Towards K-means-friendly Spaces: Simultaneous Deep Learning and Clustering

Bo Yang [1] **Xiao Fu** [1] **Nicholas D. Sidiropoulos** [1] **Mingyi Hong** [2]

Abstract

Most learning approaches treat dimensionality reduction (DR) and clustering separately (i.e., sequentially), but recent research has shown that optimizing the two tasks jointly can substantially improve the performance of both. The premise behind the latter genre is that the data samples are obtained via linear transformation of latent representations that are easy to cluster; but in practice, the transformation from the latent space to the data can be more complicated. In this work, we assume that this transformation is an unknown and possibly *nonlinear* function. To recover the 'clustering-friendly' latent representations and to better cluster the data, we propose a joint DR and K-means clustering approach in which DR is accomplished via learning a deep neural network (DNN). The motivation is to keep the advantages of jointly optimizing the two tasks, while exploiting the deep neural network's ability to approximate any nonlinear function. This way, the proposed approach can work well for a broad class of generative models. Towards this end, we carefully design the DNN structure and the associated joint optimization criterion, and propose an effective and scalable algorithm to handle the formulated optimization problem. Experiments using different real datasets are employed to showcase the effectiveness of the proposed approach.

1. Introduction

Clustering is one of the most fundamental tasks in data mining and machine learning, with an endless list of applications. It is also a notoriously hard task, whose outcome is affected by a number of factors – including data acquisition and representation, use of preprocessing such as dimensionality reduction (DR), the choice of clustering criterion and optimization algorithm, and initialization (Ertoz et al., 2003; Banerjee et al., 2005). Since its introduction in 1957 by Lloyd (published much later in 1982 (Lloyd, 1982)), K-means has been extensively used either alone or together with suitable preprocessing, due to its simplicity and effectiveness. K-means is suitable for clustering data samples that are evenly spread around some centroids (cf. the first subfigure in Fig. 1), but many real-life datasets do not exhibit this 'K-means-friendly' structure. Much effort has been spent on mapping high-dimensional data to a certain space that is suitable for performing K-means. Various techniques, including principal component analysis (PCA), canonical correlation analysis (CCA), nonnegative matrix factorization (NMF) and sparse coding (dictionary learning), were adopted for this purpose. In addition to these linear DR operators (e.g., a projection matrix), nonlinear DR techniques such as those used in spectral clustering (Ng et al., 2002) and sparse subspace clustering (Elhamifar & Vidal, 2013; You et al., 2016) have also been considered.

In recent years, motivated by the success of deep neural networks (DNNs) in *supervised* learning, *unsupervised* deep learning approaches are now widely used for DR prior to clustering. For example, the stacked autoencoder (SAE) (Vincent et al., 2010), deep CCA (DCCA) (Andrew et al., 2013), and sparse autoencoder (Ng, 2011) take insights from PCA, CCA, and sparse coding, respectively, and make use of DNNs to learn nonlinear mappings from the data domain to low-dimensional latent spaces. These approaches treat their DNNs as a preprocessing stage that is separately designed from the subsequent clustering stage. The hope is that the latent representations of the data learned by these DNNs will be naturally suitable for clustering. However, since no clustering-promoting objective is explicitly incorporated in the learning process, the learned DNNs do not necessarily output reduced-dimension data that are suitable

[1]Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis MN 55455, USA. [2]Department of Industrial and Manufacturing Systems Engineering, Iowa State University, Ames, IA 50011, USA. Correspondence to: Bo Yang <yang4173@umn.edu>, Xiao Fu <xfu@umn.edu>, Nicholas D. Sidiropoulos <nikos@ece.um.edu>, Mingyi Hong <mingyi@iastate.edu>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

for clustering – as will be seen in our experiments.

In (De Soete & Carroll, 1994; Patel et al., 2013; Yang et al., 2017), joint DR and clustering was considered. The rationale behind this line of work is that if there exists *some* latent space where the entities nicely fall into clusters, then it is natural to seek a DR transformation that reveals such structure, i.e., which yields a low K-means clustering cost. This motivates using the K-means cost in latent space as a prior that helps choose the right DR, and pushes DR towards producing K-means-friendly representations. By performing joint DR and K-means clustering, impressive clustering results have been observed in (Yang et al., 2017). The limitation of these works is that the observable data is assumed to be generated from the latent clustering-friendly space via simple linear transformation. While simple linear transformation works well in many cases, there are other cases where the generative process is more complex, involving a nonlinear mapping.

Contributions In this work, we propose a joint DR and K-means clustering framework, where the DR part is implemented through learning a DNN, rather than a linear model. Unlike previous attempts that utilize this joint DNN and clustering idea, we made customized design for this *unsupervised* task. Although implementing this idea is highly non-trivial (much more challenging than (De Soete & Carroll, 1994; Patel et al., 2013; Yang et al., 2017) where the DR part only needs to learn a linear model), our objective is well-motivated: by better modeling the data transformation process with a more general model, a much more K-means-friendly latent space can be learned – as we will demonstrate. A sneak peek of the kind of performance that can be expected using our proposed method can be seen in Fig. 1, where we generate four clusters of 2-D data which are well separated in the 2-D Euclidean space and then transform them to a 100-D space using a complex non-linear mapping [cf. (9)] which destroys the cluster structure. One can see that the proposed algorithm outputs reduced-dimension data that are most suitable for applying K-means. Our specific contributions are as follows:

• **Optimization Criterion Design**: We propose an optimization criterion for joint DNN-based DR and K-means clustering. The criterion is a combination of three parts, namely, dimensionality reduction, data reconstruction, and cluster structure-promoting regularization. We deliberately include the reconstruction part and implement it using a decoding network, which is crucial for avoiding trivial solutions. The criterion is also flexible – it can be extended to incorporate different DNN structures (e.g. convolutional neural networks (LeCun et al., 1998; Krizhevsky et al., 2012)) and clustering criteria, e.g., subspace clustering.

• **Effective and Scalable Optimization Procedure**: The formulated optimization problem is very challenging to handle, since it involves layers of nonlinear activation functions and integer constraints that are induced by the K-means part. We propose a judiciously designed solution package, including empirically effective initialization and a novel alternating stochastic gradient algorithm. The algorithmic structure is simple, enables online implementation, and is very scalable.

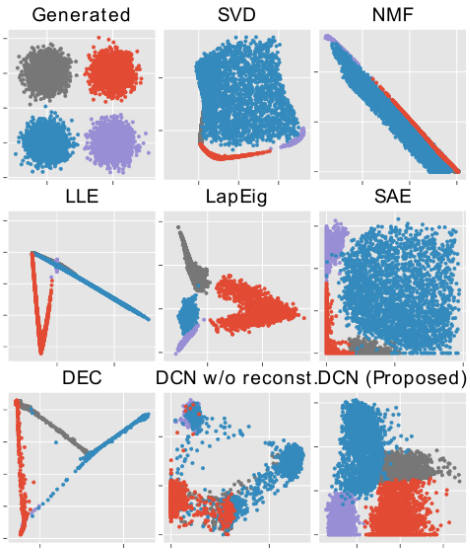


Figure 1. The learned 2-D reduced-dimension data by different methods. The observable data is in the 100-D space and is generated from 2-D data (cf. the first subfigure) through the nonlinear transformation in (9). The true cluster labels are indicated using different colors.

• **Comprehensive Experiments and Validation**: We provide a set of synthetic-data experiments and validate the method on different real datasets including various document and image copora. Evidently visible improvement from the respective state-of-art is observed for all the datasets that we experimented with.

• **Reproducibility**: The code for the experiments is available at `https://github.com/boyangumn/DCN`.

2. Background and Related Works

Given a set of data samples $\{\boldsymbol{x}_i\}_{i=1,\ldots,N}$ where $\boldsymbol{x}_i \in \mathbb{R}^M$, the task of clustering is to group the N data samples into K categories. Arguably, K-means (Lloyd, 1982) is the most widely adopted algorithm. K-means approaches this task by optimizing the following cost function:

$$\min_{\boldsymbol{M}\in\mathbb{R}^{M\times K},\{\boldsymbol{s}_i\in\mathbb{R}^K\}} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{M}\boldsymbol{s}_i\|_2^2 \qquad (1)$$
$$\text{s.t. } s_{j,i} \in \{0,1\},\ \mathbf{1}^T\boldsymbol{s}_i = 1\ \ \forall i,j,$$

where $\boldsymbol{s}_i$ is the assignment vector of data point i which has only one non-zero element, $s_{j,i}$ denotes the jth element of $\boldsymbol{s}_i$, and the kth column of $\boldsymbol{M}$, i.e., $\boldsymbol{m}_k$, denotes the centroid of the kth cluster.

K-means works well when the data samples are evenly scattered around their centroids in the feature space; we consider datasets which have this structure as being 'K-means-friendly' (cf. top-left subfigure of Fig. 1). However, high-dimensional data are in general not very K-means-friendly. In practice, using a DR pre-processing, e.g., PCA or NMF (Xu et al., 2003; Cai et al., 2011), to reduce the dimension of $\boldsymbol{x}_i$ to a much lower dimensional space and then apply K-means usually gives better results. In addition to the above classic DR methods that essentially learn a linear generative model from the latent space to the data domain, nonlinear DR approaches such as those used in spectral clustering (Ng et al., 2002; Von Luxburg, 2007) and DNN-based DR (Hinton & Salakhutdinov, 2006; Schroff et al., 2015; Hershey et al., 2016) are also widely used as pre-processing before K-means or other clustering algorithms, see also (Vincent et al., 2010; Bruna & Mallat, 2013).

Instead of using DR as a pre-processing, joint DR and clustering was also considered in the literature (De Soete & Carroll, 1994; Patel et al., 2013; Yang et al., 2017). This line of work can be summarized as follows. Consider the generative model where a data sample is generated by $\boldsymbol{x}_i = \boldsymbol{W}\boldsymbol{h}_i$, where $\boldsymbol{W} \in \mathbb{R}^{M\times R}$ and $\boldsymbol{h}_i \in \mathbb{R}^R$, where $R \ll M$. Assume that the data clusters are well-separated in latent domain (i.e., where $\boldsymbol{h}_i$ lives) but distorted by the transformation introduced by $\boldsymbol{W}$. Reference (Yang et al., 2017) formulated the joint optimization problem as follows:

$$\begin{aligned}\min_{\boldsymbol{M},\{\boldsymbol{s}_i\},\boldsymbol{W},\boldsymbol{H}} \quad & \|\boldsymbol{X}-\boldsymbol{W}\boldsymbol{H}\|_F^2+\lambda\sum_{i=1}^N\|\boldsymbol{h}_i-\boldsymbol{M}\boldsymbol{s}_i\|_2^2 \\ & +r_1(\boldsymbol{H})+r_2(\boldsymbol{W}) \\ \text{s.t.} \quad & s_{j,i}\in\{0,1\},\ \boldsymbol{1}^T\boldsymbol{s}_i=1\ \ \forall i,j,\end{aligned} \tag{2}$$

where $\boldsymbol{X} = [\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N]$, $\boldsymbol{H} = [\boldsymbol{h}_1,\ldots,\boldsymbol{h}_N]$, and $\lambda \geq 0$ is a parameter for balancing data fidelity and the latent cluster structure. In (2), the first term performs DR and the second term performs latent clustering. The terms $r_1(\cdot)$ and $r_2(\cdot)$ are regularizations (e.g., nonnegativity or sparsity) to prevent trivial solutions, e.g., $\boldsymbol{H}\to\boldsymbol{0}\in\mathbb{R}^{R\times N}$; see details in (Yang et al., 2017).

The data model $\boldsymbol{X}\approx\boldsymbol{W}\boldsymbol{H}$ in the above line of work may be oversimplified: The data generating process can be much more complex than this linear transform. Therefore, it is well justified to seek powerful non-linear transforms, e.g. DNNs, to model this data generating process, while at the same time make use of the joint DR and clustering idea. Two recent works, (Xie et al., 2016) and (Yang et al., 2016), made such attempts.

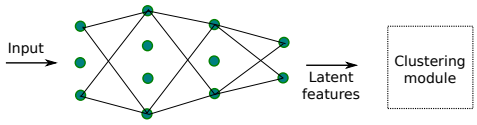


Figure 2. A problematic *joint* deep clustering structure. To avoid clutter, some links are omitted.

The idea of (Xie et al., 2016) and (Yang et al., 2016) is to connect a clustering module to the output layer of a DNN, and jointly learn DNN parameters and clusters. Specifically, the approaches look into an optimization problem of the following form

$$\min_{\mathcal{W},\Theta}\ \widehat{L}=\sum_{i=1}^N q(\boldsymbol{f}(\boldsymbol{x}_i;\mathcal{W});\Theta), \tag{3}$$

where $\boldsymbol{f}(\boldsymbol{x}_i;\mathcal{W})$ is the network output given data sample $\boldsymbol{x}_i$, $\mathcal{W}$ collects the network parameters, and Θ denotes parameters of some clustering model. For instance, Θ stands for the centroids $\boldsymbol{M}$ and assignments $\{\boldsymbol{s}_i\}$ if the K-means clustering formulation (1) is adopted. The $q(\cdot)$ in (3) denotes some clustering loss, e.g., the Kullback-Leibler (KL) divergence loss in (Xie et al., 2016) and agglomerative clustering loss in (Yang et al., 2016). An illustration of this kind of approaches is shown in Fig. 2. This idea seems reasonable, but is problematic. A *global optimal* solution to Problem (3) is $f(\boldsymbol{x}_i;\mathcal{W}) = \boldsymbol{0}$ and the optimal objective value $\widehat{L} = 0$ can always be achieved. Another type of trivial solutions are simply mapping *arbitrary* data samples to tight clusters, which will lead to a small value of $\widehat{L}$ – but this could be far from being desired since there is no provision for respecting the data samples $\boldsymbol{x}_i$'s; see the bottom-middle subfigure in Fig. 1 [Deep Clustering Network (DCN) w/o reconstruction] and the bottom-left subfigure in Fig. 1 [DEC]. This issue also exists in (Yang et al., 2016).

3. Proposed Formulation

We are motivated to model the relationship between the observable data $\boldsymbol{x}_i$ and its clustering-friendly latent representation $\boldsymbol{h}_i$ using a nonlinear mapping, i.e.,

$$\boldsymbol{h}_i=\boldsymbol{f}(\boldsymbol{x}_i;\mathcal{W}),\quad \boldsymbol{f}(\cdot;\mathcal{W}):\ \mathbb{R}^M\to\mathbb{R}^R,$$

where $\boldsymbol{f}(\cdot;\mathcal{W})$ denotes the mapping function and $\mathcal{W}$ denote the set of parameters. In this work, we propose to employ a DNN as our mapping function, since DNNs have the ability of approximating any continuous mapping using a reasonable number of parameters (Hornik et al., 1989).

We want to learn the DNN and perform clustering *simultaneously*. The critical question here is how to avoid trivial solutions in this *unsupervised* task. In fact, this can be

resolved by taking insights from (2). The key to prevent trivial solution in the linear DR case lies in the reconstruction part, i.e., the term $\|\boldsymbol{X} - \boldsymbol{W}\boldsymbol{H}\|_F^2$ in (2). This term ensures that the learned $\boldsymbol{h}_i$'s can (approximately) reconstruct the $\boldsymbol{x}_i$'s using the basis $\boldsymbol{W}$. This motivates incorporating a reconstruction term in the joint DNN-based DR and K-means. In the realm of unsupervised DNN, there are several well-developed approaches for reconstruction – e.g., the stacked autoencoder (SAE) is a popular choice for serving this purpose. To prevent trivial low-dimensional representations such as all-zero vectors, SAE uses a decoding network $\boldsymbol{g}(\cdot;\mathcal{Z})$ to map the $\boldsymbol{h}_i$'s back to the data domain and requires that $\boldsymbol{g}(\boldsymbol{h}_i;\mathcal{Z})$ and $\boldsymbol{x}_i$ match each other well under some metric, e.g., mutual information or least squares-based measures.

By the above reasoning, we come up with the following cost function:

$$\min_{\substack{\mathcal{W},\mathcal{Z},\\ \boldsymbol{M},\{\boldsymbol{s}_i\}}} \sum_{i=1}^{N} \left(\ell\left(\boldsymbol{g}(\boldsymbol{f}(\boldsymbol{x}_i)), \boldsymbol{x}_i\right) + \frac{\lambda}{2} \left\|\boldsymbol{f}(\boldsymbol{x}_i) - \boldsymbol{M}\boldsymbol{s}_i\right\|_2^2 \right) \tag{4}$$

$$\text{s.t. } s_{j,i} \in \{0,1\},\ \mathbf{1}^T s_i = 1\ \ \forall i,j,$$

where we have simplified the notation $\boldsymbol{f}(\boldsymbol{x}_i;\mathcal{W})$ and $\boldsymbol{g}(\boldsymbol{h}_i;\mathcal{Z})$ to $\boldsymbol{f}(\boldsymbol{x}_i)$ and $\boldsymbol{g}(\boldsymbol{h}_i)$, respectively, for conciseness. The function $\ell(\cdot) : \mathbb{R}^M \to \mathbb{R}$ is a certain loss function that measures the reconstruction error. In this work, we adopt the least-squares loss $\ell(\boldsymbol{x},\boldsymbol{y}) = \|\boldsymbol{x}-\boldsymbol{y}\|_2^2$; other choices such as ℓ_1-norm based fitting and the KL divergence can also be considered. $\lambda \geq 0$ is a regularization parameter which balances the reconstruction error versus finding K-means-friendly latent representations.

Fig. 3 presents the network structure corresponding to the formulation in (4). Compare to the network in Fig. 2, our latent features are also responsible for reconstructing the input, preventing all the aforementioned trivial solutions. On the left-hand side of the 'bottleneck' layer are the so-called encoding or forward layers that transform raw data to a low-dimensional space. On the right-hand side are the 'decoding' layers that try to reconstruct the data from the latent space. The K-means task is performed at the bottleneck layer. The forward network, the decoding network, and the K-means cost are optimized simultaneously. In our experiments, the structure of the decoding networks is a 'mirrored version' of the encoding network, and for both the encoding and decoding networks, we use the *rectified linear unit* (ReLU) activation-based neurons (Nair & Hinton, 2010). Since our objective is to perform DNN-driven K-means clustering, we will refer to the network in Fig. 3 as the Deep Clustering Network (DCN) in the sequel.

We should remark that the proposed optimization criterion in (4) and the network in Fig. 3 are very flexible: Other types of networks, e.g., deep convolutional neural networks

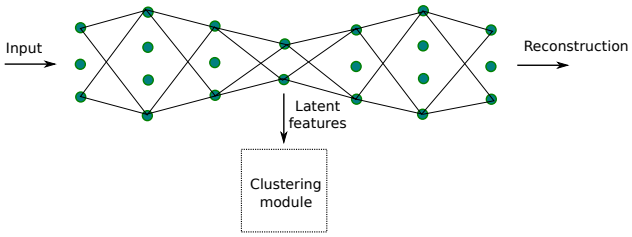


Figure 3. Proposed deep clustering network (DCN).

(LeCun et al., 1998; Krizhevsky et al., 2012), can be used. For the clustering part, other clustering criteria, e.g., K-subspace and soft K-means (Law et al., 2005; Banerjee et al., 2005), are also viable options. Nevertheless, we will concentrate on the proposed DCN in the sequel, as our interest is to provide a proof-of-concept rather than exhausting the possibilities of combinations.

4. Optimization Procedure

Optimizing (4) is highly non-trivial since both the cost function and the constraints are non-convex. In addition, there are scalability issues that need to be taken into account. In this section, we propose a pragmatic optimization procedure including an empirically effective initialization method and an alternating optimization based algorithm for handling (4).

4.1. Initialization via Layer-wise Pre-Training

For dealing with hard non-convex optimization problems like that in (4), initialization is usually crucial. To initialize the parameters of the network, i.e., $(\mathcal{W},\mathcal{Z})$, we use the layer-wise pre-training method as in (Bengio et al., 2007) for training autoencoders. This pre-training technique may be avoided in large-scale supervised learning tasks. For the proposed DCN which is completely unsupervised, however, we find that the layer-wise pre-training procedure is important no matter the size of the dataset. We refer the readers to (Bengio et al., 2007) for an introduction of layer-wise pre-training. After pre-training, we perform K-means to the outputs of the bottleneck layer to obtain initial values of $\boldsymbol{M}$ and $\{\boldsymbol{s}_i\}$.

4.2. Alternating Stochastic Optimization

Even with a good initialization, handling Problem (4) is still very challenging. The commonly used stochastic gradient descent (SGD) algorithm cannot be directly applied to jointly optimize $\mathcal{W},\mathcal{Z},\boldsymbol{M}$ and $\{\boldsymbol{s}_i\}$ because the block variable $\{\boldsymbol{s}_i\}$ is constrained on a discrete set. Our idea is to combine the insights of alternating optimization and SGD. Specifically, we propose to optimize the subproblems with respect to (w.r.t.) one of $\boldsymbol{M}$, $\{\boldsymbol{s}_i\}$ and $(\mathcal{W},\mathcal{Z})$ while keeping the other two sets of variables fixed.

4.2.1. UPDATE NETWORK PARAMETERS

For fixed $(\boldsymbol{M}, \{\boldsymbol{s}_i\})$, the subproblem w.r.t. $(\mathcal{W}, \mathcal{Z})$ is similar to training an SAE – but with an additional penalty term on the clustering performance. We can take advantage of the mature tools for training DNNs, e.g., back-propagation based SGD and its variants. To implement SGD for updating the network parameters, we look at the problem w.r.t. the incoming data $\boldsymbol{x}_i$:

$$\min_{\mathcal{W},\mathcal{Z}} L^i = \ell\left(\boldsymbol{g}(\boldsymbol{f}(\boldsymbol{x}_i)), \boldsymbol{x}_i\right) + \frac{\lambda}{2}\left\|\boldsymbol{f}(\boldsymbol{x}_i) - \boldsymbol{M}\boldsymbol{s}_i\right\|_2^2. \quad (5)$$

The gradient of the above function over the network parameters is easily computable, i.e., $\nabla_{\mathcal{X}} L^i = \frac{\partial \ell(\boldsymbol{g}(\boldsymbol{f}(\boldsymbol{x}_i)), \boldsymbol{x}_i)}{\partial \mathcal{X}} + \lambda \frac{\partial \boldsymbol{f}(\boldsymbol{x}_i)}{\partial \mathcal{X}}(\boldsymbol{f}(\boldsymbol{x}_i) - \boldsymbol{M}\boldsymbol{s}_i)$, where $\mathcal{X} = (\mathcal{W}, \mathcal{Z})$ is a collection of the network parameters and the gradients $\frac{\partial \ell}{\partial \mathcal{X}}$ and $\frac{\partial \boldsymbol{f}(\boldsymbol{x}_i)}{\partial \mathcal{X}}$ can be calculated by back-propagation (Rumelhart et al., 1988) (strictly speaking, what we calculate here is the *subgradient* w.r.t. $\mathcal{X}$ since the ReLU function is non-differentible at zero). Then, the network parameters are updated by

$$\mathcal{X} \leftarrow \mathcal{X} - \alpha \nabla_{\mathcal{X}} L^i, \quad (6)$$

where $\alpha > 0$ is a diminishing learning rate.

4.2.2. UPDATE CLUSTERING PARAMETERS

For fixed network parameters and $\boldsymbol{M}$, the assignment vector of the current sample, i.e., $\boldsymbol{s}_i$, can be naturally updated in an online fashion. Specifically, we update $\boldsymbol{s}_i$ as follows:

$$s_{j,i} \leftarrow \begin{cases} 1, & \text{if } j = \underset{k=\{1,\ldots,K\}}{\arg\min} \left\|\boldsymbol{f}(\boldsymbol{x}_i) - \boldsymbol{m}_k\right\|_2, \\ 0, & \text{otherwise.} \end{cases} \quad (7)$$

When fixing $\{\boldsymbol{s}_i\}$ and $\mathcal{X}$, the update of $\boldsymbol{M}$ is simple and may be done in a variety of ways. For example, one can simply use $\boldsymbol{m}_k = (1/|\mathcal{C}_k^i|)\sum_{i\in\mathcal{C}_k^i} \boldsymbol{f}(\boldsymbol{x}_i)$, where $\mathcal{C}_k^i$ is the recorded index set of samples assigned to cluster k from the first sample to the current sample i. Although the above update is intuitive, it could be problematic for online algorithms, since the already appeared historical data (i.e., $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_i$) might not be representative enough to model the global cluster structure and the initial $\boldsymbol{s}_i$'s might be far away from being correct. Therefore, simply averaging the current assigned samples may cause numerical problems. Instead of doing the above, we employ the idea in (Sculley, 2010) to adaptively change the learning rate of updating $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_K$. The intuition is simple: assume that the clusters are roughly balanced in terms of the number of data samples they contain. Then, after updating $\boldsymbol{M}$ for a number of samples, one should update the centroids of the clusters that already have many assigned members more gracefully while updating others more aggressively, to keep balance. To implement this, let c_k^i be the count of the number of times the algorithm assigned a sample to cluster k before handling the incoming sample $\boldsymbol{x}_i$, and update $\boldsymbol{m}_k$ by a simple gradient step:

$$\boldsymbol{m}_k \leftarrow \boldsymbol{m}_k - (1/c_k^i)\left(\boldsymbol{m}_k - \boldsymbol{f}(\boldsymbol{x}_i)\right) s_{k,i}, \quad (8)$$

where the gradient step size $1/c_k^i$ controls the learning rate. The above update of $\boldsymbol{M}$ can also be viewed as an SGD step, thereby resulting in an overall alternating block SGD procedure that is summarized in Algorithm 1. Note that an epoch corresponds to a pass of all data samples through the network.

Algorithm 1 Alternating SGD

1: Initialization {Perform T epochs over the data}
2: **for** $t = 1 : T$ **do**
3: Update network parameters by (6)
4: Update assignment by (7)
5: Update centroids by (8)
6: **end for**

Algorithm 1 has many favorable properties. First, it can be implemented in a completely online fashion, and thus is very scalable. Second, many known tricks for enhancing performance of DNN training can be directly used. In fact, we have used a mini-batch version of SGD and batch-normalization (Ioffe & Szegedy, 2015) in our experiments, which indeed help improve performance.

5. Experiments

In this section, we use synthetic and real-world data to showcase the effectiveness of DCN. We implement DCN using the deep learning toolbox Theano (Theano Development Team, 2016).

5.1. Synthetic-Data Demonstration

Our settings are as follows: Assume that the data points have K-means-friendly structure in a two-dimensional domain (cf. the first subfigure of Fig. 1). This two-dimensional domain is a latent domain which we do not observe and we denote the latent representations of the data points as $\boldsymbol{h}_i$'s in this domain. What we observe is $\boldsymbol{x}_i \in \mathbb{R}^{100}$ that is obtained via the following transformation:

$$\boldsymbol{x}_i = \sigma\left(\boldsymbol{U}\sigma(\boldsymbol{W}\boldsymbol{h}_i)\right), \quad (9)$$

where $\boldsymbol{W} \in \mathbb{R}^{10\times 2}$ and $\boldsymbol{U} \in \mathbb{R}^{100\times 10}$ are matrices whose entries follow the zero-mean unit-variance i.i.d. Gaussian distribution, $\sigma(\cdot)$ is a sigmod function to introduce nonlinearity. Under the above generative model, recovering the K-means-friendly domain where $\boldsymbol{h}_i$'s live seems very challenging.

We generate four clusters, each of which has 2,500 samples and their geometric distribution on the 2-D plane is shown in the first subfigure of Fig. 1 that we have seen before. The other subfigures show the recovered 2-D data from $\boldsymbol{x}_i$'s using a number of DR methods, namely, NMF (Lee & Seung, 1999), local linear embedding (LLE) (Saul & Roweis, 2003), Laplacian eigenmap (LapEig) (Ng et al., 2002) – the first step of spectral clustering, and DEC (Xie et al., 2016). We also present the result of using the formulation in (3) (DCN w/o reconstruction) which is a similar idea as in (Xie et al., 2016). For the three DNN-based methods (DCN, DEC, and SAE + KM), we use a four-layer forward network for dimensionality reduction, where the layers have 100, 50, 10 and 2 neurons, respectively; the reconstruction network used in DCN and SAE (and also in the per-training stage of DEC) is a mirrored version of the forward network. As one can see in Fig. 1, all the DR methods except the proposed DCN fail to map $\boldsymbol{x}_i$'s to a 2-D domain that is suitable for applying K-means. In particular, DEC and DCN w/o reconstruction indeed give trivial solutions: the reduced-dimension data are separated to four clusters, and thus $\hat{L}$ is small. But this solution is meaningless since the data partitioning is arbitrary.

In the supplementary materials, two additional simulations with different generative model than (9) are presented, and similar results are observed. This further illustrates the DCN's ability of recovering clustering-friendly structure under different nonlinear generative models.

5.2. Real-Data Validation

In this section, we validate the proposed approach on several real-data sets which are all publicly available.

5.2.1. Baseline methods

We compare the proposed DCN with a variety of baseline methods:

1) **K-means (KM)**: The classic K-means (Lloyd, 1982).

2) **Spectral Clustering (SC)**: The classic SC algorithm (Ng et al., 2002).

3) **Sparse Subspace Clustering with Orthogonal Matching Pursuit (SSC-OMP)** (You et al., 2016): SSC is considered very competitive for clustering images; we use the newly proposed greedy version here for scalability.

4) **Locally Consistent Concept Factorization (LCCF)** (Cai et al., 2011): LCCF is based on NMF with a graph Laplacian regularization and is considered state-of-the-art for document clustering.

5) **XRAY** (Kumar et al., 2013): XRAY is an NMF-based document clustering algorithm that scales very well.

6) **NMF followed by K-means** (NMF+KM): This approach applies NMF for DR, and then applies K-means to the reduced-dimension data.

7) **Stacked Autoencoder followed by K-means (SAE+KM)**: This is also a two-stage approach. We use SAE for DR first and then apply K-means.

8) **Joint NMF and K-means (JNKM)** (Yang et al., 2017): JNKM performs joint DR and K-means clustering as the proposed DCN does – but the DR part is based on NMF.

9) **Deep Embedded Clustering (DEC)** (Xie et al., 2016): DEC performs joint DNN and clustering, where the loss function contains only clustering loss, without penalty on reconstruction as in our method. We use the code[1] provided by the authors. For each experiment, we select the baselines that are considered most competitive and suitable for that application from the above pool.

5.2.2. Evaluation metrics

We adopt standard metrics for evaluating clustering performance. Specifically, we employ the following three metrics: normalized mutual information (NMI) (Cai et al., 2011), adjusted Rand index (ARI) (Yeung & Ruzzo, 2001), and clustering accuracy (ACC) (Cai et al., 2011). In a nutshell, all the above three measuring metrics are commonly used in the clustering literature, and all have pros and cons. But using them together suffices to demonstrate the effectiveness of the clustering algorithms. Note that NMI and ACC lie in the range of zero to one with one being the perfect clustering result and zero the worst. ARI is a value within -1 to 1, with one being the best clustering performance and minus one the opposite.

5.2.3. RCV1

We first test the algorithms on a large-scale text corpus, namely, the Reuters Corpus Volume 1 Version 2 (RCV1-v2). The RCV1-v2 corpus (Lewis et al., 2004) contains 804,414 documents, which were manually categorized into 103 different topics. We use a subset of the documents from the whole corpus. This subset contains 20 topics and $365,968$ documents and each document has a single topic label. As in (Nitish et al., 2014), we pick the 2,000 most frequently used words (in the tf-idf form) as the features of the documents.

We conduct experiments using different number of clusters. Towards this end, we first sort the clusters according to the number of documents that they have in a descending order, and then apply the algorithms to the first 4, 8, 12, 16, 20 clusters, respectively. Note that the first several clusters have many more documents compared to the other clusters (cf. Fig. 4). This way, we gradually increase the number of documents in our experiments and create cases with much more unbalanced cluster sizes for testing the algorithms – which means we gradually increase the difficulty of the experiments. To avoid unrealistic tuning, for all the experiments, we use a DCN whose forward network has five hidden layers which have $2000, 1000, 1000, 1000, 50$ neurons, respectively. The reconstruction network has a mirrored structure. We set $\lambda = 0.1$ for balancing the reconstruction

[1] https://github.com/piiswrong/dec

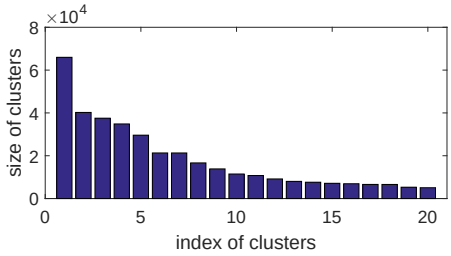


Figure 4. The sizes of 20 clusters in the experiment.

Table 1. Evaluation on the RCV1-v2 dataset

Methods		DCN	SAE+KM	KM	DEC	XRAY
4 Clust.	NMI	**0.76**	0.73	0.62	0.11	0.12
	ARI	**0.67**	0.65	0.50	0.07	-0.01
	ACC	**0.80**	0.79	0.70	0.38	0.34
8 Clust.	NMI	**0.63**	0.60	0.57	0.10	0.24
	ARI	**0.46**	0.42	0.38	0.05	0.09
	ACC	**0.63**	0.62	0.59	0.24	0.39
12 Clust.	NMI	**0.67**	0.65	0.6	0.09	0.22
	ARI	**0.52**	0.51	0.37	0.02	0.05
	ACC	**0.60**	0.56	0.54	0.18	0.29
16 Clust.	NMI	**0.62**	0.60	0.56	0.09	0.23
	ARI	**0.36**	0.35	0.30	0.02	0.04
	ACC	**0.51**	0.50	0.48	0.17	0.29
20 Clust.	NMI	**0.61**	0.59	0.58	0.08	0.25
	ARI	**0.33**	**0.33**	0.29	0.01	0.04
	ACC	**0.47**	0.46	**0.47**	0.14	0.28

error and the clustering regularization.

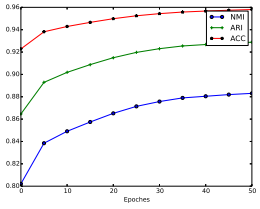
Table 1 shows the results given by the proposed DCN, SAE+KM, KM, and XRAY; other baselines are not scalable enough to handle the RCV1-v2 dataset and thus are dropped. One can see that for each case that we have tried, the proposed method gives clear improvement relative to the other methods. Particularly, the DCN approach outperforms the two-stage approach, i.e., SAE+KM, in almost all the cases and for all the evaluation metrics – this clearly demonstrates the advantage of using the joint optimization criterion. We notice that the performance of DEC in this experiment is unsatisfactory, possibly because 1) this dataset is highly unbalanced (cf. Fig. 4), while DEC is designed to produce balanced clusters; 2) DEC gets trapped in trivial solutions, as we discussed in Sec 2.

Fig. 5a shows how NMI, ARI, and ACC change when the proposed algorithm runs from epoch to epoch. One can see a clear ascending trend of every evaluation metric. This result shows that both the network structure and the optimization algorithm work towards a desired direction. In the future, it would be intriguing to derive (sufficient) conditions for guaranteeing such improvement using the proposed algorithm. Nevertheless, such empirical observation in Fig. 5a is already very interesting and encouraging.

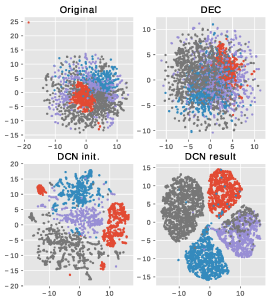
We visualize the 50-D learned embeddings of our network on the RCV1 4-clusters dataset, using t-SNE (Van der Maaten & Hinton, 2008), as shown in Fig. 5b. We can see that the proposed DCN method learns much improved results compared to the initialization. Also, the DEC method does not get a desiable clustering result, possibly due to the imbalance clusters.

Table 2. Evaluation on the 20Newsgroup dataset.

Methods	DCN	SAE+KM	LCCF	NMF+KM	KM	SC	XARY	JNKM
NMI	**0.48**	0.47	0.46	0.39	0.41	0.40	0.19	0.40
ARI	**0.34**	0.28	0.17	0.17	0.15	0.17	0.02	0.10
ACC	**0.44**	0.42	0.32	0.33	0.3	0.34	0.18	0.24



(a) Clustering performance metrics v.s. training epochs.



(b) Visualization using t-SNE. From top-left to bottom-right: Original data, DEC result, DCN initialization, DCN result

Figure 5. Visualization on the 4-clusters subset of RCV1-v2

5.2.4. 20NEWSGROUP

The 20Newsgroup corpus is a collection of 18,846 text documents which are partitioned into 20 different newsgroups. Using this corpus, we can observe how the proposed method works with a relatively small amount of samples. As the previous experiment, we use the tf-idf representation of the documents and pick the 2,000 most frequently used words as the features. Since this dataset is small, we include more baselines that are not scalable enough for RCV1-v2. Among them, both JNKM and LCCF are considered state-of-art for document clustering. In this experiment, we use a DNN with three forward layers which have 250, 100, and 20 neurons, respectively. This is a relatively 'small network' since the 20Newsgroup corpus may not have sufficient samples to fit a large network. As before, the decoding network for reconstruction has a mirrored structure of the encoding part, and the baseline SAE+KM uses the same network for the autoencoder part.

Table 2 summarizes the results of this experiment. As one can see, LCCF indeed gives the best performance among the algorithms that do not use DNNs. SAE+KM improves ARI and ACC quite substantially by involving DNN – this suggests that the generative model may indeed be nonlinear. DCN performs even better by using the proposed joint DR and clustering criterion, which supports our motivation that a K-means regularization can help discover a clustering-friendly space.

5.2.5. RAW MNIST

In this and next subsections, we present two experiments using two versions of the MNIST dataset. We first employ

Table 3. Evaluation on the raw MNIST dataset.

Methods	DCN	SAE+KM	DEC	KM	SSC-OMP
NMI	**0.81**	0.73	0.80	0.50	0.31
ARI	**0.75**	0.67	**0.75**	0.37	0.13
ACC	0.83	0.80	**0.84**	0.53	0.30

the raw MNIST dataset that has 70,000 data samples. Each sample is a 28×28 gray-scale image containing a handwritten digit, i.e., one of $\{0, 1, \ldots, 9\}$. Same as (Xie et al., 2016), we use a 4-layers forward network and set the number of neurons to be 500, 500, 2000, and 10, respectively. The reconstruction network is still a 'mirrored' version of the forward network. The hyperparameter λ is set to 1. We use SSC-OMP, which is a scalable version of SSC, and KM as a baseline for this experiment.

Table 3 shows results of applying DCN, SAE+KM, DEC, KM and SSC-OMP to the raw MNIST data – the other baselines are not efficient enough to handle 70,000 samples and thus are left out. One can see that our result is on par with the result of DEC reported in (Xie et al., 2016), and both methods outperform other methods by a large margin. The DEC method performs very competitively on this dataset, possibly because it is designed to favor balanced clusters, which is the case for MNIST dataset. On the dataset RCV1-v2 with unbalanced clusters, the result of DEC is not as satisfactory, see Fig. 5b. It is also interesting to note that our method yields approximately same results as DEC in this balanced case, but DCN also works well in unbalanced cases, as we have seen.

5.2.6. Pre-Processed MNIST

Besides the above experiment using the raw MNIST data, we also provide another interesting experiment using *pre-processed* MNIST data. The pre-processing is done by a recently introduced technique, namely, the scattering network (ScatNet) (Bruna & Mallat, 2013). ScatNet is a cascade of multiple layers of wavelet transform, which is able to learn a good feature space for clustering / classification of images. Utilizing ScatNet, the work in (You et al., 2016) reported very promising clustering results on MNIST using SSC-OMP. Our objective here is to see if the proposed DCN can further improve the performance from SSC-OMP. Our idea is simple: SSC-OMP is essentially a procedure of constructing a similarity matrix of the data; after obtaining this matrix, it performs K-means on the rows of a matrix comprising several selected eigenvectors of the similarity matrix (Ng et al., 2002). Therefore, it makes sense to treat the whole ScatNet + SSC-OMP procedure as pre-processing for performing K-means, and one can replace K-means by DCN to improve performance.

The results are shown in Table 4. One can see that the proposed method exhibits the best performance among the algorithms. We note that the result of using KM on the data processed by ScatNet and SSC-OMP is worse than that was reported in (You et al., 2016). This is possibly because we use all the 70,000 samples, while only a subset was selected for conducting the experiments in (You et al., 2016).

Table 4. Evaluation on pre-processed MNIST

Methods	DCN	SAE+KM	KM (SSC-OMP)
NMI	**0.88**	0.86	0.85
ARI	**0.89**	0.86	0.82
ACC	**0.95**	0.93	0.86

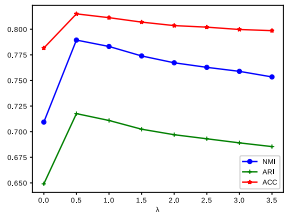


Figure 6. Clustering performance on MNIST with different λ.

This experiment is particularly interesting since it suggests that for any clustering algorithm that employs K-means as a key component, e.g., spectral clustering and sparse subspace clustering, one can use the proposed DCN to replace K-means and a better result can be expected. This is meaningful since many datasets are originally not suitable for K-means due to the nature of the data – but after pre-processing (e.g., kernelization and eigendecomposition), the pre-processed data is already more K-means-friendly, and using the proposed DCN at this point can further strengthen the result.

5.2.7. Parameter Selection

The parameter λ is important, since it trades off between the reconstruction objective and the clustering objective. As we see from the experiments, the proposed DCN works well with an appropriately chosen λ. Moreover, our experience suggests that the performance of our approach is insensitive to the exact value of λ. Fig. 6 shows how the proposed method performs with different λ on the MNIST dataset. As we can see, although there is degradation of performance as λ gets inappropriately large, the degradation is mild. The proposed method gives satisfactory result for a range of λ.

6. Conclusion

In this work, we proposed a joint DR and K-means clustering approach where the DR part is accomplished via learning a deep neural network. Our goal is to automatically map high-dimensional data to a latent space where K-means is a suitable tool for clustering. We carefully designed the network structure to avoid trivial and meaningless solutions and proposed an effective and scalable optimization procedure to handle the formulated challenging problem. Synthetic and real data experiments showed that the algorithm is very effective on a variety of datasets.

Acknowledgements

This work is supported by National Science Foundation under Projects NSF IIS-1447788, NSF ECCS-1608961, and NSF CCF-1526078. The GPU used in this work was kindly donated by NVIDIA.

References

Andrew, G., Arora, R., Bilmes, J., and Livescu, K. Deep canonical correlation analysis. In *Proceedings of the 30th International Conference on International Conference on Machine Learning*, pp. 1247–1255, 2013.

Banerjee, A., Merugu, S., Dhillon, I. S., and Ghosh, J. Clustering with bregman divergences. *Journal of Machine Learning Research*, 6:1705–1749, Oct 2005.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems 19 (NIPS 2006)*, volume 19, pp. 153–160, 2007.

Bruna, J. and Mallat, S. Invariant scattering convolution networks. *IEEE Transaction on Pattern Analysis Machine Intelligence*, 35(8):1872–1886, 2013.

Cai, D., He, X., and Han, J. Locally consistent concept factorization for document clustering. *IEEE Transaction on Knowledge and Data Engineering*, 23(6):902–913, 2011.

De Soete, G. and Carroll, J. D. K-means clustering in a low-dimensional euclidean space. In *New Approaches in Classification and Data Analysis*, pp. 212–219. Springer, 1994.

Elhamifar, E. and Vidal, R. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11): 2765–2781, 2013.

Ertoz, L., Steinbach, M., and Kumar, V. Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In *Proceedings of Second SIAM International Conference on Data Mining*, pp. 47–58, 2003.

Hershey, J. R., Chen, Z., Le Roux, J., and Watanabe, S. Deep clustering: Discriminative embeddings for segmentation and separation. In *Proceedings of 2016 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 31–35. IEEE, 2016.

Hinton, G. E. and Salakhutdinov, R. Reducing the dimensionality of data with neural networks. *Science*, 313 (5786):504–507, 2006.

Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 448–456, 2015.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Infomation Processing Systems 25 (NIPS 2012)*, pp. 1097–1105, 2012.

Kumar, A., Sindhwani, V., and Kambadur, P. Fast conical hull algorithms for near-separable non-negative matrix factorization. In *Proceedings of 30th International Conference on Machine Learning*, pp. 231–239, 2013.

Law, M. H. C., Topchy, A., and Jain, A. K. Model-based clustering with probabilistic constraints. In *Proceedings of the 2005 SIAM International Conference on Data Mining*, pp. 641–645. SIAM, 2005.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lee, D. D. and Seung, S. H. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755): 788–791, 1999.

Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, Apr 2004.

Lloyd, S. Least squares quantization in PCM. *IEEE Transaction on Information Theory*, 28(2):129–137, 1982.

Nair, V. and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*, pp. 807–814, 2010.

Ng, A. Y. Sparse autoencoder. *CS294A Lecture notes*, 72: 1–19, 2011.

Ng, A. Y., Jordan, M., and Weiss, Y. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, pp. 849–856, 2002.

Nitish, S., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Patel, V. M., Van Nguyen, H., and Vidal, R. Latent space sparse subspace clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 225–232, 2013.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Neurocomputing: foundations of research*, pp. 696–699, 1988.

Saul, L. K. and Roweis, S. T. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 4:119–155, 2003.

Schroff, F., Kalenichenko, D., and Philbin, J. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 815–823, 2015.

Sculley, D. Web-scale k-means clustering. In *Proceedings of the 19th International Conference on World Wide Web (WWW)*, pp. 1177–1178. ACM, 2010.

Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. URL http://arxiv.org/abs/1605.02688.

Van der Maaten, L. and Hinton, G. E. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9: 2579–2605, Nov 2008.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P. A. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371–3408, Dec 2010.

Von Luxburg, U. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

Xie, J., Girshick, R., and Farhadi, A. Unsupervised deep embedding for clustering analysis. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

Xu, W., Liu, X., and Gong, Y. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pp. 267–273. ACM, 2003.

Yang, B., Fu, X., and Sidiropoulos, N. D. Learning from hidden traits: Joint factor analysis and latent clustering. *IEEE Transaction on Signal Processing*, pp. 256–269, Jan. 2017.

Yang, J., Parikh, D., and Batra, D. Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5147–5156, 2016.

Yeung, K. Y. and Ruzzo, W. L. Details of the adjusted rand index and clustering algorithms, supplement to the paper an empirical study on principal component analysis for clustering gene expression data. *Bioinformatics*, 17(9): 763–774, 2001.

You, C., Robinson, D., and Vidal, R. Scalable sparse subspace clustering by orthogonal matching pursuit. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, 2016.

Supplementary material for "Towards K-means-friendly Spaces: Simultaneous Deep Learning and Clustering"

1. Additional Synthetic-Data Experiments

1.1. Additional Generative Models

In this section, we provide two more examples to illustrate the ability of DCN in recovering K-means-friendly spaces under different generative models. We first consider the transformation as follows:

$$\boldsymbol{x}_i = (\sigma(\boldsymbol{W}\boldsymbol{h}_i))^2, \tag{10}$$

where $\sigma(\cdot)$ is the sigmoid function as before and $\boldsymbol{W} \in \mathbb{R}^{100\times 2}$ is similarly generated as in the paper. We perform elementwise squaring on the result features to further complicate the generating process. The corresponding results can be seen in Fig. 1 of this supplementary document. One can see that a similar pattern as we have observed in the main text is also presented here: The proposed DCN recovers a 2-D K-means-friendly space very well and the other methods all fail.

In Fig. 2, we test the algorithms under the generative model

$$\boldsymbol{x}_i = \tanh\left(\sigma(\boldsymbol{W}\boldsymbol{h}_i)\right), \tag{11}$$

where $\boldsymbol{W} \in \mathbb{R}^{100\times 2}$. Same as before, the proposed DCN gives very clear clusters in the recovered 2-D space.

The results in this section and the synthetic-data experiment presented in main text are encouraging: Under a variety of complicated nonlinear generative models, DCN can output clustering-friendly latent representations.

2. Additional Real-Data Experiments

2.1. Pendigits

Beside the real datasets in the paper, we also conduct experiment on the Pendigits dataset. The Pendigits dataset consists of 10,992 data samples. Each sample records 8 coordinates on a tablet, on which a subject is instructed to write the digits from 0 to 9. So each sample corresponds to a vector of length 16, and represents one of the digits. Note that this dataset is quite different from MNIST – each digit in MNIST is represented by an image (pixel values) while digits in Pendigits are represented by 8 coordinates of the stylus when a person was writing a certain digit. Since each digit is represented by a very small-size vector of length 16, we use a small network who has three forward layers which are with 16, 16, and 10 neurons. Table 1 shows the results: The proposed methods give the best clustering performance compared to the competing methods, and the methods using DNNs outperform the 'shallow' ones that do not use neural networks for DR.

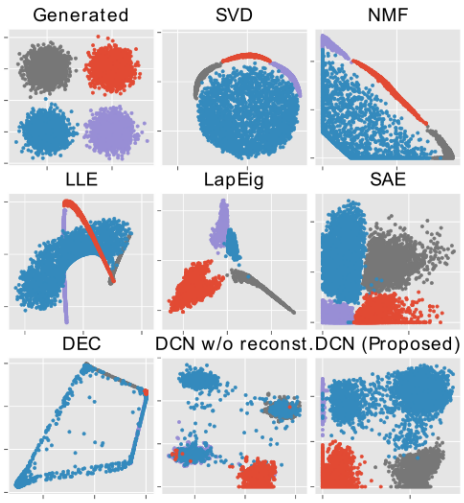


Figure 1. The generated latent representations $\{\boldsymbol{h}_i\}$ in the 2-D space and the recovered 2-D representations from $\boldsymbol{x}_i \in \mathbb{R}^{100}$, where $\boldsymbol{x}_i = (\sigma(\boldsymbol{W}\boldsymbol{h}_i))^2$.

Table 1. Evaluation on the Pendigits dataset

Methods	DCN	SAE+KM	SC	KM
NMI	**0.69**	0.65	0.67	0.67
ARI	**0.56**	0.53	0.55	0.55
ACC	**0.72**	0.70	0.71	0.69

2.2. DCN as Feature Learner

We motivate and develop DCN as a clustering method that directly works on unlabeled data. In practice, DCN can also be utilized as a feature-learning method when training samples are available – i.e., one can feed labeled training data to DCN, tune the parameters of the network to learn well clustered latent representations of the training samples, and then use the trained DCN (to be specific, the forward network) to reduce dimension of unseen testing data.

Here, we provide some additional results to showcase the feature-learning ability of DCN. We perform a 5-fold cross-validation experiment on the raw MNIST dataset, where each fold is a 80/20 training/testing random split. The performance of SAE+KM on the training sets is presented as a baseline.

The obtained NMI, ARI, and ACC (mean and standard deviation) are listed in Table. 2. One can see that the training and testing stages of DCN output similar results, which is rather encouraging. This experiment suggests that DCN is very promising as a representation learner.

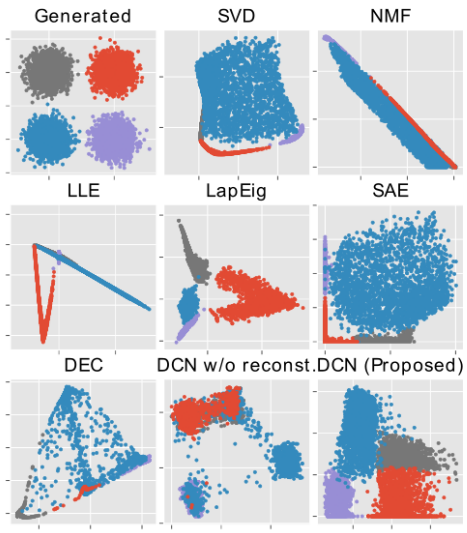


Figure 2. The generated latent representations $\{\boldsymbol{h}_i\}$ in the 2-D space of the recovered 2-D representations from $\boldsymbol{x}_i \in \mathbb{R}^{100}$, where $\boldsymbol{x}_i = \tanh(\sigma(\boldsymbol{W}\boldsymbol{h}_i))$.

Table 2. The mean (stand deviation) of the evaluation results of the 5-fold cross-validation on MNIST.

	NMI	ARI	ACC
DCN-Training	0.80 (0.001)	0.74 (0.002)	0.83 (0.002)
DCN-Testing	0.81 (0.003)	0.75 (0.005)	0.83 (0.004)
SAE+KM	0.73 (0.001)	0.67 (0.002)	0.80 (0.002)

3. Detailed Settings of Real-Data Experiments

3.1. Algorithm Parameters

There is a set of parameters in the proposed algorithm which need to be pre-defined. Specifically, the learning rate α, the number of epochs T (recall that one epoch responds to a pass of all the data samples through the network), and the balancing regularization parameter λ. These parameters vary from case to case since they are related to a number of factors, e.g., dimension of the data samples, total number of samples, scale (or energy) of the samples, etc. In practice, a reasonable way to tune these parameters is through observing the performance of the algorithm under various parameters on a small validation subset whose labels are known.

Note that the proposed algorithm has two stages, i.e., pre-training and the main algorithm and they usually use two different sets of parameters since the algorithmic structure of the two stages are quite different (to be more precise, the pre-training state does not work with the whole network but only deals with a pair of encoding-decoding layers greedily). Therefore, we distinguish the parameters of the two stages as listed in Table 3, to better describe the settings.

Table 3. List of parameters used in DCN.

Notations	Meaning
λ	regularization parameter
α_p	base pre-training stepsize
α_l	base learning stepsize
T_p	pre-traing epochs
T_l	learning epochs

We implement SGD for solving the subproblem w.r.t. $\mathcal{X}$ using the Nesterov-type acceleration (**?**), the mini-batch version, and the momentum method. Batch normalization (Ioffe & Szegedy, 2015) that is recently proven to be very effective for training supervised deep networks is also employed. Through out the experiments, the momentum parameter is set to be 0.9, the mini-batch size is selected to be $\approx 0.01 \times N$, and the other parameters are adjusted accordingly in each experiments – which will be described in detail in the next section.

3.2. Network Parameters

The considered network has two parts, namely, the forward encoding network that reduces the dimensionality of the data and the decoding network that reconstructs the data. We let two networks to have a mirrored structure of each other. There are also two parameters of a forward network, i.e., the width of each layer (number of neurons) and the depth of the network (number of layers). There is no strict rule for setting up these two parameters, but the rule of thumb is to adjust them according the amounts of data samples of the datasets and the dimension of each sample. Using a deeper and wider network may be able to better capture the underlying nonlinear transformation of the data, as the network has more degrees of freedom. However, finding a large number of parameters accurately requires a large amount of data since the procedure can be essentially considered as solving a large system of nonlinear equations – and finding more unknowns needs more equalities in the system, or, data samples in this case. Therefore, there is a clear trade-off between network depth/width and the overall performance.

3.3. Detailed Parameter Settings

The detailed parameter settings for experiments on RCV1-v2 are shown in Tables 4. Parameter settings for 20Newsgroup, raw MNIST, pre-processed MNIST, and Pendigits are shown in Tables 5, 6, 7, and 8, respectively.

4. More Discussions

We have the following several more points as further discussion:

1. We have observed that runing SAE for epochs may even worsen the clustering performance in the two-stage approach. In Fig. 3, we show how the clustering performance indexes change with the epochs when we run SAE without K-means regularization.One can see that the performance in fact becomes worse compared to merely using pre-training (i.e., initialization). This means that using the SAE does not necessarily help clustering – and this supports our motivation for adding a K-means-friendly structure-enhancing regularization.

2. To alleviate the effect brought by the intrinsic randomness of the algorithms, e.g., random initialization of pre-training, the reported results are all obtained via running the experiments several times and taking average (specifically, we run the experiments with smaller size, i.e., 20Newsgroup, raw and processed MNIST, and Pendigits for ten times and the results of the much larger dataset RCV-v2 are average of five runs; the results for DEC in Table 1 is from a single run.). Therefore, the presented results reflect the performance of the algorithms in an average sense.

3. We treat this work as a proof-of-concept: Joint DNN learning and clustering is a highly viable task according to our design and experiments. In the future, many practical issues will be investigated – e.g., designing theory-backed ways of setting up network and algorithm parameters. Another very intriguing direction is of course to design convergence-guaranteed algorithms for optimizing the proposed criterion and its variants. We leave these interesting considerations for future work.

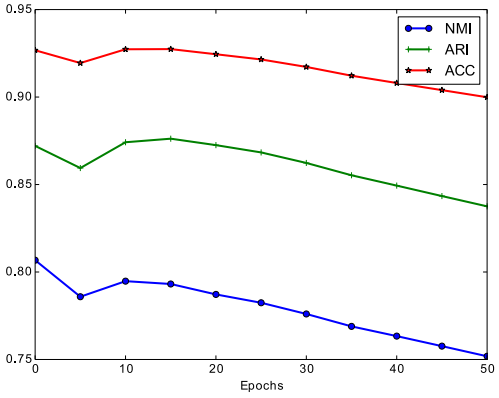


Figure 3. Clustering performance degrades when training with only reconstruction error term. This is in sharp contrast with Figure 5(a) in the paper, where clustering performance improves when training the proposed DCN model.

Table 4. Parameter settings for RCV1-v2

parameters	description
$\boldsymbol{f}(\boldsymbol{x}_i;\mathcal{W})$: $\mathbb{R}^M \to \mathbb{R}^R$	$M = 2,000$ and $R = 50$
Sample size N	178,603 or 267,466
forward net. depth	5 layers
layer width	2000/1000/1000/1000/50
λ	0.1
α_p	0.01
α_l	0.05
T_p	50
T_l	50

Table 5. Parameter settings for 20Newsgroup

parameters	description
$\boldsymbol{f}(\boldsymbol{x}_i;\mathcal{W})$: $\mathbb{R}^M \to \mathbb{R}^R$	$M = 2,000$ and $R = 20$
Sample size N	18,846
forward net. depth	3 layers
layer width	250/100/20
λ	10
α_p	0.01
α_l	0.001
T_p	10
T_l	50

Table 6. Parameter settings for raw MNIST

parameters	description
$\boldsymbol{f}(\boldsymbol{x}_i;\mathcal{W})$: $\mathbb{R}^M \to \mathbb{R}^R$	$M = 784$ and $R = 50$
Sample size N	70,000
forward net. depth	4 layers
layer width	500/ 500/ 2000/10
λ	0.05
α_p	0.01
α_l	0.05
T_p	50
T_l	50

Table 7. Parameter settings for Pre-Processed MNIST

parameters	description
$\boldsymbol{f}(\boldsymbol{x}_i;\mathcal{W})$: $\mathbb{R}^M \rightarrow \mathbb{R}^R$	$M = 10$ and $R = 5$
Sample size N	70,000
forward net. depth	3 layers
layer width	50/ 20/ 5
λ	0.1
α_p	0.01
α_l	0.01
T_p	10
T_l	50

Table 8. Parameter settings for Pendigits

parameters	description
$\boldsymbol{f}(\boldsymbol{x}_i;\mathcal{W})$: $\mathbb{R}^M \rightarrow \mathbb{R}^R$	$M = 16$ and $R = 10$
Sample size N	10,992
forward net. depth	3 layers
layer width	50/ 16/ 10
λ	0.5
α_p	0.01
α_l	0.01
T_p	50
T_l	50