Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Sparse Subspace Clustering by Learning Approximation ℓ_0 Codes

Jun Li,[1] **Yu Kong,**[1] **Yun Fu**[1,2]
[1]Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, 02115, USA.
[2]College of Computer and Information Science, Northeastern University, Boston, MA, 02115, USA.
junl.mldl@gmail.com, {yukong,yunfu}@ece.neu.edu

Abstract

Subspace clustering has been widely applied to detect meaningful clusters in high-dimensional data spaces. A main challenge in subspace clustering is to quickly calculate a "good" affinity matrix. ℓ_0, ℓ_1, ℓ_2 or nuclear norm regularization is used to construct the affinity matrix in many subspace clustering methods because of their theoretical guarantees and empirical success. However, they suffer from the following problems: (1) ℓ_2 and nuclear norm regularization require very strong assumptions to guarantee a subspace-preserving affinity; (2) although ℓ_1 regularization can be guaranteed to give a subspace-preserving affinity under certain conditions, it needs more time to solve a large-scale convex optimization problem; (3) ℓ_0 regularization can yield a tradeoff between computationally efficient and subspace-preserving affinity by using the orthogonal matching pursuit (OMP) algorithm, but this still takes more time to search the solution in OMP when the number of data points is large. In order to overcome these problems, we first propose a learned OMP (LOMP) algorithm to learn a single hidden neural network (SHNN) to fast approximate the ℓ_0 code. We then exploit a sparse subspace clustering method based on ℓ_0 code which is fast computed by SHNN. Two sufficient conditions are presented to guarantee that our method can give a subspace-preserving affinity. Experiments on handwritten digit and face clustering show that our method not only quickly computes the ℓ_0 code, but also outperforms the relevant subspace clustering methods in clustering results. In particular, our method achieves the state-of-the-art clustering accuracy (94.32%) on MNIST.

Introduction

Subspace clustering (Vidal 2011) is an important problem to deal with high-dimensional data clustering in many real-world clustering applications (Liu and Fu 2015; Liu et al. 2015; 2016; Zhao and Fu 2015), such as motion segmentation (Rao et al. 2010; Elhamifar and Vidal 2013), hand written digit clustering (You, Robinson, and Vidal 2016), and face clustering (Elhamifar and Vidal 2013). The popular methods (You, Robinson, and Vidal 2016; Elhamifar and Vidal 2013; Lu et al. 2012; Liu et al. 2013) are to apply spectral clustering to an affinity matrix obtained by solving an optimization problem with ℓ_0, ℓ_1, ℓ_2 or nuclear norm regularization because of their theoretical guarantees and empirical success.

A main challenge in applying spectral clustering to subspace clustering is to fast obtain a "good" affinity matrix, which has a subspace preserving property so that there are no connections between points from different subspaces. Most of existing methods, however, have taken expensive time to calculate the affinity matrix, such as sparse subspace clustering (SSC) (Elhamifar and Vidal 2013), least squares regression (LSR) (Lu et al. 2012; Shao et al. 2015; Li, Li, and Fu 2015) and low rank representation (LRR) (Liu et al. 2013; Li et al. 2016a; Zhao, Ding, and Fu 2016). SSC, LSR, and LRR use ℓ_1, ℓ_2 and nuclear norms to encourage the affinity matrix to be sparse or low-rank by costing more than $\mathcal{O}(n^2)$, where n is the number of data points. SSC can give a subspace-preserving affinity under broad conditions (e.g., arbitrary subspaces and corrupted data) (Elhamifar and Vidal 2013), while LRR and LSR must satisfy the extreme conditions that subspaces are independent and the data is uncorrupted (You, Robinson, and Vidal 2016). Unfortunately, SSC based on ℓ_1-norm still requires solving a large-scale convex optimization problem. To balance the subspace-preserving affinity with the computationally efficient, SSC based on ℓ_0-norm is solved by orthogonal matching pursuit (OMP) (Dyer, Sankaranarayanan, and Baraniuk 2013; You, Robinson, and Vidal 2016), called SSC-OMP. Although SSC-OMP achieves faster time and better accuracy than the original SSC (You, Robinson, and Vidal 2016), it still takes more time to search the solution in OMP when the number of data points is large.

On the other hand, ℓ_1 sparse coding in original SSC is solved by the iterative shrinkage and thresholding algorithm (ISTA) (Beck and Teboulle 2009; Elhamifar and Vidal 2013), which is referred to as SSC-ISTA in this paper. However, it leads to a expensive inference problem. To overcome this problem, training a feed-forward neural network (Li et al. 2016b) is used to produce the best approximation of the ℓ_1 sparse code (Kavukcuoglu, Ranzato, and LeCun 2008; Gregor and LeCun 2010). Learned ISTA (LISTA) is used to train the feed-forward neural network (Gregor and LeCun 2010). Given a test example, ℓ_1 code is quickly computed by the feed-forward neural network instead of ISTA.

In light of the above arguments, learning a non-linear, feed-forward neural network is an efficient method to fast infer the ℓ_0 sparse coding for the affinity matrix in sparse subspace clustering as a single hidden neural network (SHNN) with

enough hidden units can adequately fit any continuous function (Ripley 1996). The **main contributions** of this paper are as follow:

1. We first propose a learned OMP algorithm (LOMP) to train a SHNN model for approximating the ℓ_0 sparse code. We then exploit an SSC method based on ℓ_0 sparse coding which is fast computed by SHNN. This method is called SSC-LOMP.
2. Two sufficient conditions on SSC-LOMP are presented to ensure the subspace-preserving property for independent and arbitrary subspaces.
3. SSC-LOMP has faster test inference time than SSC-ISTA and SSC-OMP as the affinity matrix is quickly computed by SHNN. Compared to SSC based on LISTA (SSC-LISTA), SSC-LOMP has a fast training time because SSC-OMP is faster than SSC-ISTA.
4. SSC-LOMP outperforms the relevant subspace clustering methods in clustering results on MNIST and Extended YaleB datasets. In particular, our method achieves the state-of-the-art clustering accuracy (94.32%) on MNIST dataset.

Notation. A (data) matrix is denoted as $\mathbf{Y} = [\mathbf{y}_1, \cdots, \mathbf{y}_i, \cdots, \mathbf{y}_n] \in \mathbb{R}^{d\times n}$, where d is the dimension of a sample and n is the number of samples. A vector is denoted as $\mathbf{y}_i = [\mathbf{y}_{i1}, \cdots, \mathbf{y}_{ij}, \cdots, \mathbf{y}_{id}] \in \mathbb{R}^d$, where $\mathbf{y}_{ij}$ is the j-th dimension of i-th samples. We denote ℓ_0-norm, ℓ_1-norm, ℓ_2-norm and square of ℓ_2-norm as $\|\mathbf{y}_i\|_0 = \{\#|\mathbf{y}_{ij} \neq 0\}$, $\|\mathbf{y}_i\|_1 = \sum_j |\mathbf{y}_{ij}|$, $\|\mathbf{y}_i\|_2 = \sqrt{\sum_j \mathbf{y}_{ij}^2}$ and $\|\mathbf{y}_i\|_2^2 = \sum_j \mathbf{y}_{ij}^2$.

Related Works

In this section, we briefly review the related SSC methods.

SSC-OMP. (You, Robinson, and Vidal 2016) One way is to seek sparse solutions by ℓ_0 norm. In principle, we can find the sparsest coefficient vector by solving the following optimization problem:

$$(P_0) \quad \mathbf{x}_j^* = \arg\min_{\mathbf{x}_j} \|\mathbf{x}_j\|_0 \; s.t. \; \mathbf{y}_j = \mathbf{Y}\mathbf{x}_j, \; \mathbf{x}_{jj} = 0, \quad (1)$$

where $\mathbf{Y} \in \mathbb{R}^{d\times n}$ is the data matrix and $\mathbf{x}_j \in \mathbb{R}^n$ is the sparse representation coefficient of jth sample. The problem (P_0) is NP-hard. Usually, solutions to this problem can be found by orthogonal matching pursuit (OMP) (Mallat and Zhang 1993; Dyer, Sankaranarayanan, and Baraniuk 2013). However, it takes more time to search the solution in OMP when the number of data points is large.

SSC-ISTA. (Elhamifar and Vidal 2013) Another way to highly sparse solutions can be obtained by convex optimization. By replacing the ℓ_0 norm in (P_0) by the ℓ_1-norm, the problem can be written as the following problem:

$$(P_1) \quad \mathbf{x}_j^* = \arg\min_{\mathbf{x}_j} \|\mathbf{x}_j\|_1 \; s.t. \; \mathbf{y}_j = \mathbf{Y}\mathbf{x}_j, \; \mathbf{x}_{jj} = 0. \quad (2)$$

This problem is generally reformulated as an unconstrained problem. Iterative Shrinkage and Thresholding Algorithm (ISTA) (Beck and Teboulle 2009) is used to solve this problem. Yet, these algorithms require running many iterative operations that are always expensive.

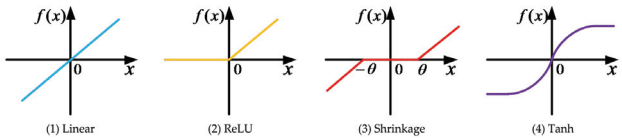

Figure 1: Four popular activation functions.

SSC-LISTA. (Gregor and LeCun 2010) In order to avoid the expensive computation, training ℓ_1 sparse code predictor is an effective method (Kavukcuoglu, Ranzato, and LeCun 2008; Gregor and LeCun 2010). The ℓ_1 sparse codes are produced by minimizing an energy function:

$$L = \|\mathbf{y}_j - \mathbf{Y}\mathbf{x}_j\|_2^2 + \alpha\|\mathbf{x}_j\|_1 + \beta\|\mathbf{x}_j - s(\mathbf{y}_j, \phi)\|_2^2, \quad (3)$$

where $s(\mathbf{y}_j, \phi)$ is a trainable feed-forward neural network, ϕ collectively designates all the trainable parameters, α and β are the regularization parameters. The last term can lead to a fast inference because the codes are obtained by only calculating a matrix multiplication and a nonlinear activation function (Kavukcuoglu, Ranzato, and LeCun 2008; Gregor and LeCun 2010). Unfortunately, LISTA is mainly applied in classification tasks (Kavukcuoglu et al. 2009; Chang et al. 2013). In this paper, we extend LISTA to do SSC experiment.

Sparse Subspace Clustering by the learned orthogonal matching pursuit (SSC-LOMP)

In this section, we firstly formulate the problem to train a single hidden neural network (SHNN) to approximate the ℓ_0 codes, and solve this problem by using orthogonal matching pursuit and gradient descent.

Problem Formulation

SSC is to fast find a sparse representation of each data point in the subspace clustering problem. Since the problem (1) is NP-hard, it is relaxed by solving the ℓ_1 problem (2). In practice, however, solving ℓ_1-minimization problem over n variables may be prohibitive if n is large (You, Robinson, and Vidal 2016). Unfortunately, it also results in that LISTA does not work in ℓ_1 sparse coding. Thus, we backtrack to the ℓ_0-minimization problem (1). In fact, this problem can be solved by using the orthogonal matching pursuit (OMP) algorithm. However, OMP still takes more time to search the solution when the number of data points is large. Inspired by the fast coding way of LISTA (Kavukcuoglu, Ranzato, and LeCun 2008; Gregor and LeCun 2010), therefore, we train a single hidden neural network (SHNN) to replace the expensive inference in the problem (1). Therefore, the optimization problem is derived as follows:

$$\{\mathbf{W}^{1*}, \mathbf{W}^{2*}, \mathbf{x}_j^*\} = \arg\min_{\mathbf{W}^1, \mathbf{W}^2} \|\mathbf{y}_j - \mathbf{Y}\mathbf{x}_j\|_2^2 \quad (4)$$
$$s.t. \; \|\mathbf{x}_j\|_0 < k, \; \mathbf{x}_{jj} = 0,$$

where $\mathbf{Y} \in \mathbb{R}^{d\times n}$ is a data matrix, $\mathbf{x}_j \in \mathbb{R}^n$ is a ℓ_0 sparse coefficient vector of jth sample, k is the maximum of this ℓ_0 vector, and $\mathbf{x}_j$ is defined as following SHNN:

$$\mathbf{x}_j = f(\mathbf{W}^2 g(\mathbf{W}^1 \mathbf{y}_j)), \quad (5)$$

Algorithm 1 Orthogonal Matching Pursuit (OMP)

Input: Dictionary $\mathbf{D} = [\mathbf{d}_1, \cdots, \mathbf{d}_N] \in \mathbb{R}^{D\times N}$, data $\mathbf{y}$, k_{max}, ϵ.
Initialize: residual $\mathbf{r}_0 = \mathbf{y}$, support set $\mathcal{T}_0 = \emptyset$, and iteration counter $k = 0$.
While $k < k_{max}$ and $\|\mathbf{r}_0\|_2 > \epsilon$ **do**
  Update $\mathcal{T}_{k+1} = \mathcal{T}_k \bigcup \{i^*\}$, where i^* is solved by

$$i^* = \arg \min_{i\in\{1,\cdots,N\}} |\mathbf{d}_i^T \mathbf{r}_k|;$$

  Update $\mathbf{r}_k = \left(1 - P_{\mathcal{T}_{k+1}}\right)\mathbf{y}$, where $P_{\mathcal{T}_{k+1}}$ is the projection onto the linear space spanned by $\{\mathbf{d}_j\}_{j\in\mathcal{T}_{k+1}}$;
  $k = k + 1$;
end While
Return solutions $\mathbf{x}^* = \arg\min_{\mathbf{x}:\text{Supp}\subseteq\mathcal{T}_{k+1}} \|\mathbf{y} - \mathbf{Y}\mathbf{x}\|_2^2$.

Algorithm 2 Learning $\mathbf{W}^1$ and $\mathbf{W}^2$ by the gradient descent (GD) algorithm

Input: Data $\mathbf{y}_j$, $\mathbf{x}_j^*$, and Parameters t_{max}, $\beta, \varepsilon, \epsilon$.
Initialize: random initialization weights $\mathbf{W}^1$, and $t = 1$.
While $t < t_{max}$ and $\|\mathbf{x}_j^* - \mathbf{W}^2 g(\mathbf{W}^1\mathbf{y}_j)\|_2^2/N > \epsilon$ **do**
  Calculate $\mathbf{h}_j$ as $\mathbf{h}_j = f(\mathbf{W}^1\mathbf{y}_j)$;
  Update $\mathbf{W}^2$ by Eq. (8);
  Update $\mathbf{W}^1$ by Eq. (9);
  $t = t + 1$;
end While
Return solutions $\mathbf{W}^1$ and $\mathbf{W}^2$.

where $\mathbf{W}^1 \in \mathbb{R}^{m\times d}$ and $\mathbf{W}^2 \in \mathbb{R}^{n\times m}$ are the trainable matrix parameters, m is the number of hidden units, and $f(\cdot)$, $g(\cdot)$ are activation functions. We mainly consider the four popular activation functions shown in Fig. 1, which are linear function and ReLU function (Nair and Hinton 2010; Glorot, Bordes, and Bengio 2011), shrinkage function (Gregor and LeCun 2010), and Tanh function. After training the weights, the ℓ_0 sparse code is obtained by fast calculating Eq. (5), not optimizing the problem (1).

Optimization

Due to the non-linear, feed-forward neural network, the problem (4) is non-convex. For simplicity, we propose a learned OMP algorithm (LOMP) to solve the problem (4). The complete optimization procedure of LOMP is summarized in **Algorithm 3**. LOMP is implemented by reasonably employing three steps to obtain a local solution. The first step is to obtain the convergent ℓ_0 sparse codes by using the OMP algorithm. The second step is to approximate the codes by using the gradient descent algorithm to learn the SHNN model (5). The three step is to fast compute the ℓ_0 sparse codes by the SHNN model. The procedures are as follows.

Update $\mathbf{x}_j^*$. $\mathbf{x}_j^*$ is solved by the following problem:

$$\mathbf{x}_j^* = \arg\min_{\mathbf{x}_j} \|\mathbf{y}_j - \mathbf{Y}\mathbf{x}_j\|_2^2 \; s.t. \; \|\mathbf{x}_j\|_0 < k, \; \mathbf{x}_{jj} = 0. \quad (6)$$

This problem can be solved by using the orthogonal matching pursuit (OMP) algorithm, which is a stepwise forward selection algorithm and is easy to implement in **Algorithm 1**. The key of OMP is to greedily select one column of $\mathbf{Y}$ at a time by maximizing the absolute value of the dot product with the residual, and compute the coefficients for the selected columns.

Algorithm 3 Learned OMP (LOMP) algorithm

Input: Dictionary $\mathbf{Y} = [\mathbf{y}_1, \cdots, \mathbf{y}_N] \in \mathbb{R}^{D\times N}$, data $\mathbf{y}$, and parameters t_{max}, k_{max}, $\beta, \varepsilon, \epsilon$.
Calculate $\mathbf{x}^*$ by **Algorithm 1**;
Calculate $\mathbf{W}^1$ and $\mathbf{W}^2$ by **Algorithm 2**;
Calculate $\mathbf{x}$ by $\mathbf{x} = f(\mathbf{W}^2 g(\mathbf{W}^1\mathbf{y}))$;
Return solutions $\mathbf{W}^1$, $\mathbf{W}^2$ and $\mathbf{x}$.

Algorithm 4 Sparse Subspace Clustering by the learned OMP Algorithm (SSC-LOMP)

Input: Data 1 $\mathbf{Y} = [\mathbf{y}_1, \cdots, \mathbf{y}_N] \in \mathbb{R}^{D\times N}$, data 2 $\widehat{\mathbf{Y}} = [\widehat{\mathbf{y}}_1, \cdots, \widehat{\mathbf{y}}_N] \in \mathbb{R}^{D\times N}$, and parameters t_{max}, k_{max}, $\beta, \varepsilon, \epsilon$.
Calculate $\mathbf{X}$, $\mathbf{W}^1$ and $\mathbf{W}^2$ by **Algorithm 3**;
Calculate $\widehat{\mathbf{X}}$ by $\widehat{\mathbf{X}} = f(\mathbf{W}^2 g(\mathbf{W}^1\widehat{\mathbf{Y}}))$;
Set $\mathbf{C} = |\mathbf{X}| + |\mathbf{X}^T|$ and $\widehat{\mathbf{C}} = |\widehat{\mathbf{X}}| + |\widehat{\mathbf{X}}^T|$.
Apply spectral clustering to $\mathbf{C}$ and $\widehat{\mathbf{C}}$.
Return Segmentation of data 1 $\mathbf{Y}$ and data 2 $\widehat{\mathbf{Y}}$.

Update $\mathbf{W}^1$ and $\mathbf{W}^2$. While $\mathbf{x}_j^*$ and $\mathbf{y}_j$ are given, $\mathbf{W}^1$ and $\mathbf{W}^2$ are calculated by minimizing the following problem

$$\mathcal{L} = \min_{\mathbf{W}^1,\mathbf{W}^2} \|\mathbf{x}_j^* - \mathbf{W}^2 g(\mathbf{W}^1\mathbf{y}_j)\|_2^2/2 + \beta\left(\|\mathbf{W}^1\|_{\text{F}}^2 + \|\mathbf{W}^2\|_{\text{F}}^2\right)/2, \quad (7)$$

where the first term is the minimal error between SHNN and the ℓ_0 codes, the second term is the regularization term, and β is the parameter. By using a gradient descent algorithm (Deng and Yu 2011; Li, Chang, and Yang 2015), when $\mathbf{W}^1$ is fixed, $\mathbf{W}^2$ has a closed-form solution:

$$\mathbf{W}^2 = \left(\mathbf{h}_j\mathbf{h}_j^T + \beta\mathbf{I}\right)^{-1}\mathbf{h}_j(\mathbf{x}_j^*)^T, \quad (8)$$

where $\mathbf{h}_j = g(\mathbf{W}^1\mathbf{y}_j)$. When $\mathbf{W}^2$ is fixed, the algorithm updates $\mathbf{W}^1$ as

$$\mathbf{W}^1 = \mathbf{W}^1 - \varepsilon\partial\mathcal{L}/\partial\mathbf{W}^1, \quad (9)$$

where ε is a learning rate, $\partial\mathcal{L}/\partial\mathbf{W}^1$ is an easily calculated gradient. The weights learning process is outlined in **Algorithm 2**. In fact, GD is difficult to guarantee the convergence in theory, but behaves well in practice (Haykin 2009).

Compute ℓ_0 sparse coding $\mathbf{x}_j$. $\mathbf{x}_j$ is calculated by $\mathbf{x}_j = f(\mathbf{W}^2 g(\mathbf{W}^1\mathbf{y}_j))$. In order to obtain more zero elements (not small numbers), the shrinkage function is selected as $f(\cdot)$ to truncate the small numbers for ℓ_0 sparse coding. We also can select the ReLU function for non-negative ℓ_0 sparse coding.

Clustering

Given a data matrix $\mathbf{Y}$, the weights $\mathbf{W}^1$ and $\mathbf{W}^2$, and the sparse coefficient matrix $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N]$ can be obtained

by **Algorithm 3**. Suppose data matrix $\widehat{\mathbf{Y}}$ is drawn from a union of the same subspaces, $\widehat{\mathbf{X}}$ is fast computed by $\widehat{\mathbf{X}} = f(\mathbf{W}^2 g(\mathbf{W}^1 \widehat{\mathbf{Y}}))$ instead of the OMP algorithm. Similar to the SSC method, spectral clustering is applied on the affinity matrix $\mathbf{C} = |\mathbf{X}| + |\mathbf{X}^T|$ and $\widehat{\mathbf{C}} = |\widehat{\mathbf{X}}| + |\widehat{\mathbf{X}}^T|$ to obtain the segmentation of the data in the low-dimensional space. The SSC-LOMP procedure are summarized in **Algorithm 4**.

Theoretical Analysis of SSC-LOMP

In this section, we discuss the convergence of **Algorithm 3** and study the subspace-preserving property.

Convergence Analysis

The optimization problem (4) is heavily non-convex due to the SHNN network. We propose the LOMP algorithm to solve this problem. **LOMP Algorithm 3** consists of three independent parts. Under certain conditions, **OMP Algorithm 1** (Wang et al. 2016) can solve the problem (6) in the first part. The second part is to use **GD Algorithm 2** to minimize the problem (7). The third part is to compute the ℓ_0 sparse codes by SHNN. The key of convergence is to approximate the ℓ_0 sparse codes by using SHNN. Next, we discuss the second part. Fortunately, $\mathbf{x}$ can be approximated uniformly on compact by $\mathbf{W}^2 g(\mathbf{W}^1 \mathbf{y})$ with enough hidden units (Ripley 1996). We use the **GD Algorithm 2** to learn the weights $\mathbf{W}^1$ and $\mathbf{W}^2$ in (7). Although it is difficult to prove the convergence, it is also considered to have converged if $\|\mathbf{x} - \mathbf{W}^2 g(\mathbf{W}^1 \mathbf{y})\|_2^2 / n < \epsilon$, where the approximation accuracy ϵ is sufficiently small (Haykin 2009).

Subspace Preserving Property

Based on the *self-expressiveness property* (Elhamifar and Vidal 2013) and the *independent (or disjoint) subspaces hypothesis* (Elhamifar and Vidal 2013; Lu et al. 2012; Liu et al. 2013), a subspace preserving (SP) property (You, Robinson, and Vidal 2016) is used to evaluate the clustering ability in SSC, which is defined as

Definition 1 *(Subspace Preserving Property (You, Robinson, and Vidal 2016).) A representation vector $\boldsymbol{x}_i \in \mathbb{R}^n$ of a point $\boldsymbol{y}_i$ belonged to ith subspace $\mathcal{S}_i$ in terms of the dictionary $\boldsymbol{Y} = [\boldsymbol{y}_1, \cdots, \boldsymbol{y}_n]$ is called subspace preserving if its nonzero entries correspond to points in $\mathcal{S}_i$, i.e.*

$$\forall j = 1, \cdots, n, \qquad \mathbf{x}_{ij} \neq 0 \Longrightarrow \mathbf{y}_j \in \mathcal{S}_i. \tag{10}$$

According to Definition 1, some sufficient conditions are used to guarantee that the sparse representation learned by OMP is subspace preserving in deterministic (independent or arbitrary) subspaces (You, Robinson, and Vidal 2016).

In Definition 1 there is a strict condition that the entries corresponded to points in other subspaces must be zeros. However, the zero entries are difficultly approximated by SHNN in the problem (4). Generally, the approximation accuracy ϵ is set to be less than a threshold value θ. For studying the SP property in the problem (4), we extend it to θ-SP property, which is defined as

Definition 2 *(θ-Subspace Preserving Property.) A representation vector $\boldsymbol{x}_i \in \mathbb{R}^n$ of a point $\boldsymbol{y}_i$ belonged to ith subspace $\mathcal{S}_i$ in terms of the dictionary $\boldsymbol{Y} = [\boldsymbol{y}_1, \cdots, \boldsymbol{y}_n]$ is called θ-subspace preserving if its entries satisfied $|\boldsymbol{x}_{ij}| > \theta$ correspond to points in $\mathcal{S}_i$. i.e.*

$$\forall j = 1, \cdots, n, \qquad |\mathbf{x}_{ij}| > \theta \Longrightarrow \mathbf{y}_j \in \mathcal{S}_i. \tag{11}$$

Compared to the SP property, Definition 2 gives a threshold value θ to softly represent the nonzero entry, that is, $\mathbf{x}_{ij}$ is a nonzero entry if it satisfies $|\mathbf{x}_{ij}| > \theta$. Next, we give two theorems on SSC-LOMP to ensure the subspace-preserving property for independent and arbitrary subspaces.

Theorem 1. *If the subspaces are independent, and the approximation accuracy ϵ is less than the threshold value θ, then after training by LOMP, the representation computed by SHNN* (5) *is subspace preserving.*

In order to study the separability among arbitrary subspaces, the residual direction set and the coherence measure are defined as follows.

Definition 3. (You, Robinson, and Vidal 2016) Let $\mathcal{R}(\mathbf{D}, \mathbf{y})$ be the set of all residual vectors computed in step 5 of OMP($\mathbf{D}$, $\mathbf{y}$). The set of OMP residual directions associated with the data matrix $\mathbf{Y}^i$ is defined as

$$\mathcal{W}^i := \bigcup_{j:\mathbf{y}_j \in \mathcal{S}_i} \mathcal{W}_j^i, \tag{12}$$

where $\mathcal{W}_j^i$ (that is the set of OMP residual directions associated with matrix $\mathbf{Y}_{-j}^i$ and point $\mathbf{y}_j \in \mathcal{S}_i$) is defined as

$$\mathcal{W}_j^i := \left\{ \mathbf{w} = \frac{\mathbf{r}}{\|\mathbf{r}\|_2} \middle| \, \mathbf{r} \in \mathcal{R}(\mathbf{Y}_{-j}^i, \mathbf{y}_j), \mathbf{r} \neq 0 \right\}. \tag{13}$$

Definition 4. (You, Robinson, and Vidal 2016) *The coherence between two sets of points of unit norm, $\mathcal{X}$ and $\mathcal{Y}$, is defined as $\mu(\mathcal{X}, \mathcal{Y}) = \max_{\boldsymbol{x} \in \mathcal{X}, \boldsymbol{y} \in \mathcal{Y}} | < \boldsymbol{x}, \boldsymbol{y} > |$.*

Theorem 2. *If the approximation accuracy ϵ is less than the threshold value θ, and*

$$\forall j = 1, \cdots, N, \qquad \max_{k:k \neq j} \mu\left(\mathcal{W}^j, \mathcal{X}^k\right) < r_j, \tag{14}$$

where $r_j = \min_i r(\mathcal{P}_{-i}^j)$, and $r(\mathcal{P}_{-i}^j) = conv(\pm \mathcal{X}_{-i}^j)$ is the symmetrized convex hull of the points in the jth subspace other than $\boldsymbol{x}_i$ then after training by LOMP, the representation computed by SHNN (5) *is subspace preserving.*

Theorem 1 and 2 show that the representation computed by SHNN (5) is subspace preserving under independent and arbitrary subspaces even if the representation learned by (7) is θ-subspace preserving. The key technology is to use the shrinkage function to forcibly set zero when $\epsilon < \theta$.

Experiments

In this section, we first introduce the experimental settings. Second, we verify that our method (SSC-LOMP) has better subspace preserving property than SSC-OMP, and achieves the state-of-the-art clustering results on MNIST dataset. Third, we conclude that SSC-LOMP and SSC-LOMP have a similar subspace preserving property, and SSC-LOMP outperforms the relevant subspace clustering methods in clustering results on Extended YaleB dataset.

Table 1: ACC (%), NMI (%) and time (s) on MNIST and Extended YaleB.

	MNIST					Extended YaleB				
No. points/classes	50	100	200	400	600	2	10	20	30	38
Clustering accuracy (ACC) (%)										
SSC-LOMP (ours)	**87.62±3.13**	**89.67±3.67**	**92.47±1.15**	**93.25±0.79**	**94.32±0.72**	99.09±1.28	**90.47±4.43**	**83.43±2.05**	**79.87±1.96**	**78.01±1.43**
SSC-OMP1 (You, Robinson, and Vidal 2016)	82.00±5.75	87.19±4.54	89.71±4.01	91.13±3.79	93.20±0.68	-	-	-	-	-
SSC-OMP (You, Robinson, and Vidal 2016)	85.07±3.82	87.62±4.57	88.99±4.53	89.83±5.32	90.56±5.48	**99.21±1.22**	86.43±6.28	81.71±3.20	78.27±3.02	77.60±1.95
SSC-LISTA (Gregor and LeCun 2010)	82.26±3.32	85.78±5.12	84.83±4.73	85.26±4.63	84.63±5.26	99.21±0.93	89.89±3.60	75.80±4.64	71.69±3.35	68.26±1.01
SSC-ISTA (Elhamifar and Vidal 2013)	82.86±2.72	84.62±4.35	85.29±4.21	85.87±4.78	85.23±5.71	99.19±0.94	89.97±3.43	76.44±4.97	70.82±2.91	67.20±1.30
LRR (Liu et al. 2013)	79.34±5.21	79.23±3.88	80.50±4.51	79.32±2.41	81.48±5.01	95.46±4.12	68.30±5.67	69.64±8.60	69.18±1.74	67.74±1.45
LSR (Lu et al. 2012)	80.14±3.26	78.83±2.23	79.14±1.92	80.39±1.19	80.42±1.13	94.50±4.86	62.68±3.38	63.38±1.43	60.38±1.43	58.62±9.82
Normalized mutual information (NMI) (%)										
SSC-LOMP (ours)	**80.19±3.41**	**83.18±2.50**	**85.05±1.63**	**86.21±1.08**	**87.89±1.09**	94.44±7.47	**86.45±4.42**	**84.07±1.64**	**82.07±1.43**	**80.94±0.62**
SSC-OMP1 (You, Robinson, and Vidal 2016)	76.43±4.08	80.17±2.85	82.58±2.11	84.14±1.67	85.70±0.91	-	-	-	-	-
SSC-OMP (You, Robinson, and Vidal 2016)	77.86±2.91	80.68±2.93	82.58±2.15	83.70±2.21	84.76±2.17	**95.19±6.66**	85.24±4.77	82.31±1.97	81.30±1.54	80.40±0.66
SSC-LISTA (Gregor and LeCun 2010)	75.91±2.62	80.49±2.93	80.37±2.23	81.69±2.11	81.86±2.20	94.75±5.66	85.02±4.20	74.45±3.94	71.96±3.18	69.99±0.60
SSC-ISTA (Elhamifar and Vidal 2013)	76.86±3.21	79.52±3.16	81.53±2.62	82.92±1.97	82.52±2.64	94.54±5.76	85.02±3.90	75.19±4.24	71.94±2.75	69.34±0.84
LRR (Liu et al. 2013)	77.61±3.15	79.15±2.21	80.37±2.23	80.07±1.38	80.26±1.61	78.88±15.3	62.52±4.77	69.67±8.33	71.54±1.24	71.20±7.37
LSR (Lu et al. 2012)	75.88±3.63	73.77±3.14	74.58±2.25	76.26±1.75	76.59±1.67	76.23±17.1	53.61±3.42	58.18±1.95	56.63±8.74	56.28±9.87
Running time ($sec.$)										
SSC-LOMP (ours test)	0.08	0.3	0.8	3.5	15.5	0.07	0.4	0.8	1.52	3.1
SSC-LOMP (ours train)	2.5	4.3	15.0	72.3	184.7	7.3	11.3	18.2	26.6	59.1
SSC-OMP (You, Robinson, and Vidal 2016)	1.1	2.5	7.0	24.8	35.9	0.3	1.5	3.9	8.1	11.3
SSC-LISTA (Gregor and LeCun 2010)	23.2	43.7	178.5	865.1	2581.3	39.2	125.9	382.1	1182.5	1432.6
SSC-ISTA (Elhamifar and Vidal 2013)	21.8	41.6	169.2	787.4	2363.2	25.9	123.7	369.6	1156.3	1353.1
LRR (Liu et al. 2013)	34.6	69.7	124.9	287.8	466.9	3.2	67.1	353.1	1148.3	1439.8
LSR (Lu et al. 2012)	0.5	1.5	14.6	96.1	196.9	0.1	0.8	4.5	14.6	35.6

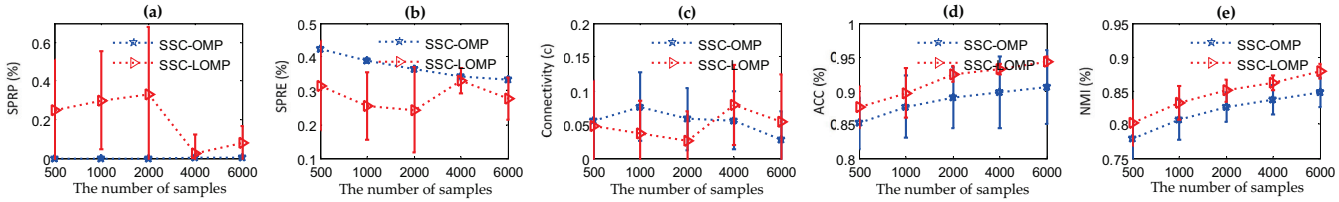

Figure 2: Performance of SSC-LOMP and SSC-OMP on MNIST. SPRP, SPRE, connectivity, ACC, and NMI are shown in (a)-(e). The overall number of points is varied from 500 to 6000.

Experimental Settings

Datasets. MNIST[1] has 70,000 examples with 28×28 pixel greyscale images of handwritten digits 0-9. We use a scattering convolution network (Bruna and Mallat 2013) to compute the feature vectors of each image by following the experimental data settings (You, Robinson, and Vidal 2016). Each feature vector is of length size 3,472 and the features are reduced to 500 dimensions by PCA. We randomly choose $\{50, 100, 200, 400, 600\}$ images from each class for all experiments. **Extended YaleB**[2] contains frontal face images of 38 individuals with 64 different illumination conditions. The size of each image is 192×168, and the original face images are downsampled to 48×42 pixels. We randomly pick $n_i \in \{2, 10, 20, 30, 38\}$ classes, and take all the images of them as the data to be clustered.

Evaluations. We follow the subspace preserving representations percentage (SPRP), subspace preserving representation error (SPRE), and connectivity to evaluate the degree of the SP property (You, Robinson, and Vidal 2016), we use the clustering accuracy (ACC) and the normalized mutual information (NMI) (Cai and Chen 2015) to evaluate the clustering results. For fair comparison, we report the average performance of 20 tests in all the experiments, and all algorithms run on Matlab 2015 and Windows 7. We experimentally select *shrinkage* for the activation function $f(\cdot)$ as the clustering results (ACC and NMI) by using *shrinkage* are better than *linear* and *ReLU* in Table 2.

Comparison methods. We compare our model, SSC-LOMP, with the state-of-the-art subspace clustering methods, including SSC-OMP, (You, Robinson, and Vidal 2016) SSC-LISTA (Gregor and LeCun 2010), SSC-ISTA (Elhamifar and Vidal 2013) (also called SSC-BP in (You, Robinson, and Vidal 2016)), LRR (Liu et al. 2013) and LSR (Lu et al. 2012). The codes provided by the respective authors are used to calculate the representation matrix $\mathbf{X}$ in all experiments, except for SSC-LISTA, which is coded by us, because it is first to apply into subspace clustering. We tune the parameters to

[1]http://yann.lecun.com/exdb/mnist/

[2]http://vision.ucsd.edu/ leekc/ExtYaleDatabase/ExtYaleB.html

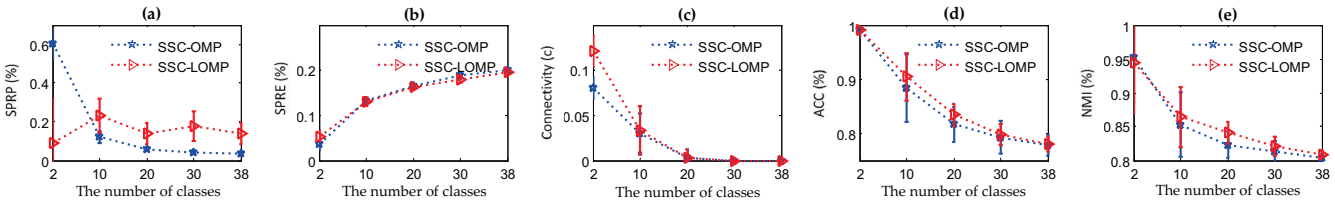

Figure 3: Performance of SSC-LOMP and SSC-OMP on Extend-YaleB. SPRP, SPRE, connectivity, ACC, and NMI are shown in (a)-(e). The overall number of classes is varied from 2 to 38.

Table 2: ACC (%) and NMI (%) with different activation functions.

$f(\cdot)$	linear	ReLU	shrinkage
MNIST (1000)			
ACC (%)	77.37±9.80	86.64±6.00	**89.67±3.67**
NMI (%)	73.18±6.92	82.16±3.02	**83.18±2.50**
Extended YaleB (10 classes)			
ACC (%)	47.48±4.19	69.51±7.47	**90.47±4.43**
NMI (%)	44.82±3.77	70.10±5.34	**86.45±4.42**

obtain the best clustering accuracy and normalized mutual information. Then we apply the normalized spectral clustering to the affinity $|\mathbf{X}| + |\mathbf{X}^T|$, except for SCC which has its own spectral clustering step.

Parameters Settings. In the proposed method (SSC-LOMP), there are three important parameters θ, m, and ε, which control the threshold value of shrinkage function, the the number of hidden units and learning rate. Actually, θ, m and ε are the typical parameters in neural networks (Gregor and LeCun 2010; Li, Chang, and Yang 2015). So, they are easy to choose in our given sets: $\theta = \{0.2, 0.1, 0.05, 0.03, 0.02, 0.01, 0.005, 0.001, 0.0001\}$, $M = \{500, 1000, 1200, 2000, 2500, 3000, 3500, 4000, 5500\}$ and $\varepsilon = \{0.01, 0.001, 0.0001, 0.00001\}$. In all experiments, k_{max} is set to 5 (Extended YaleB) or 10 (MNIST), the regularization parameter of weights β is set to 0.00001, *shrinkage* and *tanh* (or ReLU) are voted as the activation function $f(\cdot)$ and $g(\cdot)$, and the number of training iterations t_{max} is less than 10.

Results on MNIST and Extended YaleB

All results on MNIST and Extended YaleB are reported in Fig. 2, Fig. 3 and Table 1.

Subspace preserving (SP) property. The subspace-preserving representation percentage (SPRP) and subspace-preserving representation error (SPRE) are used to evaluate the degree of the SP property. A solution has a strong SP property if it is high SPRP and low SPRE.

MNIST: SPRP and SPRE on MNIST are plotted in Fig. 2(a) and Fig. 2(b), respectively. We observe that the representation learned by SSC-LOMP is higher SPRP and lower SPRE than SSC-OMP. This observation verifies that SSC-LOMP gives a subspace-preserving representation as it has a stronger SP property than SSC-OMP. The connectivity is also shown in Fig. 2(c). We found that SSC-OMP does not have as good a connectivity as SSC-BP when the number of data points is small. An important reason is that the sampling is not sufficient. In fact, SSC-LOMP has a good connectivity with the big number of data points.

Extended YaleB: Similar to the analysis on MNIST, SPRP and SPRE on Extended YaleB are plotted in Fig. 3(a) and Fig. 3(b), respectively. We observe that SSC-LOMP gives a subspace-preserving representation because the representation learned by SSC-LOMP is higher SPRP than SSC-OMP even if they obtain the same SPRE. Fig. 3(c) also shows the connectivity. We found that the connectivity of SSC-LOMP is similar to SSC-OMP. This observation validates that SSC-LOMP can effectively approximate SSC-OMP. In addition, the Algorithm 2 converges quickly after 5-10 training epochs.

Clustering results. In Table 1, we observe that SSC-LOMP outperforms all the baseline methods including SSC-OMP1[3], SSC-OMP, SSC-LISTA, SSC-ISTA, LRR and LSR, except for the 2 individuals data on Extended YaleB.

MNIST: Compared to SSC-OMP, SSC-LOMP has high ACC and NMI shown in Fig. 2(d) and Fig. 2(e). It achieves average 3.03% improvement on ACC, and reaches average 2.59% improvement on NMI. Moreover, SSC-LOMP is higher than other methods by average 6.4% ACC and 3.5% NMI improvement. *SSC-LOMP is better than the state-of-the-art method (EnSC-ORGEN) (You et al. 2016), which is* 93.79%.

Extended YaleB: Compared to SSC-OMP, SSC-LOMP has high ACC and NMI shown in Fig. 3(d) and Fig. 3(e). SSC-LOMP achieves average 0.9% improvement on ACC, and reaches average 0.7% improvement on NMI. Moreover, SSC-LOMP is higher than other methods by average 5.0% ACC and 6.0% NMI improvement.

Fast inference time. The test and training time of SSC-LOMP essentially depend on the number of hidden units in SHNN. As shown in Table 1, SSC-LOMP is always faster than SSC-OMP in test inference time even if it has the large number of hidden units such as 5500 when we select 6000 examples. Moreover, SSC-LOMP is above many times faster than LRR and SSC-ISTA. Although SSC-LOMP needs some time to train SHNN, SSC-LOMP is faster than SSC-OMP when new examples is sampled from the same subspaces.

Conclusion

In this paper, we proposed a sparse subspace clustering method by the learned orthogonal matching pursuit (SSC-LOMP), which consisted of three parts. First, the ℓ_0 sparse code is solved by orthogonal matching pursuit (OMP). Second, a single hidden neural network (SHNN) is trained to approximate the ℓ_0 sparse code. Third, a new ℓ_0 sparse code fast

[3]The data is randomly chosen from the training set (60000 examples), while we randomly choose the data from the training and test sets (70000 examples).

computed by SHNN was used to construct the affinity matrix for sparse subspace clustering. Clearly, SSC-LOMP is faster test inference time than other related methods (such as SSC-ISTA and SSC-OMP) as the affinity matrix is quickly computed by SHNN. We further provided the sufficient conditions to show that the representation learned by SSC-LOMP is subspace preserving in independent and arbitrary subspaces. Experimental results demonstrated that SSC-LOMP outperforms the relevant subspace clustering methods in clustering results on MNIST and Extended YaleB datasets.

Acknowledgments

This work is supported in part by the NSF IIS award 1651902, NSF CNS award 1314484, ONR award N00014-12-1-1028, ONR Young Investigator Award N00014-14-1-0484, and U.S. Army Research Office Young Investigator Award W911NF-14-1-0218.

References

Beck, A., and Teboulle, M. 2009. A fast iterative shrinkage thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sciences* 2(1):183–202.

Bruna, J., and Mallat, S. 2013. Invariant scattering convolution networks. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 35(8):1872–1886.

Cai, D., and Chen, X. L. 2015. Large scale spectral clustering via landmark-based sparse representation. *IEEE Trans. on Cybernetics* 45(8):1669–1680.

Chang, H.; Zhou, Y.; Spellman, P.; and Parvin, B. 2013. Stacked predictive sparse coding for classification of distinct regions in tumor histopathology. In *ICCV*.

Deng, L., and Yu, D. 2011. Deep convex networks: A scalable architecture for speech pattern classification. In *Interspeech*, 2285–2288.

Dyer, E.; Sankaranarayanan, A.; and Baraniuk, R. 2013. Greedy feature selection for subspace clustering. *Journal of Machine Learning Research* 14(1):2487–2517.

Elhamifar, E., and Vidal, R. 2013. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 35:2765–2781.

Glorot, X.; Bordes, A.; and Bengio, Y. 2011. Deep sparse rectifier neural networks. In *AISTATS*, 315–323.

Gregor, K., and LeCun, Y. 2010. Learning fast approximations of sparse coding. In *ICML*.

Haykin, S. 2009. In *Neural Networks and Learning Machines*. Pearson Education Inc.

Kavukcuoglu, K.; Ranzato, M.; Fergus, R.; and LeCun, Y. 2009. Learning invariant features through topographic filter maps. In *CVPR*.

Kavukcuoglu, K.; Ranzato, M.; and LeCun, Y. 2008. Fast inference in sparse coding algorithms with applications to object recognition. In *CBLL-TR-2008-12-01*.

Li, J.; Kong, Y.; Zhao, H.; Yang, J.; and Fu, Y. 2016a. Learning fast low-rank projection for image classification. *IEEE Trans. Image Process.* 25(10):4803–4814.

Li, J.; Zhang, T.; Luo, W.; Yang, J.; Yuan, X.; and Zhang, J. 2016b. Sparseness analysis in the pertraining of deep neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* doi:10.1109/TNNLS.2016.2541681.

Li, J.; Chang, H.; and Yang, J. 2015. Sparse deep stacking network for image classification. In *AAAI*.

Li, S.; Li, K.; and Fu, Y. 2015. Temporal subspace clustering for human motion segmentation. In *ICCV*.

Liu, H., and Fu, Y. 2015. Clustering with partition level side information. In *Proceedings of ICDM*.

Liu, G.; Lin, Z.; Yan, S.; Sun, J.; Yu, Y.; and Ma, Y. 2013. Robust recovery of subspace structures by low-rank representation. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 35:171–184.

Liu, H.; T.Liu; Wu, J.; Tao, D.; and Fu, Y. 2015. Spectral ensemble clustering. In *KDD*.

Liu, H.; Shao, M.; Li, S.; and Fu, Y. 2016. Infinite ensemble for image clustering. In *KDD*.

Lu, C. Y.; Min, H.; Zhao, Z. Q.; Zhu, L.; Huang, D. S.; and Yan, S. 2012. Robust and efficient subspace segmentation via least squares regression. In *ECCV*, 347–360.

Mallat, S., and Zhang, Z. 1993. Matching pursuits with time-frequency dictionaries. *IEEE Trans. on Signal Processing* 41:3397–3415.

Nair, V., and Hinton, G. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*, 807–814.

Rao, S.; Tron, R.; Vidal, R.; and Ma, Y. 2010. Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 32(10):1832–1845.

Ripley, B. 1996. In *Pattern Recognition and Neural Networks*. Cambridge University Press.

Shao, M.; Li, S.; Ding, Z.; and Fu, Y. 2015. Deep linear coding for fast graph clustering. In *IJCAI*, 3798–3804.

Vidal, R. 2011. Subspace clustering. *IEEE Signal Processing Magazine* 28(3):52–68.

Wang, J.; Kwon, S.; Li, P.; and Shim, B. 2016. Recovery of sparse signals via generalized orthogonal matching pursuit: A new analysis. *IEEE Trans. Signal Processing* 64(4):1076–1089.

You, C.; Li, C. G.; Robinson, D. P.; and Vidal, R. 2016. Oracle based active set algorithm for scalable elastic net subspace clustering. In *CVPR*.

You, C.; Robinson, D. P.; and Vidal, R. 2016. Scalable sparse subspace clustering by orthogonal matching pursuit. In *CVPR*.

Zhao, H. D., and Fu, Y. 2015. Dual-regularized multi-view outlier detection. In *IJCAI*, 4077–4083.

Zhao, H. D.; Ding, Z. M.; and Fu, Y. 2016. Pose-dependent low-rank embedding for head pose estimation. In *AAAI*, 1422–1428.