GAN-based clustering solution generation and fusion of diffusion

Cao, Wenming; Yu, Zhiwen; Wong, Hau-San

Published in:
Systems Science and Control Engineering

Published: 01/01/2022

Document Version:
Final Published version, also known as Publisher's PDF, Publisher's Final version or Version of Record

License:
CC BY

Publication record in CityU Scholars:
Go to record

Published version (DOI):
10.1080/21642583.2021.2024100

Publication details:
GAN-based clustering solution generation and fusion of diffusion

Wenming Cao, Zhiwen Yu & Hau-San Wong

To cite this article: Wenming Cao, Zhiwen Yu & Hau-San Wong (2022) GAN-based clustering solution generation and fusion of diffusion, Systems Science & Control Engineering, 10:1, 24-42, DOI: 10.1080/21642583.2021.2024100

To link to this article: https://doi.org/10.1080/21642583.2021.2024100

© 2022 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.

Published online: 12 Jan 2022.

Submit your article to this journal

Article views: 350

View related articles

View Crossmark data
GAN-based clustering solution generation and fusion of diffusion

Wenming Cao\textsuperscript{a}, Zhiwen Yu\textsuperscript{b} and Hau-San Wong\textsuperscript{c}

\textsuperscript{a}Department of Information and Computer Science, Chongqing Jiaotong University, Chongqing, People’s Republic of China; \textsuperscript{b}School of Computer Science and Engineering, South China University of Technology, Guangzhou, People’s Republic of China; \textsuperscript{c}Department of Computer Science, City University of Hong Kong, Hong Kong SAR, People’s Republic of China

\textbf{ABSTRACT}

In this paper, we propose a framework to generate diverse clustering solutions and conduct solution retrieval to improve performance. Specifically, we first project unlabelled data from multiple domains into a shared space while preserving the respective semantics. This space allows that representations of samples in a hard domain are recovered by a linear combination of those of others in the easy domains. Meanwhile, a clustering algorithm is adopted to provide pseudo labels for a conditional generative adversarial network to synthesize representations that in turn promote the learning of the above space. Second, we conduct the joint learning of feature projection and partition matrices on batches of representations, where the former ones are considered as clustering solutions and input into another generative adversarial network to generate more solutions. Third, we utilize the fusion of diffusion to effectively retrieve and extract the knowledge in multiple solutions to obtain the final clustering. We perform comparative experiments against other methods on multiple benchmark data sets. Experimental results demonstrate the effectiveness and superiority of our proposed method.

\textbf{1. Introduction}

Clustering splits unlabelled data into groups, so that data in the same group are more similar to each other than to those in different groups (Gu et al., 2011, August; Gu & Zhou, 2009, December; Yang et al., 2015). Clustering has been demonstrated by successful applications including image analysis (Hershey et al., 2016, March; Shaham et al., 2018; Yang et al., 2016), information retrieval (Bai et al., 2019, 2017, October; Hsu & Lin, 2018), data compression (Jiang et al., 2017, August), bio-informatics (Niu et al., 2016), etc. Conventional clustering approaches have considered that features play a key role in improving performances and focused on designing sophisticated methods of feature selection to learn discriminative features (Zhang, 2015, January; Zhang & Zhang, 2010; Zhang et al., 2016), on which clustering algorithms are adopted. However, when there is no label available in conducting feature selection (Vega-Pons & Ruiz-Shulcloper, 2011), it would lead to information loss and performance deterioration. Consensus clustering (Gionis et al., 2007; Shi et al., 2018; Yu et al., 2012, May) has provided a promising alternative by combining advantages of multiple feature transformations to explore underlying structures of data from different perspectives. Besides, consensus clustering has adopted voting schemes to fuse predictions from various clustering algorithms into a unified result (Fan et al., 2017, August; Rathore et al., 2018, June). Although consensus clustering has achieved remarkable advancements, the solutions are obtained by various clustering methods. The diversity of solutions is derived from transformations conducted on features or random feature subspace techniques (Gionis et al., 2007; Yu et al., 2012, May). Since performances of consensus clustering depend on the diversity of clustering solutions, it has become a focal point that how to design feature transformations or delicate clustering algorithms.

Another scenario is that we perform clustering on multiple related tasks in real-world applications. When handling them individually, performances are extremely poor, especially when data are high-dimensional. To address this issue, multi-task clustering methods have been proposed to cluster tasks jointly (Gu et al., 2011, August; Yang et al., 2015), where the key lies in searching a shared subspace (Gu & Zhou, 2009, December; Zhang, 2015, January) for related tasks and retaining specific information about individual tasks simultaneously. The shared subspace has captured interdependencies among related tasks to guide the knowledge transfer (Zhang et al., 2016), while specific information has clustered in the original space. In light of...
consensus clustering, multi-task consensus clustering (Niu et al., 2016) has been proposed in the context of bioinformatics, which can be considered as a combination of multi-task clustering and consensus clustering. However, limited by the disadvantages of consensus clustering, the diversity of clustering solutions still needs to be further investigated from a new perspective, beyond feature transformations or clustering algorithms.

Recently, deep clustering approaches (Kamran et al., 2017, October; Wang et al., 2016; Xie et al., 2016; Yang et al., 2016) have been proposed, which can be divided into two stages: learning deep features with auto-encoder or convolutional kernels and employing clustering algorithm on the above learnt features (Caron et al., 2018; Chang et al., 2017, October; Ji et al., 2017, December; Jiang et al., 2017, August; Peng et al., 2016). A similar yet different research field has been deep unsupervised domain adaptation (Ganin & Lempitsky, 2015, July; Long et al., 2015, 2016; Tzeng et al., 2017). The difference between deep clustering and deep unsupervised domain adaptation (Chen et al., 2018; Hu et al., 2018, June; Long et al., 2018; Volpi et al., 2018) lies in that the former has focused on single task, while the latter has utilized the knowledge from the labelled source task to handle the unlabelled target task by using deep shared features or aligning their marginal distributions (Carlucchi et al., 2017; Long et al., 2017; Sener et al., 2016; Swami et al., 2017). Furthermore, deep unsupervised domain adaptation (Herath et al., 2017; Luo et al., 2017, December; Murez et al., 2018) has been differentiated from multi-task clustering in that the former has exploited the label information from the source task, while the latter has no label information available in all the tasks.

In this paper, we focus on solving a more challenging problem that there are multiple related clustering tasks, where the clustering solutions on some tasks are difficult to obtain or the quality of clustering solutions is far away from our tolerance. Consequently, multi-task clustering or deep unsupervised domain adaptation methods fail to obtain satisfactory clustering results. To address this issue, as shown in Figure 1, we have two important considerations: first, a shared space can be learnt based on compact representations of all the tasks, on which clustering algorithms are adopted to provide pseudo labels for conditional generative adversarial networks (GANs) (Mirza & Osindero, 2014) to generate more diverse representations that in turn help the searching of the shared space; second, by combining the original and synthetic representations, we conduct the joint learning of feature projection and partition matrices on batches, and input feature projection matrices into another GANs (Goodfellow et al., 2014, December) for synthesizing more feature projection matrices. Thus it reduces the dependence on clustering algorithms and feature transformations. We adopt the fusion of diffusion to integrate the knowledge of feature projection matrices into a unified one, in which feature projection matrices are regarded as clustering solutions, and transform the original representations, followed by conducting clustering algorithms to obtain the final results.

The contributions of this paper are summarized as follows:

(1) We propose a framework to synthesize clustering solutions using GANs, which exploits the knowledge from other tasks to generate diverse enough feature projection matrices for the difficult task as the clustering solution.

(2) We design a fusion strategy of diffusion to perform consensus clustering by integrating discriminative and complementary knowledge in the affinity graphs based on feature projection matrices.

(3) We provide theoretical convergence analysis for the fusion strategy of diffusion and conduct experiments on real-world data sets to verify the effectiveness of our method.
The remainder of this paper is organized as follows. In Section 2, we review the literature on multi-task clustering, consensus clustering, deep clustering and deep unsupervised domain adaptation. In Section 3, we elaborate our proposed approach. Section 4 conducts theoretical convergence analysis for the fusion of diffusion. In Section 5, we present the comparative results between our proposed approach and other counterparts over multiple real-world data sets. Section 6 draws the conclusion for this paper.

2. Related work

In this section, we review the literature concerning multi-task clustering, consensus clustering, deep clustering and deep unsupervised domain adaptation.

Many multitask clustering approaches (Gu et al., 2011, August; Gu & Zhou, 2009, December; Yang et al., 2015; Zhang, 2015, January; Zhang & Zhang, 2010, 2013; Zhang et al., 2016) have been proposed to improve performances through jointly learning related tasks, in which the shared knowledge is explored and transferred among them. Generally, this shared knowledge is mined by searching the shared low-dimensional subspace (Gu & Zhou, 2009, December) or reproducing kernel Hilbert space (Gu et al., 2011, August), followed by conducting single-task clustering and multi-task clustering iteratively. In Yang et al. (2015), the relatedness among tasks is dissected by exploring intertask and intratask correlations with a designed $\ell_2,p$-norm regularization, in which the feature projection and partition matrices are learnt simultaneously. Since tasks have different sensitivities to the shared knowledge, it is necessary to consider negative effects brought about by the knowledge transfer. A self-adapted multi-task clustering method (Zhang et al., 2016) is proposed, in which reusable instances are identified from related clusters to construct subtasks, followed by using the information from subtasks to original tasks to avoid negative transfer. Convex discriminative multi-task feature clustering and multi-task relationship clustering methods (Zhang, 2015, January) explore the shared features embedded in a covariance matrix that models relationships between features and that models relationships between task-specific models with the Gaussian prior, respectively. Consensus clustering (Vega-Pons & Ruiz-Shulcloper, 2011) reconciles the solutions obtained on the same data sets with different clustering algorithms or different runs of the same one. Existing consensus clustering is divided into clustering ensemble (Shi et al., 2018; Yu et al., 2012, May) and clustering aggregation (Fan et al., 2017, August; Gionis et al., 2007; Rathore et al., 2018, June). The former considers the clustering problem by various formulations and cast this problem as a hyper-graph partition or correlation clustering problem, thus the solution is to compute the best k-partition of the graph. The latter fuses a collection of soft clustering solutions generated by random projections, in which each instance is denoted by a vector whose entries represent the posterior probability.

Deep learning has promoted the emerging of deep clustering (Ji et al., 2017, December; Jiang et al., 2017, August; Kamran et al., 2017, October; Peng et al., 2016; Xie et al., 2016; Yang et al., 2016). Specifically, multi-layer autoencoder (Xie et al., 2016) provides initialization to learn effective latent representations for clustering. Yang et al. (2016) build successive operations in clustering on the top of representations learnt by convolutional neural network (CNN), where image clustering and representation learning are conducted in the forward and backward pass, respectively. Kamran et al. (2017, October) propose a clustering objective function with a logistic regression imposed on deep embedding learnt by CNN. Jiang et al. (2017, August) propose a variational deep embedding method, where variational autoencoder performs unsupervised generative clustering using a Gaussian mixture model. Ji et al. (2017, December) utilize a fully-connected layer to learn self-expressive matrix in the subspace clustering. Peng et al. (2016) propose to integrate the prior information about the local and global structure into learning latent subspaces with deep autoencoder. In Peng et al. (2016), the locality for reconstructing input and structured global prior is integrated into deep subspace clustering. Chang et al. (2017, October) propose deep adaptive image clustering that formulates the clustering problem as a binary pairwise classification, where similarities are measured by the cosine distances between label features of images. Caron et al. (2018) present the deep clustering method, which allows jointly learning parameters of neural network and the clustering result on the obtained features.

Deep domain adaptation methods have emerged. Specifically, Long et al. (2016) propose to learn adaptive classifiers and transferable features from labelled source domain to unlabelled target domain, where features are embedded into a reproducing kernel Hilbert space (RKHS) to match distributions. Ganin and Lempitsky (2015, July) consider the discrimination and domain-invariance of adaptive features by jointly optimizing the label predictor and domain classifier. Long et al. (2015) generalize CNN to domain adaptation, in which all the latent task-specific representations are embedded into an RKHS, where mean embedding of distributions of different domains is matched. Tzeng et al. (2017) propose the adversarial discriminative domain adaptation framework by combining the discriminative modelling, adversarial techniques and GAN loss. Long et al. (2017) match the joint distributions
of domain-specific layers over multiple domains using the joint maximum mean discrepancy. Sener et al. (2016) propose a unified framework for unsupervised domain adaptation to learn transferable representations, in which feature representations and domain transformation are alternatively optimized. Gebru et al. (2017) design an adaptation multi-task loss based on the attributes for fine-grained recognition by leveraging labels of source domain, where the loss is composed of supervised loss, unsupervised adaptation loss and attribute consistency loss. Xu et al. (2018) propose a multi-source domain adaptation method for mitigating both the domain and class shifts, in which the distribution of target domain is represented by a combination of source domain distributions. Zhao et al. (2018) provide new generalization bounds for unsupervised multi-source domain adaptation in classification and regression, where $\mathcal{H}$-divergence is utilized to measure the distance between two distributions. Unlike deep unsupervised domain adaptations, our method does not exploit any labels from source domains. Additionally, our proposed method generates not only representations of real-like data but also feature projection matrices as clustering solutions.

3. Proposed methodology

In this section, we elaborate our clustering solution generation and retrieval approach. The overview framework of the proposed approach is in Figure 2, which includes five stages: the learning of compact representations, the generation of diverse representations, the enhancement of representations guided by clustering, the generation of clustering solutions and the fusion of solutions using the diffusion.

3.1. Problem formulation

Suppose that there are data samples from one or more than one easy tasks and one difficult task, which are denoted as $\{X^1_s \in \mathbb{R}^{d^1_s \times n^s_1}\}_{s=1}^{m}$ and $X^2 \in \mathbb{R}^{d^2 \times n^t}$, where $d^1_s$ and $d^2$ denote the dimensions of samples in the first easy and the difficult tasks. $n^s_1$ and $n^t$ denote the numbers of samples in the first easy and the difficult tasks, respectively. $m$ denotes the number of easy tasks available. The goal is to explore the feasibility that utilizes various diverse clustering solutions obtained from easy tasks to retrieve and employ them into $X^2$, when the solutions directly obtained from $X^2$ are extremely poor or far away from our capability.

3.2. Motivation

The motivation is that the knowledge embedded in the solutions obtained from multiple unlabelled data sets can be transferred to other unlabelled related data sets, on which the solutions directly obtained are extremely poor or even cannot be accessed. This concept is derived from domain adaptation. There are two key differences between our approach and domain adaption. First, our approach does not rely on any labelled information, while domain adaptation needs labelled information about the source domain or both domains. Second, our approach focuses on both the latent compact representations and the task-specific information. Besides, our approach is capable of generating clustering solutions using generative adversarial networks, which provide diverse enough solutions. With these synthetic solutions, we adopt the diffusion of fusion to integrate multiple solutions into one that achieves satisfactory performance on both data sets. Compared to clustering ensemble using the majority voting mechanism, the diffusion of fusion takes into account multiple metrics, as described below.

3.3. The details about our proposed methodology

3.3.1. The learning of compact representations for all the tasks

The learning of compact representations of data samples from multiple tasks is the cornerstone that our method works. It needs to meet the following two requirements: (1) compact representations can recover the original input in their respective task and (2) compact representations of data samples in a difficult task can be reconstructed with a linear combination of those of others in easy tasks. It is worth mentioning that when reconstructing the representation of a sample in a difficult task with
those of other samples, we do not enforce that these samples are from the same task. This means that our method is capable of dealing with a difficult task using easy tasks whose semantics are not necessary to be the same, as long as the union of the semantics of easy tasks is completely overlapped with that of the given difficult task. The loss of this learning is composed of the loss functions corresponding to the above requirements, which is defined by

$$L_1 = L_r + \alpha L_m,$$  \hspace{1cm} (1)

where \(\alpha\) is a trade-off parameter to control the relative importance of \(L_r\) and \(L_m\). They are defined as follows:

$$L_r = \sum_{s=1}^{m} \left\| \mathbf{X}_1^s - D(E(\mathbf{X}_1^s)) \right\|_2^2 + \left\| \mathbf{X}_2 - D(E(\mathbf{X}_2)) \right\|_2^2,$$  \hspace{1cm} (2)

$$L_m = \left\| \mathbf{X}_2 - D(E(\tilde{\mathbf{X}}(\mathbf{X}_1^s))) \right\|_2^2,$$  \hspace{1cm} (3)

where \(E(\cdot)\) and \(D(\cdot)\) are the encoder and decoder. \(\mathbf{X}_1^s\) is composed of the k-nearest neighbouring samples from easy tasks for each sample in \(\mathbf{X}_2 = [\mathbf{x}_1^1, \mathbf{x}_1^2, \ldots, \mathbf{x}_n^1]\). \(\mathcal{N}(\mathbf{X}_1^s) = \{\mathcal{N}(\mathbf{x}_1^1, \mathbf{X}_1^2), \mathcal{N}(\mathbf{x}_2^2, \mathbf{X}_2^2), \ldots, \mathcal{N}(\mathbf{x}_m^2, \mathbf{X}_m^2)\}\), where \(1 \leq s_i \leq m\). \(\mathcal{N}(\mathbf{x}_i^j, \mathbf{X}_i^j)\) denotes the set of samples in the \(s_i\) easy task whose compact representations in the k-nearest neighbourhood of \(x_i^j\). \(\tilde{\mathcal{N}}\) denotes the mean of each sample set in \(\mathcal{N}\).

### 3.3.2. Clustering on low-dimensional mutual representations

After obtaining the latent compact representations \(E(\mathbf{X}_1)\) of \(\mathbf{X}_1\), we perform clustering via the objective function below:

$$L_c = \min \left\| E(\mathbf{X}_1) - \mathbf{M}_1 \mathbf{P}_1 \right\|_F^2,$$  \hspace{1cm} (4)

where \(\mathbf{M}_1\) and \(\mathbf{P}_1\) are the cluster centre and partition matrices on the subspace spanned by low-dimensional representations, respectively. \(\| \cdot \|_F\) is Frobenius norm. For \(\mathbf{P}_1 \in \mathbb{R}^{N \times K}\) (\(N\) and \(K\) are the numbers of data samples and clusters respectively), the index of the largest value in each row denotes which cluster the corresponding data sample belongs to. For example, suppose the index of the largest value in the \(i\)th row vector \(\mathbf{P}_1^i \in \mathbb{R}^{1 \times K}\) of \(\mathbf{P}_1\) is \(k\), we consider that the \(i\)th data sample belongs to the \(k\)th cluster. We can easily obtain a binary matrix with the same size as that of the partition matrix. A one-hot vector can be used to denote a cluster and the index of element 1 corresponds to a specific cluster. Each pair of clusters is distinguished by the indices of element 1 in corresponding one-hot vectors.

Considering that the auto-encoder network pre-trained on the input data samples may fail to evenly cover the original space, we would utilize the pseudo-labels obtained in this stage as the condition to generate more diverse representations.

#### 3.3.3. Generating low-dimensional mutual representations

After clustering on \(E(\mathbf{X}_1)\), we use conditional generative adversarial networks (CGAN) to synthesize latent representations with the same dimension to \(E(\mathbf{X}_1)\), where the predictive labels are considered conditions. The loss function of CGAN is defined by

$$L_{CGAN} = \min_G \max_D V(D, G) = \mathbb{E}_{e_k \sim E(\mathbf{X}_1)|e_k} \log D(e_k | \tilde{y}) + \mathbb{E}_{z \sim p_z(\mathbf{z})} \log(1 - D(G(z | \tilde{y}))),$$  \hspace{1cm} (5)

where \(e_k\) denotes a real sample following the distribution of \(E(\mathbf{X}_1)\) and \(z\) is a noise variable drawn from a uniform distribution. \(\tilde{y}\) is a one-hot vector of size \(K\) derived from the clustering partition matrix, which will be fed into the first generator as pseudo-label/pseudo-supervised information to generate cluster-specific representations. Initially, the clustering performance can be poor when the learned representations are not satisfactory, resulting in undesirable synthetic representations. However, the learned representations will become more discriminative with the increase of iterations, which improves the performance of clustering and the quality of synthetic representations.

#### 3.3.4. Learning the feature projection matrix on each batch

We shuffle and combine the synthetic latent representations \(S(\mathbf{X}_1)\) and \(E(\mathbf{X}_1)\) to form a set of batches \(\{\hat{\mathbf{B}}_j\}^b_{j=1}\), where \(n_b\) is the number of batches. For each batch, we learn the feature projection and partition matrices simultaneously as follows:

$$L^j = \left\| \mathbf{P}_j - \mathbf{B}_j \mathbf{W}_j \right\|_F^2 + \lambda \text{tr}(\mathbf{P}_j^T \mathbf{L} \mathbf{P}_j) + \beta \left\| \mathbf{W}_j \right\|_{2,1}, \hspace{0.5cm} s.t., \hspace{0.2cm} (\mathbf{P}_j^T \mathbf{P}_j) = \mathbf{I},$$  \hspace{1cm} (6)

where \(\mathbf{P}_j^T\) and \(\mathbf{W}_j^T\) are the partition and the feature matrices learnt on \(\hat{\mathbf{B}}_j\), respectively. \(\mathbf{L}\) is the Laplacian matrix for \(\hat{\mathbf{B}}_j\). \(\lambda\) and \(\beta\) are two parameters. It is worth mentioning that we consider \(\mathbf{W}_j\) as the clustering solution rather than \(\mathbf{P}_j\), because the latter has stronger dependence on the specific data, compared with the former. As a result, it is necessary to align data samples to the clusters delicately.

#### 3.3.5. Generating the feature projection matrix

After completing the computation on the whole batches, we obtain a series of feature projection matrices, which are considered as the ground-truth and fed into another GAN for generating more diverse ones. Specifically, we
generate the synthetic feature projection matrix as follows:

$$\min_G \max_D V(D, G) = \mathbb{E}_{w \sim p_w(w)}[\log D(w)] + \mathbb{E}_{t \sim p_t(t)}[\log(1 - D(G(t)))] \tag{7}$$

where $p_w(w)$ denotes the distribution of the feature projection matrix, and $p_t(t)$ denotes a uniform distribution, from which the noise variable $t$ is sampled.

### 3.3.6. The fusion of clustering solutions

With the synthetic feature projection matrices and the learnt ones trained on batches, we adopt the fusion of diffusion strategy to merge the information about them via:

$$\min_{A, \Theta} \sum_{m=1}^{M} \sum_{n=1}^{M} \left( \frac{1}{2} \sum_{i,j,k,l=1}^{n_t} \Theta_{mn} S_j^{(m)} S_k^{(n)} - \frac{A_{ki} - I_{ki}}{\sqrt{D_{kk}^{(m)} D_{kk}^{(n)}}} \right)^2 + \mu \sum_{k,l=1}^{n_t} (A_{ki} - I_{ki})^2 + \frac{\tau}{2} \left\| \Theta_{mn} \right\|_2^2, \tag{8}$$

s.t., $0 \leq \Theta_{mn} \leq 1, \sum_{n=1}^{M} \Theta_{mn} = 1,$

where $A$ denotes the optimal similarity matrix that fuses all the information in $M$ feature projection matrices $W^1, W^2, \ldots, W^M$. $I$ is an identity matrix. $S_j^{(m)}$ is the weight matrix of the graph after conducting transformation on the learned representations $E(X_j)$ using the feature projection $W^{(m)}$. $S_j^{(m)}$ is the weight of the edge connecting the $i$th and $j$th samples. $D^{(m)}$ is a diagonal matrix whose entry $D_{ii}^{(m)} = \sum_{j=1}^{n_t} S_j^{(m)}$. $\Theta_{mn}$ denotes the weight that connects the $m$th and $n$th affinity graphs concerning $W^{(m)}$ and $W^{(n)}$. The first term controls the smoothness of pairwise products graphs. More specifically, if $x_i$ is similar to $x_j$ in the $m$th similarity space spanned by $W^{(m)}$, i.e. large $S_j^{(m)}$, and $x_k$ is similar to $x_l$ in the $n$th similarity space spanned by $W^{(n)}$, i.e. large $S_l^{(n)}$, the learned $A_{ki}$ and $A_{lj}$ should have a small difference. The second term indicates the learned optimal similarity should preserve self-similarity, i.e. $I_{kk}$. $\tau$ is the weight regularization parameter, controlling the distribution of $\Theta_{mn}$. If the fusion from $S^{(m)}$ to $S^{(n)}$ is non-smooth, the value of the first term will be large, and it will assign a small weight to $\Theta_{mn}$ to ensure that the objective value will decrease during the iteration.

### 3.4. Optimization process for Equation (8)

In this part, we present the process of optimizing Equation (8).

$$\min_{A, \Theta} \sum_{m=1}^{M} \sum_{n=1}^{M} \left( \Theta_{mn} Q_{mn} + \mu \sum_{k,l=1}^{n_t} (A_{ki} - I_{ki})^2 + \frac{\tau}{2} \left\| \Theta_{mn} \right\|_2^2 \right), \text{ s.t., } 0 \leq \Theta_{mn} \leq 1, \sum_{n=1}^{M} \Theta_{mn} = 1, \tag{9}$$

where

$$Q_{mn} = \frac{1}{2} \sum_{i,j,k,l=1}^{n_t} S_j^{(m)} S_k^{(n)} \left( \frac{A_{ki} - I_{ki}}{\sqrt{D_{kk}^{(m)} D_{kk}^{(n)}}} \right)^2,$$

and $\Theta^T = [\Theta_{11}^T, \ldots, \Theta_{MM}^T]$.

#### 3.4.1. Update A when $\Theta$ is fixed

In this setting, the third term in Equation (9) is a constant, and Equation (9) can be transformed into

$$\min_{A} \sum_{m=1}^{M} \sum_{n=1}^{M} \Theta_{mn} (I - S^{(m,n)})^T \bar{A} + \mu \left\| \bar{A} - \bar{I} \right\|_2^2, \tag{10}$$

where $S^{(m,n)} = S^{(m)} \otimes S^{(n)}$. The partial derivation of Equation (10) with respect to $\bar{A}$ can be denoted by

$$2 \sum_{m=1}^{M} \sum_{n=1}^{M} \Theta_{mn} (I - S^{(m,n)}) \bar{A} + \mu (\bar{A} - \bar{I})^T.$$

By setting Equation (11) to 0, we obtain

$$\bar{A} = \left( I - \sum_{m=1}^{M} \sum_{n=1}^{M} \Theta_{mn} \right)^{-1} \bar{I}, \tag{12}$$

where

$$\Theta_{mn} = \frac{\Theta_{mn}}{\mu + \sum_{m'=1}^{M} \Theta_{mm'}}. \tag{13}$$

Thus the optimal $A$ is obtained by $\text{vec}(-1)(\bar{A})$. 
3.4.2. Update $\Theta$ when $A$ is fixed

In this setting, the second term in Equation (9) is a constant, and Equation (9) is transformed into

$$\min_A \sum_{m=1}^M \sum_{n=1}^M (\Theta_{mn} Q_{mn} + \frac{\tau}{2} \|\Theta_m\|_2^2),$$

s.t.,

$$0 \leq \Theta_{mn} \leq 1,$$

$$\sum_{m=1}^M \Theta_{mn} = 1,$$

which is solved by using coordinate descent. In each iteration, we update two elements with the remainders fixed, which leads to the update scheme as follows:

$$\begin{align*}
\Theta_{ mj}^* &= \max(0, \Theta_{ mj} + \Theta_{ mj}^* - \Theta_{ mj}) \\
\Theta_{ my}^* &= \max(0, \Theta_{ mj} + \Theta_{ mj}^* - \Theta_{ mj})
\end{align*}$$

(15)

The optimization pseudo-code is shown in Algorithm 1.

4. Theoretical analysis

In this section, we perform analysis for the fusion of diffusion which can play the key role in performance improvement. First, we prove that the close-form solution of components in the square brackets of Equation (8) can be expressed by

$$A = \lim_{\tau \to \infty} A^{(t)} = (1 - \alpha)\text{vec}^{-1}((I - \alpha S)^{-1}I),$$

(16)

where $S$ denotes the Kronecker product of $S^{(1)}$ and $S^{(2)}$, i.e., $S = S^{(1)} \otimes S^{(2)}$. $S^{(1)}$ and $S^{(2)}$ are the transition matrices of two kinds of similarities. vec$(\cdot)$ denotes the vectorization of a matrix by concatenating its columns one by one, and vec$^{-1}$ denotes its inverse function. $\alpha \in (0, 1)$ is a trade-off parameter.

**Proof:** When we set all the entries in $\Theta$ to 1, the third term is a constant. In this setting, the first term in the square brackets of Equation (8) can be rewritten by

$$\frac{1}{2} \sum_{i,j,k,l=1}^{n_1} S_{ij}^{(1)} S_{kl}^{(2)} \left( \frac{A_{ki} A_{lj}}{D_{kk}^{(1)} D_{ll}^{(2)}} - \frac{A_{ij}}{D_{kk}^{(1)} D_{ll}^{(2)}} \right)^2,$$

$$= \frac{1}{2} \sum_{i,j,k,l=1}^{n_1} S_{ij}^{(1)} S_{kl}^{(2)} \left( \frac{A_{ki} A_{lj}}{D_{kk}^{(1)} D_{ll}^{(2)}} + \frac{A_{ij}}{D_{kk}^{(1)} D_{ll}^{(2)}} \right)^2 - 2 \frac{A_{ki} A_{lj}}{\sqrt{D_{kk}^{(1)} D_{ll}^{(2)}}},$$

(17)

Since

$$\sum_{i,j,k,l=1}^{n_1} S_{ij}^{(1)} S_{kl}^{(2)} A_{ki}^2 D_{kk}^{(1)} D_{ll}^{(2)} = \sum_{i,j,k,l=1}^{n_1} S_{ij}^{(1)} S_{kl}^{(2)} A_{ij} D_{kl}^{(1)} D_{kl}^{(2)},$$

we can rewrite Equation (17) as follows:

$$\sum_{i,j,k,l=1}^{n_1} S_{ij}^{(1)} S_{kl}^{(2)} \left( \frac{A_{ki} A_{lj}}{D_{kk}^{(1)} D_{ll}^{(2)}} - \frac{A_{ij}}{D_{kk}^{(1)} D_{ll}^{(2)}} \right),$$

(18)

Since

$$\sum_{i,j,k,l=1}^{n_1} S_{ij}^{(1)} S_{kl}^{(2)} A_{ki}^2 D_{kk}^{(1)} D_{ll}^{(2)} = \sum_{y,z=1}^{n_2} S_{yz} \frac{\tilde{A}_y}{D_{yy}^{(2)}} \frac{\tilde{A}_z}{D_{zz}^{(2)}},$$

we can rewrite Equation (18) by

$$\sum_{y,z=1}^{n_2} S_{yz} \frac{\tilde{A}_y}{D_{yy}^{(2)}} \frac{\tilde{A}_z}{D_{zz}^{(2)}} = \sum_{y,z=1}^{n_2} S_{yz} \frac{\tilde{A}_y}{D_{yy}^{(2)}} \frac{\tilde{A}_z}{D_{zz}^{(2)}},$$

(19)

Since $D_{yy} = D_{kk}^{(1)} D_{ll}^{(2)} = \sum_{i,j,k,l=1}^{n_1} S_{ij}^{(1)} \sum_{i,j,k,l=1}^{n_1} S_{kl}^{(2)}$, Equation (19) equals:

$$\sum_{y=1}^{n_2} \frac{\tilde{A}_y}{D_{yy}^{(2)}} \sum_{z=1}^{n_2} \frac{\tilde{A}_z}{D_{zz}^{(2)}} = \sum_{y=1}^{n_2} \frac{\tilde{A}_y}{D_{yy}^{(2)}} \sum_{z=1}^{n_2} \frac{\tilde{A}_z}{D_{zz}^{(2)}} \tilde{A}_y \tilde{A}_z - \tilde{A}_y \tilde{A}_z \tilde{A}_y \tilde{A}_z$$

$$= \tilde{A}_y \tilde{A}_z (\tilde{A} - \tilde{S}) \tilde{A} \tilde{A}$$

(20)

The second term in this bracket can be rewritten by

$$\mu \sum_{k,l=1}^{n_2} (A_{kl} - I_{kl})^2 = \mu \sum_{k,l=1}^{n_2} (A_{kl} - I_{kl})^2 = \mu \| \tilde{A} - \tilde{I} \|_2^2,$$

(21)

Therefore, the addition of these two terms would become

$$J_0 = \min_{\tilde{A}} \tilde{A} \tilde{A} + \mu \| \tilde{A} - \tilde{I} \|_2^2,$$

(22)

and the partition derivation with respect to $\tilde{A}$ can be given by

$$\frac{\partial J_0}{\partial \tilde{A}} = 2(I - \tilde{S}) \tilde{A} + 2\mu(\tilde{A} - \tilde{I}).$$

(23)
Algorithm 1 Clustering Solution Retrieval using Generative Adversarial Network and Fusion of Diffusion (CSR-GAFD)

Require:
- Easy data \( \{X_1\}_{s=1}^m \), hard data \( X_2 \), maximum iteration \( MAXITE \), trade-off parameters;

Ensure:
- Partition Matrix \( P_2 \);

1: Pre-train Feature Extractor to learn latent representations;
2: While \( c_{nt} \leq MAXITE \) do
3: Use C-GAN to fine-tune Feature Extractor.
4: for \( i=1 \) to \( N_1 \) do
5: Cluster latent representations for producing pseudo-labels via solving Equation (4);\n6: Update the Generator in the first GAN via:
\[
\nabla_{\theta_1} \frac{1}{m_1} \sum_{i=1}^{m_1} \log \left(1 - D(G(e'_i | \tilde{y}_i))\right)
\]
7: Update the Discriminator in the first GAN via:
\[
\nabla_{\theta_1} \frac{1}{m_1} \sum_{i=1}^{m_1} \left[ \log D(z'_i | \tilde{y}_i) + \log \left(1 - D(G(e'_i | \tilde{y}_i))\right)\right]
\]
8: Fine-tune Feature Extractor via solving Equation (1);
9: end for
10: % Learn the ground-truth clustering solutions
11: Shuffle and split into the real and synthetic representations batches;
12: Learn feature project matrices as ground-truth clustering solutions on batches via solving Equation (6);
13: % Generate clustering solutions using another GAN
14: for \( i=1 \) to \( N_2 \) do
15: Feed batches of clustering solutions into GAN;
16: Update the Generator in the second GAN via:
\[
\nabla_{\theta_2} \frac{1}{m_2} \sum_{i=1}^{m_2} \log \left(1 - D(G(t'_i))\right)
\]
17: Update the Discriminator in the second GAN via:
\[
\nabla_{\theta_2} \frac{1}{m_2} \sum_{i=1}^{m_2} \left[ \log D(w'_i) + \log \left(1 - D(G(t'_i))\right)\right]
\]
18: end for
19: end while
20: % Conduct the fusion of diffusion on clustering solutions via Equation (8);
21: Fix \( \Theta \) and Update \( \tilde{A} \):
\[
\tilde{A} = (1 - \sum_{m=1}^{M} \Omega_{mn}) (I - \sum_{n=1}^{M} \Omega_{mn} S^{(m,n)})^{-1} \tilde{I};
\]
\[
A = \text{vec}^{-1}(\tilde{A});
\]
22: Fix \( A \) and Update \( \Theta \) using coordinate descent via:
\[
\Theta_{mi} = \max(0, (\Theta_{mi} + \Theta_{mj} - \Theta_{mj}^*) / 2);\]
\[
\Theta_{mj}^* = \max(0, \Theta_{mi} + \Theta_{mj} - \Theta_{mj}^*);\]
23: Run clustering algorithm to generate \( P_2 \);

To find the optimal solution of Equation (22), we set \( \frac{\partial J_{A}}{\partial A} = 0 \) and obtain
\[
\tilde{A} = \mu ((1 + \mu)I + S)^{-1} \tilde{I} = \frac{\mu}{1 + \mu} \left( I - \frac{S}{1 + \mu} \right)^{-1} \tilde{I}.
\]
(24)

By setting \( \mu = \frac{1}{\alpha} - 1 \), we can obtain
\[
\tilde{A} = (1 - \alpha)(I - \alpha S)^{-1} \tilde{I},
\]
(25)
which equals Equation (16) after applying \( \text{vec}^{-1}(\cdot) \) at both sides.

Lemma 4.1. With sufficient iterations, any fusion denoted as
\[
A^{(t+1)} = \alpha S(\alpha A^{(t)} S^{(1)})^T + (1 - \alpha)I,
\]
(26)
which is derived from certain diffusion converges to Equation (16).

Proof: Since \( \text{vec}(AXB) = (B^T \otimes A)\text{vec}(X) \), we adopt \( \text{vec}(\cdot) \) operator to both sides of Equation (26) to obtain
\[
\tilde{A}^{(t+1)} = \alpha S\tilde{A}^{(t)} + (1 - \alpha)\tilde{I},
\]
(27)
where \( S = S^{(1)} \otimes S^{(2)} \). Therefore, \( \tilde{A}^{(t+1)} \) can be expanded as
\[
\tilde{A}^{(t+1)} = aS(\alpha S\tilde{A}^{(t-1)} + (1 - \alpha)I) + (1 - \alpha)I,
\]
\[
= aS(aS(\alpha S\tilde{A}^{(t-2)} + (1 - \alpha)I) + (1 - \alpha)I),
\]
\[
+ (1 - \alpha)I)
\]
\[
+ (\alpha S)^{t-1}(1 - \alpha)I,
\]
\[
= (\alpha S)^{t}A^{1} + (1 - \alpha)I + \alpha S(1 - \alpha)I + \cdots
\]
\[
+ (\alpha S)^{t-1}(1 - \alpha)I,
\]
\[
= (\alpha S)^{t}A^{1} + (1 - \alpha)\sum_{i=1}^{t-1}(\alpha S)^{i}I. \tag{28}
\]

Because the spectral radius of both \( S^{(1)} \) and \( S^{(2)} \) are no larger than 1, all the eigenvalues of \( S \) are in \([-1, 1]\). Since \( 0 < \alpha < 1 \), we can find that
\[
\lim_{t \to \infty} (\alpha S)^{i}A^{1} = 0, \tag{29}
\]
\[
\lim_{t \to \infty} (1 - \alpha)\sum_{i=0}^{t-1}(\alpha S)^{i}I = (1 - \alpha)(I - \alpha S)^{-1}I. \tag{30}
\]
Adding Equation (29) into Equation (30), we can easily obtain that
\[
\lim_{t \to \infty} \tilde{A}^{(t+1)} = (1 - \alpha)(I - \alpha S)^{-1}I, \tag{31}
\]
which equals Equation (16) after we apply \( \text{vec}^{-1}(\cdot) \) to both sides.

5. Experiment

In this section, we conduct comparative experiments, where accuracy (AC) and normalized mutual information (NMI) are adopted to quantify the clustering quality. Second, we divide our method into five stages and analyse their contributions to improvements. Third, we explore effects of network architecture of the feature extractor on performance and effects of trade-off parameters in the fusion of diffusion. Finally, we conduct exploration a situation where the label space of easy data sets is not fully overlapped while the union of their labels is the same as that of difficult data set.

5.1. Data set

In this part, we introduce the real-world data sets to evaluate the effectiveness of our approach. These data sets include Amazon reviews, news texts, travel reviews, autistic spectral disorder screening data, KEGG metabolic relation network data, letter recognition and EMNIST. Their descriptions are in Table 1. Specifically, Amazon reviews are concerning four kinds of products: books, dvds, electronics and kitchen appliances, denoted by B, D, E and K, respectively. Each one consists of 2000 samples with 5000 features. News texts include documents selected from RCV1, 20 Newsgroups, Reuters and TDT2, from which we select four categories, respectively. They are denoted by RCV1-4 with 9625 samples, 20NG-4 with 3978 samples, Reuters-4 with 6387 samples and TDT2-4 with 5705 samples, respectively. Travel reviews include Tr-Re1 and Tr-Re2, where the former has 980 samples with 11 features and the latter has 5456 samples with 25 features. Autistic spectral disorder screening data includes ASDS-adolescent and ASDS-children, where the former has 104 samples of 21 features and the latter has 292 samples of 21 features. KEGG metabolic relation network data includes K-directed with 53,414 samples over 24 features and K-undirected with 65554 samples over 29 features. Letter recognition data has 20,000 samples with 16 features and EMNIST has 145,600 samples with 256 features.

5.2. Parameter setting

In this part, we describe parameter settings of our approach and comparative counterparts. Concerning the Amazon Review and News Texts, the Feature Extractor consists of four hidden fully-connected layers with 2500, 1000, 500, 100 neurons, respectively; the generator in C-GAN consists of four layers with 10, 25, 50 and 100 neurons, respectively, while + the discriminator in the second GAN used to synthesize the feature projection matrices are symmetrical in the structures, where the former one consists of three layers with 40, 80 and 200 neurons, and the latter one consists of

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Data set</th>
<th>( d )</th>
<th>( n )</th>
<th>( c )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon Review (T1)</td>
<td>Amazon-books (B)</td>
<td>5000</td>
<td>2000</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Amazon-dvds (D)</td>
<td>5000</td>
<td>2000</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Amazon-electronics (E)</td>
<td>5000</td>
<td>2000</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Amazon-kitchen (K)</td>
<td>5000</td>
<td>2000</td>
<td>2</td>
</tr>
<tr>
<td>News Texts (T2)</td>
<td>RCV1-4</td>
<td>5000</td>
<td>9625</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>20NG-4</td>
<td>5000</td>
<td>3978</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Reuters-4</td>
<td>5000</td>
<td>6387</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>TDT2-4</td>
<td>5000</td>
<td>5705</td>
<td>4</td>
</tr>
<tr>
<td>Tr-Re1</td>
<td>Directed</td>
<td>24</td>
<td>53414</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Undirected</td>
<td>29</td>
<td>65554</td>
<td>2</td>
</tr>
<tr>
<td>T6</td>
<td>LetterRecg</td>
<td>16</td>
<td>20000</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>EMNIST</td>
<td>256</td>
<td>145600</td>
<td>26</td>
</tr>
</tbody>
</table>
three layers with 200, 80 and 40 neurons. All the activation functions are set to ReLu. The detailed architectures for other data sets are shown in Table 2. The trade-off parameters $\alpha$, $\lambda$, $\beta$, and $\mu$ in our proposed method are set in the range $[0.1, 0.2, \ldots, 0.9]$, while the parameter $\tau$ is searched in the range $[15, 30, \ldots, 150]$. For a better reproduce of our model, we have included the detailed network architectures for T1/T2, which is shown in Figure 3.

The trade-off parameter in LSSMCT for controlling relative importance between single-task clustering and multi-task clustering is in the range $[0.1, 0.2, \ldots, 0.9]$, and the dimension of the shared embedding space is searched across in $\{2, 2^2, \ldots, 2^6\}$. The number of neighbours for spectral clustering is in the range $[3, 6, \ldots, 15]$. The number of clustering solutions in consensus clustering is in the range $[2, 4, \ldots, 8]$. For Amazon Review and News Texts, the sample rate of features selected in consensus clustering is in the range $[0.1, 0.2, \ldots, 0.9]$.

Before illustrating experimental results, we first show trends of cost function values concerning the iteration. From Figure 4, we observe that values of the cost function can quickly converge within 10 iterations over all the data sets. It indicates that the stage fusion of diffusion costs little running time to obtain the final clustering result.

### 5.3. Comparative experiments with other related methods

In this section, we present and analyse the comparative experiments on the above data sets against related methods including K-means, spectral clustering, consensus clustering, multi-task clustering, deep clustering and adversarial multi-source domain adaptation (MDAN) (Zhao et al., 2018). We choose LSSMTC (Gu & Zhou, 2009, December) and deep embedding clustering (DEC) (Xie et al., 2016) as multi-task clustering and deep clustering, respectively. The comparison results are shown in Tables 3 and 4. From these tables, we have the following observations:

1. Consensus clustering has achieved better performances than K-means and spectral clustering since it can combine and fuse the multi-view informative knowledge in multiple clustering solutions. The solutions are obtained by conducting random subspace tricks or random transformation on the original features.

2. LSSMTC performs better than consensus clustering, in which the knowledge has been explored and transformed among tasks. It indicates that compared with the informative knowledge in multiple solutions, the knowledge discovered from other tasks can play a more important role in performance improvement.

3. DEC achieves better or comparable performance, compared with LSSMTC, although it does not utilize multiple solutions or tasks. It indicates that a neural network with appropriate structure has enough capability to learn effective information beneficial to downstream tasks.

4. Both MDAN-hard and MDAN-soft significantly outperform LSSMTC and DEC, which indicate that combining the knowledge from other tasks with the powerful learning capability of neural networks can further enrich the discriminative and complementary features.

5. Our proposed method has achieved better performance than MDAN-hard and MDAN-soft in the majority of cases. The advantage of our method over MDAN-hard (soft) can attribute to the clustering-guided feature extractor for learning the discriminative information, the exploration of common knowledge of related tasks and the fusion of clustering solutions generated by GAN.

From the comparison results, we observe that although consensus clustering can generate diverse clustering solutions by adopting random subspace or transformation on the original features, it may suffer from information loss and ignore different robustness of tasks to transfer of the mined common information. LSSMCT achieves performance improvement via searching a shared subspace across related tasks. However, it works under the assumption that related works share the same semantic space, limiting the applications. DEC has tried to identify a clustering-friendly embedding space via optimizing the representations and clustering assignment alternately. However, its capability of mining

### Table 2. The detailed network architectures of the Feature Extractor and two Generative Adversarial Networks, where Input-Dim depends on the specific dimension of the original input.

<table>
<thead>
<tr>
<th>Component</th>
<th>Detailed Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1 &amp; T2</td>
<td>Feature Extractor 5000, 2500, 1000, 500, 100 (neurons)</td>
</tr>
<tr>
<td></td>
<td>First Generator 10, 25, 50, 100 (neurons)</td>
</tr>
<tr>
<td></td>
<td>Second Generator 40, 80, 200 (neurons)</td>
</tr>
<tr>
<td></td>
<td>First Discriminator 100, 50, 25, 10 (neurons)</td>
</tr>
<tr>
<td></td>
<td>Second Discriminator 200, 80, 40 (neurons)</td>
</tr>
<tr>
<td>T3, T4 &amp; T5</td>
<td>Feature Extractor (Input-Dim), 50, 30, 20 (neurons)</td>
</tr>
<tr>
<td></td>
<td>First Generator 10, 20, 20 (neurons)</td>
</tr>
<tr>
<td></td>
<td>Second Generator 40, 80, 200 (neurons)</td>
</tr>
<tr>
<td></td>
<td>First Discriminator 20, 20, 10 (neurons)</td>
</tr>
<tr>
<td></td>
<td>Second Discriminator 200, 80, 40 (neurons)</td>
</tr>
<tr>
<td>T6</td>
<td>Feature Extractor 256/16, 160, 100, 50 (neurons)</td>
</tr>
<tr>
<td></td>
<td>First Generator 10, 25, 50 (neurons)</td>
</tr>
<tr>
<td></td>
<td>Second Generator 200, 400, 800 (neurons)</td>
</tr>
<tr>
<td></td>
<td>First Discriminator 50, 25, 10 (neurons)</td>
</tr>
<tr>
<td></td>
<td>Second Discriminator 800, 400, 200 (neurons)</td>
</tr>
</tbody>
</table>
In this part, we divide our proposed method into five stages and explore their contributions to performance improvement. Specifically, in the first stage, we denote the clustering result obtained from the learnt mutual representations by $S1$. In the second stage, we denote the clustering result with the help of synthetic mutual representations by $S2$, since we assume that the clustering provides pseudo-labels, based on which more representations can be synthesized to in turn improve the learning of representations. In the third stage, we denote the clustering result by considering feature projection matrices learnt on batches of mutual representations as $S3$. In the fourth stage, we denote the clustering results of combining the feature matrices learnt on batches and those generated by GANs using a certain voting scheme as $S4$. In the fifth stage, we denote the clustering results by fusing feature projections.
Figure 4. The trends of the cost function values concerning iterations.

with diffusion as S5. Experimental results are shown in Tables 5-6.

From these tables, we have the following observations: all the five stages have brought about positive effects on improvement, and there exist some contribution differences among these stages. For example, on the data set Amazon Review, the performance improvement mainly attributes to S2 and S4. It means that the increasing amount of latent representations does help the learning of discriminative informative knowledge, no matter they are from the original input or GANs. This is the key why S4 and S5, since both of them need more diverse and discriminative information. However, on data set travel review TrRe-1 and TrRe-2, these stages play the approximately same role in enhancement. Besides, S5 outperforms S4 all the data sets, indicating that the fusion of diffusion can better integrate effective knowledge from multiple affinity matrices into a unified solution rather than the voting scheme (S4). The reason is that the order of information fusion makes a difference, while the order has little effect on the voting scheme.

Apart from dissecting contributions of each stage, we shall investigate the effects of the number of clustering solutions on performance over data set Amazon Review by comparing consensus clustering with our proposed method, as shown in Figure 5. From this figure, it is observed that when the number of clustering solutions increases, performances obtained by consensus clustering and our proposed method show upward trends. However, the improvement gain brought about by the
clustering solutions becomes diminishing in the case of consensus clustering, while the improvement gain can be lost over a larger range of solution numbers. In addition, with the same increment of clustering solutions, our method enjoys a larger improvement, compared with consensus clustering. We deduce that the reason is that the generated diverse representations in the latent space provide the feasibility to analyze the data from multiple perspectives. Meanwhile, more clustering solutions are obtained before feeding into the second GAN to synthesize such solutions that contain complementary information.

Besides, we explore the effects of the number of easy data sets on clustering performance over Amazon Review and News Texts, as shown in Figure 6. It can be observed that when the number of easy data sets increases, the performance shows a significant improvement over all the cases. It indicates that with a larger number of easy data sets available, much richer complementary information can be provided, thus the feature extractor can learn more informative and discriminative representations which are beneficial to downstream tasks. As a result, more diverse clustering solutions can be generated.

### 5.5. Effects of the feature extractor’s architectures and trade-off parameters

In this section, we first investigate the effects of the feature extractor’s architecture that includes the network depth and the network width on clustering performance. Second, we explore the effects of tradeoff parameters used in the fusion of diffusion.

#### 5.5.1. Effects of the feature extractor’s architecture

We have explored the effects of the network depth and width in the Feature Extractor on clustering performance in terms of AC and NMI over data sets Amazon review and News texts, as shown in Figures 7 and 8. Here the

### Table 4. Clustering performance in terms of normalized mutual information (NMI), where SC and CC denote spectral clustering and consensus clustering.

<table>
<thead>
<tr>
<th>Task</th>
<th>Kmeans</th>
<th>SC</th>
<th>CC</th>
<th>LSSMTNC</th>
<th>DEC</th>
<th>MDAN-hard</th>
<th>MDAN-soft</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>D,E,K→B</td>
<td>43.22±3.38</td>
<td>46.28±2.67</td>
<td>48.78±1.96</td>
<td>49.22±2.28</td>
<td>52.05±1.98</td>
<td>54.66±3.22</td>
<td>55.68±2.18</td>
<td>59.63±2.04</td>
</tr>
<tr>
<td>B,E→K</td>
<td>46.44±2.97</td>
<td>48.67±2.10</td>
<td>50.28±1.86</td>
<td>51.65±2.06</td>
<td>52.84±1.54</td>
<td>53.14±1.79</td>
<td>54.26±2.63</td>
<td>58.48±2.54</td>
</tr>
<tr>
<td>B,D,K→E</td>
<td>41.36±2.83</td>
<td>45.34±1.92</td>
<td>48.14±2.43</td>
<td>50.45±1.48</td>
<td>49.86±2.16</td>
<td>62.11±2.56</td>
<td>64.84±2.07</td>
<td>66.95±1.68</td>
</tr>
<tr>
<td>B,D,E→K</td>
<td>50.15±1.69</td>
<td>53.26±1.64</td>
<td>56.47±1.52</td>
<td>60.62±1.64</td>
<td>55.89±1.26</td>
<td>60.42±1.89</td>
<td>66.64±2.45</td>
<td>76.00±1.59</td>
</tr>
</tbody>
</table>

####定向 5.5.2. Effects of the feature extractor’s architectures and trade-off parameters

We have also explored the effects of the feature extractor’s architecture and trade-off parameters over data sets Amazon review and News texts, as shown in Figure 7. Here the

### Table 5. Clustering performance in terms of AC.

<table>
<thead>
<tr>
<th>Task</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
</tr>
</thead>
<tbody>
<tr>
<td>D,E,K→B</td>
<td>71.34±1.96</td>
<td>73.45±2.10</td>
<td>76.05±1.98</td>
<td>77.68±2.10</td>
<td>79.34±1.93</td>
</tr>
<tr>
<td>B,E→K</td>
<td>75.68±1.86</td>
<td>77.65±2.06</td>
<td>78.84±1.54</td>
<td>82.14±1.79</td>
<td>83.48±2.04</td>
</tr>
<tr>
<td>B,D→E</td>
<td>82.14±2.43</td>
<td>83.45±1.48</td>
<td>84.86±2.16</td>
<td>85.84±2.07</td>
<td>85.95±1.68</td>
</tr>
<tr>
<td>B,D,E→K</td>
<td>83.47±1.52</td>
<td>85.62±1.64</td>
<td>86.89±1.26</td>
<td>88.20±1.89</td>
<td>89.09±1.59</td>
</tr>
<tr>
<td>RCV1-4</td>
<td>80.42±1.87</td>
<td>83.24±2.07</td>
<td>85.48±1.84</td>
<td>86.85±2.04</td>
<td>87.04±1.97</td>
</tr>
<tr>
<td>RCV2-4</td>
<td>89.78±2.08</td>
<td>91.75±1.98</td>
<td>92.48±1.86</td>
<td>93.00±2.05</td>
<td>93.48±1.68</td>
</tr>
<tr>
<td>RCV3-4</td>
<td>80.74±1.96</td>
<td>82.47±1.68</td>
<td>85.83±2.04</td>
<td>86.40±1.89</td>
<td>87.05±1.46</td>
</tr>
<tr>
<td>RCV4-4</td>
<td>89.75±1.92</td>
<td>91.58±2.02</td>
<td>92.67±1.83</td>
<td>93.54±2.18</td>
<td>94.79±1.59</td>
</tr>
<tr>
<td>TrRe-1</td>
<td>88.45±2.64</td>
<td>90.25±2.08</td>
<td>91.05±2.14</td>
<td>91.85±1.68</td>
<td>92.01±1.83</td>
</tr>
<tr>
<td>TrRe-2</td>
<td>81.23±2.52</td>
<td>83.44±2.83</td>
<td>85.65±2.07</td>
<td>86.08±2.05</td>
<td>88.43±2.07</td>
</tr>
<tr>
<td>ASDS-adolescent</td>
<td>73.14±2.58</td>
<td>75.09±2.04</td>
<td>76.54±1.99</td>
<td>77.57±2.08</td>
<td>78.64±1.65</td>
</tr>
<tr>
<td>ASDS-children</td>
<td>76.98±2.68</td>
<td>78.67±2.47</td>
<td>80.44±2.09</td>
<td>82.74±2.37</td>
<td>83.19±1.93</td>
</tr>
<tr>
<td>Directed</td>
<td>73.59±1.03</td>
<td>75.19±2.47</td>
<td>76.84±2.09</td>
<td>77.68±1.93</td>
<td>79.82±1.68</td>
</tr>
<tr>
<td>Undirected</td>
<td>77.68±1.18</td>
<td>78.43±2.74</td>
<td>79.97±2.62</td>
<td>80.67±2.00</td>
<td>82.06±1.87</td>
</tr>
<tr>
<td>LetterRecg</td>
<td>81.65±3.12</td>
<td>84.35±2.59</td>
<td>86.38±2.88</td>
<td>86.86±2.08</td>
<td>87.69±1.99</td>
</tr>
<tr>
<td>EMNIST</td>
<td>60.47±3.79</td>
<td>62.67±3.21</td>
<td>64.27±3.19</td>
<td>66.84±2.79</td>
<td>67.58±2.52</td>
</tr>
</tbody>
</table>
Table 6. Clustering performance in terms of NMI.

<table>
<thead>
<tr>
<th>Task</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
</tr>
</thead>
<tbody>
<tr>
<td>D,E,K → B</td>
<td>50.62±2.16</td>
<td>53.60±3.48</td>
<td>56.35±2.89</td>
<td>56.98±2.99</td>
<td>59.63±2.04</td>
</tr>
<tr>
<td>B,E,K → D</td>
<td>53.48±2.18</td>
<td>55.99±3.50</td>
<td>57.99±3.02</td>
<td>58.54±3.10</td>
<td>58.48±2.54</td>
</tr>
<tr>
<td>B,D,K → E</td>
<td>60.46±1.97</td>
<td>63.85±1.48</td>
<td>64.45±3.56</td>
<td>66.58±1.98</td>
<td>67.76±2.45</td>
</tr>
<tr>
<td>B,D,E → K</td>
<td>58.57±2.68</td>
<td>60.56±1.64</td>
<td>65.27±3.45</td>
<td>67.62±2.45</td>
<td>68.00±1.59</td>
</tr>
<tr>
<td>→ RCV1-4</td>
<td>60.59±2.18</td>
<td>63.74±2.56</td>
<td>65.89±2.37</td>
<td>66.51±2.83</td>
<td>69.04±2.26</td>
</tr>
<tr>
<td>→ 20NG-4</td>
<td>58.88±3.05</td>
<td>60.25±2.42</td>
<td>62.34±2.68</td>
<td>64.30±3.58</td>
<td>67.40±1.88</td>
</tr>
<tr>
<td>→ Reuters-4</td>
<td>60.78±1.96</td>
<td>62.57±3.48</td>
<td>66.07±3.58</td>
<td>67.40±3.40</td>
<td>62.20±1.88</td>
</tr>
<tr>
<td>→ TDT2-4</td>
<td>59.56±2.95</td>
<td>62.58±2.02</td>
<td>63.58±2.97</td>
<td>65.68±3.50</td>
<td>70.69±3.55</td>
</tr>
<tr>
<td>→ TrRe-1</td>
<td>60.55±3.28</td>
<td>62.45±1.98</td>
<td>63.05±2.54</td>
<td>65.25±1.88</td>
<td>66.20±2.87</td>
</tr>
<tr>
<td>→ TrRe-2</td>
<td>62.53±2.52</td>
<td>63.47±2.03</td>
<td>65.47±1.89</td>
<td>66.57±1.97</td>
<td>68.40±1.68</td>
</tr>
<tr>
<td>→ ASDS-adolescent</td>
<td>53.28±4.53</td>
<td>55.83±5.07</td>
<td>56.75±3.97</td>
<td>57.89±3.47</td>
<td>58.67±2.05</td>
</tr>
<tr>
<td>→ ASDS-children</td>
<td>60.87±1.75</td>
<td>62.60±2.74</td>
<td>64.55±3.42</td>
<td>66.08±2.53</td>
<td>67.10±2.67</td>
</tr>
<tr>
<td>→ Directed</td>
<td>53.78±2.87</td>
<td>54.74±3.17</td>
<td>56.95±3.40</td>
<td>58.50±3.87</td>
<td>59.82±1.69</td>
</tr>
<tr>
<td>→ Undirected</td>
<td>61.45±3.49</td>
<td>63.85±2.98</td>
<td>66.87±2.57</td>
<td>68.84±2.54</td>
<td>69.27±2.54</td>
</tr>
<tr>
<td>→ LetterRecg</td>
<td>50.58±3.68</td>
<td>54.88±2.68</td>
<td>56.84±3.74</td>
<td>56.88±2.70</td>
<td>57.02±2.15</td>
</tr>
<tr>
<td>→ EMNIST</td>
<td>27.85±3.84</td>
<td>29.67±3.21</td>
<td>30.81±3.73</td>
<td>31.68±2.95</td>
<td>32.53±2.73</td>
</tr>
</tbody>
</table>

Figure 5. Effects of the number of clustering solutions on performance in terms of AC and NMI over the data set Amazon Review D,E,K → B (a)–(b), B,E,K → D (c)–(d), B,D,K → E (e)–(f) and B,D,E → K (g)–(h).

depth denotes the number of hidden layers, and the width denotes the number of neurons or the number of convolutional layers. From these figures, we observe that with the increase of depth, performances show a rapid upward trend before remaining steady. The performance gains become diminishing when the number of hidden layers is larger than a certain number. Similar observations concerning the effects of the network width have been found. It indicates that when the network structures become more complex, it requires an increasing amount of data to train this network for achieving satisfactory performance. It is worth mentioning that the width scale in

Figure 6. Effects of the number of data sets on performance in terms of AC and NMI over Amazon Review and News texts, where RC, NG, Reu and TD stand for RCV1-4, 20NG-4, Reuters-4 and TDT2-4, respectively.
Figure 7. Effects of the network depth and width on clustering performance in terms of AC and NMI over the dataset Amazon Review.

Figure 8. Effects of the network depth and width on clustering performance in terms of AC and NMI over the dataset News texts.

Figure 9. Effects of $\mu$ on performance in terms of AC and NMI over Amazon review and News texts.
5.5.2. Effects of trade-off parameters in the fusion of diffusion

We have explored the effects of $\mu$ and $\tau$ on Amazon review and News texts. $\mu$ controls the importance between the smoothness of pairwise affinity graphs and the self-similarity preservation. $\tau$ controls the distribution of the weights of pairwise affinity graphs. Figure 9 shows trends of performance when $\mu$ changes in the range between 0.1 and 0.9. Generally, when $\mu$ increases gradually, performances show upward trends before remaining steady or downward trends to various extents. The optimal $\mu$ of Amazon review is 0.5, which indicates that the smoothness of affinity graphs and the self-similarity preservation plays an approximately equal role in performance improvement. While the optimal $\mu$ of News texts is 0.6, which means that the self-similarity preservation makes a larger contribution than the above smoothness.

Figure 10 displays effects of $\tau$ on performance in terms of AC and NMI, where $\tau$ varies in the range from 15 to 135. From this figure, we observe that when $\tau$ is smaller than 75 on Amazon review or 60 on News texts, clustering performances remain steady with slight fluctuations. However, with the continuous increase of $\tau$, the performances become significantly poorer. It indicates that $\tau$ can not only preserve the discriminative knowledge in the informative affinity graphs but also suppress the negative effects of non-informative affinity graphs in a relatively wide range. Overall, compared to $\mu$, performances show more robust to $\tau$.

Besides, we have conducted experiments concerning determining the trade-off parameter. The detailed experimental results are shown in Figure 11. As described in Equation (1), $\alpha$ controls relative importance between the learning of reconstructive representations in respective tasks and the learning of common representations. From Figure 11, one can observe that with the increase of $\alpha$, the performance shows an obvious upward trend, which indicates that the learning of common representations is
beneficial to knowledge mining and transfer from easy to difficult tasks. When $\alpha$ is larger than a certain value, performance shows various trends with the increase of $\alpha$. For example, on data Amazon review, when $\alpha$ is larger than 0.6, performances of the cases ‘$D, E, K \rightarrow B$’ and ‘$B, E, K \rightarrow D$’ show significant downward trends, while those of the cases ‘$B, D, K \rightarrow E$’ and ‘$B, D, E \rightarrow K$’ remains relatively steady. It indicates different tasks show various robust to knowledge transfer. Differently, when $\alpha$ is larger than 0.6, performances for all the cases become much worse on data News text. Overall, a large $\alpha$ encourages the learning of common representations, from which the shared knowledge is mined and transferred. However, a large $\alpha$ results in the poor learning of reconstructive representation, decreasing the performance of each clustering task. Since tasks have various sensitivities to knowledge transfer, it is reasonable to set a small value to initially and increase its value delicately with the continuity of iterations.

5.6. Extendable exploration

We explore the feasibility in a more challenging application where the label spaces of easy data sets are not completely overlapped, but the union of unique classes from these data sets fully covers the classes of hard data set. To verify the extendability, we construct the following scenarios: (1) Easy data sets Letter1 and EMNIST1 are generated from Letter Recognition and EMNIST Letter by randomly selecting 20 classes and the number of unique classes is 26. The hard data set is Chars74K-E, which contains 26 classes. (2) Easy data sets Pen-based1 and Optical1 are generated from Pen-based Recognition and Optical Recognition by randomly selecting 8 classes and the number of unique classes is 10. The hard data set is Semion handwritten digit. (3) Easy data sets USPS1 and MNIST1 are generated from USPS and MNIST by randomly selecting eight classes and the number of unique classes is 10. The hard data set is SVHN. The extendable exploration experiments in the above scenarios are shown in Figure 12, which compares our proposed method with DEC, MDAN-hard and MDAN-soft. From this figure, we observe that DEC achieves the worst performances. The reasons are that (1) DEC fails to learn the discriminative features when the label spaces of data sets are not overlapped and (2) DEC does not contain the adversarial mechanism to generate diverse representations for information enrichment. On the contrary, our method outperforms the other competitors over all the cases. It indicates that our method can not only utilize the knowledge from multiple data sets by exploring the common features as what MDAN-hard and MDAN-soft do, but also better integrate the knowledge from multi-affinity graphs obtained by the feature projection matrices.

6. Conclusion

In this paper, we propose a framework for generating diverse clustering solutions, followed by conducting solution retrieval to improve the clustering performance. More specifically, we first utilize a feature extractor to learn compact representations for multiple domains, thus each representation in a difficult domain is represented with a combination of others in easy domains. Second, we conduct clustering on these representations, based on which a conditional GAN is introduced to generate more diverse representations conditioned on the clustering pseudo-labels. The synthetic representations facilitate finding a better embedding space. Third, we learn the feature projection and partition matrices on batches of the above representations and consider feature projection matrices as clustering solutions that are input into another GAN to generate more clustering solutions. Finally, these solutions are merged by the fusion of diffusion, followed by performing a transformation on latent representations. We conduct extensive comparative experiments against state-of-the-art methods on multiple real-world applications, and experimental results demonstrate the effectiveness of our proposed framework. Since the proposed approach is not an end-to-end clustering solution generation scheme, resulting in lower efficiency, we will design an end-to-end scheme to mitigate it. Meanwhile, we shall improve a better fusion strategy to integrate the information of clustering solution to promote knowledge transfer from easy source tasks to hard ones.

Disclosure statement

No potential conflict of interest was reported by the author(s).

References


of the IEEE international conference on computer vision (pp. 5067–5075).


