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 (Ertöz 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

1Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis MN 55455, USA. 2Department 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.umn.edu>, Mingyi Hong <mingyi@iastate.edu>.

Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).
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.

- **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 corpora. 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 \( \{x_i\}_{i=1}^{N} \) where \( x_i \in \mathbb{R}^d \), 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:

\[
min_{M \in \mathbb{R}^{MK}, \{s_i \in \mathbb{R}^K\}} \sum_{i=1}^{N} ||x_i - Ms_i||^2_2 \\
\text{s.t. } s_{j,i} \in \{0, 1\}, 1^T s_i = 1 \forall i, j,
\]
where $s_i$ is the assignment vector of data point $i$ which has only one non-zero element, $s_{j,i}$ denotes the $j$th element of $s_i$, and the $k$th column of $M$, i.e., $m_k$, denotes the centroid of the $k$th 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 $x_i$ 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 $x_i = W h_i$, where $W \in \mathbb{R}^{M \times R}$ and $h_i \in \mathbb{R}^R$, where $R \ll M$. Assume that the data clusters are well-separated in latent domain (i.e., where $h_i$ lives) but distorted by the transformation introduced by $W$. Reference (Yang et al., 2017) formulated the joint optimization problem as follows:

$$
\min_{M, \{s_i\}, W, H} \|X - WH\|_F^2 + \lambda \sum_{i=1}^{N} \|h_i - M s_i\|_2^2 \\
+ r_1(H) + r_2(W)
$$

s.t. $s_{j,i} \in \{0, 1\}$, $1^T s_i = 1 \forall i, j$, (2)

where $X = [x_1, \ldots, x_N]$, $H = [h_1, \ldots, h_N]$, 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., $H \rightarrow 0 \in \mathbb{R}^{R \times N}$; see details in (Yang et al., 2017).

The data model $X \approx WH$ 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.

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

$$
\min_{W, \Theta} \tilde{L} = \sum_{i=1}^{N} q(f(x_i; W); \Theta),
$$

where $f(x_i; W)$ is the network output given data sample $x_i$, $W$ collects the network parameters, and $\Theta$ denotes parameters of some clustering model. For instance, $\Theta$ stands for the centroids $M$ and assignments $\{s_i\}$ 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 $f(x_i; W) = 0$ and the optimal objective value $\tilde{L} = 0$ 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 $\tilde{L}$ – but this could be far from being desired since there is no provision for respecting the data samples $x_i$’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 $x_i$ and its clustering-friendly latent representation $h_i$ using a nonlinear mapping, i.e.,

$$
h_i = f(x_i; W), \quad f(\cdot; W) : \mathbb{R}^M \rightarrow \mathbb{R}^R,
$$

where $f(\cdot; W)$ denotes the mapping function and $W$ denotes 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
Towards K-means-friendly Spaces: Simultaneous Deep Learning and Clustering

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 \( \|X - WH\|_F^2 \) in (2). This term ensures that the learned \( h_i \)'s can (approximately) reconstruct the \( x_i \)'s using the basis \( W \). 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 \( g(\cdot; Z) \) to map the \( h_i \)'s back to the data domain and requires that \( g(h_i; Z) \) and \( x_i \) 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:

\[
\min_{W, Z} \sum_{i=1}^{N} \left( \ell(g(f(x_i)), x_i) + \frac{\lambda}{2} \|f(x_i) - Ms_i\|^2 \right)
\]

\[\text{s.t. } s_{j,i} \in \{0, 1\}, 1^T s_i = 1 \quad \forall i, j,\]

where we have simplified the notation \( f(x_i; W) \) and \( g(h_i; Z) \) to \( f(x_i) \) and \( g(h_i) \), respectively, for conciseness. The function \( \ell(\cdot) : \mathbb{R}^M \rightarrow \mathbb{R} \) is a certain loss function that measures the reconstruction error. In this work, we adopt the least-squares loss \( \ell(x, y) = \|x - y\|^2_2 \); other choices such as \( \ell_1 \)-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 (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., \((W, Z)\), 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 \( M \) and \( \{s_i\} \).

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 \( W, Z, M \) and \( \{s_i\} \) because the block variable \( \{s_i\} \) 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 \( M, \{s_i\} \) and \((W, Z)\) while keeping the other two sets of variables fixed.
4.2.1. Update network parameters

For fixed \( (M, \{ s_i \}) \), 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 \( x_i \):

\[
\min_{\mathcal{W}, \mathcal{Z}} L^i = \ell (g(f(x_i)), x_i) + \gamma \cdot \frac{1}{2} \|f(x_i) - Ms_i\|^2_2.
\]

The gradient of the above function over the network parameters is easily computable, i.e., \( \nabla_{\mathcal{W}, \mathcal{Z}} L^i = \frac{\partial \ell (g(f(x_i)), x_i)}{\partial \mathcal{X}} + \gamma \frac{\partial f(x_i)}{\partial \mathcal{X}} (f(x_i) - Ms_i) \), where \( \mathcal{X} = (\mathcal{W}, \mathcal{Z}) \) is a collection of the network parameters and the gradients \( \frac{\partial \ell}{\partial \mathcal{X}} \) and \( \frac{\partial f(x_i)}{\partial \mathcal{X}} \) 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-differentiable at zero). Then, the network parameters are updated by

\[
\mathcal{X} \leftarrow \mathcal{X} - \alpha \nabla_{\mathcal{X}} L^i,
\]

where \( \alpha > 0 \) is a diminishing learning rate.

4.2.2. Update clustering parameters

For fixed network parameters and \( M \), the assignment vector of the current sample, i.e., \( s_i \), can be naturally updated in an online fashion. Specifically, we update \( s_i \) as follows:

\[
s_{j,i} \leftarrow \begin{cases} 
1, & \text{if } j = \arg \min_{k=1, \ldots, K} \|f(x_i) - m_k\|_2^2, \\
0, & \text{otherwise}.
\end{cases}
\]

When fixing \( \{ s_i \} \) and \( \mathcal{X} \), the update of \( M \) is simple and may be done in a variety of ways. For example, one can simply use \( m_k = \frac{1}{|C_k^i|} \sum_{i \in C_k^i} f(x_i) \), where \( C_k^i \) 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., \( x_1, \ldots, x_i \)) might not be representative enough to model the global cluster structure and the initial \( s_i \)’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 \( m_1, \ldots, m_K \). The intuition is simple: assume that the clusters are roughly balanced in terms of the number of data samples they contain. Then, after updating \( M \) 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 \( x_i \), and update \( m_k \) by a simple gradient step:

\[
m_k \leftarrow m_k - \frac{1}{c_k^i} (m_k - f(x_i)) s_{k,i},
\]

where the step size \( 1/c_k^i \) controls the learning rate. The above update of \( M \) 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 \( h_i \)'s in this domain. What we observe is \( x_i \in \mathbb{R}^{100} \) that is obtained via the following transformation:

\[
x_i = \sigma (U \sigma (W h_i)),
\]

where \( W \in \mathbb{R}^{10 \times 2} \) and \( U \in \mathbb{R}^{100 \times 10} \) are matrices whose entries follow the zero-mean unit-variance i.i.d. Gaussian distribution, \( \sigma (\cdot) \) is a sigmoid function to introduce nonlinearity. Under the above generative model, recovering the K-means-friendly domain where \( h_i \)’s live seems very challenging.
Towards K-means-friendly Spaces: Simultaneous Deep Learning and Clustering

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 $x_i$’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 $x_i$’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 $L$ 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.

5.2.2. 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.

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 code1 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.

1https://github.com/piiswrong/dec
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 desirable clustering result, possibly due to the imbalance clusters.

Table 2. Evaluation on the 20Newsgroup dataset.

<table>
<thead>
<tr>
<th>Methods</th>
<th>DCN</th>
<th>SAE+KM</th>
<th>LCCF</th>
<th>NMF+KM</th>
<th>KM</th>
<th>SC</th>
<th>XRAY</th>
<th>JNKM</th>
</tr>
</thead>
<tbody>
<tr>
<td>NMI</td>
<td>0.48</td>
<td>0.47</td>
<td>0.46</td>
<td>0.39</td>
<td>0.41</td>
<td>0.40</td>
<td>0.19</td>
<td>0.40</td>
</tr>
<tr>
<td>ARI</td>
<td>0.34</td>
<td>0.28</td>
<td>0.17</td>
<td>0.17</td>
<td>0.15</td>
<td>0.17</td>
<td>0.02</td>
<td>0.10</td>
</tr>
<tr>
<td>ACC</td>
<td>0.44</td>
<td>0.42</td>
<td>0.32</td>
<td>0.33</td>
<td>0.3</td>
<td>0.34</td>
<td>0.18</td>
<td>0.24</td>
</tr>
</tbody>
</table>

Fig. 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 non-linear. 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
the raw MNIST dataset that has 70,000 data samples. Each sample is a $28 \times 28$ 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 $\lambda$ 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 or 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).

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 $\lambda$ 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 $\lambda$. Moreover, our experience suggests that the performance of our approach is insensitive to the exact value of $\lambda$. Fig. 6 shows how the proposed method performs with different $\lambda$ on the MNIST dataset. As we can see, although there is degradation of performance as $\lambda$ gets inappropriately large, the degradation is mild. The proposed method gives satisfactory result for a range of $\lambda$.

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

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


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


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


