When to Learn What: Deep Cognitive Subspace Clustering

Yangbangyan Jiang[1,2], Zhiyong Yang[1,2], Qianqian Xu[3], Xiaochun Cao[1,2], Qingming Huang[3,4,5*]
[1]State Key Laboratory of Information Security, Institute of Information Engineering, CAS, Beijing, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3]Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, CAS, Beijing, China
[4]School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing, China
[5]Key Laboratory of Big Data Mining and Knowledge Management, CAS, Beijing, China
{jiangyangbangyan,yangzhiyong,caoxiaochun}@iie.ac.cn,xuqianqian@ict.ac.cn,qmhuang@ucas.ac.cn

ABSTRACT

Subspace clustering aims at clustering data points drawn from a union of low-dimensional subspaces. Recently deep neural networks are introduced into this problem to improve both representation ability and precision for non-linear data. However, such models are sensitive to noise and outliers, since both difficult and easy samples are treated equally. On the contrary, in the human cognitive process, individuals tend to follow a learning paradigm from easy to hard and less to more. In other words, human beings always learn from simple concepts, then absorb more complicated ones gradually. Inspired by such learning scheme, in this paper, we propose a robust deep subspace clustering framework based on the principle of human cognitive process. Specifically, we measure the easinesses of samples dynamically so that our proposed method could gradually utilize instances from easy to more complex ones in a robust way. Meanwhile, a promising solution is designed to update the weights and parameters using an alternative optimization strategy, followed by a theoretical analysis to demonstrated the rationality of the proposed method. Experimental results on three popular benchmark datasets demonstrate the validity of the proposed method.

KEYWORDS

Subspace clustering; Self-paced learning; Deep learning

ACM Reference Format:
Yangbangyan Jiang, Zhiyong Yang, Qianqian Xu, Xiaochun Cao, Qingming Huang. 2018. When to Learn What: Deep Cognitive Subspace Clustering. In *MM '18: 2018 ACM Multimedia Conference, Oct. 22–26, 2018, Seoul, Republic of Korea.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3240508.3240582

1 INTRODUCTION

The emergence of subspace clustering presents us new possibilities to deal with more challenging tasks even in the absence of supervision. Recently, there arises a plethora of high-dimensional data in computer vision and multimedia systems, such as motion segmentation [33], foreground segmentation [11] and image clustering [40], where data points are often assumed to lie in a union of low-dimensional subspaces. In such a case, subspace clustering is well known as a solution. Unlike standard clustering methods, subspace clustering [6, 25, 30, 37] takes advantage of multi-subspace structure of data and thus can obtain better performance. Among existing subspace clustering methods, spectral clustering-based methods [6, 23, 32] have gained much attention in recent years. Such methods are essentially based on the hypothesis that a data point can be represented by a linear combination of other data points in the same subspace. Under this condition, we can find the representation under certain constraints and build an affinity matrix to recover the subspace structure.

It is common to assume that data points lie in linear or affine subspaces, yet in practice, a large amount of data are better modeled by non-linear ones. In this case, such linear methods are no longer suited. By applying kernel methods, linear models can be extended to handle non-linear data in subspace clustering [28, 34]. Nevertheless, an appropriate kernel must be selected empirically. Another approach to dealing with non-linear data is to use deep neural networks (DNNs), which have made a great progress in various areas such as computer vision [9], natural language processing [16] and multimedia processing [27] with their remarkable representation ability on large-scale dataset. Such methods reduce the clustering error significantly as proposed in [13]. However, existing DNN-based models are mainly limited by the non-convexity of their objective functions. This deficiency often causes such methods to stuck into bad local minima, especially in the presence of heavy noises and gross errors. This motivates us to investigate robust schemes for such methods.

However, in the aforementioned approaches, all the samples are learned without considering their difficulties. This may cause unstable learning dominated by hard samples either far away from the corresponding subspaces, or very close to subspaces they do not belong to, due to noises and corruptions. Moreover, this procedure is contrary to human cognitive process, where human often start learning from simple and easy concepts (e.g., natural numbers), then build up to complex and difficult ones (e.g., complex numbers). Inspired by this behavior, we can design an algorithm to measure the easiness of each sample and accordingly build an easy-to-hard learning sequence. During the early stage of learning, only easy instances are taught to the learner. Then harder samples are gradually included in learning according to the learner's ability. Such learning scheme, called *self-paced learning*, is proposed in [19] and

*Corresponding author.

then applied to alleviate the local minimum issue in various areas [15, 24, 36]. Furthermore, the combination of self-paced learning and deep networks has been proved to benefit from each other in [22], which provides a possible solution to our problem.

In this paper, we propose a *deep cognitive subspace clustering* (DeepCogSC) framework to improve the robustness of subspace clustering. In the core of the framework, each sample is dynamically assigned with a weight to reflect its easiness. And then according to these values, a DNN learns from easy samples to hard ones. Specifically, a deep auto-encoder is employed to map the data nonlinearly, inserted with a fully-connected layer to self-express. DeepCogSC aims at minimizing the reconstruction loss, self-expressive loss with the self-paced regularizer simultaneously. To solve this problem, we then implement an alternative optimization strategy which updates network parameters and weights iteratively. Besides, theoretical analysis and experimental results both rationalize and validate our proposed method.

The main contributions of this paper are listed as follows:

- We propose a novel deep cognitive subspace clustering approach. The proposed method benefits from both self-paced learning and deep learning, thus is more robust to noise and outliers than previous DNN-based methods.
- We conduct theoretical analysis to show some properties of the proposed method, by which we justify our method.
- We evaluate the proposed method on three benchmark datasets to demonstrate its effectiveness.

The rest of the paper is organized as follows. We first briefly review the related work of subspace clustering and self-paced learning in Section 2. Then we systematically introduce our proposed DeepCogSC algorithm, together with a theoretical analysis of the optimization procedure. After that, we present experimental results and the conclusion in the last two sections. Extensive experiment validation based on three datasets are demonstrated in Section 4. Finally, Section 5 gives the concluding remarks.

2 RELATED WORK

In this section, we briefly review the development of subspace clustering and self-paced learning.

2.1 Subspace Clustering

There are four main categories of subspace clustering methods: iterative, algebraic, statistical, and spectral clustering-based methods.

Iterative methods like K-subspaces [10] and median K-flats [37] repeat the step of dividing each point into a subspace and fitting subspaces to clusters alternatively. Algebraic methods [7, 25] build a similarity matrix by factorizing the data matrix and then construct a partition. Some statistical methods apply the expectation maximization algorithm to the data under the assumption that the data in subspaces subject to Gaussian distribution [8]. Some others are based on the information theory [30].

Spectral clustering-based methods first build an affinity matrix by estimating the similarity between each pair of data points and then perform spectral clustering [26] on it to obtain the partition for subspaces. The key step of such approaches is to build a favorable affinity matrix, where data points lying in the same subspace have higher similarities and vice versa. In practice, we find a "self-represent" coefficient matrix, by which the data can be represented using themselves, to construct this matrix. The coefficient matrix is often constrained by different regularization terms to preserve specific properties. To name a few, Liu et al. [23] proposes Low Rank Representation (LRR) which assumes data points are drawn from low-rank subspaces and uses the nuclear norm of the coefficient matrix as the constraint. Then some improvements on noisy LRR have been achieved like Low Rank Subspace Clustering [32]. Sparse Subspace Clustering (SSC) [6] is developed where the principle of sparsity is invoked, and thus ℓ_1 norm is utilized to ensure the sparsity of coefficients. Accordingly, many variants appear to improve its efficiency. For example, [35] implements orthogonal matching pursuit method to improve the computational efficiency of SSC. As for dense-connected graphs, Efficient Dense Subspace Clustering [12] formulates subspace clustering with the Frobenius norm to solve the problem. However, since these methods assume all of the subspaces are linear or affine, they do not apply to non-linear data. Hence, some non-linear models are proposed to deal with this problem. A typical kind of algorithms like Kernel Sparse Subspace Clustering [28] integrate the kernel trick with linear methods to address the problem. On the other hand, DNN is also implemented to introduce non-linear mappings for the data in [13]. Such methods improve the performance significantly but suffer from the local minimum problem.

2.2 Self-paced Learning

Self-paced learning (SPL) derives from curriculum learning [3], a human-like learning paradigm where the easier parts of the task are learned first, and the difficulty level gradually increases during the learning process. The difficulty of each sample is predefined manually and fixed in the learning process. In [19], Kumar et al. give the definition of self-paced learning, which learns from the easy samples to the complex ones. However, unlike curriculum learning, self-paced learning adjusts the sample difficulty dynamically. Instances of high-confidence are regarded as easy ones. In the learning process, only samples that are "easy" to learn for current model are included into training. This character of SPL eliminates the effect of noisy points and outliers, and also prevents the model from being stuck into bad local minima as much as possible. Therefore, we can improve the generalization ability of machine learning methods via SPL.

Choosing a good sample weighting scheme is the main concern for SPL. When first proposed in [19], SPL has a hard weighting scheme that divides samples only into two parts, easy or hard. To reflect the easinesses of samples, [14] develops some soft weighting schemes where these weights are assigned as real-valued numbers ranging from 0 to 1, such as linear weighting, logarithmic weighting, and mixture weighting. Moreover, Jiang et al. [15] introduce diversity into SPL by proposing a general regularizer which presents the preference for easy and diverse samples.

Because of the notable robustness and generalization ability, SPL is promoted in various machine learning applications. To name a few, [36] addresses co-saliency detection via the combination of multi-instance learning and SPL. By regarding the elements of a matrix as samples and assigning them with different weights,

[39] deals with matrix factorization problem in a self-paced manner. [29] integrates SPL into boosting to capture the intrinsic inner-class discriminative patterns with higher sample reliability. [21] develops a task-oriented self-paced regularizer for multi-task learning, which weights tasks and samples simultaneously. [24] performs self-paced co-training helping to reveal the easy-to-hard insights in the learning process and this method improves the performance of person re-identification. [38] conducts SPL for crowdsourcing quality control where evident samples make better efforts. SPL is integrated into a deep convolutional network to avoid local minima in [22]. Most of these studies integrate SPL paradigm into supervised machine learning models. However, there are few applications for SPL theory in unsupervised learning.

3 METHODOLOGY

In this section, we propose a deep cognitive learning framework for subspace clustering. Firstly, we introduce the preliminary, including notations, general self-paced learning paradigm and subspace clustering problem. Then the proposed method is detailed, together with the solution to this optimization problem. Finally, we conduct theoretical analysis to illustrate some properties of our method. Although the fundamental idea applies to data in any dimensions, we focus on subspace clustering on 2-D data in this paper.

3.1 Preliminary

Notation: In this paper, scalars, vectors and matrices are denoted as lowercase letters a, bold lowercase letters $\boldsymbol{a}$ and bold uppercase letters $\boldsymbol{A}$, respectively. $\boldsymbol{a}_i$ denotes the i-th column of the matrix $\boldsymbol{A}$. a_{ij} denotes the (i,j) element of $\boldsymbol{A}$. $\boldsymbol{I}$ denotes the identity matrix. $\|\cdot\|_p$ denotes the ℓ_p norm: $\|\boldsymbol{A}\|_p = (\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^p)^{1/p}$. And $\|\cdot\|_F$ denotes the Frobenius norm of a matrix: $\|\boldsymbol{A}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$. $Tr(\cdot)$ denotes the trace of a matrix. $diag(\cdot)$ denotes the diagonal vector of a matrix. n denotes the number of samples in a dataset. a^t and $\boldsymbol{A}^t$ denote the value of a and $\boldsymbol{A}$ in t-th iteration or stage.

Self-paced learning: Given a set of images $\mathcal{D} = \{\boldsymbol{X}_i\}_{i=1}^n$ where $\boldsymbol{X}_i \in \mathbb{R}^{w\times h}$, we assign each sample with a real-valued weight $v_i \in [0,1]$ to reflect its easiness (e.g., 0.01 means very hard). Let Θ denote the model's parameter and $L(\boldsymbol{X}_i;\Theta)$ denote the loss of $\boldsymbol{X}_i$. Self-paced learning aims at jointly learning the parameter Θ and the sample weights $\boldsymbol{v} = [v_1, v_2, \dots, v_n] \in \mathbb{R}^n$. The objective of SPL is a weighted sum of loss and weight regularizer of each sample. Thus we optimize SPL by the following minimization problem:

$$\min_{\Theta, \boldsymbol{v}\in[0,1]^n} E(\boldsymbol{v}, \Theta, \zeta) = \sum_{i=1}^{n} v_i L(\boldsymbol{X}_i;\Theta) + f(v_i, \zeta) \tag{1}$$

where the age parameter ζ controls the learning pace, and $f(v_i,\zeta)$ represents the self-paced regularizer. Note that ζ determines the number of samples selected in each learning stage and anneals gradually to involve more samples in training. At the beginning of training, ζ is small and only "simple" samples with small losses are selected for training. As the learning progresses, the increasing of ζ leads to more inclusion of "complex" samples, in which way the model learns from easy to hard.

The learning process of SPL complies with the cognitive process of human being, in which easy samples are learned and mastered first, and then difficult ones are learned. Learning from easy to hard ensures the model to capture the intrinsic patterns of samples with high confidence and avoids bad local minima. Therefore SPL is more robust to hard samples like noisy points and outliers than other methods. Consequently, we introduce SPL into subspace clustering to improve the performance.

Subspace clustering: Given a set of n data points $\mathcal{Z} = \{\boldsymbol{z}_i\}_{i=1}^n$ drawn from an unknown union of k low-dimensional subspaces $\mathcal{S} = \{S_i\}_{i=1}^k$ with their dimensions $\dim(S_i)$ unknown, the goal of subspace clustering is to divide them into k clusters, namely k subspaces correctly.

Let $\boldsymbol{Z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \dots, \boldsymbol{z}_n]$ be the data matrix. Observing the fact that each data point can be expressed as a linear combination of other points in the same subspace, we can derive the constraint of self-expressiveness $\boldsymbol{Z} = \boldsymbol{ZC}$, where $\boldsymbol{C}$ is the self-expressive coefficient matrix and its element c_{ij} indicates whether i-th and j-th point possess the membership of the same cluster. c_{ij} is expected to be zero if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ lie in different subspaces. In this way, it is desired that the coefficient matrix $\boldsymbol{C}$ has a block-diagonal form through proper permutation. The number of blocks is equivalent to the number of subspaces. In [12], it has been proved that this feature can be ensured by minimizing certain norms of $\boldsymbol{C}$ under the assumption that all the subspaces are independent, which can be formulated as the following optimization problem:

$$\min_{\boldsymbol{C}} \|\boldsymbol{C}\|_p \qquad s.t.\ \boldsymbol{Z} = \boldsymbol{ZC}, diag(\boldsymbol{C}) = \boldsymbol{0} \tag{2}$$

Specifically, the constraint $diag(\boldsymbol{C}) = \boldsymbol{0}$ is used to avoid the trivial solution that each data point is represented by itself, so that we can capture the global structure of the dataset. Meanwhile, the ℓ_p norm of $\boldsymbol{C}$ is exploited to preserve certain properties for subspaces.

Practically, we could relax (2) via replacing the self-expressive constraint as a penalty term in the objective function. Ultimately, the resulting problem becomes:

$$\min_{\boldsymbol{C}} \|\boldsymbol{C}\|_p + \frac{\lambda}{2}\|\boldsymbol{Z} - \boldsymbol{ZC}\|_F^2 \qquad s.t.\ diag(\boldsymbol{C}) = \boldsymbol{0} \tag{3}$$

where the hyperparameter λ controls the tradeoff between the two terms.

3.2 Model Formulation

The major issue of traditional subspace clustering methods is that most of them work under the assumption that all the subspaces are linear or affine. By contrast, a considerable quantity of data lies in non-linear subspaces in practice, and linear models do not apply well to them. With the help of deep learning, Ji et al. [13] has introduced effective non-linear transformation into clustering to address this problem. Nevertheless, such methods are still troubled by outliers and confusing data points. To make the framework more robust, we take a step further by performing deep subspace clustering in a self-paced manner.

Following the previous work, a deep neural network is utilized in our method. Specifically, as depicted in Figure 1, we integrate a self-expressive layer into a deep auto-encoder. In our model, input data is mapped onto a non-linear latent space, self-expressed in this space, and, again, mapped onto the original space. Let Θ_E, Θ_D, Θ_S denote the parameters of the encoder, decoder and self-expressive layer, respectively. With the three components given, we denote

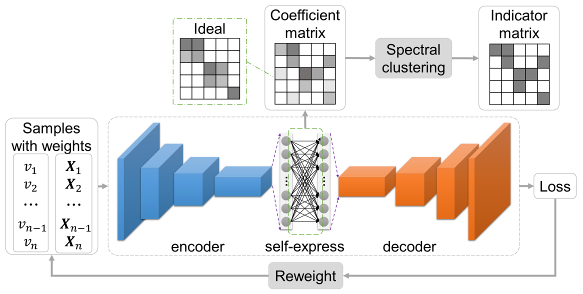


Figure 1: The framework of Deep Cognitive Subspace Clustering. Weighted samples are fed into the deep subspace clustering network, and then their losses are used to reweight the samples according to our weighting scheme. In the network, input data are mapped onto the latent space by the encoder, passed through a fully-connected layer to represent the data by itself, and finally reconstructed by the decoder.

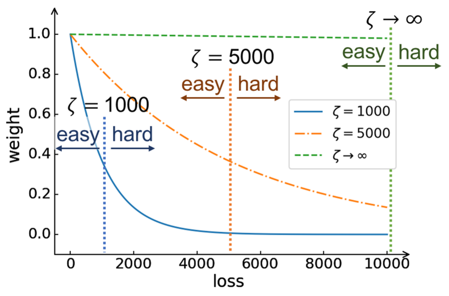


Figure 2: The change of weight v with respect to the age parameter ζ and loss ℓ. An instance is considered to be "simple" if its loss is less than ζ and "hard" if not. As ζ increases, more and more hard samples become "easy" for the model, and the weights for all the samples intend to be identical.

the whole parameter set as $\Theta = \left\{\Theta_E, \Theta_D, \Theta_S\right\}$. The roles played by these modules are described in the following.

Data Reconstruction with Θ_E and Θ_D. The deep convolutional auto-encoder performs a series of non-linear transformations to map the input data into the latent space through the encoder and tries to recover data from this space via the decoder. To guarantee the recovering performance, we minimize the reconstruction loss defined in the following:

$$\ell_{\text{rec},i} = \frac{1}{2}\|X_i - \hat{X}_i\|_F^2 \tag{4}$$

Latent Self-expressiveness with Θ_S. With the reconstruction performance guaranteed, we could assume that, after a series of highly non-linear transformations produced by the encoder, the data points eventually lie in a union of linear subspaces in the resulting latent space. Consequently, we pose the constraint of subspace clustering right after the encoder module. To be concrete, we replenish a fully-connected layer without bias and activation function after the encoder to leverage "self-expressiveness". Before fed into this layer, the encoder's output for X_i is flattened as a feature vector z_i. These vectors form a feature matrix Z. Furthermore, we model the parameter of this layer as the expression coefficient matrix $\Theta_S \in \mathbb{R}^{n\times n}$. Similar to Eq. (3), we formulate the self-expressive loss of each sample as follows:

$$\ell_{\exp,i} = \|z_i - Z\Theta_{S,i}\|_2^2 \tag{5}$$

where $\Theta_{S,i}$ denotes the i-th column of the coefficient matrix Θ_S.

Above all, it is easy to see that DeepCogSC should simultaneously guarantee the feature learning quality to obtain a correct linear latent space and a self-expressive structure to obtain a promising clustering performance. Therefore, the resulting loss function for a specific instance (*say for the i-th instance in training data*) thus becomes a weighted sum of the reconstruction loss and the self-expressive loss:

$$\ell_i(\Theta) = \ell_{\text{rec},i} + \frac{\lambda_1}{2}\ell_{\exp,i} \tag{6}$$

where the hyperparameter λ_1 controls the tradeoff of the two losses.

Based on the aforementioned clustering network, we weight the losses to adjust the contribution of samples to the final objective. Meanwhile, the self-paced regularizer $f(v_i,\zeta)$ is applied to each weight to control the learning pace of the model. Therefore, the self-paced loss for each sample is written as:

$$\mathcal{L}_i(\Theta,\zeta) = v_i\ell_i(\Theta) + f(v_i,\zeta) \tag{7}$$

Furthermore, we employ a regularization term to preserve certain structure of Θ_S:

$$\Omega(\Theta_S) = \|\Theta_S\|_p \tag{8}$$

Putting all these together, we eventually reach our objective for DeepCogSC as:

$$\begin{aligned} E(\boldsymbol{v},\Theta,\zeta) &= \sum_{i=1}^{n}\mathcal{L}_i(\Theta,\zeta) + \lambda_2\Omega(\Theta_S) \\ &= \sum_{i=1}^{n}\left[v_i\left(\frac{1}{2}\|X_i - \hat{X}_i\|_F^2 + \frac{\lambda_1}{2}\|z_i - Z\Theta_{S,i}\|_2^2\right)\right. \\ &\qquad \left. + f(v_i,\zeta)\right] + \lambda_2\|\Theta_S\|_p \end{aligned} \tag{9}$$

It is easy to see that the objective function above works generally for all possible self-paced regularizers $f(v,\zeta)$. Now we specify $E(\boldsymbol{v},\Theta,\zeta)$ via defining a novel regularizer in the following:

$$f(v,\zeta) = \zeta(v\log v - v) \tag{10}$$

Summing all the regularizers together, we have $\sum_{i=1}^{n} f(v_i,\zeta) = \sum_{i=1}^{n}\zeta(v_i\log v_i - v_i)$. Especially, this regularizer imposes a penalty on the weights to induce a cognition based learning process. At the beginning phase, the weights of hard samples are assigned to almost zero so that only easy samples could be learned. As the learning process goes on, hard samples are assigned with higher weights and gradually involved in training. Since all the samples are included eventually, we expect the weights to be non-zero and identical at the end of learning. The minimization of $\sum_i v_i\log v_i$, namely the maximization of entropy, could balance those different weights and lead to identical values finally. On the other hand, $-\sum_i v_i$ actually represents the minus ℓ_1 norm of v for v is not less than zero. Minimizing this term thus suppresses the sparsity of v and prevents merely including simple instances in training. Moreover, when the learner becomes sophisticated, we could increase ζ to accelerate this process. To see this, it is easy to find that, as ζ

increases, the growth of the weights of the hard examples becomes faster. To sum up, we illustrate the resulting process in Figure 2.

3.3 Optimization

Since the objective function is non-convex and intractable, we employ an alternative optimization strategy (AOS) to solve the problem. According to the optimization rules of AOS, the minimization of Eq. (9) can be solved via iteratively minimizing a part of parameters while keeping the other part fixed.

Update $\boldsymbol{v}$: First, we fix Θ at Θ^k, the parameter in k-th iteration. Ignoring those terms invariable to $\boldsymbol{v}$, the sub-minimization problem can be written as follows:

$$\min_{v_i \in [0,1]} \sum_{i=1}^{n} \mathcal{L}_i(\Theta^k, \zeta) = \min_{v_i \in [0,1]} \sum_{i=1}^{n} v_i \ell_i(\Theta^k) + f(v_i, \zeta) \tag{11}$$

Note that according to Eq. (10), each regularizer $f(v_i, \zeta)$ is convex with respect to v_i, consequently $\mathcal{L}_i(\Theta^k, \zeta)$ is convex with respect to v_i. The closed-form optimal solution of v_i can be deduced by:

$$\nabla_{v_i} \mathcal{L}_i(\Theta^k, \zeta) = \ell_i(\Theta^k) + \zeta \log v_i = 0 \tag{12}$$

Therefore we get the solution:

$$v_i^*(\Theta^k, \zeta) = e^{-\frac{\ell_i(\Theta^k)}{\zeta}} \tag{13}$$

For convenience, we simplify $v_i^*(\Theta^k, \zeta)$ as v_i^*. Ignoring the subscripts and superscripts, we denote $v^* = e^{-\ell/\zeta}$. Here, v^* is monotonically decreasing with respect to ℓ and it holds that $\lim_{\ell \to 0} v^* = 1$, $\lim_{\ell \to \infty} v^* = 0$. It is indicated that easy samples are often preferred by the model because of their smaller losses. On the other hand, v^* is monotonically increasing with respect to ζ and it holds that $\lim_{\zeta \to 0} v^* = 0$, $\lim_{\zeta \to \infty} v^* \leq 1$. As the "age" of the model grows, more hard samples are included into training.

Update ζ: Since the value of v^* depends on ζ, we should update ζ before v. In order to control the learning procedure explicitly, we predefine an ascending sequence $N = \{N_1, N_2, \ldots, N_m\}$ to set the number of selected samples in each training stage. Besides, to guarantee that all the samples are included in the last stage, we have $N_m = n$. For t-th stage, ζ is set as the N_t-th value of the loss sequence $\tilde{\ell}$ sorted in ascending order. Let ζ^t be the value of ζ in the t-th stage. Since ζ is increasing with respect to t, we expand the value of ζ by a factor $\tau \in (0, 1]$ and take the maximum value between them to update ζ as follows:

$$\zeta^{t+1} = \max\{\tilde{\ell}_{N_t}, (1+\tau)\zeta^t\} \tag{14}$$

Update Θ: Then, we fix v_i at v_i^*. Ignoring those terms invariable to Θ, the minimization subproblem becomes as follows:

$$\min \sum_{i=1}^{n} v_i^* \ell_i(\Theta) + \lambda_2 \|\Theta_S\|_p \tag{15}$$

Here we can update Θ through back-propagation algorithm. In practice, Θ can be updated using any classical gradient-based solver such as Stochastic Gradient Descent [2], Adam [18], etc.

On the other hand, it obviously holds that

$$\frac{\partial(v_i^* \ell_i(\Theta))}{\partial \Theta} = v_i^* \frac{\partial \ell_i(\Theta)}{\partial \Theta} \tag{16}$$

Hence in back-propagation, the weight v_i^* helps choosing the learning rates adaptively according to the sample easinesses.

After iterating these steps, we obtain the final coefficient matrix Θ_S. Using that, we can construct the affinity matrix $\boldsymbol{Z}$ as follows:

$$\boldsymbol{Z} = |\Theta_S| + |\Theta_S^\top| \tag{17}$$

Then we perform spectral clustering on $\boldsymbol{Z}$. In this problem, the input matrix is regarded as the adjacent matrix in a graph and the problem can be transformed into the minimization of normalized cut in this graph. Let $\boldsymbol{H}$ denote the indicator matrix where h_{ij} indicates whether the i-th sample belongs to the j-th cluster, the problem can be formulated as follows:

$$\begin{aligned} &\min_{\boldsymbol{H}} \; Tr\,(\boldsymbol{H}^\top \boldsymbol{L} \boldsymbol{H}) \\ &s.t. \; \boldsymbol{H}^\top \boldsymbol{D} \boldsymbol{H} = \boldsymbol{I} \end{aligned} \tag{18}$$

where $\boldsymbol{D} \in \mathbb{R}^{n \times n}$ is a diagonal matrix with $d_{ii} = \sum_j z_{ij}$, and $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{Z}$ is the Laplacian matrix for this graph.

To solve this problem, we first normalize the Laplacian matrix by $\tilde{\boldsymbol{L}} = \boldsymbol{D}^{-1/2} \boldsymbol{L} \boldsymbol{D}^{-1/2}$. Then we obtain its eigenvectors with k smallest eigenvalues by solving $\tilde{\boldsymbol{L}} \boldsymbol{x} = \zeta \boldsymbol{x}$. These eigenvectors are fed into a k-means algorithm [17] to derive the final group labels $\boldsymbol{y} = [y_1, \cdots, y_n]$.

Algorithm 1 Self-paced deep subspace clustering algorithm

```
Input: Dataset $\mathcal{D} = \{X_i\}_{i=1}^n$
Output: Parameters $\Theta$, clustering labels $y = [y_1, \cdots, y_n]$
 1: Initialize $\Theta$
 2: $N \leftarrow \{N_1, N_2, \ldots, N_m\}$
 3: $\zeta \leftarrow 0$
 4: for $t \leftarrow 1$ to $m$ do
 5:     Update $\zeta$ with Eq. (14)
 6:     while not converged do
 7:         Update each $v$ with Eq.(13)                          ▷ Fix $\Theta$
 8:         while not converged do
 9:             Update $\Theta$ with a gradient-based solver      ▷ Fix $v$
10:         end while
11:     end while
12: end for
13: $Z \leftarrow |\Theta_S| + |\Theta_S^\top|$
14: $y$ = SpectralClustering($Z$)
```

Our algorithm is shown in Algorithm 1. At first, we initialize the parameters in line 1–3. Then line 4–12 shows the AOS steps where ζ, v and Θ are updated alternatively until convergence. Note that, since Θ are updated by a gradient-based solver, we add another *while* loop (in line 8–10) inside the *for* loop to ensure that the model converges when $\boldsymbol{v}$ is fixed. Since v_i is directly solved by Eq (13), we do not explicitly initialize them. Finally, we obtain the clustering results from the adjacent matrix by spectral clustering in line 13–14.

3.4 Theoretical Analysis

Now in this subsection, we propose some important properties concerning our objective function which may shed light on the rationality of DeepCogSC.

Before analyzing the objective function, we first introduce two important variables: $F_\zeta(\ell)$ and $Q_\zeta(\Theta|\Theta^*)$. $F_\zeta(\ell)$ is defined as the integration of $v^*(\ell, \zeta)$ with respect to ℓ:

$$F_\zeta(\ell) = \int_0^\ell v^*(l,\zeta)\mathrm{d}l = \zeta(1 - e^{-\frac{\ell}{\zeta}}) \tag{19}$$

Then we define $Q_\zeta(\Theta|\Theta^*)$ as the first order expansion of $F_\zeta(\ell(\Theta))$ at $\ell(\Theta^*)$:

$$Q_\zeta(\Theta|\Theta^*) = F_\zeta(\ell_i(\Theta^*)) + v^*(\ell(\Theta^*),\zeta)\ (\ell(\Theta) - \ell(\Theta^*)) \tag{20}$$

According to (9), we see that our proposed objective function is hard to be analyzed, in the sense that both the self-paced weights v and the network parameters Θ should be solved. In the following proposition, we show that DeepCogSC is actually minimizing a much more simplified implicit objective function where the self-paced weights v are completely eliminated.

Proposition 1 (Implicit Objective Function). *With fixed ζ, the alternative optimization strategy for minimizing Eq. (9) is equivalent to the majorization-minimization algorithm for solving $\sum_{i=1}^n F_\zeta(\ell_i(\Theta)) + \lambda_2\|\Theta_S\|_p$.*

Proof Sketch. It can be proved that the following holds:

$$\begin{aligned} F_\zeta(\ell_i(\Theta)) &\le Q_\zeta^{(i)}(\Theta|\Theta^*) \\ &= F_\zeta(\ell_i(\Theta^*)) + v_i^*(\ell_i(\Theta^*),\zeta)\ (\ell_i(\Theta) - \ell_i(\Theta^*)) \end{aligned} \tag{21}$$

Hence $Q_\zeta^{(i)}(\Theta|\Theta^*)$ is a tractable surrogate function of $F_\zeta(\ell_i(\Theta))$. Then we could prove the equivalence by applying a majorization-minimization algorithm based on Eq. (21). Due to the limited space, we attach the detailed proof to the supplementary materials. □

According to (19), the proposition above implies that DeepCogSC minimizes an implicit loss function where v is eliminated. With the implicit loss revealed, we could now propose another proposition to show that DeepCogSC is robust toward hard examples. For the sake of simplicity, we denote $\ell_i(\Theta)$ as ℓ_i in the rest of this subsection.

Proposition 2 (robustness). *Suppose that $\min_k \ell_k > B$ and $B < \infty$, we have, for any pair of distinct instances (i,j) in training dataset $\mathcal{D}$:*

$$\left|F_\zeta(\ell_i) - F_\zeta(\ell_j)\right| \le e^{-\frac{B}{\zeta}} \cdot \left|\ell_i - \ell_j\right|$$

Proof. Let $a = \max\{\ell_i, \ell_j\}$, $b = \min\{\ell_i, \ell_j\}$. According to the mean value theorem, $\exists\ \xi \in [a,b]$, s.t. :

$$\frac{F_\zeta(\ell_i) - F_\zeta(\ell_j)}{\ell_i - \ell_j} = \frac{\partial F_\zeta(\ell)}{\partial l}\bigg|_{l=\xi} = e^{-\frac{\xi}{\zeta}} \tag{22}$$

Accordingly, we have:

$$\begin{aligned} \left|F_\zeta(\ell_i) - F_\zeta(\ell_j)\right| &\le \Big(\sup_{\ell\in[a,b]} \left|e^{-\frac{\ell}{\zeta}}\right|\Big) \cdot \left|\ell_i - \ell_j\right| \\ &\le e^{-\frac{B}{\zeta}} \cdot \left|\ell_i - \ell_j\right| \end{aligned} \tag{23}$$

□

By the virtue of Proposition 2, we see that, compared with $\ell(\cdot)$ (*i.e. the original loss*), $F_\zeta(\ell(\cdot))$ is more robust toward hard instances equipped with large loss. To see this, let i be a hard example and j an easy one. Then the loss difference $\left|F_\zeta(\ell_i) - F_\zeta(\ell_j)\right|$ in DeepCogSC is much smaller than the original loss difference $\left|\ell_i - \ell_j\right|$ (since $e^{-B/\zeta} < 1$). Accordingly, $F_\zeta(\ell(\cdot))$ is more robust in the sense that $F_\zeta(\ell(\cdot))$ is less sensitive toward large loss.

Finally, we show that when the learner grows sufficiently sophisticated, DeepCogSC tends to be consistent with the original learning paradigm where all the instances are treated equally.

With the Taylor expansion of $F_\zeta(\cdot)$, we have:

$$F_\zeta(\ell) = \zeta\left(1 - \sum_{n=0}^{\infty} \frac{\left(-\frac{\ell}{\zeta}\right)^n}{n!}\right) = \ell - o(\frac{1}{\zeta}) \tag{24}$$

From Algorithm 1, we know that ζ keeps increasing as the algorithm evolves. Since $\lim_{\zeta\to\infty} F_\zeta(\ell) = \ell$, it is easy to see that $F_\zeta(\ell)$ degenerates to the original ℓ after sufficient steps of iterations. At this time, all the samples are involved in training and they contribute to the learning process equally.

4 EXPERIMENTS

To evaluate the performance of our method, we conduct experiments on image datasets Extended Yale B, ORL, and COIL20. The competitors are listed as follows:

- Low Rank Representation (LRR) [23]: it assumes the data points are drawn from low-rank subspaces and utilizes the nuclear norm of the coefficient in the optimization problem.
- Low Rank Subspace Clustering (LRSC) [32]: a noisy variant for LRR.
- Sparse Subspace Clustering (SSC) [6]: the principle of sparsity is invoked, thus ℓ_1 norm is utilized to imply the sparse constraint.
- Kernel Sparse Subspace Clustering (KSSC) [28]: it introduces the nonlinear transformation into the problem through the kernel strategy.
- SSC by Orthogonal Matching Pursuit (SSC-OMP) [35]: Orthogonal matching pursuit method is employed in SSC.
- Efficient Dense Subspace Clustering (EDSC) [12]: it solves subspace clustering for dense-connected graphs and formulates it with the Frobenius norm.
- SSC with pre-trained convolutional auto-encoder features (AE+SSC): Features extracted by a pre-trained deep convolutional auto-encoder is used in SSC.
- EDSC with pre-trained convolutional auto-encoder features (AE+EDSC)
- Deep Subspace Clustering Network with ℓ_1 norm (DSC-Net-L1) [13]: A deep auto-encoder based network is used to obtain the coefficient matrix with the ℓ_1 norm regularization.
- Deep Subspace Clustering Network with ℓ_2 norm (DSC-Net-L2) [13]: DSC-Net with the ℓ_2 norm regularization.

All the results of these competitors are reported in [13]. Like DSC-Nets, we also employ our framework with ℓ_1 and ℓ_2 norm respectively using the same network architectures, which are denoted as DeepCogSC-L1 and DeepCogSC-L2, to demonstrate our performance under different constraints. The network structures are depicted in Table 1. We pre-train the network without the self-expressive layer and weighting scheme, and then finetune our models from the weights. Our methods are implemented with Tensorflow [1] and the code is run on a server with an NVIDIA Titan X GPU. Adam is used as the optimization method for DNN. Since

Table 1: Network structures for three datasets. We manifest "kernel size@channels" for conv and deconv layers, and the size of the coefficient matrix for fc layers.

	Extend Yale B	ORL	COIL20
	5×5@10	5×5@5	3×3@15
encoder	3×3@20	3×3@3	-
	3×3@30	3×3@3	-
fc	2432×2432	400×400	1440×1440
	3×3@30	3×3@3	3×3@15
decoder	3×3@20	3×3@3	-
	5×5@10	5×5@5	-

Table 2: Clustering error rate (%) on Extended Yale B. The bold value holds the lowest error rate or the second.

	10 subjects	20 subjects	30 subjects	all
LRR	22.22	30.23	37.98	34.87
LRSC	30.95	28.76	30.64	29.89
SSC	10.22	19.75	28.76	27.51
AE+SSC	17.06	18.23	19.99	25.33
KSSC	14.49	16.55	20.49	27.75
SSC-OMP	12.08	15.16	20.75	24.71
EDSC	5.64	9.30	11.24	11.64
AE+EDSC	5.46	7.67	11.56	12.66
DSC-Net-L1	2.23	2.17	2.63	3.33
DSC-Net-L2	**1.59**	**1.73**	2.07	2.67
DeepCogSC-L1 (Ours)	1.89	1.96	**1.93**	**2.38**
DeepCogSC-L2 (Ours)	**1.46**	**1.54**	**1.83**	**2.18**

the datasets are not that large, we use the full batch of the data as the input for our network.

We use *clustering error rate* as the evaluation metric in our experiments, which can be calculated as follows:

$$\text{err\%} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}[y_i \neq \hat{y}_i] \times 100\% \qquad (25)$$

where y_i is the cluster label predicted by spectral clustering and $\hat{y}_i$ is the corresponding ground truth.

4.1 Extended Yale B Dataset

Dataset description. The Extended Yale B dataset [20] is a popular dataset used for clustering. It contains 2432 grayscale face images of 38 subjects each seen under 64 illumination conditions. Each image has a size of 192×168.

Settings. The images are down-sampled to the size of 48×42 for preprocessing according to the experimental setup in [6]. To evaluate the performance with respect to increasing cluster numbers, we set the subject number K to 10, 20, 30, 38 and report their results. For computational convenience, we perform clustering on every sequence of continuous K subjects rather than all the combinations of K subjects, and then compute the average error rate.

For regularization parameters, we fix λ_1 at 1.0, τ at 0.15 and set λ_2 to $1.0 \times 10^{\frac{K}{10}-3}$. For the learning pace sequence, we set $N_1 = \lfloor \frac{n}{2} \rfloor$ and increase it by the number of images in a cluster for each step. Then we use grid search to tune other parameters for different K.

Results. The performance of all the methods is recorded in Table 2. It is shown that our method using ℓ_2 norm outperforms all the other competitors. We can see that deep subspace clustering network methods DSC-Net-L1 and DSC-Net-L2 achieve much lower error rate than other baselines. Yet our DeepCogSC-L1 could still outperform DSC-Net-L1, and likewise, DeepCogSC-L2 outperforms DSC-Net-L2, confirming that the introduction of self-paced learning paradigm could enhance the learning ability of subspace clustering. Similar as DSC-Net, the error rate of our model using ℓ_1 norm is slightly higher than the model using ℓ_2 norm, which is probably caused by the optimization difficulty of ℓ_1 norm. On the other hand, the clustering accuracy of our methods exceeds others' for each number of subjects. Hence it is demonstrated that the stability of the model to different numbers of subjects still holds after the self-paced paradigm is conducted. Specifically, we can observe that our method with ℓ_2 outperforms DSC-Net-L2 by 0.13% at 10 subjects, 0.19% at 20 subjects, 0.24% at 30 subjects and 0.49% at 38 subjects. The difference gets larger as the number of subjects increases, which implies that the larger the number of clusters gets, the more effective our method is.

4.2 ORL Dataset

Dataset description. The ORL dataset [31] consists of 400 face images collected from 40 subjects, each of which has 10 images taken at different times, lighting, and facial expressions. The image size is 112×92. This dataset is much smaller than Extended Yale B and has more variant conditions, accordingly is harder to cluster.

Settings. The images are down-sampled to the size of 32×32 for preprocessing like [5]. Unlike in Extended Yale B, we only evaluate this dataset for all 40 subjects. Considering that the dataset is small, we decrease the number of channels in our auto-encoder. We set the learning rate to 5.5×10^{-4} for our methods. Then we fix λ_1 at 1.0, λ_2 at 0.2 and τ at 0.15. The sequence $\{N_t\}$ is constructed like Section 4.1. The number of inner loops is 6. The number of outer loops is 4 for DeepCogSC-L1 and 6 for DeepCogSC-L2, respectively.

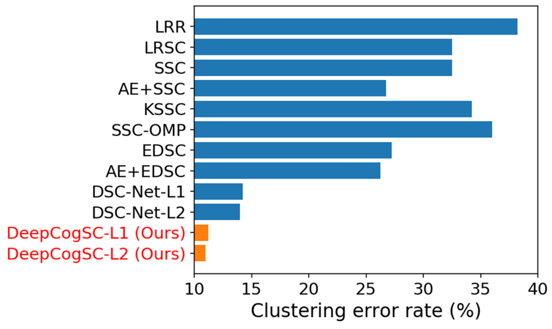
Results. We report the results for all the methods for ORL in Figure 3(a). Our DeepCogSC-L2 achieves lowest clustering error rate and DeepCogSC-L1 achieves second lowest. The performance of all the competitors on ORL is worse than that on Extend Yale B since ORL has a much smaller cluster size. Nevertheless, both our methods improve the performance of DSC-Net by 3% error rate, where the improvements of introducing self-paced learning paradigm become larger.

To show the strength of our method, the matrices $\boldsymbol{F}$ where $f_{ij} = \mathbb{I}[\hat{y}_i = \hat{y}_j]$ obtained by our DeepCogSC-L2 and DSC-Net-L2 are visualized in Figure 4. The block-diagonal structure is clearly exhibited in both Figure 4(a)(b). Moreover, comparing with Figure 4(b), there are less noisy points in Figure 4(a), indicating that the SPL improves deep subspace clustering effectively.

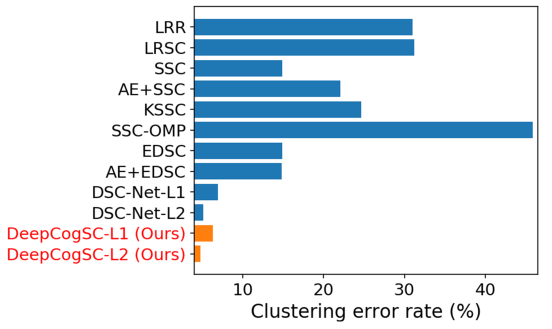
We then visualize the losses and assigned weights of samples at the first iteration and the fourth one. According to Figure 5, our method only focuses on easy samples at the beginning of the learning. As the training goes on and ζ increases, more and more hard instances get higher weights and are included in the learning.

4.3 COIL20 Dataset

Dataset description. Different from previous face datasets, the COIL20 dataset [31] consists of 1440 object images collected from



(a) ORL



(b) COIL20

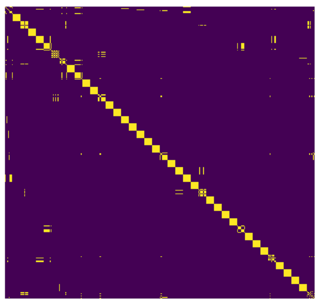
Figure 3: Clustering error rate (%) on ORL and COIL20. The smaller the better.

20 subjects, each of which has 72 images with the size of 128×128 taken at pose intervals of 5 degrees on a turntable. Note that various objects in COIL20 make this dataset more challenging to cluster.
Settings. Following the experimental setup in [4], we down-sample those images to the size of 32×32 for preprocessing. Same as ORL, we only evaluate this dataset for all the subjects.

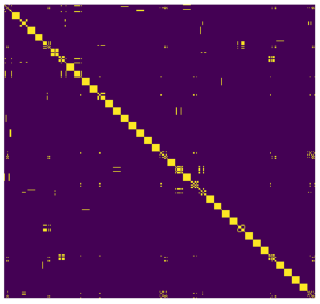
As depicted in Table 1, the network depth is decreased, while the number of channels becomes larger than that in ORL to obtain more features. The learning rate is set to 4.5×10^{-4}. We fix λ_1 at 1.0, λ_2 at 150 and τ at 0.25. The number of inner loops is 2. The number of outer loops is 2 for DeepCogSC-L1 and 3 for DeepCogSC-L2. Since the dataset is small, we construct the learning pace sequence by $N_1 = \lfloor \frac{3}{4} n \rfloor$ and $N_{t+1} = N_t + 36$.
Results. The clustering error rates on COIL20 are demonstrated in Figure 3(b). Our methods also achieve the lowest error rate. Specifically, the result of our DeepCogSC-L1 is 0.42% less than DSC-Net-L1. And our DeepCogSC-L2 decreases 0.25% error rate by DSC-Net-L2. Since the dataset is much challenging, the improvement obtained by SPL is less than on previous datasets.

5 CONCLUSIONS

In this paper, we introduce a novel deep subspace clustering method called Deep Cognitive Subspace Clustering (DeepCogSC). Inspired by the human cognitive process, DeepCogSC takes sample difficulty into consideration, where subspace clustering is performed in a self-paced manner based on a deep auto-encoder. An objective function is proposed thereafter which simultaneously considers the reconstruction performance of the auto-encoder and the clustering performance. Subsequently, we propose an alternative optimization strategy to solve the model parameters. Theoretical analysis shows that DeepCogSC is more robust toward hard examples and outliers. Moreover, DeepCogSC tends to be consistent with the original subspace clustering algorithm after sufficient steps of iterations. Furthermore, experiments are carried out on three benchmark datasets, Extend Yale B, ORL, and COIL20. The corresponding results demonstrate the superiority of DeepCogSC. Typically, we achieve up to 3% improvement with respect to the original deep subspace clustering method on ORL dataset.

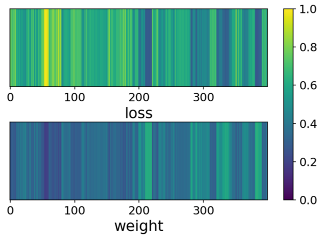


(a) DeepCogSC-L2

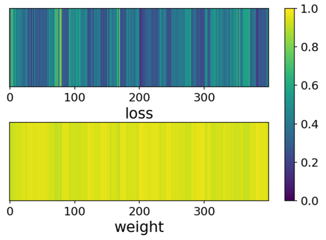


(b) DSC-Net-L2

Figure 4: Visualization of F on ORL. f_{ij} is 1 if i-th and j-th samples belong to the same class and 0 if not. The number of points clustered incorrectly in (a) is much less than (b).



(a) 1st iteration



(b) 4th iteration

Figure 5: Visualization of the weights and losses at 1st and 4th iteration. The losses are normalized.

ACKNOWLEDGMENTS

The research of Yangbangyan Jiang and Xiaochun Cao was supported by the National Key R&D Program of China (No.2016YFB08-00603), National Natural Science Foundation of China (No.61733007, 61650202), Beijing Natural Science Foundation (No.4172068). The research of Zhiyong Yang and Qingming Huang was supported in part by National Natural Science Foundation of China: 61332016, 61620106009 and U1636214, in part by National Basic Research Program of China (973 Program): 2015CB351800, in part by Key Research Program of Frontier Sciences, CAS: QYZDJ-SSW-SYS013. The research of Qianqian Xu was supported in part by National Natural Science Foundation of China (No.61672514, 61390514, 61572042), Beijing Natural Science Foundation (4182079), Youth Innovation Promotion Association CAS, and CCF-Tencent Open Research Fund.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A System for Large-Scale Machine Learning. In *USENIX Symposium on Operating Systems Design and Implementation*. 265–283.

[2] Pierre Baldi. 1995. Gradient descent learning algorithm overview: a general dynamical systems perspective. *IEEE Transactions on Neural Networks* 6, 1 (1995), 182–195.

[3] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *International Conference on Machine Learning*. 41–48.

[4] Deng Cai, Xiaofei He, Jiawei Han, and Thomas S. Huang. 2011. Graph Regularized Nonnegative Matrix Factorization for Data Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 8 (2011), 1548–1560.

[5] Deng Cai, Xiaofei He, Yuxiao Hu, Jiawei Han, and Thomas S. Huang. 2007. Learning a Spatially Smooth Subspace for Face Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[6] Ehsan Elhamifar and René Vidal. 2013. Sparse Subspace Clustering: Algorithm, Theory, and Applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 11 (2013), 2765–2781.

[7] C. William Gear. 1998. Multibody Grouping from Motion Images. *International Journal of Computer Vision* 29, 2 (1998), 133–150.

[8] Amit Gruber and Yair Weiss. 2004. Multibody Factorization with Uncertainty and Missing Data Using the EM Algorithm. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 707–714.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

[10] Jeffrey Ho, Ming-Hsuan Yang, Jongwoo Lim, Kuang-Chih Lee, and David J. Kriegman. 2003. Clustering Appearances of Objects Under Varying Illumination Conditions. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 11–18.

[11] Sajid Javed, Arif Mahmood, Thierry Bouwmans, and Soon Ki Jung. 2017. Background-Foreground Modeling Based on Spatiotemporal Sparse Subspace Clustering. *IEEE Transactions on Image Processing* 26, 12 (2017), 5840–5854.

[12] Pan Ji, Mathieu Salzmann, and Hongdong Li. 2014. Efficient dense subspace clustering. In *IEEE Winter Conference on Applications of Computer Vision*. 461–468.

[13] Pan Ji, Tong Zhang, Hongdong Li, Mathieu Salzmann, and Ian D. Reid. 2017. Deep Subspace Clustering Networks. In *Advances in Neural Information Processing Systems*. 23–32.

[14] Lu Jiang, Deyu Meng, Teruko Mitamura, and Alexander G. Hauptmann. 2014. Easy Samples First: Self-paced Reranking for Zero-Example Multimedia Search. In *ACM International Conference on Multimedia*. 547–556.

[15] Lu Jiang, Deyu Meng, Shoou-I Yu, Zhen-Zhong Lan, Shiguang Shan, and Alexander G. Hauptmann. 2014. Self-Paced Learning with Diversity. In *Advances in Neural Information Processing Systems*. 2078–2086.

[16] Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. *Transactions of the Association for Computational Linguistics* 5 (2017), 339–351.

[17] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. 2002. An Efficient k-Means Clustering Algorithm: Analysis and Implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24, 7 (2002), 881–892.

[18] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). arXiv:1412.6980 http://arxiv.org/abs/1412.6980

[19] M. Pawan Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-Paced Learning for Latent Variable Models. In *Advances in Neural Information Processing Systems*. 1189–1197.

[20] Kuang-Chih Lee, Jeffrey Ho, and David J. Kriegman. 2005. Acquiring Linear Subspaces for Face Recognition under Variable Lighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27, 5 (2005), 684–698.

[21] Changsheng Li, Junchi Yan, Fan Wei, Weishan Dong, Qingshan Liu, and Hongyuan Zha. 2017. Self-Paced Multi-Task Learning. In *AAAI Conference on Artificial Intelligence*. 2175–2181.

[22] Hao Li and Maoguo Gong. 2017. Self-paced Convolutional Neural Networks. In *International Joint Conference on Artificial Intelligence*. 2110–2116.

[23] Guangcan Liu, Zhouchen Lin, Shuicheng Yan, Ju Sun, Yong Yu, and Yi Ma. 2013. Robust Recovery of Subspace Structures by Low-Rank Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 1 (2013), 171–184.

[24] Fan Ma, Deyu Meng, Qi Xie, Zina Li, and Xuanyi Dong. 2017. Self-Paced Co-training. In *International Conference on Machine Learning*. 2275–2284.

[25] Yi Ma, Allen Y. Yang, Harm Derksen, and Robert M. Fossum. 2008. Estimation of Subspace Arrangements with Applications in Modeling and Segmenting Mixed Data. *SIAM Rev.* 50, 3 (2008), 413–458.

[26] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. 2001. On Spectral Clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*. 849–856.

[27] Lei Pang, Shiai Zhu, and Chong-Wah Ngo. 2015. Deep Multimodal Learning for Affective Analysis and Retrieval. *IEEE Transactions on Multimedia* 17, 11 (2015), 2008–2020.

[28] Vishal M. Patel and René Vidal. 2014. Kernel sparse subspace clustering. In *IEEE International Conference on Image Processing*. 2849–2853.

[29] Te Pi, Xi Li, Zhongfei Zhang, Deyu Meng, Fei Wu, Jun Xiao, and Yueting Zhuang. 2016. Self-Paced Boost Learning for Classification. In *International Joint Conference on Artificial Intelligence*. 1932–1938.

[30] Shankar R. Rao, Roberto Tron, René Vidal, and Yi Ma. 2010. Motion Segmentation in the Presence of Outlying, Incomplete, or Corrupted Trajectories. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32, 10 (2010), 1832–1845.

[31] Ferdinand Samaria and Andy Harter. 1994. Parameterisation of a stochastic model for human face identification. In *IEEE Workshop on Applications of Computer Vision*. 138–142.

[32] René Vidal and Paolo Favaro. 2014. Low rank subspace clustering (LRSC). *Pattern Recognition Letters* 43 (2014), 47–61.

[33] Guiyu Xia, Huaijiang Sun, Lei Feng, Guoqing Zhang, and Yazhou Liu. 2018. Human Motion Segmentation via Robust Kernel Sparse Subspace Clustering. *IEEE Transactions on Image Processing* 27, 1 (2018), 135–150.

[34] Ming Yin, Yi Guo, Junbin Gao, Zhaoshui He, and Shengli Xie. 2016. Kernel Sparse Subspace Clustering on Symmetric Positive Definite Manifolds. In *IEEE Conference on Computer Vision and Pattern Recognition*. 5157–5164.

[35] Chong You, Daniel P. Robinson, and René Vidal. 2016. Scalable Sparse Subspace Clustering by Orthogonal Matching Pursuit. In *IEEE Conference on Computer Vision and Pattern Recognition*. 3918–3927.

[36] Dingwen Zhang, Deyu Meng, Chao Li, Lu Jiang, Qian Zhao, and Junwei Han. 2015. A Self-Paced Multiple-Instance Learning Framework for Co-Saliency Detection. In *IEEE International Conference on Computer Vision*. 594–602.

[37] Teng Zhang, Arthur Szlam, and Gilad Lerman. 2009. Median k-flats for hybrid linear modeling with many outliers. In *IEEE International Conference on Computer Vision Workshops*. 234–241.

[38] Xianchao Zhang, Heng Shi, Yuangang Li, and Wenxin Liang. 2017. SPGLAD: A Self-paced Learning-Based Crowdsourcing Classification Model. In *Trends and Applications in Knowledge Discovery and Data Mining Workshops*. 189–201.

[39] Qian Zhao, Deyu Meng, Lu Jiang, Qi Xie, Zongben Xu, and Alexander G. Hauptmann. 2015. Self-Paced Learning for Matrix Factorization. In *AAAI Conference on Artificial Intelligence*. 3196–3202.

[40] Wencheng Zhu, Jiwen Lu, and Jie Zhou. 2017. Nonlinear subspace clustering for image clustering. *Pattern Recognition Letters* (2017).