Deep Embedded Subspace Clustering with Hard-Sample Mining

Li Zou[1], Tingting Leng[1], Rui Xie[1], Jiaxiong Liu[1] and Jun Zhou[1](✉)

[1] College of Computer and Information Science, Southwest University, China
zhouj@swu.edu.cn

Abstract. In recent years, subspace clustering has received increasing attention for its ability to accurately discover the underlying subspace in high-dimensional data. Among them, end-to-end subspace clustering methods compute cluster assignments by mapping points to subspaces. However, such methods ignore hard samples when computing the clustering assignment, i.e., low-value samples in the correct clustering assignment and high-value samples in the incorrect clustering assignment. To address this problem, in contrast to the previous instance-level hard samples, we mine hard samples at the subspace cluster-level. We first construct a deep embedded subspace clustering framework as the clustering target and learn subspace bases in iterations to obtain clustering assignments. Secondly, we utilize pseudo-supervised information and clustering assignments to mine hard samples at subspace cluster-level. Finally, a weight modulation strategy is proposed to dynamically focus the hard samples and obtain more accurate subspace clustering assignments. Through extensive experiments, we show that our method outperforms state-of-the-art subspace clustering algorithms on four benchmark datasets.

Keywords: Subspace Clustering, End To End Clustering, Hard-Sample Mining.

1. Introduction

Clustering is the process of achieving high intra-class similarity and low inter-class similarity by some measure of similarity in the absence of labelling information. Subspace clustering assumes that high-dimensional data tends to exist in low-dimensional structures. With the emergence of high dimensional data, subspace clustering is often proposed for clustering high dimensional data and is a practical approach in various clustering tasks such as motion segmentation, face clustering and image segmentation. Classical subspace clustering algorithms such [1,2,3] are based on self-expressive models. These algorithms are based on a two-stage framework. The first step in each iteration first learns an affinity matrix and the second step is to perform spectral clustering on the affinity matrix. These algorithms do not give fast clustering results and cannot be applied to large datasets.

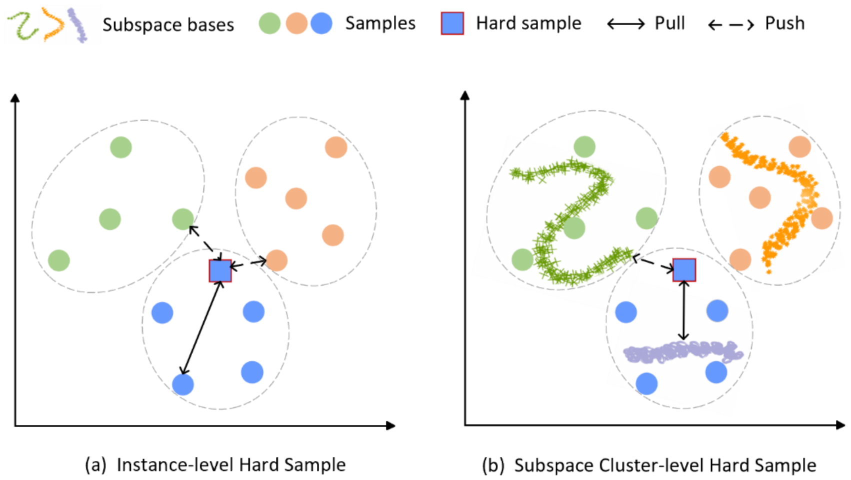

Subspace bases
Samples
Hard sample
Pull
Push
(a) Instance-level Hard Sample
(b) Subspace Cluster-level Hard Sample

Fig. 1. (a) Input samples are compared with hard-positive samples belonging to the same cluster and hard-negative samples from other clusters to learn to distinguish confusing samples. (b) Input samples are compared with the subspace bases to which they belong and with other subspace bases, with samples being more compact with the correct subspace and more distant from the other subspace bases, thus obtaining more discriminative cluster assignments.

There are now many subspace clustering methods that move away from the self-expressive framework [4,5,6] . SENet [5] uses an appropriately designed neural network to learn the self-expressive representation of the data. Jicong Fan et al. [4] which pursues structured sparsity in the matrix decomposition model by directly decomposing the data into K-groups. End-to-end deep embedded clustering [6,7,8] can move away from self-expressive models, e.g., EDESC [6] learns a set of subspace bases from extracted latent features based on end-to-end embedded clustering and optimizes the clustering objective together with the network. Although these algorithms enable end-to-end subspace clustering, none of them take hard samples into account.

Hard samples are considered an important part of many deep learning algorithms [9]. Hard samples can provide more information, by focusing more on hard samples rather than simple samples, more discriminative results can be obtained. In clustering, hard samples are generally used for the selection of positive and negative sample pairs for contrastive learning. However, all these methods mine hard samples at instance-level [10,11], ignoring the fact that hard positive and negative samples also exist at cluster-level.

Unlike previous instance-level hard sample mining, we perform hard sample mining at the subspace cluster-level. Fig. 1 illustrates the comparison between previous instance-level hard samples and our proposed subspace cluster level hard samples. In subspace clustering, since we are able to capture low-dimensional subspace bases, these low-dimensional subspace bases can approximately characterize subspaces in high-dimensional data [13]. Sample points and subspace bases can be assigned to clusters by some metric. Specifically we utilize the subspace bases to represent the clusters with the sample points to compute the cluster assignment probability values. The probability distribution between pseudo-supervised information and soft assignments is utilized to mine hard samples and dynamically adjust the weights, guided by high-confidence clustering results. Hard samples mining at subspace cluster-level makes the samples closer to the correct subspace and farther from the incorrect subspace. Compared to considering only hard positive and negative sample pairs at instance-level, we can capture the assignment of samples to clusters more accurately with hard sample mining at subspace cluster-level, which improves the subspace clustering performance. The main contributions of the proposed method in this paper are summarised as follows:

1. We construct an deep embedded subspace clustering framework that generates subspace bases through iterative refinement while obtaining soft assignments for clustering.
2. We propose a hard sample mining method at subspace cluster-level that combines pseudo-supervision infilormation and clustering soft assignment under high confidence guidance to discover hard samples.
3. We propose a sample weight modulation strategy to dynamically focus hard samples in a subspace clustering framework to enhance the discriminative power of subspace clustering assignment.
4. Experimental results are reported on four benchmark scale datasets to demonstrate the clustering performance of our method.

2. Related Works

2.1 End to End clustering

The end-to-end clustering method aggregates feature learning and clustering processes to obtain clustering results. DEC [7] is an important end-to-end deep embedding clustering method. It learns the embedding representation of the samples through an autoencoder and learns a set of prime centers to compute the soft assignments,and continuously optimizes the prime centers of the clusters to obtain the soft assigments. IDEC [8] integrates autoencoder reconstruction loss and clustering loss into a unified framework. JULE [14] uses the aggregated clustering result of an image as a supervised signal to learn the embedding representation, which in turn facilitates image clustering. DAC [15] treats pairs of images as a binary classification problem, where the cosine distance between image labeled features is used as a similarity, making the learned labeled features tend to be a hotspot vector. DSEC [16] Defines the clustering task as a binary pairwise classification problem to estimate whether the pairwise patterns are similar or not. All of these methods allow for fast end-to-end clustering, but all ignore the problem of hard samples.

2.2 Hard Sample Mining

In recent years, hard samples are commonly used in clustering tasks. Chuang et al. [16]proposed that hard negative samples of images should be considered, i.e., samples with higher similarity in the negative sample pairs. Xia. et al. [10] established a new measure suitable for negative samples by designing probabilistic ability estimators.

Jang. et al. [18] went beyond statistical methods to explore the link between hard negative samples and data bias. All these methods use hard samples for negative sample pairs. Liu. et al. [11] proposed that not only hard negative sample pairs should be considered, but also hard positive sample pairs, i.e., samples with lower similarity in the same category, should be carefully learned. All these methods mine hard samples at instance-level, we mine hard samples at subspace cluster-level in subspace clustering.

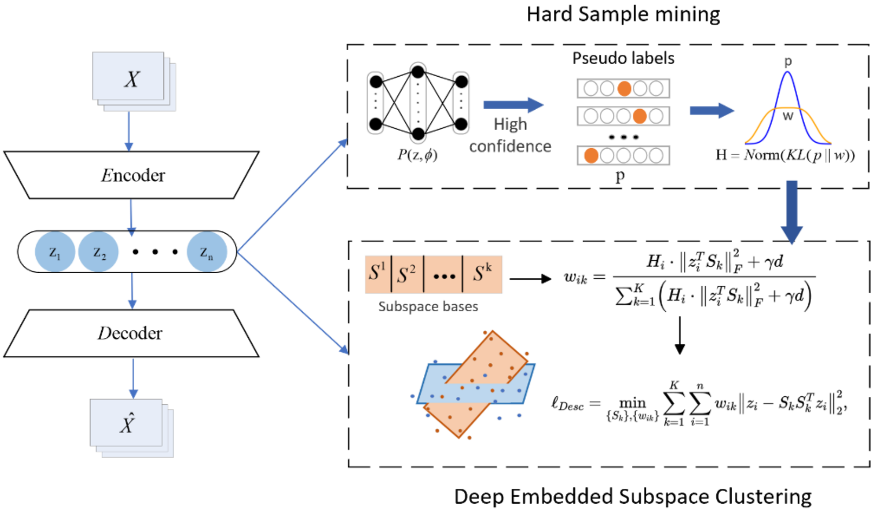

Hard Sample mining
X
Encoder
z_1
z_2
z_n
Decoder
$\hat{X}$
$P(z,\phi)$
High confidence
Pseudo labels
p
p
w
$H = Norm(KL(p \| w))$
S^1 S^2 ••• S^k
Subspace bases
$w_{ik} = \frac{H_i \cdot \|z_i^T S_k\|_F^2 + \gamma d}{\sum_{k=1}^K \left(H_i \cdot \|z_i^T S_k\|_F^2 + \gamma d\right)}$
$\ell_{Desc} = \min_{\{S_k\},\{w_{ik}\}} \sum_{k=1}^{K} \sum_{i=1}^{n} w_{ik} \|z_i - S_k S_k^T z_i\|_2^2,$
Deep Embedded Subspace Clustering

Fig. 2. Architecture of the proposed method. The input X are mapped by the encoder to Z, which is then reconstructed to $\hat{X}$ by the decoder. p is the output of the classifier module $P(z, \phi)$. w is the soft assignment of the deep embedded subspace clustering algorithm. H is a sample weight modulation function constructed by finding Kullback-Leibler Divergence between pseudo-labels and soft assignment.

3. Methods

In this section, we construct a deep embedded subspace clustering framework based on hard samples. The framework is shown in Fig. 2.

3.1 Feature Extractor

Autoencoders [19] are deep feature learning architectures commonly used for unsupervised tasks. We use an autoencoder as a feature extractor to extract features suitable for subspace clustering from the raw data. The encoder F and decoder G are parameterised by θ_e and θ_d respectively. The input to the feature extraction network is $X \in R^{n \times d}$, $\hat{X}_i = G_{\Theta_d}\left(F_{\Theta_e}(X_i)\right)$ is a reconstruction of X_i, specifically, AE can be optimised by the following objective function:

$$\ell_{ae} = \min_{\theta_e,\theta_d} \sum_{i=1}^{n} \| X_i - G_{\theta_e}\left(F_{\theta_e}\left(X_i\right)\right) \|_2^2 = \min_{\theta_e,\theta_d} \sum_{i=1}^{n} \| X_i - \hat{X}_i \|_2^2 \qquad (1)$$

3.2 Deep embedded subspace clustering

The problem of subspace clustering can be considered as follows:

$$\min_{\mathrm{W},\{\mathbf{z}_1,\ldots,\mathbf{z}_n\}} \frac{1}{2n}\sum_{i=1}^{n} \| \mathbf{x}_i - h_{\mathrm{W}}\left(\mathbf{z}_i\right) \|^2, \quad \mathbf{z}_i \in \mathrm{S}_i, i = 1,\ldots,n. \tag{2}$$

where h_W denotes a multilayer neural network with a parameter set W, n denotes the number of samples and S_i denotes the real clusters to which X_i belongs, Z_i is an embedded representation that X_i has undergone an autoencoder. However, it is not possible to solve this problem directly due to the unknown S_i. Now we introduces a new variable $S = \left[\mathbf{S}_{(1)}, \mathbf{S}_{(2)}, \ldots, \mathbf{S}_{(k)}\right]$ which contains k blocks, $S_{(j)} \in \mathbb{R}^{p\times d}$, and $\|S^u_{(j)}\| = 1, u = 1,\ldots,d$, $j = 1,\ldots,k$. For all $\mathrm{j} \neq l$, $\|S^T_{(j)} S_{(l)}\|_F$ should be small enough in order to ensure that there is no similarity between the different subspaces:

$$\| S^T_{(j)} S_{(l)} \|_F \leq \tau, \quad j \neq l \tag{3}$$

where τ is a small constant. Each sample Z_i is assigned to a subspace $S_{(j)}$, Z_i is highly correlated with only one block of S.

$$\left\| \mathbf{z}_i^{\text{晻}} \mathbf{S}_{(\alpha_i)} \right\| \gg \max_{j \neq \alpha_j} \left\| \mathbf{z}_i \ \mathbf{S}_{(j)} \right\|, \quad i = 1,\ldots,n. \tag{4}$$

The goal of subspace clustering is to learn the optimal k subspaces on the set of samples and assign them to the nearest subspace. Unlike previous two-stage subspace clustering algorithms, we fuse the computation of high-dimensional subspace bases and the assignment of samples to the k subspaces in a single framework using an iterative refinement:

$$\ell_{Desc} = \min_{\{S_k\},\{w_{ik}\}} \sum_{k=1}^{K}\sum_{i=1}^{n} w_{ik} \left\| z_i - S_k S_k^T z_i \right\|_2^2 \tag{5}$$

$$s.t.\ S_k^T S_k = I,\ w_{ik} \in \{0,1\}\ and \sum_{k=1}^{K} w_{ik} = 1$$

where $S_k \in R^{d\times q}$, $k \in \{1,\ldots,K\}$ is the k-th subspace basis, $w \in R^{n\times \mathrm{K}}$, w_{ik} denotes the assignment of x_i to S_k. H is the weight modulation function we introduce. In this paper, We use k-means clustering to initialise the subspace base $\{S_1,\ldots,S_k\}$, defining w_{ik} as the probability value, samples can be assigned to the nearest subspace by the following equation:

$$w_{ik} = \frac{H_i \cdot \left\| z_i^T S_k \right\|_F^2 + \gamma d}{\sum_{k=1}^{K}\left(H_i \cdot \left\| z_i^T S_k \right\|_F^2 + \gamma d \right)} \tag{6}$$

where γ is the parameter controlling the smoothness, d is the dimension of the subspace, K is the number of clusters. Thus w_{ik} denotes the probability that the embedding vector z_i belongs to the k-th subspace S_k, which will be used as the result of our deep embedded subspace clustering. H helps us to keep adjusting the sample weights during the iteration process so as to focus the clustering target on the samples that are hard to separate at subspace level. We write the constraints of (3) and (4) on the subspace basis as loss terms as follows:

$$\ell_{\text{basis}} = \left\| \mathbf{S}^{\bullet} \mathbf{S} \odot \mathbf{I} - \mathbf{I} \right\|_F^2 + \left\| S^T S \odot O \right\|_F^2 \tag{7}$$

where $\odot$ represents the Hadamard product, and I is an identity matrix of size kd by kd. O is a matrix in which all d-size diagonal block elements are zeros and all others are ones.

When training is complete, the final clustering labels can be obtained in the following ways:

$$y_{pred}^{(i)} = \arg\max_k w_{ik} \tag{8}$$

3.3 Hard-Sample Mining

After obtaining the latent representations from the feature extractor, we can compute the pseudo-labels by introducing a classification layer designed as a fully connected layer on top of the feature extractor module:

$$p_i = \arg\max_k \left[P_\phi\left(z_i\right) \right]_k \tag{9}$$

where $P_\phi(\cdot)$ denotes the transformation of the fully connected layer, which can produce one-hot pseudo-labels.

However, it is not possible to directly solve for the optimal one-hot pseudo-labels due to its non-convexity. To avoid obtaining a mundane solution, we select high-confidence pseudo-labels to participate in the subsequent mining of hard samples by setting a threshold for the probability [20]. We do this by setting a threshold σ for the probability $p_i = \max\left[P_\phi\left(z_i\right)\right]_k$. Samples above the threshold are used as the high-confidence sample set V. Unlike the comparative clustering approach of finding pairs of positive and negative hard samples [11,21], in this paper we try to mine hard samples by finding Kullback-Leibler Divergence between the soft assignment probability distribution of the subspace and pseudo-labels distribution guided by high confidence. The difference between the two probability distributions will be higher for hard samples than for simple samples. We propose a weight modulation function H to dynamically adjust the weights of sample during the training process. H is formulated as follows:

$$H(i) = \begin{cases} 1, & \neg\left(p_i, w_i \in \mathbf{V}\right), \\ \text{Norm}\left(KL\left(p_i \| \, w_i\right)\right), & \text{others} \end{cases} \tag{10}$$

where p_i is the clustering pseudo labels of the i-th sample, $p \in R^{n \times k}$. w_i is the clustering soft assignment of the i-th sample, $w \in R^{n \times k}$. $Norm$ denotes the min-max normalization. Specifically, when neither distribution satisfies high confidence, H keeps the sample weights constant. When both distributions satisfy high confidence, H dynamically adjusts the sample weights for the next iteration, up-weight to hard sample while down-weighting the easy ones.

3.4 Training Settings

To train our network, we propose a two-stage training strategy. The first stage is pre-training using only to initialize the network. The second stage fine-tunes the entire network by (11), where λ_1, λ_2 are tuning parameters. The training flows of the proposed method is presented in Algorithm 1.

$$\ell = \ell_{\text{ae}} + \lambda_1 \ell_{\text{basis}} + \lambda_2 \ell_{\text{Desc}} \tag{11}$$

Table 1. Detailed information of the benchmark datasets

Dataset name	Total samples	Classes	Size
Fashion-MNIST	70,000	10	28×28
CIFAR-10	6,0000	10	32×32×3
REUTERS-10K	10,000	4	2,000
STL-10	1,3000	10	96×96×3

Table 2. Clustering performance compared with the baseline methods. Note that the best results are marked in bold.

	Fashion-MNIST		CIFAR-10		REUTERS-10K		STL-10	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
DEC	0.590	0.601	0.301	0.257	0.618	0.314	0.359	0.276
IDEC	0.592	0.604	0.316	0.273	0.684	0.351	0.378	0.324
JULE	0.563	0.608	0.272	0.192	0.626	0.405	0.277	0.182
DAC	0.615	0.632	0.522	0.396	-	-	0.470	0.366
DSEC	-	-	0.477	0.437	0.783	0.708	0.481	0.403
k-SCN	0.600	0.623	0.601	0.515	0.801	0.598	-	-
k-FSC	0.727	0.692	-	-	0.798	0.573	0.759	0671
EDESC	0.631	0.670	0.627	0.464	0.825	0.611	0.745	0.687
Ours	**0.731**	**0.693**	**0.672**	**0.610**	**0.833**	**0.634**	**0.751**	**0.689**

4. EXPERIMENTATS

4.1 Datasets and Evaluation Metrics

To evaluate the clustering performance of our method, we conduct experiments on four widely used benchmark datasets. Including three image datasets (Fashion-MNIST, CIFAR-10, STL-10) and one text dataset (REUTERS-10K).The details of the datasets are given in Table 1.We quantify the clustering performance using clustering accuracy ACC, normalized mutual information NMI and adjusted rand index ARI.

4.2 Experiment Settings

In the experiments, for Fashion-MNIST and REUTERS-10K, the encoder and decoder consist of fully connected networks of m-500-500-1000 and 1000-500-500-m. For CIFAR-10 and STL-10, we applied ResNet50 [22] to extract their 2048-dimensional features. We set $\gamma = 5$ for all datasets. We use the Adam optimizer [23] to minimize the objective function and set the learning rate to 0.001. k-means algorithm is used to to initialize the subspace.

4.3 Comparisons with Baseline Methods

End-to-end clustering methods are used as comparison methods, including DEC[7], IDEC[8], JULE[1], DAC[14], DSEC[15], kSCN [23], k-FSC[4], EDESC[6]). The results of our method and the compared methods are listed in Table 2. From the Table 2, we can see that our method significantly outperforms other baselines on the four datasets. For example, on the Fashion-MNIST dataset, the method proposed in this paper improves 1.4% and 0.6% on ACC and NMI, respectively, compared to k-FSC. On the text dataset Reuters-10K we outperform the recent work EDESC, where we improve 0.8% and 2.3% on ACC and NMI, respectively. On the large datasets CIFAR-10 and STL-10, our method similarly outperforms other methods. The reason why our method outperforms other end-to-end subspace clustering methods is that we perform hard sample mining to obtain better cluster assignments.

Table 3. Results of ablation experiments on two benchmark datasets.

	CIFAR-10		REUTERS-10K	
	ACC	NMI	ACC	NMI
$w/o\ \ell$	0.672	0.610	0.833	0.634
$w/o\ \ell_{ae}$	0.610	0.479	0.771	0.539
$w/o\ \ell_{Desc}$	0.560	0.379	0.752	0.474
$w/o\ \ell_{basis}$	0.654	0.603	0.829	0.590

4.4 Ablation Study

In this section, we conduct an ablation study to explore the impact of each loss term in the proposed method on the clustering performance, and we construct four degenerate models through the remaining loss terms. Since l_{Desc} is an indispensable prerequisite for clustering, we remove only the sample weight modulation function H in w_{ik}. We conducted ablation experiments on the CIFAR-10 and REUTERS-10K datasets. Table 3 summarizes the experimental results of the ablation study. Firstly, the reconstruction loss can maintain data structure information, which is a necessary part of deep embedded clustering. Secondly, the sample weight modulation function plays an important role to handle hard samples. Finally the constraints on the subspace bases further improve the clustering performance.

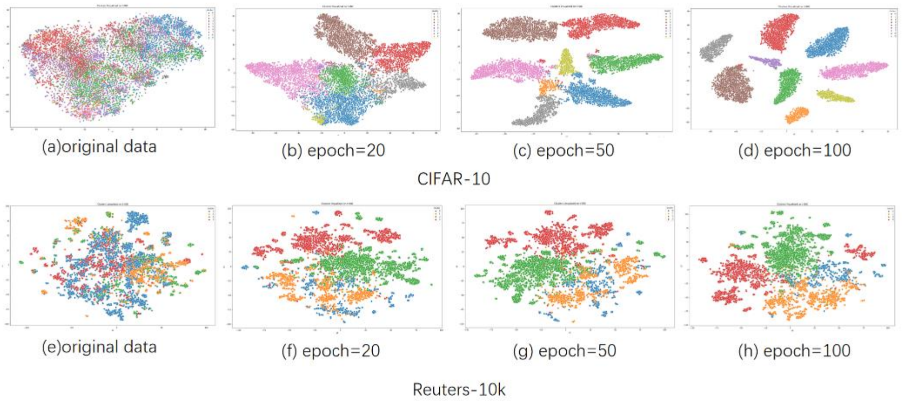


Fig. 3. 2D Visualization of the embedded representations using t-SNE on CIFAR-10 and REUTERS-10K.

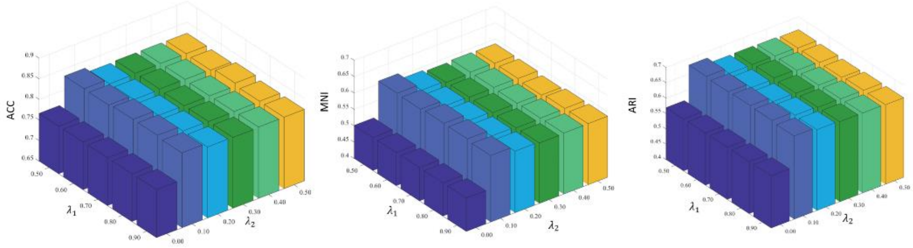


Fig. 4. Model sensitivity varies with the two hyperparameters λ1,λ2. The x-axis and y-axis denote λ1,λ2, from left to right z-axis are ACC,NMI,ARI.

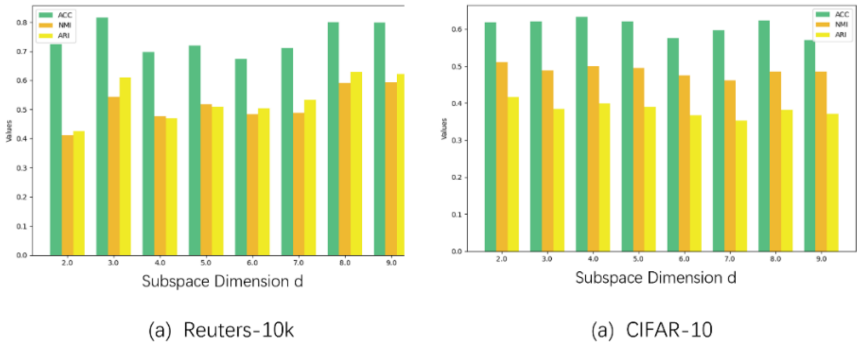
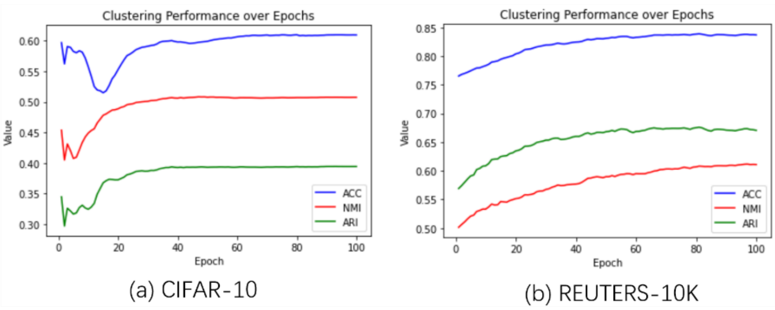


Fig. 5. Parameter sensitivity of subspace dimension d of the proposed method on REUTERS-10K and CIFAR-10.



(a) CIFAR-10 (b) REUTERS-10K

Fig. 6. Online clustering performance on two datasets.

4.5 Qualitative Study

The first row of Fig. 3 shows the t-SNE visualization of the embedding representation learned from 10,000 images randomly sampled on the real-world dataset CIFAR-10. The second row of Fig. 3 shows the t-SNE visualization of the embedding representation of REUTERS-10K. We show the embedding representation at different stages, and it can be observed that similar samples are discriminatively assigned to a group during the training process, and the clustering results become increasingly clear.

4.6 Parameter Sensitiveness Analysis

We tested the sensitivity of the hyperparameters λ_1 and λ_2 in the method on Reuters-10K. We search for λ_1 from $\{0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ and λ_2 from $\{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$. The results from the grid search are shown in Fig. 4. We can observe that our method is not sensitive to the hyperparameter λ_1 and it is easy to set in real applications, we set $\lambda_1 = 1$. For λ_2, the clustering accuracy first rises and then there is a slow decrease, so we set λ_2 to 0.1.

In addition, we also tested the effect of subspace dimension d on clustering. We conducted experiments on Reuters-10K and CIFAR-10, respectively, and the results are shown in Fig. 5, which shows that the subspace dimensionality has an effect on clustering. For Reuters-10K, the best ACC and ARI are achieved at d =3. For CIFAR-10, we achieve the best performance at d =4. Good performance can also be achieved when d =8, 9, but there will be a large expenses, so in this paper we set d =3 on Reuters-10K, d =4 on CIFAR-10.

4.7 Online Clustering

In this section, we conducted online clustering experiments on CIFAR-10 and REUTERS-10K. Fig. 6 records the results of ACC, NMI, and ARI changes during the training process. We can observe that the REUTERS-10K results in a continuous steady increase before reaching convergence around the 80th round. While CIFAR-10 will show an increase and then a rapid decrease due to the randomness of the k-means algorithm at the initialization, and then gradually increase to reach the convergence state during the training process.

5. Conclusion

In this paper, we construct a deep embedded subspace clustering framework with hard sample mining. We construct subspace bases to obtain clustering assignments and combine them with a sample weight modulation strategy to solve the hard sample problem of deeply embedded clustering. Experimental results are reported on four benchmark-sized datasets to demonstrate the clustering performance and efficiency of our approach.

References

1. Xu, Y., Chen, S., Li, J., & Qian, J. (2022). Linearity-Aware Subspace Clustering. AAAI, 8770–8778.
2. Peng, Z., Liu, H., Jia, Y., & Hou, J. (2022). Adaptive attribute and structure subspace clustering network. IEEE Transactions on Image Processing, 31, 3430–3439.
3. Qin, Y., Zhang, X., Shen, L., & Feng, G. (2022). Maximum block energy guided robust subspace clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(2), 2652–2659.
4. Fan, J. (2021). Large-Scale Subspace Clustering via k-Factorization. KDD, 342–352.
5. Zhang, S., You, C., Vidal, R., & Li, C.-G. (2021). Learning a self-expressive network for subspace clustering. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 12393–12403.
6. Cai, J., Fan, J., Guo, W., Wang, S., Zhang, Y., & Zhang, Z. (2022). Efficient deep embedded subspace clustering. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 1–10.
7. Xie, J., Girshick, R. B., & Farhadi, A. (2016). Unsupervised Deep Embedding for Clustering Analysis. ICML, 48, 478–487.
8. Guo, X., Gao, L., Liu, X., & Yin, J. (2017). Improved deep embedded clustering with local structure preservation. Ijcai, 17, 1753–1759.
9. Wu, Z., Xiong, Y., Yu, S. X., & Lin, D. (2018). Unsupervised feature learning via non-parametric instance discrimination. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 3733–3742.
10. Xia, W., Wang, Q., Gao, Q., Yang, M., & Gao, X. (2022). Self-consistent contrastive attributed graph clustering with pseudo-label prompt. IEEE Transactions on Multimedia.
11. Liu, Y., Yang, X., Zhou, S., Liu, X., Wang, Z., Liang, K., Tu, W., Li, L., Duan, J., & Chen, C. (2023). Hard sample aware network for contrastive deep graph clustering. Proceedings of the AAAI Conference on Artificial Intelligence, 37(7), 8914–8922.
12. Qu, W., Xiu, X., Chen, H., & Kong, L. (2023). A survey on high-dimensional subspace clustering. Mathematics, 11(2), 436.
13. Yang, J., Parikh, D., & Batra, D. (2016). Joint Unsupervised Learning of Deep Representations and Image Clusters. CVPR, 5147–5156.
14. Chang, J., Wang, L., Meng, G., Xiang, S., & Pan, C. (2017). Deep Adaptive Image Clustering. ICCV, 5880–5888.
15. Chang, J., Meng, G., Wang, L., Xiang, S., & Pan, C. (2020). Deep Self-Evolution Clustering. IEEE Trans. Pattern Anal. Mach. Intell., 42(4), 809–823.
16. Chuang, C.-Y., Robinson, J., Lin, Y.-C., Torralba, A., & Jegelka, S. (2020). Debiased contrastive learning. Advances in Neural Information Processing Systems, 33, 8765–8775.

17. Jang, T., & Wang, X. (2023). Difficulty-Based Sampling for Debiased Contrastive Representation Learning. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 24039–24048.
18. Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2006). Greedy Layer-Wise Training of Deep Networks. NIPS, 153–160.
19. Lv, J., Kang, Z., Lu, X., & Xu, Z. (2021). Pseudo-supervised deep subspace clustering. IEEE Transactions on Image Processing, 30, 5252–5263.
20. Hu, Z., Zhu, C., & He, G. (2021). Hard-sample guided hybrid contrast learning for unsupervised person re-identification. 2021 7th IEEE International Conference on Network Intelligence and Digital Content (IC-NIDC), 91–95.
21. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 770–778.
22. Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. ICLR (Poster).
23. Zhang, T., Ji, P., Harandi, M., Hartley, R., & Reid, I. (2019). Scalable deep k-subspace clustering. Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part V 14, 466–481.