Large-scale Subspace Clustering by Fast Regression Coding

Jun Li[1], **Handong Zhao**[1], **Zhiqiang Tao**[1], and **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, {hdzhao,zqtao,yunfu}@ece.neu.edu

Abstract

Large-Scale Subspace Clustering (LSSC) is an interesting and important problem in big data era. However, most existing methods (i.e., sparse or low-rank subspace clustering) cannot be directly used for solving LSSC because they suffer from the high time complexity-*quadratic* or *cubic* in n (the number of data points). To overcome this limitation, we propose a Fast Regression Coding (FRC) to optimize regression codes, and simultaneously train a nonlinear function to approximate the codes. By using FRC, we develop an efficient **R**egression **C**oding **C**lustering (**RCC**) framework to solve the LSSC problem. It consists of *sampling*, *FRC* and *clustering*. RCC randomly samples a small number of data points, quickly calculates the codes of all data points by using the non-linear function learned from FRC, and employs a large-scale spectral clustering method to cluster the codes. Besides, we provide a theorem guarantee that the non-linear function has a first-order approximation ability and a group effect. The theorem manifests that the codes are easily used to construct a dividable similarity graph. Compared with the state-of-the-art LSSC methods, our model achieves better clustering results in large-scale datasets.

1 Introduction

Subspace clustering, as a fundamental problem, has attracted much attention due to its success in the data mining [Zhao and Fu, 2015a] and computer vision, e.g. image clustering [Lu *et al.*, 2012; Li *et al.*, 2017a], and segmentation of images, video and motion [Liu *et al.*, 2013; Zhao and Fu, 2015b]. The classical subspace clustering methods, such as sparse subspace clustering (SSC) [Elhamifar and Vidal, 2013], low-rank representation (LRR) [Liu *et al.*, 2013; Xiao *et al.*, 2014; Zhang *et al.*, 2015; Lee *et al.*, 2015] and least squares regression (LSR) [Lu *et al.*, 2012] are based on *self-expressiveness* (SE) property, which states that each data point in a union of subspaces can be efficiently represented as a linear or affine combination of other points [Elhamifar and Vidal, 2013]. These methods have already provided theoretical guarantees to recover the subspace representations (codes), and shown the state-of-the-art performances on small datasets. As data (e.g. image, video, and gene) grow rapidly [Liu *et al.*, 2007], subspace clustering (LSSC) with large number of data points becomes an important challenging problem, called Large-Scale Subspace Clustering (LSSC), which is formally defined as follow:

Definition 1 (Large-scale Subspace Clustering (LSSC)). *Given a set of data vectors* $\boldsymbol{X} = [\boldsymbol{X}^1, \cdots, \boldsymbol{X}^i, \cdots, \boldsymbol{X}^k] \in \mathbb{R}^{d \times n}$*, where* d *is the feature dimension,* k *is the number of subspaces,* $\boldsymbol{X}_i = [\boldsymbol{x}_1^i, \cdots, \boldsymbol{x}_j^i, \cdots, \boldsymbol{x}_{n_i}^i] \in \mathbb{R}^{d \times n_i}$ *is drawn from the* i*-th subspace* $\mathcal{S}_i$*,* n_i *is the number of data points of* $\mathcal{S}_i$*, and* $\sum_{i=1}^k n_i = n$*. Moreover,* $r \ll d \ll n$*, where* $r = \sum_{i=1}^k r_i$*, and* r_i *is the number of the bases of* $\mathcal{S}_i$*. The task is to segment the large-scale data according to the underlying subspaces they are drawn from.*

However, the traditional subspace clustering methods cannot well handle the LSSC problem because of the following limitations. First, the high time cost results in that these methods (e.g. SSC, LRR and LSR) are impossible to solve the LSSC problem. According to the SE property, the coding methods usually first select the whole data points as a dictionary to learn the sparse, low-rank and regression codes, and then employ Normalized Cuts (NCuts) [Shi and Malik, 2000] to segment the codes for the clustering tasks. Unfortunately, these reasonable methods generally employ an iterative optimization manner to learn the sparse/low-rank codes, which suffers from a high time complexity-*quadratic* or *cubic* in n (the number of data points) [Wang *et al.*, 2014]. Thus, existing coding-based methods are difficult to directly apply for LSSC.

Second, it is difficult to classify data points by a linear classifier in the sampling-clustering-classification (S-C-C) strategy [Peng *et al.*, 2016; Wang *et al.*, 2014]. The key of S-C-C uses a small number of data points sampled from the large-scale data points to preform subspace clustering by the coding-based methods (e.g. SSC, LRR and LSR), and learn a linear classifier for the LSSC problem by the clustering results. Clearly, the final clustering results heavily depend on the classifier. However, the linear classifier is difficult to classify the complex data points [Bengio *et al.*, 2013]. Therefore, S-C-C is not suitable for LSSC.

To overcome the time limitation, we propose a Fast Regression Coding (FRC) algorithm which directly trains a nonlinear function to approximate regression codes learned from

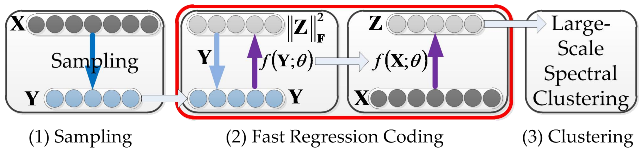

Figure 1: Regression coding clustering framework.

LSR. This function can quickly compute the codes. Based on FRC, we build an efficient **R**egression **C**oding **C**lustering (**RCC**) framework to solve the LSSC problem. As shown in Fig 1, RCC mainly consists of three steps. (1) To reduce the data size, we randomly sample a small data matrix $\mathbf{Y} \in \mathbb{R}^{d\times m} (r < m \ll n)$ from $\mathbf{X}$. (2) The sampled data points are used to train a non-linear function by FRC, and codes of all data points can be quickly calculated by this function. (3) The large-scale spectral clustering (LSSpC) method, such as landmark-based spectral clustering (LSC) [Chen and Cai, 2011; Cai and Chen, 2015], is employed to cluster the non-linear codes, which only needs $\mathcal{O}(n)$ time. The **major contributions** of this paper are two-fold.

1. We propose a fast regression coding (FRC) to quickly compute the codes of large-scale data points. We prove that FRC has a grouping effect that tends to group highly correlated data together.
2. By using FRC, we develop an efficient RCC framework: *sampling*, *FRC* and *LSSpC* to solve the LSSC problem. Moreover, the complexity of RCC is $\mathcal{O}(n)$.

2 Related Works

The subspace clustering methods have shown excellent performance in many real-world applications (also see [Vidal, 2011] for a review). In this section, we mainly review the classical coding (CCod) methods, and the large-scale spectral clustering (LSSpC) methods.

CCod usually selected the observed data matrix $\mathbf{X}$ itself as the dictionary in the independent and disjoint subspaces, such as LSR [Lu *et al.*, 2012], SSC [Elhamifar and Vidal, 2013], and LRR [Liu *et al.*, 2013]. However, this leads to the fact that the coding methods are difficult to compute the codes for the LSSC problems because they spend a lot of time to optimize the codes. In addition, many improved coding methods are proposed in the literatures, such as structured sparse subspace clustering [Li and Vidal, 2015], divide-factor-combine LRR [Talwalkar *et al.*, 2013], and Laplacian regularized LRR [Yin *et al.*, 2016]. However, they are still difficult to solve the LSSC problem due to the expensive inference time. Compared to CCod, our method (FRC) can quickly calculate the codes by using a non-linear projection function.

LSSpC can be roughly divided into two main ways. The first way is to reduce the computational cost of eigen-decomposition over the Laplacian matrix, such as the Nyström method for the approximate eigenvectors [Fowlkes *et al.*, 2004], and parallel eigenvalue computations in distributed systems [Chen *et al.*, 2011]. The second way is to reduce the data size by selecting a small number of samples to replace the original data, for example, k-means methods [Yan *et al.*, 2009], the out-of-sample data [Nie *et al.*, 2011], and LSC [Cai and Chen, 2015]. Although those LSSpC algorithms can solve the LSSC problem, they will lead to poor results as the complex data structure easily results in an indivisible affinity matrix. Compared to LSSpC, our method fast computes the divisible codes, and preform clustering by LSC.

3 Regression Coding Clustering (RCC)

In this section, we develop an efficient **R**egression **C**oding **C**lustering (**RCC**) framework, which consists of *sampling, FRC* and *LSSpC*, to largely cluster a collection of multi-subspace data. Specifically, we formulate a fast regression coding (FRC) model to learn the non-linear function, and then adopt an analytical method and a gradient descent algorithm to solve the FRC model. The landmark-based spectral clustering (LSC) [Cai and Chen, 2015] is used to fast segment the codes of all data points computed by the function. Finally, we provide some theoretical guarantees to show that FRC recovers the subspace representations (codes) effectively.

3.1 Problem Formulation.

To address the LSSC problem in Definition 1, we reduce the data size by selecting a small data matrix $\mathbf{Y}^i \in \mathbb{R}^{d\times m_i}$ drawn from the data matrix $\mathbf{X}^i \in \mathbb{R}^{d\times n_i}$ in the ith subspace $\mathcal{S}_i$ $(1 \le i \le k)$, where m_i is the number of the samples of $\mathbf{Y}_i$, and $r_i < m_i \ll n_i$. Then $\mathbf{Y} = [\mathbf{Y}^1, \cdots, \mathbf{Y}^i, \cdots, \mathbf{Y}^k]$ $(m = \sum_{i=1}^k m_i)$ is used as a small dictionary, and $r = \sum_{i=1}^k r_i < m = \sum_{i=1}^k m_i \ll n = \sum_{i=1}^k n_i$. In order to show the fast coding ability of the small data matrix $\mathbf{Y}$, we extend the SE property [Elhamifar and Vidal, 2013] to a fast self-expressiveness property, which is formally defined as:

Definition 2 (Fast Self-Expressiveness (FSE) Property). *For the large-scale data matrix $\boldsymbol{X} \in \mathbb{R}^{d\times n}$, there exists a non-linear function $f(\cdot;\theta) \in \mathbb{R}^{m\times n}$ such that $\boldsymbol{X} = \boldsymbol{Y} f(\boldsymbol{X};\theta)$, where a small data matrix $\boldsymbol{Y} \in \mathbb{R}^{d\times m}$ is drawn from $\boldsymbol{X}$, and the parameter θ is learned from the following system:*

$$\mathbf{Y} = \mathbf{Y}\mathbf{Z},\ \mathbf{Z} = f(\mathbf{Y};\theta). \tag{1}$$

Remark 1: In general, the traditional coding methods consider $\mathbf{Z}_{ii} = 0$ which aims to avoid the self-representation [Lu *et al.*, 2012; Elhamifar and Vidal, 2013]. However, we use $\mathbf{Y}$ as a self-expressive dictionary to learn the projective function to compute the representations of the large data matrix $\mathbf{X}$. Thus, $\mathbf{Z}_{ii} = 0$ is not restricted in (1).

To fast train the non-linear function to approximate the codes, we exploit a three-layer neural networks (NN) as:

$$\mathbf{Z} = f(\mathbf{Y};\theta) = \mathbf{W}_2 g(\mathbf{W}_1\mathbf{Y}), \tag{2}$$

where $g(\cdot)$ is a nonlinear activation function (that is also a element-wise transformation), $\mathbf{W}_1 \in \mathbb{R}^{h\times d}$ and $\mathbf{W}_2 \in \mathbb{R}^{m\times h}$ are the projective weights, h is the number of hidden units. In this paper we mainly consider the *tanh* activation function $f(a) = (e^a - e^{-a})/(e^a + e^{-a})$, while other nonlinearities (i.e., sigmoid $f(a) = 1/(1 + e^{-a})$, rectified linear unit $max(0, a)$, and rectifier piecewise linear units [Li *et al.*, 2017b]) can also be used.

The Frobenius norm $\|\mathbf{Z}\|_F^2$ in LSR has the power to capture the strong correlations in the same subspace, and does not take more time to carry out many iterations for convergence [Lu *et al.*, 2012]. Moreover, it is used to control the over-fitting problem. To replace $f(\mathbf{Y};\theta)$ by (2), therefore, we propose a FRC model defined as:

$$\min_{\mathbf{Z},\mathbf{W}_1,\mathbf{W}_2} \|\mathbf{Z}\|_{\mathrm{F}}^2 \ s.t.\ \mathbf{Y}=\mathbf{Y}\mathbf{Z},\ \mathbf{Z}=\mathbf{W}_2 g(\mathbf{W}_1\mathbf{Y}). \quad (3)$$

In addition, data are often corrupted by noise due to measurement/process noise and collection techniques in real-world problems [Liu *et al.*, 2013; Elhamifar and Vidal, 2013], for example, the small and independent and identically distributed (i.i.d.) Gaussian noise. In general, the Frobenius norm is used to measure the quality of approximation for fitting the Gaussian noise [Elhamifar and Vidal, 2013]. By using the Frobenius norm to penalize the noise, the problem (3) is written as the following optimization:

$$\min_{\mathbf{Z},\mathbf{W}_1,\mathbf{W}_2} \|\mathbf{Y}-\mathbf{Y}\mathbf{Z}\|_{\mathrm{F}}^2+\alpha\|\mathbf{Z}\|_{\mathrm{F}}^2 \quad (4)$$
$$s.t.\ \mathbf{Z}=\mathbf{W}_2 g(\mathbf{W}_1\mathbf{Y}),$$

where α is a parameter to balance the effects of two terms.

Remark 2: In fact, when $\mathbf{Y}$ is selected as the dictionary, LSR also can be considered as the following problem:

$$\mathbf{C}^\star=\operatorname{argmin}\|\mathbf{C}\|_{\mathrm{F}}^2\ s.t.\ \mathbf{X}=\mathbf{Y}\mathbf{C}. \quad (5)$$

Usually, it has a solution $\mathbf{C}^\star=(\mathbf{Y}^T\mathbf{Y}+\alpha\mathbf{I})^{-1}\mathbf{Y}^T\mathbf{X}$, where α is a regularization parameter. Clearly, this solution is a linear function. However, the linear projection is very limited for clustering tasks as they cannot extract more abstract and non-linear representations [Bengio *et al.*, 2013; Li *et al.*, 2017b]. Therefore, we train a non-linear function (2) to approximate the codes. The approximate accuracy will be studied in subsection 3.4. Moreover, the empirical evidence in Table 6 shows that our non-linear model is better than LSR.

3.2 Optimization.

The problem (4) is non-convex due to the nonlinear function $g(\cdot)$ in NN (2). Fortunately, NN with the suitable number[1] of hidden units can approximate any continuous function [Haykin, 2009]. Then the problem (4) can be written as:

$$\min_{\mathbf{Z},\mathbf{W}_1,\mathbf{W}_2} \mathcal{L}=\|\mathbf{Y}-\mathbf{Y}\mathbf{Z}\|_{\mathrm{F}}^2+\alpha\|\mathbf{Z}\|_{\mathrm{F}}^2 + \beta\|\mathbf{Z}-\mathbf{W}_2 g(\mathbf{W}_1\mathbf{Y})\|_{\mathrm{F}}^2, \quad (6)$$

where β is a regularization parameter. The problem (6) is solved by iteratively updating the variable $\mathbf{Z}$, and the weights $\{\mathbf{W}_1,\mathbf{W}_2\}$. Thus, we first adopt the ridge regression [Hoerl and Kennard, 2000] to solve $\mathbf{Z}$, and then use a gradient descent algorithm to train $\{\mathbf{W}_1,\mathbf{W}_2\}$ to approximate $\mathbf{Z}$. The iteration will lead to be time consuming because it needs to train $\{\mathbf{W}_1,\mathbf{W}_2\}$ many times after each updating $\mathbf{Z}$. Clearly, the high time cost is not beneficial to LSSC. Therefore, to avoid the iterative time cost, we use the ridge regression to solve $\mathbf{Z}$ without the last term of (6). The scheme is as follows:

Computing Z: $\mathbf{Z}$ is calculated by minimizing $\mathcal{L}$ without the last term of (6). By removing the last term, the problem (6) is written as the following subproblem:

$$\min_{\mathbf{Z}} \mathcal{L}_{\mathbf{Z}}=\|\mathbf{Y}-\mathbf{Y}\mathbf{Z}\|_{\mathrm{F}}^2+\alpha\|\mathbf{Z}\|_{\mathrm{F}}^2. \quad (7)$$

[1]The number of hidden units h is selected by depending on the approximating accuracy between $\mathbf{Z}$ and $\mathbf{W}_2 g(\mathbf{W}_1\mathbf{Y})$.

Algorithm 1 FRC via the gradient descent algorithm

```
1: input: data $\mathbf{Y}$, the number of hidden units $h$, maximal training epochs $T_m$, regularization parameters $\alpha,\beta,\gamma$, learning rate $\varepsilon$, and approximation accuracy $\epsilon$.
2: initialize: random initialization $\mathbf{W}_1$, and $t=1$.
3: compute $\mathbf{Z}^\star$ by $\mathbf{Z}^\star=\left(\mathbf{Y}^T\mathbf{Y}+\alpha\mathbf{I}\right)^{-1}\mathbf{Y}^T\mathbf{Y}$.
4: while $t<T_m$ and not converged do
5:     compute $\mathbf{H}$ by $\mathbf{H}=g(\mathbf{W}_1\mathbf{Y})$;
6:     update $\mathbf{W}_2$ by (9);
7:     update $\mathbf{W}_1$ by $\mathbf{W}_1=\mathbf{W}_1-\varepsilon\frac{\partial\mathcal{L}_{\mathbf{W}_1,\mathbf{W}_2}}{\partial\mathbf{W}_1}$, where $\frac{\partial\mathcal{L}_{\mathbf{W}_1,\mathbf{W}_2}}{\partial\mathbf{W}_1}$ is defined in (10);
8:     check the condition: $\|\mathbf{Z}^\star-\mathbf{W}_2 g(\mathbf{W}_1\mathbf{Y})\|_{\mathrm{F}}^2/m<\epsilon$.
9:     $t=t+1$;
10: end while
11: return solutions $\mathbf{W}_1$ and $\mathbf{W}_2$.
```

Clearly, this subproblem is easily solved by a ridge regression [Hoerl and Kennard, 2000].

Updating $\mathbf{W}_1$ and $\mathbf{W}_2$: $\mathbf{W}_1$ and $\mathbf{W}_2$ are calculated by minimizing $\mathcal{L}$ with respect to $\mathbf{W}_1$ and $\mathbf{W}_2$, when $\mathbf{Z}$ is fixed. In general, the Frobenius norm is used to prevent the over-fitting of weights in neural networks. So, by adding two regularization terms[2] $\|\mathbf{W}_1\|_{\mathrm{F}}^2$ and $\|\mathbf{W}_2\|_{\mathrm{F}}^2$, the optimal solution is to solve the following subproblem:

$$\min_{\mathbf{W}_1,\mathbf{W}_2} \mathcal{L}_{\mathbf{W}_1,\mathbf{W}_2}=\|\mathbf{Z}^\star-\mathbf{W}_2 g(\mathbf{W}_1\mathbf{Y})\|_{\mathrm{F}}^2+ \gamma(\|\mathbf{W}_1\|_{\mathrm{F}}^2+\|\mathbf{W}_2\|_{\mathrm{F}}^2), \quad (8)$$

where γ is a regularization parameter. The gradient descent algorithm is used to learn $\mathbf{W}_1$ and $\mathbf{W}_2$. Once the weight $\mathbf{W}_1$ is fixed, $\mathbf{H}=tanh(\mathbf{W}_1\mathbf{Y})$ is also determined uniquely. Then, solving $\mathbf{W}_2$ can be formulated as a convex optimization problem $\|\mathbf{Z}^\star-\mathbf{W}_2\mathbf{H}\|_{\mathrm{F}}^2$, which has a closed-form solution:

$$\mathbf{W}_2=(\mathbf{H}\mathbf{H}^T+\gamma\mathbf{I})^{-1}\mathbf{H}\left(\mathbf{Z}^\star\right)^T. \quad (9)$$

By using a gradient descent (GD) algorithm [Li *et al.*, 2015; Li *et al.*, 2016] to minimize the the least squares objective in (8), deriving the gradient of $\mathbf{W}_1$ obtains

$$\frac{\partial\mathcal{L}_{\mathbf{W}_1,\mathbf{W}_2}}{\partial\mathbf{W}_1}=2\mathbf{X}\Big[\left(\mathbf{1}-\mathbf{H}\circ\mathbf{H}\right)^T\circ \left(\mathbf{W}_2\mathbf{W}_2^T\mathbf{H}-\mathbf{W}_2\mathbf{Z}^\star\right)^T\Big]+\gamma\mathbf{W}_1, \quad (10)$$

where $\circ$ denotes element-wise multiplication, $\mathbf{1}$ is an identical matrix, and $\mathbf{1}-\mathbf{H}\circ\mathbf{H}$ is the gradient of $\mathbf{H}=tanh(\mathbf{W}_1\mathbf{Y})$. The complete optimization procedure of FRC is summarized in **Algorithm 1**, which includes the details of updating gradients and the convergence conditions.

Time Complexity and Convergence Analysis. The computational bottlenecks of FRC lies in the matrix inversion in step 3, of which the maximum complexity is $\mathcal{O}(m^3)$. Considering the cost of the gradient descent algorithm in steps

[2]In order to easy understand our main idea, we do not add the regularization into the main objective function (4) as this regularization is a common trick in neural networks.

Algorithm 2 Regression Coding Clustering.

```
1: input: large-scale data $\mathbf{X}$ with $k$ subspaces ;
2: select $\mathbf{Y}$ from $\mathbf{X}$;
3: solve $\mathbf{W}_1$ and $\mathbf{W}_2$ by FRC in Algorithm 1;
4: compute $\mathbf{Z_X}$ by $\mathbf{Z} = \mathbf{W}_2 g(\mathbf{W}_1\mathbf{X})$;
5: apply LSC to segment $\mathbf{X}$ into $k$ subspaces by $\mathbf{Z_X}$;
6: output: segmentation of the data: $\mathbf{X}^1, \mathbf{X}^2, \cdots, \mathbf{X}^k$.
```

Table 1: Computational complexities of the coding methods (SSC, LRR and LSR), and our FRC.

Mehtods	training time	coding time
FRC (ours)	$\mathcal{O}(m^3 + T_m(m^3 + dhm + hm^2))$	$\mathcal{O}(dhn + hmn)$
SSC [Elhamifar and Vidal, 2013]	$\mathcal{O}(0)$	$\mathcal{O}(t_1(d^2n^2 + dn^3))$
LRR [Liu *et al.*, 2013]	$\mathcal{O}(0)$	$\mathcal{O}(t_2(d^2n + n^3))$
LSR [Lu *et al.*, 2012]	$\mathcal{O}(0)$	$\mathcal{O}(n^3)$
n:♯ of samples; d:♯ of dimensionality of sample; m:♯ of the selected samples; h:♯ of hidden units of non-linear function; t_1, t_2, T_m:♯ of the number of iterations in l_1 solver, rank-minimizer, training the non-linear function; $\mathcal{O}(0)$: no training time.		

4-10 and the number of iterations needed to converge, the overall computational complexity of FRC is $\mathcal{O}(m^3 + T_m(m^3 + dhm + hm^2)))$, where $L \leq 10$ in this paper. The complexities of FRC and the coding methods (e.g. SSC, LRR, and LSR) are summarized in Table 1. In fact, the training complexity can be ignored due to $n \gg m$. So, our coding complexity $\mathcal{O}(mn)$ is *linear* in terms of n. Thus, our method is easily scalable for computing the codes in large-scale datasets. When $n = m$, we can directly use LSR to obtain the codes.

The theoretical convergence of FRC is difficultly guaranteed because the gradient descent (back-propagation, BP) algorithm cannot be proved to converge [Haykin, 2009]. Fortunately, BP have been widely used in many applications as they empirically converges well in general [Haykin, 2009]. The convergence is reached when the change in objective function is below a user-defined threshold $\epsilon = 10^{-4}$.

3.3 Clustering.

Our RCC procedure summarized in **Algorithm 2** is available for both large and small datasets. To large datasets (e.g. MNIST, and JCNORB), we select a small data $\mathbf{Y} \in \mathbb{R}^{d\times m}$ from $\mathbf{X} \in \mathbb{R}^{d\times n}$ to train the weights $\mathbf{W}_1$ and $\mathbf{W}_2$. After training $\mathbf{W}_1$ and $\mathbf{W}_2$ by FRC, the codes $\mathbf{Z_X}$ of $\mathbf{X}$ can be fast computed by $f(\mathbf{X}; \{\mathbf{W}_1, \mathbf{W}_2\})$ in (2). We cluster $\mathbf{Z_X}$ by employing LSC [Cai and Chen, 2015] to obtain the final results.

To small datasets (e.g. Extend-YaleB and AR), we select all data points $\mathbf{X}$ as $\mathbf{Y}$. Similar to the SSC method, spectral clustering is applied on the affinity matrix $\mathbf{C} = |\mathbf{Z_X}| + |\mathbf{Z_X}^T|$ to segment the data into k subspaces by NCuts [Shi and Malik, 2000]. Clearly, it cannot segment the large-scale datasets.

3.4 Theoretical Analysis.

To solve the LSSC problem, FRC trains the function (2) to approximate the codes learned by (5). In this subsection, we show an approximation condition and a grouping effect to verify that FRC is fit for large-scale subspace clustering.

Approximate the subspace representations by FRC. A sufficient condition is to show that FRC can effectively approximate the subspace representations.

Table 2: Databases.

Data set	# samples	Dimension	# classes
Extend-YaleB	2,414	32,256	38
AR	1,400	19,800	100
MNIST	70,000	784	10
JCNORB	349,920	2,048	6

Theorem 1[3]**.** *Assume a large-scale set of data points* $\boldsymbol{X}$ *drawn from* k *independent subspaces* $\{\mathcal{S}_i\}_{i=1}^k$ *with* r_i *bases, and a small data points* $\boldsymbol{Y}$ *drawn from the large data points* $\boldsymbol{X}$*, where* $rank(\boldsymbol{X}) = rank(\boldsymbol{Y}) = r$*. For any* $\boldsymbol{U}$ *drawn from* $\boldsymbol{X}$*, if* $\boldsymbol{U}$ *belongs to the neighbourhood of* $\boldsymbol{Y}$ *with radius* $\rho > 0$*, that is,* $\boldsymbol{U} \in \{\boldsymbol{P} \in \boldsymbol{X} | \|\boldsymbol{P} - \boldsymbol{Y}\|_F^2 < \rho\}$*, and*

$$\partial f(\mathbf{Y}; \theta)/\partial \mathbf{Y} = \mathbf{Y}^\dagger, \tag{11}$$

then we have $\|\boldsymbol{C}^\star - \boldsymbol{Z}\|_F^2 \leq o(\rho)$*, where* $\mathbf{Y}^\dagger$ *is the pseudoinverse of* $\boldsymbol{Y}$*,* $\mathbf{C}^\star$ *is the code of* $\boldsymbol{U}$ *learned by* (5)*,* $\boldsymbol{Z} = f(\boldsymbol{U}; \theta)$*,* $f(\cdot; \theta)$ *is a projective function (i.e.* (2)*), and* θ *is trained by the FRC problem* (3)*.*

Remark 3: The condition (11) is easily satisfied because $f(\cdot; \theta)$ is learned from the FRC problem (3). Hence, the projective codes can approximate to the subspace representations by a first-order accuracy as long as $\mathbf{U}$ drawn from $\mathbf{X}$ is belonging to the neighbourhood of $\mathbf{Y}$.

Grouping effect. The grouping effect means that the correlated data has the approximately equal coefficients, which implies these coefficients are easy to construct the similarity graph to divide the data for LSSC. We prove that FRC exhibits this grouping effect, which is stated as following:

Theorem 2. *Given a data vector* $\boldsymbol{x} \in \mathbb{R}^d$ *selected from large-scale data* $\boldsymbol{X}$*, a dictionary* $\boldsymbol{Y} \in \mathbb{R}^{d\times m}$ *and two parameters* α, β*. Assume each data point of* $\boldsymbol{Y}$ *is normalized and* $\boldsymbol{x} \in \boldsymbol{U}$*, where* $\boldsymbol{U} \in \{\boldsymbol{P} \in \boldsymbol{X} | \|\boldsymbol{P} - \boldsymbol{Y}\|_F^2 < \rho\}$*. Let* $\boldsymbol{z} = [z_1, \cdots, z_m] = f(\boldsymbol{x}; \theta)$ *be the non-linear code, where* θ *is trained by the FRC problem* (3)*. If the condition* (11) *is satisfied, then we have*

$$\sqrt{(z_i - z_j)^2} \leq \sqrt{2(1-\nu)}\|\mathbf{x}\|_2/\alpha + 2\sqrt{o(\rho)}. \tag{12}$$

where $\nu = \mathbf{y}_i^T \mathbf{y}_j$ *is the dictionary correlation.*

Remark 4: Theorem 2 shows that the solution is dependent correlation. If $\mathbf{y}_i$ and $\mathbf{y}_j$ are highly correlated (that is $\nu = 1$), then the difference between the coefficient of $\mathbf{y}_i$ and $\mathbf{y}_j$ tends to be 0 due to the higher-order infinitesimal $o(\rho)$. Hence, naturally, they will be grouped in the same cluster.

4 Experiments

In this section we evaluate our approach on four databases: Extended-YaleB[4], AR[5], MNIST[6], and JCNORB[7] in Table 2.

Datasets: Extended-YaleB contains 2,414 frontal face images of 38 people. There are about 64 images for each person. **AR** consists of over 4,000 color images of 126 people. Each

[3]The proofs of Theorems 1 and 2 are provided in https://www.researchgate.net/profile/Jun_Li7

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

[5]http://www2.ece.ohio-state.edu/ aleix/ARdatabase.html

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

[7]http://www.cs.nyu.edu/ ylclab/data/norb-v1.0/

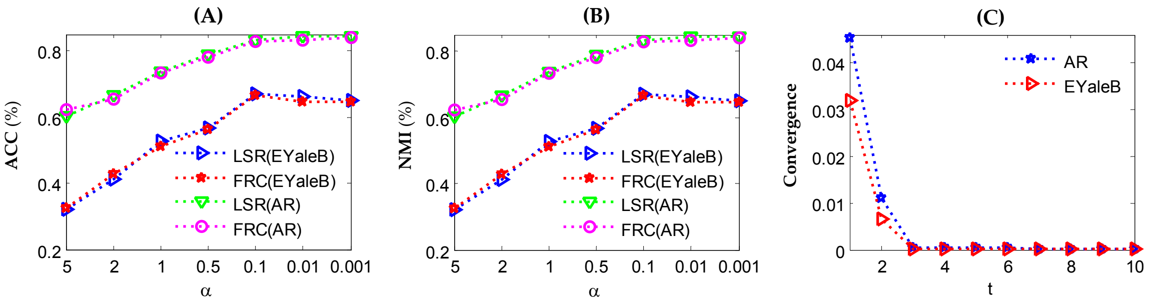

Figure 2: (A) Clustering accuracy (ACC) with different parameter α on AR and Extend-YaleB. (B) Normalized mutual information (NMI) with different parameter α on AR and Extend-YaleB. (C) Convergence of FRC on AR and Extended YaleB.

person has 26 face images taken during two sessions. We use a subset of the dataset consisting of 2,600 images from 50 male subjects and 50 female subjects. For computational efficiency, the original images of Extended-YaleB and AR were cropped and normalized to 48×42 and 55×40 pixels, respectively. Moreover, the features are reduced to 167 and 114 dimensions by PCA. **MNIST** has 70,000 examples with 28×28 pixel greyscale images of handwritten digits 0-9. **JCNORB** contains images of 50 toys belonging to a background and 5 generic categories: four-legged animals, human figures, airplanes, trucks, and cars. There are 349,920 training and testing examples with six classes. The original $2 \times 96 \times 96$ images were subsampled and scaled to $2 \times 32 \times 32$. The features are reduced to 500 dimensions by PCA.

4.1 Baselines and Evaluation Criterion.

The proposed clustering framework, RCC, employs LSC-K to segment the data points, based on the codes computed by our FRC. To evaluate the clustering performance of RCC, we compare our method with the classical coding (CCod) methods, the fast coding (FCod) methods, the large-scale spectral clustering (LSSpC) methods, and the *sampling, clustering* and *classification* (S-C-C), as well as the k-means [Cai, 2011]. Specifically, they are described as follow:

- *CCod* respectively uses LSR [Lu *et al.*, 2012], SSC [Elhamifar and Vidal, 2013], and LRR [Liu *et al.*, 2013] to calculate the codes of data samples, which are used to construct an affinity matrix, and employs NCuts to segment the data samples.
- *FCod* is a related method, which also learns a projective function to approximate the (sparse) codes, such as, Denoising autoencoders (DAE)[8] [Vincent *et al.*, 2010] and predictive sparse decomposition (PSD) [Kavukcuoglu *et al.*, 2008; Gregor and LeCun, 2010]. In reality, LSR with a small dictionary [Lu *et al.*, 2012] becomes a linear coding function shown in Remark 2. To compare these models, they are used to fast calculate the codes in our RCC framework, and also employ LSC-K [Cai and Chen, 2015] to cluster the data points.
- *LSSpC* directly uses the data points to construct an similarity matrix and segment the data points. Nyström [Chen *et al.*, 2011] finds an approximate eigendecomposition of the similarity matrix, and NyströmO [Chen *et al.*, 2011] is constrained in the orthogonal eigenvectors. LSR-R [Cai and Chen, 2015] randomly selects a few samples as the landmarks to compute the similarity matrix, while LSR-K [Cai and Chen, 2015] uses k-means to select the landmarks.
- *S-C-C* follows a strategy: *sampling, clustering* and *classification*. Specifically, select+SSC (selSSC) and select+LRR (selLRR) [Wang *et al.*, 2014] respectively use SSC and LRR to cluster the sampling data, and learn a linear classifier to segment the rest data, while SLSR, SSSC and SLRR [Peng *et al.*, 2016] respectively cluster the sampling data based on LSR, SSC and LRR, and use SRC or CRC to classify the rest data.

[8]We use the codes from https://github.com/kyunghyuncho/deepmat.

Table 3: ACC (%) and NMI (%) on Extend-YaleB and AR.

	Extended-YaleB		AR	
	ACC	NMI	ACC	NMI
RCC (ours)	66.5±1.74	70.5±0.79	82.8±0.80	91.1±0.36
DAE [Vincent *et al.*, 2010]	59.6±1.25	61.1±0.73	62.3±0.95	70.2±0.59
PSD [Gregor and LeCun, 2010]	72.2±0.87	73.0±0.61	80.8±1.05	90.5±0.44
LSR [Lu *et al.*, 2012]	67.0±1.17	70.7±0.51	**83.1±1.21**	**91.3±0.28**
SSC [Elhamifar and Vidal, 2013]	**76.5±1.22**	**78.4±0.35**	78.1±1.72	88.3±0.95
LRR [Liu *et al.*, 2013]	67.2±0.98	70.4±0.55	82.0±1.16	**91.3±0.51**
LSR-R [Cai and Chen, 2015]	43.4±2.08	55.2±0.87	33.9±1.02	62.0±0.49
LSR-K [Cai and Chen, 2015]	43.4±1.21	54.5±0.63	34.4±0.72	62.5±0.67
Nyström [Chen *et al.*, 2011]	24.5±1.16	44.2±1.02	61.8±2.40	83.0±0.90
NyströmO [Chen *et al.*, 2011]	21.5±0.98	41.4±1.05	57.6±2.20	79.8±1.22
k-means [Cai, 2011]	8.6±0.55	10.4±0.66	28.0±0.98	58.2±0.51

The clustering quality is measured by Clustering Accuracy (ACC) and Normalized Mutual Information (NMI) [Cai and Chen, 2015] between the produced clusters and the ground truth categories. ACC and NMI both range from 0 to 1, where 1 indicates perfect matching with the true subspace distribution, whereas 0 indicates totally mismatch. To obtain reliable results, each experiment is repeated 10 times, and the final results are reported by the mean and standard deviation of ACC and NMI.

In our FRC, there are three parameters ε, α, and β. The learning rate ε, as a typical parameter in neural networks, is set to 0.0001 in all experiments. Fig. 2 (A) and (B) show ACC and NMI with different α on AR and Extend-YaleB, and the best results is shown in $\alpha = 0.1$. We set $\alpha = 25$ and 250 respectively for MNIST and JCNORB to obtain the best results. Due to the limited space, we do not show the varying results with different α for these two datasets. The parameter β does not affect FRC as it is solved by minimizing $\mathcal{L}$ without the last term of (6). In all experiments, the number of hidden units h is set to 1000, the parameter γ is set to 0.0001, *tanh*

Table 4: Inference time (second) comparison between RCC and CCod on Extend-YaleB and AR.

	Extend-YaleB		AR	
RCC (ours)	training	coding	training	coding
	23	**0.4**	49	**0.2**
LSR [Lu *et al.*, 2012]	20		48	
SSC [Elhamifar and Vidal, 2013]	56		72	
LRR [Liu *et al.*, 2013]	41		63	

Table 5: All clustering times (second) compared RCC with S-C-C on MNIST, and JCNORB.

	MNIST	JCNORB
RCC (ours)	**30**	**262**
SLSR [Peng *et al.*, 2016]	541	1365
SLRR [Peng *et al.*, 2016]	638	2617
SSSC [Peng *et al.*, 2016]	613	2603
selSSC [Wang *et al.*, 2014]	469	3629
selLRR [Wang *et al.*, 2014]	1569	4489

is voted as the activation function, and the number of training epochs T_m is less than 10.

4.2 Comparison on Small-scale Datasets.

We verify that our approach (RCC) efficiently approximate the clustering results of the classical coding methods. When all samples are selected as training samples in Extended-YaleB and AR, the results are reported in Table 3. We observe that the performance of RCC is comparable to LSR as RCC trains a non-linear function to effectively approximate LSR. Moreover, RCC has a similar clustering result to other CCod methods, such as SSC and LRR. In addition, Table 3 also shows that CCod is much better than k-means and LSSpC (such as LSR-R, LSR-K, Nyström and NyströmO). Clearly, this observation drives us to apply the coding methods to fast compute the codes of the data samples for LSSC.

Table 4 shows that the coding time of RCC are 0.4, and 0.2 second in Extended-YaleB and AR, which are 50 times faster than LSR, SSC and LRR at least. Hence, RCC is able to apply for large-scale datasets when we randomly select the training samples. Moreover, Fig. 2 (C) does not only show the approximation accuracy between the codes and the non-linear function, but also illustrates that Algorithm 1 converges quickly. Although RCC spends some time to train the function, it can infer the codes in a rapid way.

4.3 Comparison on Large-scale Datasets.

It is worth noting that the CCod methods (LSR, LRR and SSC) have difficultly (or impossibly) to infer the codes in large-scale datasets with over 10,000 samples. Thus, we develop FRC to fast compute the codes for large-scale clustering. For large-scale datasets, 2000 and 2400 samples are respectively selected as the training data in MNIST, and JCNORB. The clustering times and results are reported in Table 5 and Table 6, respectively. We have the following observations:

Table 6 shows that our proposed RCC outperforms all the baseline methods in the scenario of large-scale datasets. Compared to the S-C-C methods (for example, SLSR, SLRR, SSS-C, selSSC, and selLRR), FRC has significantly better clustering performance, where we achieve at least 18.2% (MNIST), and 7.4% (JCNORB) improvement by ACC; and also reaches 15.2% (MNIST), and 8.8% (JCNORB) improvement by NMI.

Table 6: Clustering accuracy (ACC) (%) and normalized mutual information (NMI) (%) on MNIST and JCNORB.

	MNIST (2000)		JCNORB (2400)	
	ACC	NMI	ACC	NMI
FCod				
RCC (ours)	**73.6±3.79**	**69.6±2.15**	**32.6±1.00**	**15.4±0.66**
LSR [Lu *et al.*, 2012]	69.8±2.35	67.3±1.03	31.5±0.68	14.5±1.56
PSD [Gregor and LeCun, 2010]	50.1±1.79	43.7±0.38	29.3±2.85	9.7±3.06
DAE [Vincent *et al.*, 2010]	68.9±3.71	68.6±1.67	26.6±2.59	9.4±3.28
S-C-C				
SLSR [Peng *et al.*, 2016]	54.1±1.56	48.1±0.87	25.4±0.67	6.6±0.41
SLRR [Peng *et al.*, 2016]	50.0±3.87	49.1±2.27	22.9±0.22	5.0±0.27
SSSC [Peng *et al.*, 2016]	54.9±1.89	49.9±1.15	20.9±0.46	2.0±0.37
selSSC [Wang *et al.*, 2014]	55.2±1.63	54.4±2.19	20.6±0.67	2.1±0.69
selLRR [Wang *et al.*, 2014]	55.4±5.11	52.1±2.50	23.4±0.30	4.1±0.18
LSSpC				
LSR-R [Cai and Chen, 2015]	58.5±4.19	55.8±2.65	24.4±0.32	6.5±2.20
LSR-K [Cai and Chen, 2015]	68.3±4.99	67.7±2.29	24.6±0.36	7.8±0.40
Nyström [Chen *et al.*, 2011]	52.7±1.46	47.4±0.38	25.6±0.29	7.8±0.23
NyströmO[Chen *et al.*, 2011]	52.1±3.48	46.9±1.49	27.3±0.72	8.6±0.30
k-means [Cai, 2011]	55.3±0.06	52.6±0.19	23.1±0.00	5.3±0.00

The fundamental reason is that the linear classifier cannot handle the complex image data in the S-C-C methods. In contrast, our method trains the non-linear function to fast capture the excellent codes of the complex image data, and then perform LSC-K. Compared to LSSpC without coding (such as LSR-R, LSR-K, Nyström, and NyströmO), RCC still shows higher ACC and NMI as RCC can learn the excellent codes of the original data points. In addition, RCC is better and faster than DAE and PSD in the FCod methods. Besides, RCC is also better than LSR as RCC trains a non-linear function.

The overall clustering time consists of the training, coding, and clustering times. RCC is faster than S-C-C since RCC can very fast infer the codes by only computing a non-linear function. As shown in Table 5, RCC is round five times faster than SLSR, SLRR, SSSC, selLRR, and selSSC. On the other hand, although the LSSpC methods are faster than RCC, their clustering results perform worse than RCC. In fact, the time complexities of RCC is comparable to LSSpC as the number of data points increases. Moreover, RCC has a faster training time than other FCod methods.

5 Conclusion

To effectively solve the LSSC problem, we presented an efficient RCC framework: *sampling, FRC* and *LSC*. Firstly, a small dataset is (randomly) sampled from the large-scale data points due to the effectiveness of sampling. Secondly, based on fast self-expressiveness property, we proposed an FRC model to train a non-linear function from the sampled data for fast computing the codes of all the data points. Thirdly, the LSC method was employed to cluster the codes computed by the non-linear function. Besides, we theoretically proved that the non-linear function can approximate the codes learned from LSR by a first-order accuracy, and have a grouping effect that tends to group highly correlated data together. Extensive experimental results verified that our method can be successfully applied into the LSSC problem.

Acknowledgments

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

References

[Bengio *et al.*, 2013] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, Aug. 2013.

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

[Cai, 2011] Deng Cai. Litekmeans: the fastest matlab implementation of kmeans. In *Available at: http://www.zjucadcg.cn/dengcai/Data/Clustering.html*, 2011.

[Chen and Cai, 2011] Xinlei Chen and Deng Cai. Large scale spectral clustering with landmark-based representation. In *Proc. of the AAAI Conf. on Artif. Intell.*, pages 313–318, 2011.

[Chen *et al.*, 2011] Wen-Yen Chen, Yangqiu Song, Hongjie Bai, Chih-Jen Lin, and Edward Y. Chang. Parallel spectral clustering in distributed systems. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(3):568–586, 2011.

[Elhamifar and Vidal, 2013] Ehsan Elhamifar and Rene Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35:2765–2781, 2013.

[Fowlkes *et al.*, 2004] Charless Fowlkes, Serge Belongie, Fan Chung, and Jitendra Malik. Spectral grouping using the nystrom method. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(2):214–225, 2004.

[Gregor and LeCun, 2010] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proc. of Int. Conf. Mach. Learn.*, pages 399–406, 2010.

[Haykin, 2009] Simon Haykin. In *Neural Networks and Learning Machines*. Pearson Education Inc., 2009.

[Hoerl and Kennard, 2000] Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 42(1):80–86, 2000.

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

[Lee *et al.*, 2015] Minsik Lee, Jieun Lee, Hyeogjin Lee, and Nojun Kwak. Membership representation for detecting block-diagonal structure in low-rank or sparse subspace clustering. In *Proc. of IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1648–1656, 2015.

[Li and Vidal, 2015] Chun-Guang Li and Rene Vidal. Structured sparse subspace clustering: A unified optimization framework. In *Proc. of IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 277–286, 2015.

[Li *et al.*, 2015] Jun Li, Heyou Chang, and Jian Yang. Sparse deep stacking network for image classification. In *Proc. of the AAAI Conf. on Artif. Intell.*, pages 3804–3810, 2015.

[Li *et al.*, 2016] Jun Li, Yu Kong, Handong Zhao, Jian Yang, and Yun Fu. Learning fast low-rank projection for image classification. *IEEE Trans. Image Process.*, 25(10):4803–4814, 2016.

[Li *et al.*, 2017a] Jun Li, Yu Kong, and Yun Fu. Sparse subspace clustering by learning approximation ℓ_0 codes. In *Proc. of the AAAI Conf. on Artif. Intell.*, pages 2189–2195, 2017.

[Li *et al.*, 2017b] Jun Li, Tong Zhang, Wei Luo, Jian Yang, Xiaotong Yuan, and Jian Zhang. Sparseness analysis in the pertraining of deep neural networks. *IEEE Trans. Neural Netw. Learn. Syst.*, 28(6):1425–1438, 2017.

[Liu *et al.*, 2007] Ting Liu, Charles Rosenberg, and Henry A. Rowley. Clustering billions of images with large scale nearest neighbor search. In *Proc. of IEEE Workshop on App. of Comput. Vis.*, 2007.

[Liu *et al.*, 2013] Guangcan Liu, Zhouchen Lin, Shuicheng Yan, Ju Sun, Yong Yu, and Yi Ma. Robust recovery of subspace structures by low-rank representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35:171–184, 2013.

[Lu *et al.*, 2012] Can-Yi Lu, Hai Min, Zhong-Qiu Zhao, Lin Zhu, De-Shuang Huang, and Shuicheng Yan. Robust and efficient subspace segmentation via least squares regression. In *Proc. of Euro. Conf. Comput. Vis.*, pages 347–360, 2012.

[Nie *et al.*, 2011] Feiping Nie, Zinan Zeng, Ivor W. Tsang, Dong Xu, and Changshui Zhang. Spectral embedded clustering: A framework for in-sample and out-of-sample spectral clustering. *IEEE Trans. on Neural Netw.*, 22(11):1796–1808, 2011.

[Peng *et al.*, 2016] Xi Peng, Huajin Tang, Lei Zhang, Yi Zhang, and Shijie Xiao. A unified framework for representation-based subspace clustering of out-of-sample and large-scale data. *IEEE Trans. Neural Netw. Learn. Syst.*, 27(12):2499–2512, 2016.

[Shi and Malik, 2000] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, 2000.

[Talwalkar *et al.*, 2013] Ameet Talwalkar, Lester Mackey, Yadong Mu, Shih-Fu Chang, and Michael I. Jordan. Distributed low-rank subspace segmentation. In *Proc. of IEEE Int. Conf. Comput. Vis.*, pages 3543–3550, 2013.

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

[Vincent *et al.*, 2010] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, 2010.

[Wang *et al.*, 2014] Shusen Wang, Bojun Tu, Congfu Xu, and Zhihua Zhang. Exact subspace clustering in linear time. In *Proc. of the AAAI Conf. on Artif. Intell.*, pages 2113–2120, 2014.

[Xiao *et al.*, 2014] Shijie Xiao, Mingkui Tan, and Dong Xu. Weighted block-sparse low rank representation for face clustering in videos. In *Proc. of Euro. Conf. Comput. Vis.*, pages 123–138, 2014.

[Yan *et al.*, 2009] Donghui Yan, Ling Huang, and Michael I. Jordan. fast approximate spectral clustering. In *Proc. of ACM SIGKDD Int. Conf. Knowl. Dis. and Data Min.*, pages 907–916, 2009.

[Yin *et al.*, 2016] Ming Yin, Junbin Gao, and Zhouchen Lin. Laplacian regularized low-rank representation and its applications. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(3):504–517, 2016.

[Zhang *et al.*, 2015] Changqing Zhang, Huazhu Fu, Si Liu, Guangcan Liu, and Xiaochun Cao. Low-rank tensor constrained multiview subspace clustering. In *Proc. of IEEE Int. Conf. Comput. Vis.*, pages 1582–1590, 2015.

[Zhao and Fu, 2015a] Handong Zhao and Yun Fu. Dual-regularized multi-view outlier detection. In *Proc. of of Int. Joint Conf. on Artif. Intell.*, pages 4077–4083, 2015.

[Zhao and Fu, 2015b] Handong Zhao and Yun Fu. Semantic single video segmentation with robust graph representation. In *Proc. of of Int. Joint Conf. on Artif. Intell.*, pages 2219–2226, 2015.