = S_0 = 0$, $E_0 = 0$, $Y^0_0 = 0$, $Y^0_1 = 0$, $\max_\mu = 10^6$, $\mu_\nu = 1.12$, $\epsilon = 10^{-7}$.

While not converged do
  fix the other variables and update $L_{\mu}$ with
  \[ L_{\mu} = \arg \min_{L \geq 0} \left( \frac{1}{2} \| \beta \| L + \frac{1}{\mu} \right) \left\| L - (X - S_{\mu} X - E_{\mu}) \right\|_F^2 ; \]
  fix the other variables and update $W_{\mu}$ with
  \[ W_{\mu} = \arg \min_{W \geq 0} \left( \frac{1}{2} \| \beta \| W + \frac{1}{\mu} \right) \left\| W - (S_{\mu} + Y^0_1) \right\|_F^2 ; \]
  fix the other variables and update $S_{\mu}$ with
  \[ S_{\mu} = \left[ W_{\mu} + (X - L_{\mu} - E_{\mu}) X^T (Y^0_1 X^T - Y^0_1) / \mu \right] (I + \xi G^{(\mu)} + XX^T)^{-1} ; \]
  fix the other variables and update $E_{\mu}$ with
  \[ E_{\mu} = \arg \min_{E \geq 0} \left( \frac{1}{2} \| \beta \| E + \frac{1}{\mu} \right) \left\| E - (X - L_{\mu} - S_{\mu} X + Y^0_1) \right\|_F^2 ; \]
  update the multipliers $\lambda$ and $\mu$ with
  \[ Y^0_1 = Y^0_1 + \mu_\nu (S_{\mu} - W_{\mu}), Y^0_1 = Y^0_1 + \mu_\nu (X - L_{\mu} - S_{\mu} X - E_{\mu}) ; \]
  update the parameter $\mu$ with $\mu_{\nu} = \min(\eta_\mu, \max_\mu)$;
  check for convergence through: supposing $\max \left( \| W_{\mu} - S_{\mu} \|, \| X - L_{\mu} - S_{\mu} X - E_{\mu} \| \right) < \epsilon$, stop;
else $k = k + 1$.
End while

\[ L_{\mu+1} = \arg \min_{L \geq 0} \left\{ \beta \| L \| \right\} \left\{ L - (X - S_{\mu}, Y^0_0, Y^0_1) \right\} \]
\[ W_{\mu+1} = \arg \min_{W \geq 0} \left\{ \beta \| W \| \right\} \left\{ W - (S_{\mu}, Y^0_0, Y^0_1) \right\} \]
\[ S_{\mu+1} = \arg \min_{S \geq 0} \left\{ \beta \| S \| \right\} \left\{ S - (X - L_{\mu} - S_{\mu}, Y^0_1) \right\} \]
\[ E_{\mu+1} = \arg \min_{E \geq 0} \left\{ \beta \| E \| \right\} \left\{ E - (X - L_{\mu} - S_{\mu}, X - E_{\mu}) \right\} \]

where each step corresponds to a convex problem that can be effectively solved. For computational efficiency, the inexact ALM is adopted in this paper. For the competence of presentation, we detail the optimization procedures of applying the inexact ALM in Algorithm 1. The convergence properties of SPLRR are similar as that of [6], [12], [22]. Note that Step 2, Step 3 and Step 5 all have closed-form solutions. Step 2 is solved via the Singular Value Thresholding (SVT) operator [10]. Step 3 is solved by the shrinkage operator [9]. According to [6], [8], the $i$-th column $E^*_{\mu}$ of solution $E_{\mu}$ at the $(k+1)$-iteration in Step 5 can be calculated as
A Sparse Projection and Low-Rank Recovery Framework for Handwriting Representation and Salient Stroke Feature Extraction

39:9

ACM Transactions on xxxxxxxx, Vol. xx, No. xx, Article xx, Publication date: Month YYYY

\[ E_{\iota,1} = \begin{cases} \frac{L_{i,1}}{\mu_{i}} \hat{\phi}_{i}^\top & \text{if } \frac{\lambda}{\mu_{i}} < \frac{1}{\lambda} \\ \frac{1}{\mu_{i}} \hat{\phi}_{i}^\top & \text{otherwise} \end{cases}, \]  

\[ \lvert \lvert E \rvert \rvert_2^2 = \frac{1}{\lambda} \sum_{i=1}^{L} \frac{\lambda}{\mu_{i}} \| \hat{\phi}_{i}^\top \|^2_{2} + \frac{1}{\mu_{i}} \| \hat{\phi}_{i}^\top \|^2_{2}, \]  

(17)

where \( \hat{\phi}^\top = X - L_{i,1} - S_{i,1}X + Y_{i}^* / \mu_{i} \), and \( \hat{\phi}_{i}^\top \) is the \( i \)-th column of matrix \( \hat{\phi}^\top \). The major computation of Algorithm 1 is at Step 2, which requires calculating the SVD of matrices, thus the computational complexity of the proposed SPLRR is the same as that of the inexact ALM based RPCA [9]. Note that calculating \( \mu_{i} \) via \( \text{SVT} \) at the \((k+1)\)-iteration requires to compute the singular vectors of \( X - S_{i}X - X_{i} / \mu_{i} \) according to the singular values that exceed the threshold \((1/\mu_{i})\), so a good initialization of \( \mu_{i} \) is very important [8], [11], [22]. In this paper, we observe that \( \mu_{i} = 2.3 \times 10^3 \) and \( \eta = 1.12 \) are good choices for our SPLRR. It is worth noting that the convergence analysis of the inexact ALM based RPCA has been well established in [9], but when there are more than two terms in the alternating minimizations, such as [6], [12], [22] and our proposed formulation, a rigorous proof of convergence still remains a difficult issue [6], [22], [48]. In our simulations, we can observe that our presented SPLRR algorithm always converges to the optimization problem with the iteration number ranged from 30 to 280.

4. DISCUSSION: CONNECTION, COMPARISON AND EXTENSION

4.1 Relationship Analysis

In this subsection, we discuss some connections by comparing our SPLRR algorithm with other related studies.

(1) **Connection with RPCA.** Our SPLRR formulation is a generalization of RPCA. Clearly, by setting \( \beta = \xi = 0 \), the criterion of SPLRR is reduced to RPCA if the same norm is imposed on \( E \) for modeling the errors.

(2) **Connection with GoDec [5].** The formulation of our SPLRR is a generalization of the GoDec criterion. Since \( X - L - SX \) identifies the sparse errors \( E \) in SPLRR, by imposing the squared Frobenius norm on the sparse error term \( E \) to characterize the random noise [6] and applying similar formulation approach as GoDec, the objective function of our SPLRR technique can be reformulated as the following framework:

\[
\arg \min_{i, S} \{ \|X - L - SX + \xi \rangle \| + \sum_{i} \xi \} \]  

(18)

Clearly, if \( B = X \), the above formulation corresponds to our SPLRR problem. By setting \( B = I \) (i.e., standard basis) and \( \xi = 0 \), the above formulation identifies GoDec, so GoDec is regarded as a special case of our SPLRR.

(3) **Connection with SR [17], Graph Regularized Sparse Coding (GraphSC) [18] and Laplacian Sparse Coding (LSc) [20].** In this case, we consider \( \beta = 1 \), that is SPLRR focuses on sparse coding. By imposing the squared Frobenius norm on the error term \( E \), SPLRR is equivalent to the following sparse coding formulation:

\[
S = \arg \min_{S} \{ \|X - SX \| + \mu \sum_{i} \| \rangle \| + \mu \sum_{i} \xi \rangle \} \]  

(19)
where \( \hat{\sigma}_i = 1 / \lambda, \hat{\gamma}_i = \xi / \lambda \) are tuning parameters, and \( \hat{f}_i(X) = (1 / 2) \sum_i W_{ij}^2 \| x_i - S_{ij} \| + \hat{\sigma}_i \sum_j W_{ij} \sum_l \| \tilde{x}_{ijl} \| + \hat{\sigma}_i f_i(X) \). Motivated by [12], the above problem can be reformulated as the following equivalent one:

\[
S^* = \arg \min \left\{ \sum \| x^* - X^* s \| + \gamma \sum_i \| x_i \| + \lambda \sum_i f_i(X) \right\},
\]

where term \( f_i(X) \) denotes the transpose of \( f_i(X) \). Note that setting the data matrix \( X^* \) itself as the dictionary \( D = [D_1, D_2, \ldots, D_N] \) may not be optimal in sparse coding formulation [18], [20], so one usually alternately learns the dictionary and the sparse codes. Next we reformulate the above problem as

\[
\left( D', S' \right) = \arg \min \left\{ \sum \| x^* - D S \| + \gamma \sum_i \| x_i \| + \lambda \sum_i f_i(X) \right\},
\]

Subj \( \| D \| \leq c, i = 1, 2, \ldots, N \)

where \( c \) is a constant, for instance constant 1. In this case, we can equivalently minimize \( f_i(X) = (1 / 2) \sum_i W_{ij}^2 \| D s_i^* - D S \| = (1 / 2) \sum_i W_{ij}^2 \| D (s_i^* - D S) \| = \text{tr} \left( S D F^T D S \right) \) instead of preserving the similarity or locality of features in the sparse coding process with \( D \) fixed. Also, by applying the \( k \)-neighborhood to find the neighbors of each sample and constructing the weight matrix \( W \) with the simple-minded method or the heat kernel [25], i.e., \( W_{ij} \) are weighted if \( x_i \) belongs to the \( k \)-nearest-neighbors of \( x_j \), the above objective function further becomes

\[
\left( D', S' \right) = \arg \min \left\{ \| x^* - D S \| + \gamma \sum_i \| x_i \| + \lambda \sum_i f_i(X) \right\},
\]

Subj \( \| D \| \leq c, i = 1, 2, \ldots, N \)

Clearly, the above problem is equivalent to the GraphSC formulation when \( D \) is known, especially when \( D = I \). By further imposing a constraint \( \| D \| = 1 \) to avoid the scaling problem of \( D \), the above problem is also identified to the formulation of LSC. Thus, the solution \( S' \) to the above problem can be similarly calculated as GraphSC and LSC. Note that if the regularization term \( f_i(X) \) and the weight matrix \( W \) are defined as above, and each row of the data matrix \( X \) corresponds to a sample point, by setting the dictionary \( D = X^* \), the above formulation is converted to the optimization problem of Eq.19. That is, our SPLRR formulation with \( \beta = 1 \) can be considered as a special case of the GraphSC and LSC formulations. But note that, different from the problems of GraphSC and LSC, our SPLRR algorithm avoids estimating the optimal model parameters, e.g., the \( k \) number and the heat kernel width, for similarity preservation, since this issue is very difficult in reality. By setting \( \hat{\sigma}_i = 0 \) or \( \xi = 0 \), the above formulation can be further transformed to the following sparse coding problem:

\[
\left( D', S' \right) = \arg \min \left\{ \| x^* - D S \| + \gamma \sum_i \| x_i \| \right\}, \text{Subj} \| D \| \leq c, i = 1, 2, \ldots, N \).
\]

Since \( X^* - D S \) identifies the errors, if we similarly consider each row of the data matrix \( X \) as an instance, and define \( D = X^* \) and \( \hat{\sigma}_i = \lambda = 1 \), the above sparse coding formulation is identified to the standard SR problem in Eq.4. In other words, the formulation of our SPLRR with \( \beta = 1 \) and \( \hat{\gamma}_i = 0 \) can also be regarded as a special example of the SR formulation in Eq.4 when imposing the same constraint.
4.2 Supervised Extension of SPLRR (S-SPLRR)

We also consider the supervised extensions of SPLRR for discriminant stroke feature extraction. In the previous sections, we have proposed an unsupervised SPLRR for handwriting representation and stroke feature extraction without considering any supervised information of the handwritings. For classification, utilizing certain supervised knowledge of samples, e.g., class information or pairwise constraints (PC), can boost the performance. Note that our proposed SPLRR algorithm can be easily extended for discriminative learning if class information of data is available. Next, we show two ways. First, we can incorporate the class information of samples into the similarity preserving term \( \hat{f}_s(X) \), which can improve the discriminating power of the projection matrix \( S \) and enhance subsequent performance. More specifically, for a given set of handwritten samples with their class labels known, we aim to encode the similarity among samples from the same class. Let \( l(x_i) \) be the class label of \( x_i \), then the weight matrix \( W^{(c)} \) in \( \hat{f}_s(X) \) is weighted as

\[
W^{(c)}_{i,j} = \begin{cases} 
\exp(\cos(\theta)) \cdot \cos(\theta), & \text{if } l(x_i) = l(x_j) \\
0, & \text{if } l(x_i) \neq l(x_j)
\end{cases}
\]  

(24)

In the above S-SPLRR setting, minimizing \( \hat{f}_s(X) \) is equivalent to pushing samples from the same class close together, thus the representations of features can be improved. But note that high separation between points from different classes should also be guaranteed. In what follows, we introduce another effective approach for implementing S-SPLRR by using the pairwise constraints [45-47]. Denote the must-link (ML) set by \( ml = \{ (x_i, x_j) | x_i, x_j \in X, l(x_i) = l(x_j) \} \) and the cannot-link (CL) set by \( cl = \{ (x_i, x_j) | x_i, x_j \in X, l(x_i) \neq l(x_j) \} \), we define the traces of the scatter matrices over data pairs constrained by ML and CL as follows for S-SPLRR:

\[
\text{Tr}(U_{ml}) = \sum_{(x_i, x_j) \in ml} \| x_i - x_j \| W_{i,j}^{ml} , \quad \text{Tr}(U_{cl}) = \sum_{(x_i, x_j) \in cl} \| x_i - x_j \| W_{i,j}^{cl}
\]

(25)

where the weight matrices \( W^{ml} \) and \( W^{cl} \) are respectively constructed as

\[
W_{i,j}^{ml} = \begin{cases} 
\exp(\cos(\theta)) \cdot \cos(\theta), & \text{if } (x_i, x_j) \in ml \\
-\exp(\cos(\theta)), & \text{if } (x_i, x_j) \notin cl \\
0, & \text{if } (x_i, x_j) \notin ml
\end{cases}, \\
W_{i,j}^{cl} = \begin{cases} 
\exp(\cos(\theta)), & \text{if } (x_i, x_j) \in cl \\
0, & \text{if } (x_i, x_j) \notin cl
\end{cases}
\]  

(26)

where \( \cos(\theta) \) is similarly defined as above. Note that minimizing \( \text{Tr}(U_{ml}) \) and maximizing \( \text{Tr}(U_{cl}) \) at the same time are equivalent to pushing intra-class features close with each other and separating inter-class features at each iteration, so a natural choice for \( \hat{f}_s(X) \) in this supervised S-SPLRR setting is defined as

\[
\hat{f}_s(X) = \text{Tr}(U_{ml}) - \text{Tr}(U_{cl})
\]  

(27)

Note that the above S-SPLRR settings optimize the same problem as SPLRR, and the solutions can be similarly obtained, but S-SPLRR can deliver more discriminant stroke features for boosting subsequent classification.

5. SIMULATION RESULTS AND ANALYSIS

In this section, we test the effectiveness of SPLRR for handwriting representation and recognition, along with illustrating the simulation results.
5.1 Baselines and Simulation Settings

5.1.1 Compared algorithms and parameter settings. Since labeled samples are expensive to obtain, we mainly test our unsupervised SPLRR for stroke recovery and handwriting recognition. For handwriting recognition, we compare SPLRR with 8 unsupervised methods, i.e., Principal Component Analysis (PCA) [23], Neighborhood Preserving Embedding (NPE) [27], Locality Preserving Projections (LPP) [25], Sparse LPP (SLPP), Euclidian Projective Nonnegative Matrix Factorization (PNMF) [24], Sparsity Preserving Projections (SPP) [19], IRPCA and LatLRR. The codes of LPP and NPE are available at http://www.cad.zju.edu.cn/home/dengcai/Data/.

(a) PNMF: Euclidean PNMF approximates a given set of data vectors \(\{x_i\}_{i=1}^N\) by its nonnegative subspace projection, and approximately factorizes a projection matrix \(P\) with given rank \(r\) into a non-negative low-rank matrix \(H \in \mathbb{R}^{r \times n}\) and its transpose by minimizing the following reconstruction error [24]:

\[
H^* = \arg\min_{H} \frac{1}{2} \|X - HH^T X\|_F, \quad \text{Subj} \quad H^T H, H > 0 .
\]  

(28)

After calculating the factor \(H\), a nonnegative projection matrix can be constructed as \(P = HH^T\) for projecting new data and \(PX\) can also be treated as the reconstruction of the original data.

(b) SPP and SLPP: SPP is one representative SR based subspace learning criterion. With \(R^* = [r_1, r_2, \ldots, r_k]\) obtained from Eq.4, SPP seeks a projection matrix \(P \in \mathbb{R}^{d \times r}\) that best preserve the optimal weight vector \(r_i^*\) by minimizing the following objective function:

\[
F^* = \arg\max_{P \in \mathbb{R}^{d \times r}} \text{Tr}\{P^T XX^T P\}, \quad \text{Subj} \quad P^T XX^T P = I ,
\]  

(29)

where \(W_{sr} = R^* + R^{*T} - R^* R^*\). With \(P\) obtained, new data can be embedded by projecting it onto \(P\). The only difference between LPP and SPP is the construction of weight matrix for measuring the locality of data pairs. Similar to [40], this work defines the weight matrix in SLPP as \(\|R^* + R^{*T}\|_2/2\).

(c) Parameter Settings. For fair comparison, the \(l^2\)-norm is regularized on the error term \(E\) of SR, IRPCA, LatLRR and SPLRR. Note that these methods have a common parameter \(\lambda\). According to [6], a relatively large \(\lambda\) is used if errors are slight and otherwise one should tune \(\lambda\) to be relatively small. This work regulates \(\lambda \in \{4 \times 10^7 \cdot i = -4, -3, \ldots, 1\}\). There is a common parameter in LPP and NPE, i.e., the \(k\) number in \(k\)-neighborhood. The heat kernel [25] is used to define the weight matrix for LPP and the kernel width is similarly estimated as [32]. This paper regulates \(k\) from \(\{i \mid i = 5, 7, \ldots, 19\}\). For each algorithm, various parameters are tested and the best results over parameters are reported. The number \(r\) in PNMF is set to the subject number as [24]. SPLRR has three parameters, i.e. \(\beta, \xi, \lambda\). In the simulations, if \(X\) itself is an image, we set \(\xi\) to 0 and apply the idea of “grid-search” [30] on \(\beta\) and \(\lambda\) by repeating experiments. Various pairs of \((\beta, \lambda)\) values are tested and the one according to the best result is picked. Otherwise if \(X\) is a set of data vectors, we fix \(\xi\) and perform the grid-search on \(\beta\) and \(\lambda\). In this case, we choose \(\xi \in \{10^2 \cdot i = -4, -3, \ldots, 2\}\) and pick the pairs of \((\beta, \lambda)\) values according to the best performance. We perform all simulations on a PC with Intel (R) Core (TM) Duo CPU E8400 @ 3.00 GHz 2.99 GHz 1.96G.

---

1 http://users.ics.aalto.fi/rozyang/pnmf/index.shtml
2 http://perception.csl.illinois.edu/matrix-rank/sample_code.html
3 https://sites.google.com/site/guangcanliu/LatLRR
5.1.2 Data preparation. Four handwriting digits databases, i.e., MNIST [34], USPS [35], Optical Recognition of Handwritten Digits (ORHD)\(^4\) and CASIA-HWDB1.1 [44], are evaluated:

(a) The CASIA-HWDB1.1 database consists of 3755 Chinese characters and 171 alphanumeric and symbols, collected from 300 writers. A subset referred to as HWDB1.1-D, including 2381 handwritten digits (‘0’-’9’), from CASIA-HWDB1.1 is also sampled for the evaluations. Since the sizes of the handwritten images in CASIA-HWDB1.1 are inconsistent, we resize all the images to 14×14 pixels.

(b) The MNIST database has a training set of 60000 samples and a testing set of 10000 samples. As a common practice, all images of the set are resized to 16×16 pixels due to the computational consideration.

(c) The USPS database consists of 9298 handwritten digits (‘0’-’9’). In this study, the publicly available set from http://cs.nyu.edu/~roweis/data.html is used. This set includes 16×16 pixels in 8-bit grayscale images of ‘0’-’9’. Each digit has 1100 images and each image is represented by a 256-dimensional vector.

(d) The ORHD database from the UCI machine learning (ML) repository includes 5620 digital samples, where each element is an integer ranged from 0 to 16, from a total of 43 people.

We have shown typical sample images of each database in Figure 1 and described the detailed information of used datasets in Table 1.

5.1.3 Evaluation metrics. For recovery, we visually evaluate the experimental results. For handwriting feature extraction and recognition, K-nearest-neighbor (K-NN) classifier with Euclidean metric is used for quantitatively evaluating the results. For handwriting recognition, each dataset will be randomly split into training and test sets. After calculating the projection matrix from the training set, test data are embedded onto the projection directions. Finally, the projected samples are identified by a K-NN classifier and the averaged performance is reported. PCA is employed to eliminate the null space of the training sets before feature extraction.

\[\text{Fig.1. Typical sample images of (a) MNIST, (b) USPS, (c) ORHD, and (d) HWDB1.1-D.}\]

<table>
<thead>
<tr>
<th>Dataset Name</th>
<th># Class</th>
<th># Num.</th>
<th># Dim.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>C=10</td>
<td>70000</td>
<td>28×28</td>
</tr>
<tr>
<td>USPS</td>
<td>C=10</td>
<td>9298</td>
<td>16×16</td>
</tr>
<tr>
<td>ORHD</td>
<td>C=10</td>
<td>5620</td>
<td>8×8</td>
</tr>
<tr>
<td>HWDB1.1-D</td>
<td>C=10</td>
<td>2381</td>
<td>14×14</td>
</tr>
</tbody>
</table>

\(^4\) http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits

ACM Transactions on xxxxxxxx, Vol. xx, No. xx, Article xx, Publication date: Month YYYY
5.2 Handwriting Recovery

This study visualizes the performance of SPLRR for document representation, including data recovery and feature extraction. Two simulations over different vision data are conducted. For a given data matrix $X$ corrupted by some errors or noise, SPLRR handles it by recovering the low-rank principal components $L^*$, extracting the saliency features $S^*X$, and delivering a sparse part $E^*$ fitting errors. We first examine the effectiveness of SPLRR for recovering the background from heavy pixel corruptions, including the handwritten digits, Chinese and English characters. The corruptions in $X$ are denoted by using red rectangles, and the recovery results are illustrated in Figure 2(a). As observed from Figure 2(a), SPLRR delivers accurate recoveries, i.e., it effectively removes the corruptions and recovers the background. But conversely, the corruptions included in the background may be regarded as salient discriminative information for identifying different backgrounds in the tasks of classification. Note that our developed SPLRR also provides a solution to this issue. More specifically, based on the sparse projection $S^*$, the visually salient discriminant features can be efficiently extracted by SPLRR. Moreover, the sparse projection matrix $S^*$ can be used to represent new test samples in classification.

Fig. 2. Recovery by SPLRR on: (a) Background corrupted by documents, (b) MNIST, and (c) HWDB1.1-D.

We next address another simulation to examine SPLRR for processing handwriting digits with certain missing values or errors. This task is to evaluate SPLRR for recovering the digits and extracting the salient stroke features at the same time. In this simulation, the MNIST and HWDB1.1-D databases are tested and we aim at visualizing 100 images of the handwritten digit '0' of each dataset. Note that the matrix composed by transforming the images as column vectors is naturally of low-ranked. The results are visualized in Figures 2(b) and 2(c), where we denote the digits with missing pixel values in the original matrix using the rectangles. We observe from the results that our SPLRR successfully recovers the missing parts and removes the noise, along with delivering saliency features.
5.3 Handwritten Stroke Correction

In this simulation, we mainly test SPLRR for isolated handwriting (e.g., handwritten digits or characters) stroke correction by low-rank recovery. It is worth noting that handwritten stroke correction is vital and challenging in isolated handwriting image analysis because of the variability of the writing styles of different persons. That is, the stroke shapes of the handwritten digits or characters are usually irregular and un-uniformly curved [26][28]. Thus, those originally similar digits or characters, e.g., (digits ‘1’ vs. ‘7’) and (letters ‘i’ vs. ‘j’) from the CASIA-HWDB1.1 database, are easy to be misclassified. We have shown some irregular examples in Figure 3.

Fig. 3. Examples of irregular, similar handwritten digits and characters.

\[ X = L^* + S^*Y + E^* \]

Fig. 4. Handwritten stroke correction results of SPLRR on: (a) USPS; (b) MNIST; (c) HWDB1.1-D.

This study is mainly prepared to test SPLRR for correcting the incorrect or irregular strokes. In this study, we aim at visualizing the stroke correction results of 100 images of handwritten digit ‘7’ from the MNIST, USPS and HWDB1.1-D databases respectively, in Figure 4. In all our simulations, the publically available USPS dataset at http://cs.nyu.edu/~roweis/data.html is tested. This sample set has 16×16 pixels in 8-bit grayscale images of ‘0’-‘9’. Each digit, denoted by a 256-dimensional vector, has 1100 images. We also show some examples of the irregular or
incorrect strokes using the rectangular boxes in the first column of Figure 4. Observing from the results, we find clearly that the incorrect or irregular strokes of the handwritings from each database are all accurately corrected and recovered by our SPLRR algorithm, as illustrated by $I'$. But once again, one must notice that these irregular strokes may also be useful in some specific applications, e.g., personal handwriting identification [41], [42], since it can precisely highlight the individuality of the writing styles of a person, which is a factor that is as important as successfully repairing the irregular or even incorrect strokes of the handwritings. In other words, effectively extracting the salient stroke features of the handwritings and performing stroke correction is equally important. As observed from Figure 4, applying our SPLRR method can enhance the visual saliency of the writing strokes of different persons in addition to recover the irregular strokes and remove the sparse errors.

5.4 Handwritten Digits Recognition

5.4.1 Arrangements of experiments. In this study, we test SPLRR for handwritten digits recognition on four real databases, i.e., MNIST, HWDB1.1-D, ORHD and USPS. Considering that samples from the same subject are usually compactly distributed exhibiting similar structures, for MNIST dataset we randomly choose 2000 samples from the training set and 2000 samples from the test set to form our sample set, in which each digit has 400 samples. For USPS, we randomly select 4000 samples from each digit to form experiments, in which each digit has 400 samples. For each database, we normalize the pixel values of the handwritten images to [0, 1]. Two experimental configurations are tested and the 1NN classifier, i.e., $K=1$ is used. The first one is to compare the performance of SPLRR with IRPCA, LatLRR and PNMF, as given in subsection 5.4.2. In this study, we use the projection matrix (i.e., $S$, $Q$, $P$ and $P$ respectively) of SPLRR, IRPCA, LatLRR and PNMF learnt from training samples for embedding the test data for classification by projecting the test data onto the projection directions. Considering that the dimension of the extracted features by SPLRR is the same as that of the original samples, we in subsection 5.4.3 consider SPLRR for unsupervised subspace learning [25], called SPLRR-SL. Given a set of handwriting samples, resembling [2], SPLRR-SL considers the salient stroke features and low-rank principal components simultaneously, and calculates an orthogonal projection matrix $P\in\mathbb{R}^{n\times d}$ ($d\leq n$) onto which dissimilar handwriting features can be maximally separated, from the following objective function:

$$
\tilde{P} = \arg\max_{P\succeq 0} \frac{1}{2} \sum_{i,j} |\tilde{x}_i - \tilde{P}^T \tilde{x}_i - \tilde{x}_i |^2 + \frac{1}{2} \sum_{i,j} \left( P^T L - \tilde{P}^T L \right) \left( \tilde{w}^{(i)}_{i,j} \right), \
$$

with $\tilde{x}$ and $L$ being $S\tilde{x}$, and the $i$-th column of low-rank $L'$ respectively, where $\tilde{w}^{(i)}_{i,j} = \max \left( w^{(i)}_{i,j} \right) - w^{(i)}_{i,j}$ and $\max \left( w^{(i)} \right)$ denote the maximal value among weights $w^{(i)}_{i,j}$, i.e., $c_i$. Let $\hat{D}$ denote a diagonal matrix with entries $\hat{D}_{ii} = \sum_{j=1}^{n} \tilde{w}^{(i)}_{i,j}$, based on using the matrix expressions, the above problem becomes

$$
\tilde{P} = \arg\max_{P\succeq 0} \text{Tr} \left( P^T \left( L F^{(i)} L^T + \hat{X} F^{(i)} \hat{X}^T \right) P \right), \text{Subj} \tilde{P}^T \tilde{P} = I, 
$$

where $\hat{X} = S'X$, from which the optimal $\tilde{P}$ can be obtained as eigenvectors according to $d$ leading eigenvalues of the eigenvalue problem: $\left( L F^{(i)} L^T + S'X F^{(i)} \hat{X}^T \right) \tilde{\omega}_j = \hat{X} \tilde{\omega}_j$. With the projection matrix $\tilde{P}$ learnt from the training samples, the dimensionality of test data can be reduced to $d$ by projecting them onto $\tilde{P}$. In this study, the
performance of our SPLRR-SL algorithm is mainly compared with those of PCA, LPP, NPE, SLPP and SPP.

5.4.2 Handwriting recognition by feature representation. We test the projection matrix of SPLRR, IRPCA, LatLRR and PNMF for handwriting representation. First, we explore the effects of the parameters $\beta$, $\xi$ and $\lambda$ on the recognition performance of our SPLRR. We only report the parameter analysis results on the HWDB1.1-D and USPS databases due to page limitation. For each set, four settings are tested. Figures 5 and 6 show the results of SPLRR on HWDB1.1-D and USPS respectively, where the training numbers per digit are also given. We firstly vary the pairs of $(\beta, \lambda)$ values with the values of $\xi$ fixed, where $\beta \in \{0.005, 0.01, 0.05, 0.1, 0.3, 0.5, 0.7, 0.9\}$, and experimentally find the best choices of the pairs of $(\beta, \lambda)$. To avoid the bias caused by the splits of training/test samples, for each fixed pair of $(\beta, \lambda)$, we ran our SPLRR 15 times with different splits and show the averaged accuracies. Observing from the results for this setting on HWDB1.1-D, MNIST and ORHD, we find our SPLRR with pairs of $(0.01, 2 \times 10^{-2})$ and $(0.005, 2 \times 10^{-2})$ can deliver the highest accuracies in most cases under parameters $\xi$. We also note that the performance of SPLRR with pair of $(0.05, 2 \times 10^{-2})$ is better than with other combinations on USPS. Figure 5(a) and Figure 6(a) illustrate the results of our SPLRR with $10^{-1}$ on HWDB1.1-D and USPS as examples. In the other three settings, we alternately fix two of the three parameters and test another one. For each setting, we fix the training numbers from each digit and report the averaged results over 15 splits. For different numbers of training samples from each database, we can have the following observations. (1) When testing $\xi$ by fixing $\beta$ and $\lambda = 4 \times 10^{-2}$, our SPLRR always delivers satisfactory results when $\xi \geq 10^{-2}$. More specifically, our proposed SPLRR with $\xi = 10^{-1}$ delivers the best results in most cases. (2) When testing $\beta$ by setting $\xi = 10^{-1}$ and $\lambda = 4 \times 10^{-2}$, we similarly find our SPLRR algorithm with $\beta = 0.005$ and $\beta = 0.01$ can deliver comparable highest records in most cases on the HWDB1.1-D, ORHD and MNIST databases, and our SPLRR with $\beta = 0.05$ achieves the satisfactory results on USPS in most cases. (3) When testing $\lambda$ by fixing the values of $\xi$ and $\beta$, the results delivered by our SPLRR initially go up when the values of $\lambda$ increase, and then start to decrease at $\lambda = 4 \times 10^{-2}$ in most cases, i.e., $\lambda = 4 \times 10^{-2}$ is a good choice for our SPLRR on each dataset in most cases, especially when the training number is relatively small. When we increase the training number to a larger one, our SPLRR, with less $\lambda$ values than $4 \times 10^{-2}$, delivers comparable results to that of SPLRR with $\lambda = 4 \times 10^{-2}$ on each set. This is because increasing the numbers of training samples per digit can increase the noise level of the training set to some extent. It is also noticed that increasing the values of $\lambda$ may cause the results of the embeddings of our SPLRR algorithm to deteriorate. Note that the selection of $\lambda$ depends on the error level of the datasets [6], so the above results can provide a guideline to select the $\lambda$ values for SR, IRPCA and LatLRR. In the simulations below, we set $\lambda \in \{4 \times 10^{-1} \mid i = -3, -2, -1\}$ for SR, IRPCA and LatLRR, and illustrate the best records. In conclusion, our proposed SPLRR algorithm with $\xi = 10^{-1}$ and $\lambda = 4 \times 10^{-2}$ is used in the following handwriting recognition. The value of $\beta$ is set to 0.01 for HWDB1.1-D, ORHD and MNIST, and 0.05 for USPS, respectively.
Fig. 5. Parameter analysis of $\beta$, $\xi$, and $\lambda$ for our SPLRR on HWDB1.1-D database, where: (a) Fix $\xi$, vary the pairs of ($\beta$, $\lambda$); (b) Fix $\beta$ and $\lambda$, vary $\xi$; (c) Fix $\lambda$ and $\xi$, vary $\beta$; (d) Fix $\beta$ and $\xi$, vary $\lambda$.

Fig. 6. Parameter analysis of $\beta$, $\xi$, and $\lambda$ for our SPLRR on USPS database, where: (a) Fix $\xi$, vary the pairs of ($\beta$, $\lambda$); (b) Fix $\beta$ and $\lambda$, vary $\xi$; (c) Fix $\lambda$ and $\xi$, vary $\beta$; (d) Fix $\beta$ and $\xi$, vary $\lambda$.
We then compare SPLRR with IRPCA, LatLRR and PNMF for handwritten digits recognition. We examine each method by varying the number of training samples per digit from 20 to 160 with step 20. For each training number, the recognition results are averaged over 15 runs. The results on the four datasets are plotted in Figure 7. The mean and best results according to Figure 7 are summarized in Table 2. We have the following observations. (1) The overall performance of each method goes up as the number training samples is increased. (2) Our SPLRR algorithm delivers comparable or even high accuracy than its competitors across training numbers in most cases. The advantages of our SPLRR mainly lie in its ability to automatically extract the salient stroke features from the training data. (3) For HWDB1.1-D, PNMF initially performs better than IRPCA and LatLRR, and then starts to decrease if the training number passed 100, and finally continues to go up. IRPCA performs better than LatLRR in all cases. (4) For MNIST, IRPCA and LatLRR are highly competitive with our SPLRR by delivering comparable results in most cases, that is, the advantage of SPLRR to IRPCA and LatLRR is not obvious for this set. It is also observed that PNMF is inferior to IRPCA, LatLRR and our SPLRR method in most cases. (5) For USPS, IRPCA, LatLRR and SPLRR outperform PNMF for recognition on this dataset. Among the first three winners, our SPLRR method ranks first, followed by LatLRR and IRPCA respectively. (6) For ORHD, the figures of results can be divided into two groups. The first group includes PNMF and our SPLRR, and the second one consists of IRPCA and LatLRR. From the results, the first group can achieve higher accuracy by using relatively smaller number of training samples than another group. When the training number is increased to 15, the gap between the two groups is reduced. (7) Similar conclusions can be obtained from the results in Table 2.

![Figure 7](image_url)
5.4.3 Handwriting recognition by subspace learning. In this study, we compare SPLRR-SL with PCA, LPP, NPE, SLPP and SPP for unsupervised subspace learning and apply the reduced features for recognition. The simulation setting over each database is shown as follows. We fix the training numbers (i.e. 40, 80 and 120) for HWDB1.1-D, (i.e. 60, 90 and 120) for MNIST, (i.e. 60, 90 and 120) for USPS and (i.e. 12, 15 and 18) for ORHD. For each case based on the HWDB1.1-D, MNIST, and USPS databases, we fix the training number of each digit and vary the number of reduced dimensions from 5 to 80 with step 5. For ORHD, we aim to vary the reduced dimensions from 5 to 50 with step 5 in each case. For each setting of simulations, the results averaged over 15 runs are described in Tables 3-6, in which the highest accuracies and the standard deviation (STD) are also list. We see from Tables 3-6 that: (1) The performance of each method can be boosted by the increasing number of training samples over each database. Our SPLRR-SL algorithm is superior to other related methods over each database in most cases. (2) PCA, NPE, SLPP and SPP are always competitive with each other and their results are better than LPP in most cases. (3) For MNIST, we observe from the results that SLPP and SPP are comparative with our SPLRR method for subspace learning in most cases, and their accuracies are higher than other methods in each case. PCA is superior to LPP and NPE for this simulation. (4) For USPS, SPP and SLPP outperform PCA, LPP and NPE in most cases, and they always deliver comparable results to our SPLRR-SL algorithm. (5) For ORHD, SPP obtains comparable results to our SPLRR-SL in each case, and both are superior to the other methods in most cases. PCA delivers close results to SLPP, and both are better than the remaining methods.

Table 2. Performance comparison of IRPCA, LatLRR, PNMF and SPLRR on each database.

<table>
<thead>
<tr>
<th>Result</th>
<th>HWDB1.1-D</th>
<th>MNIST digits</th>
<th>USPS digits</th>
<th>ORHD digits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Mean</td>
<td>Best</td>
<td>Mean</td>
<td>Best</td>
</tr>
<tr>
<td>IRPCA</td>
<td>87.95%</td>
<td>91.06%</td>
<td>87.54%</td>
<td>90.76%</td>
</tr>
<tr>
<td>LatLRR</td>
<td>87.56%</td>
<td>90.99%</td>
<td>87.8%</td>
<td>91.05%</td>
</tr>
<tr>
<td>PNMF</td>
<td>87.99%</td>
<td>90.50%</td>
<td>86.45%</td>
<td>90.04%</td>
</tr>
<tr>
<td>SPLRR</td>
<td>89.38%</td>
<td>91.70%</td>
<td>88.27%</td>
<td>91.24%</td>
</tr>
</tbody>
</table>

Table 3. Performance comparison of subspace learning algorithms on the HWDB1.1-D database.

<table>
<thead>
<tr>
<th>Result</th>
<th>HWDB1.1-D (40 train)</th>
<th>HWDB1.1-D (80 train)</th>
<th>HWDB1.1-D (120 train)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Mean</td>
<td>STD</td>
<td>Best</td>
</tr>
<tr>
<td>PCA</td>
<td>83.38%</td>
<td>0.0251</td>
<td>85.56%</td>
</tr>
<tr>
<td>LPP</td>
<td>79.69%</td>
<td>0.0492</td>
<td>82.87%</td>
</tr>
<tr>
<td>NPE</td>
<td>83.02%</td>
<td>0.0448</td>
<td>85.42%</td>
</tr>
<tr>
<td>SLPP</td>
<td>83.52%</td>
<td>0.0269</td>
<td>86.58%</td>
</tr>
<tr>
<td>SPP</td>
<td>83.99%</td>
<td>0.0583</td>
<td>86.91%</td>
</tr>
<tr>
<td>SPLRR-SL</td>
<td>87.20%</td>
<td>0.0354</td>
<td>89.06%</td>
</tr>
</tbody>
</table>

Table 4. Performance comparison of subspace learning algorithms on the MNIST database.

<table>
<thead>
<tr>
<th>Result</th>
<th>MNIST (60 train/digit)</th>
<th>MNIST (90 train/digit)</th>
<th>MNIST (120 train/digit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Mean</td>
<td>STD</td>
<td>Best</td>
</tr>
<tr>
<td>PCA</td>
<td>78.18%</td>
<td>0.0659</td>
<td>80.73%</td>
</tr>
<tr>
<td>LPP</td>
<td>69.81%</td>
<td>0.0532</td>
<td>72.53%</td>
</tr>
<tr>
<td>NPE</td>
<td>73.42%</td>
<td>0.0665</td>
<td>76.79%</td>
</tr>
<tr>
<td>SLPP</td>
<td>80.72%</td>
<td>0.0829</td>
<td>84.24%</td>
</tr>
<tr>
<td>SPP</td>
<td>80.95%</td>
<td>0.0854</td>
<td>85.00%</td>
</tr>
<tr>
<td>SPLRR-SL</td>
<td>82.93%</td>
<td>0.0775</td>
<td>85.65%</td>
</tr>
</tbody>
</table>
5.5 Handwritten Digital Binary Classification

We also address a binary classification problem to examine the recognition ability of SPLRR. This task is prepared to separate the odd numbers (i.e., positive class, including ‘1’, ‘3’, ‘5’, ‘7’ and ‘9’) from even numbers (i.e., negative class, including ‘0’, ‘2’, ‘4’, ‘6’ and ‘8’). We show the specifications of the sampled binary datasets in Table 7. We mainly evaluate the projection matrices of IRPCA, LatLRR, PNMF and our SPLRR for binary classification on MNIST, USPS, ORHD and HWDB1.1-D. For each set, fixed number of training samples is selected for learning the projection matrices, and the rest are treated as testing samples. In this simulation, the ROC graph [38], [39] is applied for measuring the classification performance of each algorithm.

Note that in binary classification, the outcomes of a classifier for test data are either positive (Pos.) or negative (Neg.). If a positive test sample is classified as positive, it is counted as a true positive (TP); otherwise it will be counted as a false negative (FN). Conversely, if a negative sample is classified as negative, it is counted as a true negative (TN); otherwise it is counted as a false positive (FP). Then we can have the following evaluation metrics, including Sensitivity or True Positive Rate (TPR), Specificity or True Negative Rate (TNR), Precision or Positive Predictive Value (PPV), Negative Predictive Value (NPV), Accuracy, and F-score [38], [39]:

\[
TPR = \frac{TP}{(TP + FN)}, \quad PPV = \frac{TP}{(TP + FP)} \\
TNR = \frac{TN}{(TN + FP)}, \quad NPV = \frac{TN}{(TN + FN)}.
\]

\[
Accuracy = \frac{(TP + TN)}{N}, \quad F-score = TPR \times PPV
\] (32)

Denote by Tcl and Pcl to be the true class labels provided by the data corpus and the predicted labels from a classifier respectively. By comparing Tcl with Pcl, the above metrics can be calculated. In this study, a KNN classifier with K=13 is used. For each test data, classification is performed by finding the K nearest samples in the training set. To generate the ROC graph, the outputs of the classifier include a
predicted class label and a score that is calculated dividing by \( K \) the number of positive samples included in the \( K \) nearest training samples. Based on the resulting scores and \( T_c l \), the ROC graphs can be illustrated. In our simulations, for each training number, we average the results over 15 random splits of training and test samples. The result of each algorithm combined with a NN classifier is summarized in Table 8, in which the training number per class is also described. From Table 8, we find that the above evaluation metrics, including sensitivity (TPR), specificity (TNR), precision, accuracy and F-score, etc., delivered by our SPLPP are comparable or even better than other methods in most cases.

<table>
<thead>
<tr>
<th>Dataset Name</th>
<th># Total</th>
<th>Class</th>
<th># Num. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HWDB1.1-D</td>
<td>2381 samples</td>
<td>Neg.</td>
<td>1188 (49.90)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pos.</td>
<td>1193 (50.10)</td>
</tr>
<tr>
<td>MNIST</td>
<td>4000 samples</td>
<td>Neg.</td>
<td>1956 (48.90)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pos.</td>
<td>2044 (51.10)</td>
</tr>
<tr>
<td>USPS</td>
<td>4000 samples</td>
<td>Neg.</td>
<td>2000 (50.00)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pos.</td>
<td>2000 (50.00)</td>
</tr>
<tr>
<td>ORHD</td>
<td>5620 samples</td>
<td>Neg.</td>
<td>2791 (49.66)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pos.</td>
<td>2829 (50.34)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Result Method</th>
<th>HWDB1.1-D (training num.: 100, test num.: remains)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPR</td>
<td>TNR</td>
</tr>
<tr>
<td>IRPCA</td>
<td>0.9035</td>
</tr>
<tr>
<td>LatLRR</td>
<td>0.8871</td>
</tr>
<tr>
<td>PNMF</td>
<td>0.8809</td>
</tr>
<tr>
<td>SPLRR</td>
<td>0.9447</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Result Method</th>
<th>USPS (training num.: 100, testing num.: remains)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPR</td>
<td>TNR</td>
</tr>
<tr>
<td>IRPCA</td>
<td>0.8716</td>
</tr>
<tr>
<td>LatLRR</td>
<td>0.8654</td>
</tr>
<tr>
<td>PNMF</td>
<td>0.8819</td>
</tr>
<tr>
<td>SPLRR</td>
<td>0.8919</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Result Method</th>
<th>MNIST (training num.: 200, testing num.: remains)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPR</td>
<td>TNR</td>
</tr>
<tr>
<td>IRPCA</td>
<td>0.8877</td>
</tr>
<tr>
<td>LatLRR</td>
<td>0.8868</td>
</tr>
<tr>
<td>PNMF</td>
<td>0.8666</td>
</tr>
<tr>
<td>SPLRR</td>
<td>0.8933</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Result Method</th>
<th>ORHD (training num.: 90, testing num.: remains)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPR</td>
<td>TNR</td>
</tr>
<tr>
<td>IRPCA</td>
<td>0.9552</td>
</tr>
<tr>
<td>LatLRR</td>
<td>0.9557</td>
</tr>
<tr>
<td>PNMF</td>
<td>0.9466</td>
</tr>
<tr>
<td>SPLRR</td>
<td>0.9414</td>
</tr>
</tbody>
</table>

We illustrate the ROC curves and ROC convex hulls [38], [39] of the algorithms on each dataset in Figure 8 and Figure 9, respectively. ROC convex hull is formed by the smallest convex set of points in the ROC curves [49]. The threshold is varied between the minimum and maximum values of the classifier scores. The best accuracies of each algorithm over various thresholds are also exhibited in Figure 8. To quantitate
the classification performance, the averaged area under ROC curve (AUC) and area under the ROC convex hull (AUCH) over splits are described. As the ROC graph describes the trade-offs between TP and FP, the nearer to the upper left corner of the diagram, the better the performance (i.e., higher TPR and/or lower FPR are desired). For better presenting the comparison results, the horizontal-axis-scale command of the ROC graphs is set to log scale in MATLAB figures. Observing from Figure 8 and Figure 9, we see that: (1) Our SPLRR technique works better than other methods in most cases, including the highest overall accuracies, since the ROC curves and the ROC convex hulls delivered by our SPLRR are nearer to the upper left corner of the ROC diagrams. Note that IRPCA, LatLRR and PNMF obtain comparable results to our SPLRR on each dataset; (2) It is also observed from the values of AUC and AUCH that our SPLRR with a NN classifier maximizes the AUC and AUCH in most cases, compared with other combinations.

![Fig.8](image.png)  
Fig.8. The ROC curves of IRPCA, LatLRR, PNMF and our SPLRR algorithm on each database.
6. CONCLUDING REMARKS

In this paper, we have presented a new Sparse Projection and Low-Rank Recovery (SPLRR) framework. SPLRR is a generalization of recent RPCA by incorporating a similarity-preserving salient feature extraction term. SPLRR clearly considers the sparsity and low-rank properties of high-dimensional handwriting features, and is originally proposed for handwriting representation and recognition by considering the properties of the handwriting strokes. SPLRR can perform handwriting repairing, deliver a sparse projection for extracting salient stroke features and at the same time detect errors. For recovery, SPLRR can effectively handle the corruptions, repair the missing pixel values and correct the irregular handwriting strokes. For feature extraction, SPLRR is capable of automatically extracting salient stroke features from training images, which can best distinguish handwritings for classification. For handwriting digits recognition, we experimentally observe that the sparse projection delivered by our SPLRR is as much powerful as the low-rank projection of the other existing criteria for feature extraction and recognition. We have also considered SPLRR for reducing the dimensionality of data.

Results on benchmark problems verified the validity of our SPLRR for processing handwriting digits, compared with several state-of-the-art methods. Thus it is interesting in our future work to test the performance of SPLRR in representing and recognizing Chinese or English handwriting characters. Also, the developed SPLRR method can effectively enhance the visual saliency of handwriting strokes and repair...
the irregular strokes or missing values, so extending SPLRR to other application areas, e.g., writer identification, is another interesting future direction.

ACKNOWLEDGMENTS

The authors would like to express our sincere thanks to the anonymous reviewers’ comments and suggestions which have made the paper a higher standard.

REFERENCES


Received May 2013; revised January 2014; accepted February 2014

ACM Transactions on xxxxxxxx, Vol. xx, No. x, Article x, Publication date: Month YYYY