Large-scale Subspace Clustering by Fast Regression Coding

Jun Li1, Handong Zhao1, Zhiqiang Tao1, and Yun Fu1,2

1Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, 02115, USA.
2College 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 non-linear function to approximate the codes. By using FRC, we develop an efficient Regression Coding Clustering (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 $X = \{X_1, \ldots, X_i, \ldots, X_k\} \in \mathbb{R}^{d \times n}$, where $d$ is the feature dimension, $k$ is the number of subspaces, $X_i = \{x_i^1, \ldots, x_i^r, \ldots, x_i^n\} \in \mathbb{R}^{d \times n_i}$ is drawn from the $i$-th subspace $S_i$, $n_i$ is the number of data points of $S_i$, 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 $S_i$. 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 [Benigo 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 non-linear function to approximate regression codes learned from
LSR. This function can quickly compute the codes. Based on FRC, we build an efficient Regression Coding Clustering (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 \( O(n) \) 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 \( O(n) \).

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 Regression Coding Clustering (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 \( \mathbf{Y} \in \mathbb{R}^{d \times m} \), drawn from the data matrix \( \mathbf{X} \in \mathbb{R}^{d \times n} \), in the \( i \)-th subspace \( \mathcal{S}_i \) (\( 1 \leq i \leq k \)), where \( m_i \) is the number of the samples of \( \mathbf{Y}_i \), and \( r_i < m_i \ll n_i \). Then \( \mathbf{Y} = [\mathbf{Y}_1, \ldots, \mathbf{Y}_i, \ldots, \mathbf{Y}_k] \) \((m = \sum_{i=1}^k m_i)\) is used as a small dictionary, and \( r = \sum_{i=1}^k r_i \ll m = \sum_{i=1}^k m_i < n = \sum_{i=1}^k n_i \). 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 \( \mathbf{X} \in \mathbb{R}^{d \times n} \), there exists a non-linear function \( f(\cdot ; \theta) \in \mathbb{R}^{m \times n} \) such that \( \mathbf{X} = \mathbf{Y} f(\mathbf{X}; \theta) \), where a small data matrix \( \mathbf{Y} \in \mathbb{R}^{d \times m} \) is drawn from \( \mathbf{X} \), and the parameter \( \theta \) is learned from the following system:

\[
\mathbf{Y} = \mathbf{YZ}, \quad \mathbf{Z} = f(\mathbf{Y}; \theta).
\]  

**Remark 1:** In general, the traditional coding methods consider \( Z_{ij} = 0 \) 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, \( Z_{ij} = 0 \) 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:

\[
\mathbf{Z} = f(\mathbf{Y}; \theta) = \mathbf{W}_2 g(\mathbf{W}_1 \mathbf{Y}),
\]  

where \( g(\cdot) \) is a non-linear 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 \( f(Y; \theta) \) by (2), therefore, we propose a FRC model defined as:

\[
\min_{Z, W_1, W_2} \|Z\|_F^2 \quad \text{s.t.} \quad Y = YZ, \quad Z = W_2g(W_1Y).
\] (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:

\[
\min_{Z, W_1, W_2} \|Y - YZ\|_F^2 + \alpha \|Z\|_F^2
\] s.t. \( Z = W_2g(W_1Y) \),

where \( \alpha \) is a parameter to balance the effects of two terms.

**Remark 2:** In fact, when \( Y \) is selected as the dictionary, LSR also can be considered as the following problem:

\[
C^* = \arg \min \|C\|_2^2 \quad \text{s.t.} \quad X = YC.
\] (5)

Usually, it has a solution \( C^* = (Y^TY + \alpha I)^{-1}Y^TX \), where \( \alpha \) 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:

\[
\min_{Z, W_1, W_2} \mathcal{L} = \|Y - YZ\|_F^2 + \alpha \|Z\|_F^2
\]

\[+ \beta \|Z - W_2g(W_1Y)\|_F^2, \] (6)

where \( \beta \) is a regularization parameter. The problem (6) is solved by iteratively updating the variable \( Z \), and the weights \([W_1, W_2]\). Thus, we first adopt the ridge regression [Hoerl and Kennard, 2000] to solve \( Z \), and then use a gradient descent algorithm to train \([W_1, W_2]\) to approximate \( Z \). The iteration will lead to be time consuming because it needs to train \([W_1, W_2]\) many times after each updating \( Z \). Clearly, the high time cost is not beneficial to LSSC. Therefore, to avoid the iterative time cost, we use the ridge regression to solve \( Z \) without the last term of (6). The scheme is as follows:

**Computing Z:** \( Z \) 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:

\[
\min_Z \mathcal{L}_Z = \|Y - YZ\|_F^2 + \alpha \|Z\|_F^2.
\] (7)

\(^1\)The number of hidden units \( h \) is selected by depending on the approximating accuracy between \( Z \) and \( W_2g(W_1Y) \).

### Algorithm 1 FRC via the gradient descent algorithm

1. **input:** data \( Y \), 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 \( W_1 \), and \( t = 1 \).
3. compute \( Z^* \) by \( Z^* = (Y^TY + \alpha I)^{-1}Y^TY \).
4. while \( t < T_m \) and not converged do
5. compute \( H \) by \( H = g(W_1Y) \);
6. update \( W_2 \) by (9);
7. update \( W_1 \) by \( W_1 = W_1 - \varepsilon \frac{\partial \mathcal{L}_{W_1, W_2}}{\partial W_1} \), where \( \frac{\partial \mathcal{L}_{W_1, W_2}}{\partial W_1} \) is defined in (10);
8. check the condition: \( \|Z^* - W_2g(W_1Y)\|_F^2/m < \epsilon \).
9. \( t = t + 1; \)
10. end while
11. return solutions \( W_1 \) and \( W_2 \).

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

**Updating \( W_1 \) and \( W_2 \):** \( W_1 \) and \( W_2 \) are calculated by minimizing \( \mathcal{L} \) with respect to \( W_1 \) and \( W_2 \), when \( Z \) 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\(^2\) \( \|W_1\|_F^2 \) and \( \|W_2\|_F^2 \), the optimal solution is solved by the following subproblem:

\[
\min_{W_1, W_2} \mathcal{L}_{W_1, W_2} = \|Z^* - W_2g(W_1Y)\|_F^2 + \gamma (\|W_1\|_F^2 + \|W_2\|_F^2),
\] (8)

where \( \gamma \) is a regularization parameter. The gradient descent algorithm is used to learn \( W_1 \) and \( W_2 \). Once the weight \( W_1 \) is fixed, \( H = \text{tanh}(W_1Y) \) is also determined uniquely. Then, solving \( W_2 \) can be formulated as a convex optimization problem \( \|Z^* - W_2H\|_F^2 \), which has a closed-form solution:

\[
W_2 = (HH^T + \gamma I)^{-1}H(Z^*)^T.
\] (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 \( W_1 \) obtains

\[
\frac{\partial \mathcal{L}_{W_1, W_2}}{\partial W_1} = 2X \left[ (1 - H \odot H)^T \odot \left(W_2W_2^T H - W_2Z^*)^T \right) + \gamma W_1 \right],
\] (10)

where \( \odot \) denotes element-wise multiplication, \( I \) is an identical matrix, and \( I - H \odot H \) is the gradient of \( H = \text{tanh}(W_1Y) \). 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 \( O(m^3) \). 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 X with k subspaces ;
2: select Y from X;
3: solve W_1 and W_2 by FRC in Algorithm 1;
4: compute Z_X by Z = W_2T(W_1X);
5: apply LSC to segment X into k subspaces by Z_X;
6: output: segmentation of the data: X_1, X_2, \ldots , X_k.

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

<table>
<thead>
<tr>
<th>Methods</th>
<th>training time</th>
<th>coding time</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRC (ours)</td>
<td>O(m^3 + T_m(m^3 + dhm + h^2n))</td>
<td>O(dhm + hmn)</td>
</tr>
<tr>
<td>SSC [Elhamifar and Vidal, 2013]</td>
<td>O(0)</td>
<td>O(t_1(d^2n^2 + dn^3))</td>
</tr>
<tr>
<td>LRR [Liu et al., 2013]</td>
<td>O(0)</td>
<td>O(t_2(d^2n + n^3))</td>
</tr>
<tr>
<td>LSR [Lu et al., 2012]</td>
<td>O(0)</td>
<td>O(n^3)</td>
</tr>
</tbody>
</table>

n: \% of samples; d: \% of dimensionality of sample; m: \% of selected samples; h: \% of hidden units of non-linear function; t_1, t_2, T_m, \% of the number of iterations in \% solver, rank minimizer, training the non-linear function, O(0): no training time.

4–10 and the number of iterations needed to converge, all computational complexity of FRC is O(m^3 + T_m(m^3 + dhm + h^2n)), 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 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 has 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 Y \( \in \mathbb{R}^{d \times m} \) from X \( \in \mathbb{R}^{d \times n} \) to train the weights W_1 and W_2. After training W_1 and W_2 by FRC, the codes Z_X of X can be fast computed by f(X; W_1, W_2) in (2). We cluster 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 X as Y. Similar to the SSC method, spectral clustering is applied on the affinity matrix C = \( |Z_X| + |Z_0_X| \) 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.

<table>
<thead>
<tr>
<th>Data set</th>
<th># samples</th>
<th>Dimension</th>
<th># classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extend-YaleB</td>
<td>2,414</td>
<td>32,256</td>
<td>38</td>
</tr>
<tr>
<td>AR</td>
<td>1,400</td>
<td>19,800</td>
<td>100</td>
</tr>
<tr>
<td>MNIST</td>
<td>70,000</td>
<td>784</td>
<td>10</td>
</tr>
<tr>
<td>JCNORB</td>
<td>349,920</td>
<td>2,048</td>
<td>6</td>
</tr>
</tbody>
</table>

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

\[
\partial f(Y; \theta)/\partial Y = Y^T, \tag{11}
\]

then we have \( \|C^* - Z\|_F^2 < \alpha \rho \), where Y^T is the pseudo-inverse of Y, C^* is the code of U learned by (5), Z = f(U; \theta), f(\cdot; \theta) is a projective function (i.e. (2)), and \theta 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 U drawn from X is belonging to the neighbourhood of 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 x \( \in \mathbb{R}^d \) selected from large-scale data X, a dictionary Y \( \in \mathbb{R}^{d \times m} \) and two parameters \( \alpha, \beta \). Assume each data point of Y is normalized and x \( \in U \), where U \( \in \{P \in \mathbb{R}^d | \|P - Y\|_F^2 < \rho \} \). Let z = [z_1, \ldots , z_m] = f(x; \theta) be the non-linear code, where \theta is trained by the FRC problem (3). If the condition (11) is satisfied, then we have

\[
\sqrt{(z_i - z_j)^2} \leq \sqrt{2(1 - \nu)} \|x\|_2/\alpha + 2\sqrt{\nu}(\rho), \tag{12}
\]

where \( \nu = y_y^T y_j \) is the dictionary correlation.

Remark 4: Theorem 2 shows that the solution is dependent correlation. If y_i and y_j are highly correlated (that is \( \nu = 1 \)), then the difference between the coefficient of y_i and y_j tends to be 0 due to the higher-order infinitesimal \( \alpha \rho \). 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

^3The proofs of Theorems 1 and 2 are provided in https://www.researchgate.net/profile/Jun_Li7
^4http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html
^5http://www2.ece.ohio-state.edu/~aleix/ARdatabase.html
^6http://yann.lecun.com/exdb/mnist/
^7http://www.cs.nyu.edu/ylclab/data/norb-v1.0/
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 × 96 × 96 images were subsampled and scaled to 2 × 32 × 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

---

\(^8\)We use the codes from https://github.com/kyunghyuncho/deepmat.
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 has much better than \( k \)-means and LSSpC (such as LSR-K, LSR-R, Nyström and Nyström). 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 difficulty (or impossibility) 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 results and times 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, SSSC, 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.

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 Nyström), 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 SLRR, SSC, 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 the 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


