Improving Spectral Clustering with Deep Embedding and Cluster Estimation

Liang Duan*†, Charu Aggarwal‡, Shuai Ma*†, Saket Sathe‡
*SKLSDE Lab, Beihang University, China
†Beijing Advanced Innovation Center for Big Data and Brain Computing, Beijing, China
‡IBM T. J. Watson Research Center, New York, USA
{duanliang, mashuai}@buaa.edu.cn, {charu, ssathe}@us.ibm.com

***Abstract*—Spectral clustering is one of the most popular modern clustering algorithms. It is easy to implement, can be solved efficiently, and very often outperforms other traditional clustering algorithms such as k-means. However, spectral clustering would be insufficient when dealing with most datasets which have complex statistical properties and requires the user to specify the number of clusters (called k). To address these two problems, in this paper, we propose an approach to extending spectral clustering with deep embedding and estimation of the number of clusters. Specifically, we first generate the deep embedding via learning a deep autoencoder, which transforms the raw data into the lower dimensional representations that suitable for clustering. We then provide an effective method to estimate the number of clusters by learning a softmax autoencoder from the deep embedding. We finally extend spectral clustering with the learned embedding and the estimated number. An extensive experimental study on several image and text datasets illustrates the effectiveness and efficiency of our approach.**

***Index Terms*—spectral clustering, deep embedding, autoencoder, number of clusters, clustering**

I. Introduction

Clustering can be considered as one of the most fundamental unsupervised learning techniques, which has been widely used in various fields from computer science to social science [1]. The goal of clustering is to group a set of data points into multiple groups or clusters so that points within a cluster have high similarity, but are very dissimilar to points in other clusters [2]. Thus, a notion of dissimilarity or distance is central to clustering algorithms and most of the existing methods focus on modeling the dissimilarity relationship among data points based on the data representation in a feature space. For example, the traditional clustering method k-means [3] uses the Euclidean distance between data points in a given feature space, which might be raw pixels for images or TF-IDF representations for text documents.

Different from k-means that directly clusters on the given feature space, Spectral Clustering (SC) [4] works by embedding the data into the eigenspace of the Laplacian matrix that derived from the pairwise similarities between data points, and applying k-means on this embedding to obtain the clusters. In fact, SC has many fundamental advantages, such as it is very simple to implement, can be solved efficiently by standard linear algebra methods, and often performs better than other traditional clustering methods. However, the pairwise similarities are typically constructed on Euclidean distance and this might make SC work poorly on high dimensional data due to the curse of dimensionality [5]. Furthermore, SC requires the user to supply the number k of clusters in the input data, which is not always clear what is the best value for k in practical applications [6].

Recently, deep learning has achieved widespread success in numerous machine learning tasks [7], where learning powerful feature representations by deep neural networks (DNN) lies in the core. Thus, it is conceivable to conduct clustering on the powerful representations rather than on the raw data. The deep autoencoder [8], [9] is one of the most popular architectures of DNN for learning good representations and several deep-autoencoder-based methods have been proposed to improve the clustering performance. Deep Embedded Clustering (DEC) [10] was proposed to simultaneously learn feature representations and cluster assignments. It pretrains a multilayer autoencoder to generate deep embedding and then finetunes the parameters of the autoencoder and cluster centroids simultaneously by the defined clustering loss with an iterative approach. However, the clustering loss can not guarantee good embedding and might lead to corruption of embedded space [11], [12]. Therefore, we adopt the deep autoencoder in DEC to learn the embedded features, but after that we apply spectral clustering on the embedding to obtain clusters. We refer to this clustering method as Spectral Clustering with Deep Embedding (SCDE).

Another limitation of spectral clustering is that the number of clusters has to be supplied. In the last decades, several methods have been proposed to determine the value of k automatically and most of them are wrappers around k-means or some other clustering algorithms for fixed k [6]. They usually use splitting or merging rules for cluster centroids to increase or decrease k as the algorithm proceeds and estimate the optimum number by applying different clustering evaluation criteria, such as Calinski-Harabasz Index [13], Davies-Bouldin Index [14], Silhouette Statistic [15], Bayesian Information Criterion [16], etc. However, these methods depend on the clustering algorithm in use, i.e., they may find an incorrect number if the clustering algorithm performs poorly on the data. Furthermore, they would be inefficient when a large range of k are considered. Therefore, we propose a new method to find the correct number k by learning a special autoencoder, referred to as Softmax Autoencoder (SA), which can estimate

2374-8486/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDM.2019.00027

the number of clusters directly rather than runs multiple k-means on the data. Moreover, this method could be conducted on the deep embedding, which effectively improves the estimation accuracy. To this end, we integrate this estimation method into SCDE, referred to as SCDE+.

Contributions. Our major contributions are follows:

1) We first provide an effective extension of spectral clustering with deep embedding by utilizing a deep autoencoder to learn the representations from the raw data and then applying spectral clustering to cluster.
2) We then propose a novel method to estimate the number of clusters using a softmax autoencoder and integrate it into the extension of spectral clustering. By incorporating with the learned deep embedding, it is effective for estimating the number for clustering.
3) We finally conduct a set of experiments on several image and text datasets, and show that our proposed approach for clustering is both effective and efficient.

Organization. This paper is organized as follows. In the next section, we provide the basic notations for clustering and describe spectral clustering with deep embedding. Section III discusses how to determine the optimal number of clusters and integrate it into the proposed clustering method. Section IV presents the experimental results, followed by related work in Section V and conclusions in Section VI.

II. SCDE: Spectral Clustering with Deep Embedding

In this section, we describe Spectral Clustering with Deep Embedding (SCDE), an effective method for extending spectral clustering with deep embedding via learning a deep autoencoder from the raw data.

Assuming that the data set $X = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$ contains n data points and each point $\boldsymbol{x} = (x_1, ..., x_d)$ is a d-dimensional row vector. The clustering problem is to organize the set X into k partitions ($k \leq n$), where each partition represents a cluster [1]. For example, the classic clustering method k-means [3] divides X into k disjoint clusters $\mathcal{C} = \{C_1, ..., C_k\}$ by choosing centroids that minimize the within-cluster sum-of-squares criterion $\sum_{j=1}^{k} \sum_{\boldsymbol{x}_i \in C_j} \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|^2$, where $\boldsymbol{\mu}_j$ is the centroid of cluster C_j. Note that k-means are suitable for clustering the data scattered around their centroids, but would be poorly to elongated clusters or manifolds with irregular shapes. Moreover, high dimensional data are in general not very friendly to k-means [11]. In order to solve this problem, several methods have been proposed by using a dimensionality reduction technique, such as principal component analysis (PCA) [17] or nonnegative matrix factorization (NMF) [18], [19], to reduce the original input data into a lower dimensional space and then applying k-means, which usually obtain better results. However, most of these methods are linear embedding and insufficient for more complex data.

Spectral clustering makes use of a nonlinear embedding to reduce the dimensionality of the data. The process of spectral clustering is shown in Algorithm 1. Indeed, SC first constructs a similarity matrix A (also called affinity matrix) from X and then runs an eigenvalue decomposition on the normalized Laplacian matrix L_n. After that, SC finds the k largest eigenvectors of L_n to form new representations Y and applies k-means on Y to obtain the final clusters [20]. SC takes $O(n^2 d)$ time to compute A and $O(n^2 k)$ time to find eigenvectors. Thus, the complexity of SC is $O(n^2(d + k))$.

Algorithm 1 SC Algorithm

```
Input:
    $X$, a data set; $k$, the number of clusters
Output:
    $\mathcal{C}$, a set of $k$ clusters $\{C_1, ..., C_k\}$
Construct the similarity matrix $A = (a_{ij})_{i,j=1,...,n}$ from the raw data $X$, where $a_{ij}$ is the pairwise similarity between point $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$;
Form the normalized Laplacian matrix $L_n = I - D^{-1/2} A D^{-1/2}$, where $D$ is the diagonal matrix whose $(i, i)$-element is the sum of $A$'s $i$-th row;
Find the $k$ largest eigenvectors $\boldsymbol{u}_1, ..., \boldsymbol{u}_k$ of $L_n$ and then form a new matrix $Y = [\boldsymbol{u}_1 ... \boldsymbol{u}_k]$;
Normalize the rows of $Y$ by $y_{ij} = y_{ij} / \sqrt{\sum_{j=1}^{k} y_{ij}^2}$;
$\mathcal{C} = \{C_1, ..., C_k\} \leftarrow$ cluster $Y$ with $k$-means.
```

SC often outperforms other traditional approaches due to the good representations generated by the top eigenvectors of the Laplacian matrix of the similarity matrix [20]. Note that SC does not directly cluster on the raw data but on the similarity matrix A. Thus, the performance of SC relies on the similarity matrix. However, the similarity matrix is usually constructed by a K-nearest neighbor graph based on Euclidean distance or a fully connected graph based on Gaussian similarity function [4]. This would be ineffective when the input dataset consists of complex statistical properties [21]. Therefore, we extract more effective representation for SC by learning a deep autoencoder from the raw data.

A traditional autoencoder is a neural network for dimensionality reduction, which aims to learn a compressed representation for an input by minimizing its reconstruction error [7]. Internally, it has a hidden layer $\boldsymbol{h}$ that describes a code used to represent the input. The network usually consists of two parts: an encoder function $\boldsymbol{h} = f(\boldsymbol{x})$ and a decoder that produces a reconstruction $\boldsymbol{x}' = g(\boldsymbol{h})$. The learning process of the autoencoder is to minimize a loss function $L(\boldsymbol{x}, g(f(\boldsymbol{x})))$, where L is a loss function penalizing $g(f(\boldsymbol{x}))$ for being dissimilar from $\boldsymbol{x}$. If we constrain $\boldsymbol{h}$ to have a smaller dimension than $\boldsymbol{x}$, it will force the autoencoder to capture the most salient features of the training data and obtain useful representations of the data.

Several methods based on multilayer autoencoders (also called deep autoencoder) have been proposed to learn powerful feature representations and achieve great success in many fields [8], [10], [11], [21]. DEC is one of these successful methods that use an iterative way to update the parameters of the encoder and cluster centroids jointly after obtain the deep embedding. The clustering loss of DEC is the Kullback-Leibler

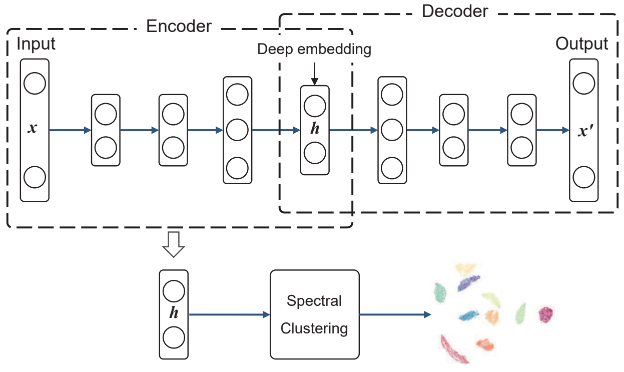


Fig. 1. The framework of SCDE. The top row is the network architecture of the deep autoencoder and the bottom row is the clustering phase of SCDE that applies spectral clustering on the deep embedding to obtain clusters.

```
Algorithm 2 SCDE Algorithm
Input:
    $X$, a data set; $k$, the number of clusters
Output:
    $\mathcal{C}$, a set of $k$ clusters $\{C_1, ..., C_k\}$
train a deep autoencoder: $\boldsymbol{x}_i \xrightarrow{f} \boldsymbol{h}_i \xrightarrow{g} \boldsymbol{x}'_i$ with data reconstruction loss in (1);
obtain deep embedding: $\boldsymbol{h}_i = f(\boldsymbol{x}_i)$;
$\mathcal{C} = \{C_1, ..., C_k\} \leftarrow$ cluster $\{\boldsymbol{h}_i\}$ with SC (Algorithm 1).
```

(KL) divergence loss $\sum_{i=1}^{n} KL(f(\boldsymbol{x};\omega);\theta)$, where ω is the encoder parameters and θ is the cluster centroids. Optimizing this loss might lead to a trivial solution $f(\boldsymbol{x};\omega) = 0$ which distorts the embedding. IDEC [12] improves DEC by taking the reconstruction loss into consideration, but can not solve this problem completely.

Inspired by these work, we adopt the deep autoencoder in DEC to obtain good embedding of the raw data. After that, we apply SC on the embedding to cluster the data. We refer to this method as Spectral Clustering with Deep Embedding (SCDE). Fig. 1 shows the network architecture of the deep autoencoder and the framework of SCDE. The complete procedure of SCDE is shown in Algorithm 2.

We first learn a deep autoencoder from X by using stochastic gradient descent (SGD) algorithm to minimize the data reconstruction loss L_r, which is defined as follows:

$$L_r = \sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{x}'_i\|^2 \tag{1}$$

We then obtain the deep embedding $\boldsymbol{h}_i$ for each point $\boldsymbol{x}_i$ using the encoder f on X and finally apply SC on this embedding to finish the clustering. SCDE takes $O(nwt)$ time to train the deep autoencoder, where w and t are the number of weights in the autoencoder and the total training epochs. Thus, the complexity of SCDE is $O(nwt + n^2(h+k))$, where h is the dimension of the embedding layer.

Discussions. Using deep embedding for spectral clustering has several advantages. The deep autoencoder is a nonlinear transformation to extract more powerful features from the raw data and generates better data representations, which significantly improve the clustering accuracy of spectral clustering. Moreover, compared with the iterative clustering methods such as DEC, our approach can not distort the deep embedding. However, SC requires to supply the number of clusters, which is not always clear for users in practical applications. In order to handle this problem, we propose an effective and efficient method to estimate the number of clusters.

III. SCDE+: Improving SCDE with Estimation of the Number of Clusters

In this section, we first propose a novel method to estimate the number of clusters based on a softmax autoencoder. We then integrate this method into SCDE to estimate the number of clusters automatically.

A. Estimation of the Number of Clusters

The basic idea of this estimation method is that using the softmax autoencoder as a clustering method and counting the number of cluster labels assigned by the softmax autoencoder as the estimation.

The softmax autoencoder derives its name from the fact that the innermost hidden layer uses the softmax activation function. Note that the softmax activation function is often (almost exclusively) reserved for the output layer in multiway prediction problems; therefore, its use in a hidden layer seems somewhat unusual at first sight. However, the use of the softmax activation within the hidden layer is actually quite logical in this setting, when one considers the fact that it is intended for the innermost hidden layer to yield probabilistic cluster memberships. Although this can be viewed as a clustering method, we use these cluster memberships to estimate the number of clusters rather than cluster the data.

The softmax autoencoder contains a total of $2m+1$ layers, including the input layer. The innermost layer, which is the $(m+1)$-th layer, contains k_u units, and it also represents the upper bound of the number of clusters into which we wish to partition the data points. This layer uses the softmax activation function, and is not truly a hidden layer, because its output is visible, and used to infer the probabilistic assignments of the points to clusters. The estimated number is the total number of clusters assigned to the points. The encoder-decoder architecture is symmetric in terms of the number of units in the matching layer, but not necessarily in terms of the computations performed in those layers. In other words, for $r \leq m$, the r-th layer matches up with the $(2m+2-r)$-th layer in terms of the number of units. However, it is possible for the activation functions to be different in these layers. For example, the first layer is a non-computational input layer, whereas the $(2m+1)$-th layer is computational in nature. The $(2m+1)$-th layer contains linear activations because it is possible for the inputs to take on arbitrary real values. All other hidden layers (except for the innermost layer) use the

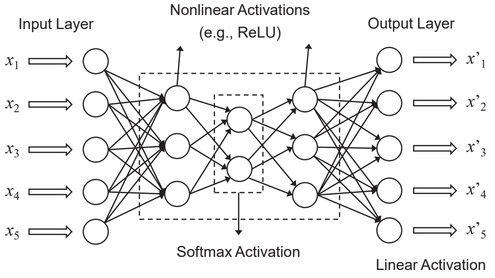


Fig. 2. The architecture of softmax autoencoder.

Algorithm 3 SCDE+ Algorithm

```
Input:
    $X$, a data set; $k_u$, the upper bound of the estimation of the number of clusters
Output:
    $\mathcal{C}$, a set of $k'$ clusters $\{C_1, ..., C_{k'}\}$
train a deep autoencoder: $\boldsymbol{x}_i \xrightarrow{f} \boldsymbol{h}_i \xrightarrow{g} \boldsymbol{x}'_i$ with data reconstruction loss in (1);
obtain deep embedding: $\boldsymbol{h}_i = f(\boldsymbol{x}_i)$;
train a softmax autoencoder: $\boldsymbol{h}_i \xrightarrow{f'} \boldsymbol{z}_i \xrightarrow{g'} \boldsymbol{h}'_i$ with loss in (4); // the dimension of $\boldsymbol{z}_i$ is $k_u$
obtain clustering membership probabilities: $\boldsymbol{z}_i = f'(\boldsymbol{h}_i)$;
$k' \leftarrow$ count the distinct labels $\arg\max_{j=1}^{k_u}\{z_{ij} \in \boldsymbol{z}_i\}$;
$\mathcal{C} = \{C_1, ..., C_{k'}\} \leftarrow$ cluster $\{\boldsymbol{h}_i\}$ with SC (Algorithm 1).
```

ReLU activation function. The innermost layer, which is the $(m+1)$-th layer, uses the softmax activation function, and therefore its k_u outputs sum to 1. The overall architecture of softmax autoencoder is schematically shown in Fig. 2.

1) Loss Function: A key part of the workings of the softmax autoencoder is the loss function, which ensures that the autoencoder creates representations that are good enough to estimate the number of clusters. The principle behind the loss function depends on viewing the estimation as a specific type of dimensionality reduction technique:

> One can view estimating the number of clusters as a dimensionality reduction technique in which the reduced representation of the data corresponds to the probabilities of memberships of points to clusters, which is used for the estimation. Furthermore, a high-quality clustering will assign membership probabilities that are spread out in an uneven way across clusters. In other words, low entropy in cluster assignments is encouraged.

Therefore, the loss function contains two parts. The first part is a standard reconstruction loss on the final output layer, which ensures that the reduced representation can reconstruct the data. This is a standard squared loss and defined in (1). The second part of the loss uses the Gini index of the activations $z_1, ..., z_{k_u}$ in the innermost layer containing k_u units. Note that $z_1, ..., z_{k_u}$ sum to 1 because of the use of softmax activation, and the Gini index G is defined as follows:

$$G = 1 - \sum_{i=1}^{k_u} z_i^2 \tag{2}$$

Note that the constant value of 1 can be ignored. Therefore, the second part of the loss, referred to as the *cluster coherence* loss L_c defined as follows:

$$L_c = -\sum_{i=1}^{k_u} z_i^2 \tag{3}$$

Interestingly, this is also a squared loss, albeit with a negative sign in front of it. However, from a practical point of view, this squared loss does help in easily computing gradients with existing deep learning tools, e.g., SGD, and relatively few customized changes. The overall loss is a combination of the reconstruction and the cluster coherence loss:

$$L = L_r + L_c \tag{4}$$

Although the approach naturally yields a soft clustering in terms of membership probabilities, it is also possible to convert it into a hard clustering by assigning each point to the cluster with the highest membership probability. In practice, however, many applications might have naturally overlapping clusters. In such cases, the soft clustering approach seems more reasonable from a practical point of view.

After learning the softmax autoencoder from the data, we estimate the number of clusters based on the clustering membership probabilities obtained from the autoencoder. We first generate the cluster labels for all points by assigning each point to the cluster with the highest probability, and then count the number of distinct elements in the labels as the estimation of the number of clusters in the input data.

B. SCDE+ Framework

One advantage of the softmax autoencoder estimation method is that it can be easily incorporated with the deep embedding, i.e., taking the deep embedding as the input instead of the raw data. As a result, the good representations of the deep embedding help the estimation method to find the correct number. Thus, we integrate this estimation method into the SCDE clustering to determine the number of clusters for spectral clustering. We now refer to SCDE with this technique of estimating the number of clusters as SCDE+. The procedure of SCDE+ is shown in Algorithm 3.

We learn a deep autoencoder to obtain the deep embedding as the same as SCDE, and then we learn a softmax autoencoder from the embedding to generate the clustering membership probabilities of each point. After that, we assign each point to the cluster with the highest probability and count the number of distinct labels as the estimation k'. Finally, we apply spectral clustering with the embedding and k' to cluster. The complexity of SCDE+ is $O(n(wt+w't')+n^2(h+k'))$, where w' and t' are the number of weights and the total training epochs of the softmax autoencoder.

Discussions. Different from traditional estimation methods that run multiple k-means to determine the optimal number of clusters, our method takes less time to estimation, especially for the lower dimensional embedding, which also help to find a better estimation. It is worth mentioning that our estimation method is not limited to SCDE, and may be applied to other clustering methods that require the number of clusters, such as k-means, SC and DEC.

IV. Experimental Study

In this section, we present experimental results on several real-life datasets to evaluate our proposed method. We first introduce the experimental settings, and then we conduct three sets of experiments: (1) clustering with different numbers of clusters, (2) estimation of the number of clusters, and (3) clustering with estimation of k, to evaluate our method compared with the comparison methods.

A. Experimental Settings

We first present our experimental settings.

1) Datasets: We performed experiments on four image datasets and two text datasets, which are widely used for evaluating the performance of clustering methods.

- MNIST [22]: Consists of total 70,000 handwritten digits ($0 \sim 9$) of 28×28 pixel size. We reshaped each gray image to a 784 dimensional vector.
- Fashion-MNIST[1]: Consists of total 70,000 data samples of Zalando's article images, which is often served as a direct drop-in replacement for the MNIST dataset for benchmarking machine learning algorithms because MNIST might be too easy and overused. Each sample is a 28×28 gray-scale image, associated with a label from 10 classes. Similar to MNIST, we reshaped each image to a 784 dimensional vector.
- USPS[2]: Consists of total 11,000 gray-scale handwritten digits ($0 \sim 9$) with size of 16×16 pixels. We reshaped each gray image to a 256 dimensional vector.
- STL-10[3]: Consists of 13,000 color images of 96×96 pixel size and grouped into 10 classes. Since clustering directly on the high resolution images is rather difficult, we extracted the image features by VGG16 [23] and the dimensionality of the extracted features is 4096.
- Reuters-8 [24]: A text corpus that contains 804,414 English documents categorized into 103 different topics. Restricted by computational resources, we used a subset of the corpus that contains 8 topics and 10,000 documents with a single topic label. As in DEC [10], we computed TF-IDF features on the 2,000 most frequently occurring word stems for clustering.
- 20Newsgroups[4]: A collection of 18,846 text documents which are partitioned into 20 different newsgroups. We also extracted the TF-IDF features on the 2,000 most frequently used words for clustering.

We summarize the important statistics about these datasets in Table I.

TABLE I
Datasets Statistics

Dataset	# Samples	Dimension	# Clusters
MNIST	70,000	784	10
Fashion-MNIST	70,000	784	10
USPS	11,000	256	10
STL-10	13,000	4,096	10
Reuters-8	10,000	2,000	8
20Newsgroups	18,846	2,000	20

2) Evaluation Metrics: Since all datasets have the ground truth assignments of clusters, we evaluate the performance of a clustering algorithm by two standard metrics: Normalized Mutual Information (NMI) [18] and Adjusted Rand Index (ARI) [25]. MNI has a range of [0,1] with one being the perfect clustering and zero the worst, and ARI has a range of [-1,1] with one being the best clustering performance and minus one the opposite.

3) Algorithms for Comparison: We have carefully chosen several methods for comparison. For clustering, we compared our clustering methods SCDE and SCDE+ with k-means, spectral clustering (SC) [20], Spectral Embedding Clustering (SEC) that combines SC and linear embedding to improve the clustering performance [26], the classic dimensionality reduction method NMF followed by SC (NMF+SC) and the autoencoder-based method DEC [10]. For estimating the number of clusters, we compared our softmax autoencoder method (denoted as SA) with two splitting-rule-based methods X-means [16] and G-means [6] and other two methods that run multiple k-means in an increasing sequence of k to determine the best value of k by following criteria: Davies-Bouldin Index (DB) [14] and Silhouette Statistic (SS) [15].

4) Implementation: We implemented all algorithms based on Python and Keras[5]. For deep embedding, we adopted the same network architecture of the autoencoder in DEC. Specifically, we set the network dimensions of the autoencoder to d-500-500-2000-10-2000-500-500-d, where d is the dimension of the input data. All layers are densely connected and all hidden layers (except for the innermost layer) use the ReLU activation function. All the autoencoders on different datasets are trained for 50 epochs and the mini-batch size is fixed to 256. For estimating the number of clusters, we set the network architecture of the SA to d_s-50-k_u-50-d_s, where d_s is the dimension of the input and k_u is the upper bound of the estimation of the number of clusters. For spectral clustering, we constructed the similarity matrix by Gaussian similarity function for 20Newsgroups and K-nearest neighbor graph

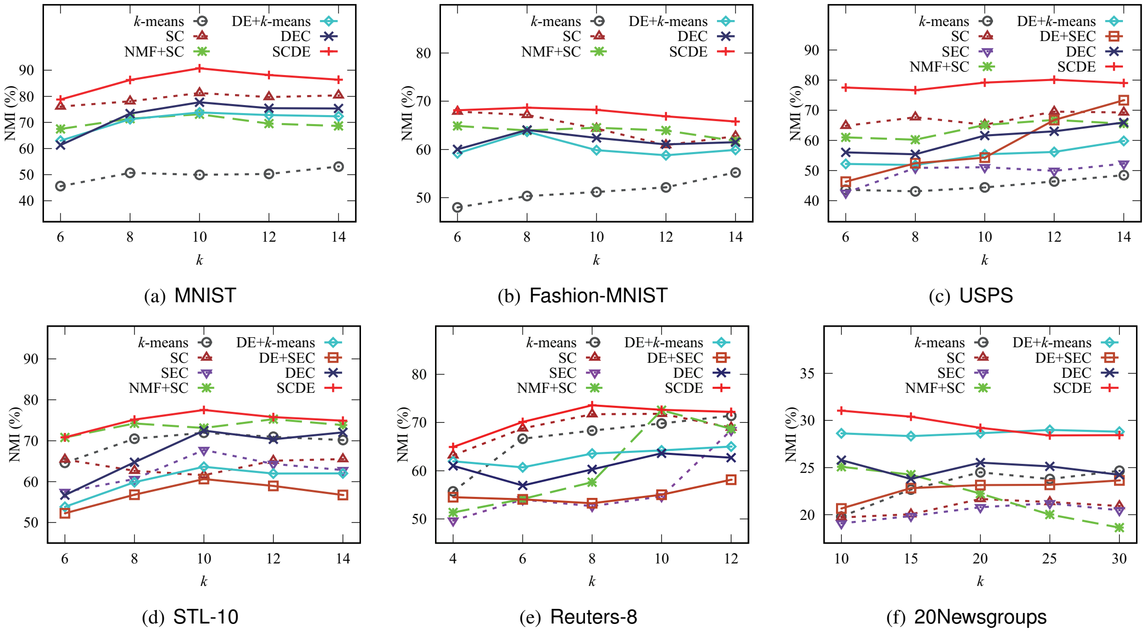
[1] https://github.com/zalandoresearch/fashion-mnist

[2] https://cs.nyu.edu/%7Eroweis/data.html

[3] https://cs.stanford.edu/%7Eacoates/stl10/

[4] http://qwone.com/%7Ejason/20Newsgroups/

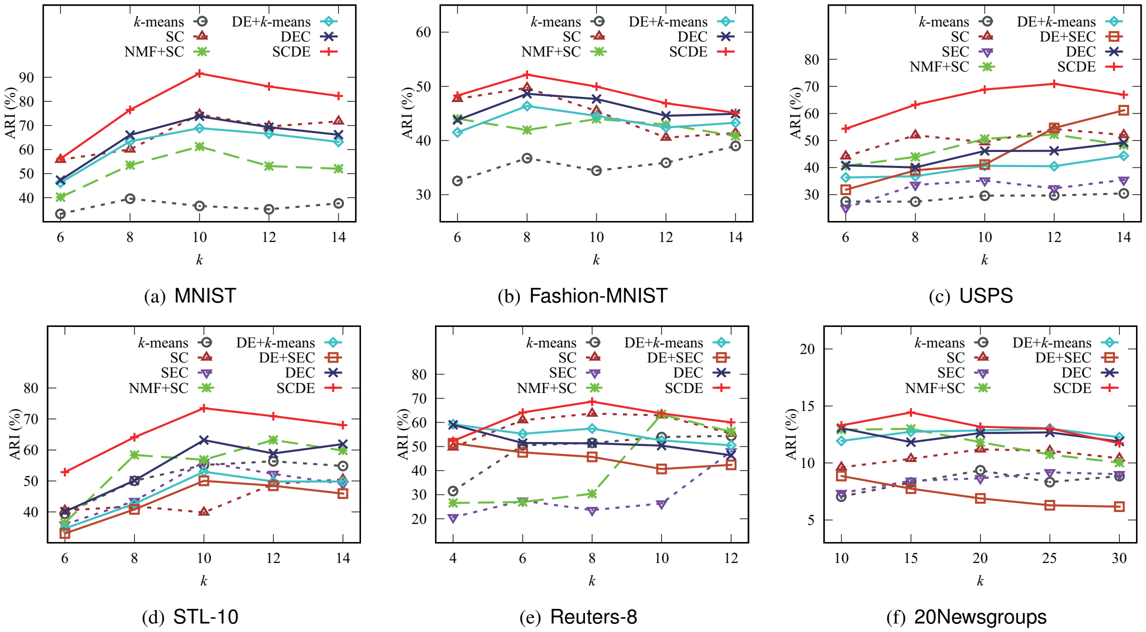
[5] https://github.com/keras-team



(a) MNIST (b) Fashion-MNIST (c) USPS

(d) STL-10 (e) Reuters-8 (f) 20Newsgroups

Fig. 3. Clustering comparison on NMI: with respect to the number k of clusters. It is better to view the figure in color.



(a) MNIST (b) Fashion-MNIST (c) USPS

(d) STL-10 (e) Reuters-8 (f) 20Newsgroups

Fig. 4. Clustering comparison on ARI: with respect to the number k of clusters. It is better to view the figure in color.

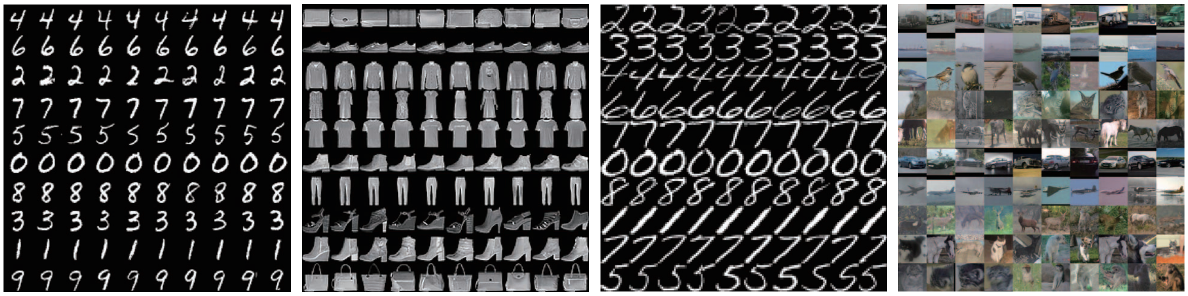
for other datasets. To guarantee the K-nearest neighbor graph is fully connected, we fixed $K = 10$ for STL-10, $K = 50$ for Reuters-8 and $K = 5$ for the others.

All experiments were conducted on a machine with 2 Intel Xeon E5-2630 2.3GHz CPUs and 64GB of Memory, running 64 bit Windows 10 Professional system. Each experiment was repeated 5 times, and the average is reported here.

B. Experimental Results

We next present our findings.

1) Clustering with Different Numbers of Clusters: In the first set of tests, we deliberately chose different numbers of clusters k to evaluate the effectiveness of our clustering method SCDE compared with k-means, SC, SEC and DEC. To better understand the contribution of deep embedding, we



(a) MNIST (b) Fashion-MNIST (c) USPS (d) STL-10

Fig. 5. Clustering results of our approach SCDE. Each row contains the top 10 scoring elements from one cluster. Best viewed in color.

compared SCDE with NMF+SC that uses the classical dimensionality reduction method NMF to generate low dimensional embedding of the raw data. We also tested k-means and SEC on the deep embedding, denoted as DE+k-means and DE+SEC, respectively. For a fair comparison, we used the same embedding for DE+k-means, DE+SEC, DEC and our method. The number k is fixed to [6, 8, 10, 12, 14] for MNIST, Fashion-MNIST, USPS and STL-10, [4, 6, 8, 10, 12] for Reuters-8 and [10, 15, 20, 25, 30] for 20Newsgroups, respectively. The results of clustering comparison on NMI and ARI are reported in Fig. 3 and Fig. 4, respectively.

The results tell us that (a) our proposed method SCDE outperforms other methods by large margins on all datasets except slightly weaker than DE+k-means on 20Newsgroups when k is large and slightly weaker than DEC and DE+k-means with ARI on Reuters-8 when k is small, (b) the deep embedding significantly improve the NMI and ARI of SCDE and DE+k-means compared with their counterparts SC and k-means on most of the datasets, which reveals the usefulness of the deep embedding for clustering, (c) SC is the second best method on NMI while DEC is the second best method on ARI, and (d) NMF+SC works well on Fashion-MNIST, USPS and STL-10, but performs poorly on other datasets, which means that NMF cannot generate good enough representations for clustering compared with deep embedding. This verifies the effectiveness of our approach.

To better illustrate the performance of SCDE, we plot the top 10 scoring images of each cluster from the results of SCDE on MNIST, Fashion-MNIST, USPS and STL-10, shown in Fig. 5. Each row corresponds to a cluster and images are sorted from left to right based on their distance to the cluster centroid obtained by the k-means in SCDE.

From Fig. 5 we can see that (a) SCDE clusters very well on MNIST and assigns each point to the correct cluster, (b) SCDE also works well on Fashion-MNIST, which serves as a direct drop-in replacement for MNIST but is more difficult to cluster, (c) the clustering results on USPS are as well as on MNIST, with the exception of confusing 4 and 9, which also exists in DEC [10], and (d) for STL-10, SCDE is mostly correct on truck, ship, bird, car and airplane categories, and provides interesting clustering assignments on other categories. For instance, all the animals in the cluster of the 5-th row of Fig. 5(d) have four legs.

2) Estimation of the Number of Clusters: In the second set of tests, we evaluated the effectiveness and efficiency of our method SA for estimating the number of clusters compared with X-means, G-means, DB and SS. To better understand the effectiveness of deep embedding for estimation, we also tested SA, X-means, G-means, DB and SS on the deep embedding, denoted as DE+SA, DE+X-means, DE+G-means, DE+DB and DE+SS, respectively. Similar to X-means, we fixed the range of k to $[2, 2k_g]$ for all datasets, where k_g is the ground truth number of clusters of each dataset. We fixed the number $k_u = 2k_g$ in SA on each dataset. The optimal values obtained by each method and the corresponding running time are reported in Table II.

The estimation results tell us that (a) our method DE+SA does the best at finding the correct k and outperforms other methods on all datasets, (b) when the deep embedding is not available, our method SA also performs better than X-means, G-means, DB and SS on most of the datasets, (c) the deep embedding significantly improves the estimation accuracy of SA, DB and SS, and (d) X-means and G-means perform worse than other methods on most of the datasets and their estimation accuracy cannot be improved by the deep embedding. Specifically, DE+SA obtains the optimal values (10.2, 9.2, 9.6, 9.8, 8, 20.8) on MNIST, Fashion-MNIST, USPS, STL-10, Reuters-8 and 20Newsgroups, which are very close to the ground truth numbers of clusters on these datasets. X-means and G-means overestimate the numbers of true clusters and their performance cannot be improved by deep embedding. However, DE+SA performs consistently better than SA when incorporating with deep embedding. This verifies the effectiveness of our method.

The running time results tell us that (a) our method SA outperforms other methods on all datasets, (b) our method DE+SA is faster than other methods when using the deep embedding, (c) most of the methods become slower when incorporating with deep embedding because generating the embedding takes much time, while some methods become faster due to the dimension of the embedding is lower than that of the raw data, (d) DB is faster than SS, and SS is the most time consuming method since it needs to calculate the

TABLE II
ESTIMATION OF THE NUMBER OF CLUSTERS AND RUNNING TIME (SEC.) OF EACH METHOD ON ALL DATASETS.

Method	MNIST		Fashion-MNIST		USPS		STL-10		Reuters-8		20Newsgroups	
	k	Time	k	Time	k	Time	k	Time	k	Time	k	Time
Ground Truth	10	-	10	-	10	-	10	-	8	-	20	-
X-means	20	357	20	321	20	19	20	235	16	56	40	470
G-means	15	732	16	647	20	41	20	605	16	121	40	1892
DB	20	912	3	785	18.8	37	6.6	588	2	114	34.8	1840
SS	2	2972	2	2935	5	88	3	664	15.4	150	2	2232
SA (ours)	11.2	**85**	12	**88**	6.6	**10**	2.8	**56**	12.6	**24**	31.6	**59**
DE+X-means	20	1304	20	1328	20	169	20	474	16	164	40	508
DE+G-means	9	1301	19.4	1352	20	172	16	475	16	165	40	550
DE+DB	10.8	1313	6.2	1340	17.8	169	8.8	473	14.4	162	25	536
DE+SS	9	2475	5.2	2413	12.2	222	7.8	512	14.8	196	2	701
DE+SA (ours)	**10.2**	1301	**9.2**	1327	**9.6**	167	**9.8**	469	**8**	162	**20.8**	494

The best estimation and running time are highlighted in boldface. The k is a floating number since it is the average of found ks.

distance between a point and all other points, and (e) X-means is faster than G-means, DB and SS. Note that our method SA directly estimates the number k by training a softmax autoencoder rather than running multiple k-means to find the best number. Thus, SA is faster than other methods. Indeed, SA is (11, 9, 4, 11, 5, 31) and (35, 33, 9, 12, 6, 38) times faster than DB and SS on (MNIST, Fashion-MNIST, USPS, STL-10, Reuters-8, 20Newsgroups), respectively. Since X-means and G-means cannot find the correct number and often hit our limit of $2k_g$ clusters, we omit the comparison with them. When incorporating with deep embedding, our method DE+SA also runs faster than DE+DB and DE+SS. This verifies the efficiency of our method.

3) Clustering with Estimation of k: In the last set of tests, we evaluated the effectiveness of our clustering method SCDE+. Since k-means, SC, SEC, NMF+SC and DEC need to specify the number of clusters, we revised them with the best method obtained in the previous test, i.e., SA for the raw data and DE+SA for the embedding. Thus, we adopted SA for k-means, SC, SEC, NMF+SC (denoted as SA+k-means, SA+SC, SA+SEC and SA+NMF+SC) on the raw data, and DE+SA for DE+k-means, DE+SEC and DEC (denoted as DE+SA+k-means, DE+SA+SEC and DEC+SA) on the deep embedding. The results of NMI and ARI are reported in Table III. Note that the classic density-based clustering DBSCAN [2] does not need to specify the number k, but it performs poorly on these datasets. Thus, we do not choose it for comparison.

The results tell us that (a) our proposed method SCDE+ outperforms other methods by large margins on all datasets, (b) SCDE+ significantly improves the NMI and ARI over its counterpart SA+SC by incorporating with deep embedding, which is also the same to DE+SA+k-means and DE+SA+SEC on most of the datasets, (c) SCDE+ performs better than SC+NMF+SC due to the deep autoencoder can learn more powerful representations for estimating the number of clusters and clustering than NMF, and (d) DEC+SA achieves the second best performance by jointly learning the deep embedding and cluster assignments in an iterative way. Indeed, our method SCDE+ improves NMI by (13%, 7%, 31%, 9%, 20%, 15%) over the second best method DEC+SA on MNIST, Fashion-MNIST, USPS, STL-10, Reuters-8 and 20Newsgroups, respectively. Moreover, our method also obtains higher ARI than other methods on all datasets. This verifies the effectiveness of our proposed approach.

4) Summary: From these experimental results on several image and text datasets, we find the following.

1) Our clustering method SCDE has significantly improved the clustering accuracy and performs better than other methods, including k-means, SC, SEC, NMF+SC and DEC, on a large range of the number of clusters. Actually, SCDE achieves the highest NMI and ARI score on most of the datasets with different values of k.
2) Our estimation method SA is effective and efficient to find the correct number of clusters, especially when incorporating with the deep embedding. For instance, DE+SA finds the values of k (10.2, 9.2, 9.6, 9.8, 8, 20.8) on MNIST, Fashion-MNIST, USPS, STL-10, Reuters-8 and 20Newsgroups, which are very close to the ground truth numbers of clusters. Furthermore, our method also has the efficiency advantage of linear complexity in the number of data points, which makes it faster than DB and SS that have to run multiple k-means to determine the number of clusters.
3) Incorporating with the estimation method SA, our proposed method SCDE+ could estimate the number k automatically and cluster data effectively. It improves MNI by (13%, 7%, 31%, 9%, 20%, 15%) over the second best method DEC+SA on MNIST, Fashion-MNIST, USPS, STL-10, Reuters-8 and 20Newsgroups, respectively.

TABLE III
Comparison of clustering NMI (%) and ARI (%) with estimation of the number of clusters on all datasets.

Method	MNIST		Fashion-MNIST		USPS		STL-10		Reuters-8		20Newsgroups	
	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI
SA+k-means	50.19	35.90	51.90	35.84	43.35	27.53	53.41	23.99	71.12	51.01	24.15	08.41
SA+SC	79.43	69.37	62.17	41.57	65.67	46.49	53.91	21.49	70.91	57.51	20.98	10.49
SA+SEC	N/A	N/A	N/A	N/A	45.23	27.96	51.65	24.02	67.48	45.90	20.80	08.91
SA+NMF+SC	70.67	54.81	63.21	42.60	61.00	42.13	56.59	21.74	69.96	58.12	18.50	09.72
DE+SA+k-means	74.81	70.01	61.12	44.95	53.00	38.10	62.38	49.74	62.82	55.62	28.41	12.79
DE+SA+SEC	N/A	N/A	N/A	N/A	54.69	41.31	58.54	45.89	54.31	45.97	23.13	06.58
DEC+SA	78.63	74.80	63.32	48.16	58.86	43.32	69.68	58.47	59.57	49.32	25.20	12.55
SCDE+ (ours)	**89.02**	**86.31**	**67.81**	**50.11**	**77.04**	**63.97**	**75.91**	**68.61**	**71.61**	**64.56**	**28.92**	**13.44**

The best NMI and ARI are highlighted in boldface. N/A means that not enough memory for the method on the dataset.

V. Related work

Clustering Methods. Clustering is one of the most fundamental tasks in data mining and machine learning [1], [27], and a large number of clustering algorithms have been developed and successfully applied in enormous real world applications [2]. These methods can be classified into feature-based clustering and similarity-based clustering. A feature-based method takes a $n \times d$ matrix as input, where n is the number of samples and d is the dimension of features. One famous feature-based method is the k-means [3], which partitions the samples into k clusters so as to minimize the sum of the Euclidean distance between samples to the corresponding centroids. However, the Euclidean distance metrics are limited to the raw data space and make k-means ineffective when the input data is high dimensional [10]. Therefore, several variants of k-means have been proposed, including principal component analysis (PCA) [17], nonnegative matrix factorization (NMF) [18], [19], canonical correlation analysis (CCA) [28] and sparse coding [29], to reduce the high dimensional data into a much lower dimensional data space that more suitable for performing k-means [11]. However, most of these methods are linear embedding and not sufficient for more complex data.

Different from feature-based methods, similarity-based methods construct a $n \times n$ similarity matrix that measures the distance between each pair of the samples. Spectral clustering (SC) is a classical similarity-based method that leverages the Laplacian spectra of the similarity matrix to generate low dimensional embedding of samples and runs a k-means in the embedding to get the clusters [4], [20]. Compared with k-means, SC has the advantage that kernel functions or domain-specific similarity can be incorporated into the construction of the similarity matrix and generally performs better than k-means [20]. Thus, we adopt SC in our method.

Several methods have been proposed to improve the performance of SC. Spectral Embedded Clustering (SEC) combines linear embedding and spectral clustering [26]. Another improved SC to replace the eigenvalue decomposition with deep autoencoder has been proposed in [30], but it increases memory consumption. Different from the above extensions, we learn a deep autoencoder to generate better embedding, which is not only used to improve the clustering accuracy of SC but also to estimate the number of clusters.

Deep Embedded Clustering. Recently, some clustering methods based on deep neural networks (DNNs) have been proposed due to their high representational power. By using DNNs, it is possible to learn non-linear mappings that transform the raw data into more clustering-friendly representations [31]. Deep Embedded Clustering (DEC) is proposed to learn feature representations and cluster assignments simultaneously using DNNs [10]. DEC learns a mapping from data space to a lower-dimensional feature space in which it iteratively optimizes a clustering objective. Deep Clustering Network (DCN) is a joint dimensionality reduction (DR) and k-means clustering approach in which DR is accomplished via learning a DNN [11]. Variational Deep Embedding (VaDE) is a generative model for clustering by modeling the data generative procedure with a Gaussian Mixture Model and a DNN [32]. Most of these methods use a two-phased training procedure. In the first phase, pre-training an autoencoder with the standard reconstruction loss. In the second phase, combining the autoencoder with a clustering method (e.g., k-means or agglomerative clustering) and then fine-tuning the joint model with a loss function consisting of the reconstruction loss and a clustering loss iteratively [31]. Note that the use of an iterative approach to fine-tune the joint model already makes several assumptions and an optimal solution to this problem is not easy to achieve, which might distort the embedding and cost much time to find a well solution [12]. Therefore, we adopt this two-phased procedure, but we apply spectral clustering on the deep embedding without fine-tuning, which is more effective and efficient for clustering.

Estimating the Number of Clusters. Estimating the optimal number of clusters is an important and yet unsolved problem in unsupervised clustering and attracts considerable interest in the literature (for reviews, see [33]). An effective solution for this problem is to run k-means clustering on the input data for a range of values k (e.g., $k \in [2, 20]$), and for each value of k calculate a cost function that incorporates the k and the error in clustering [34]. A number of measure criteria for such cost function have been proposed, such as Calinski-Harabasz Index [13], Davies-Bouldin Index [14], Silhouette Statistic [15], etc. X-means uses a splitting rule for k-means centroids to search the optimal k based on Bayesian Information Criterion [16]. G-means runs k-means with increasing k in a hierarchical fashion until the test accepts the hypothesis that the data assigned to each centroid are Gaussian [6]. Different from these methods, our estimation method does not need to run multiple k-means for a range of k, which is an efficiency advantage. Moreover, by incorporating with the deep embedding, our method can find a number that very close to the ground truth number of clusters.

VI. Conclusions and Future Work

In this paper, we have proposed an approach to improving spectral clustering with deep embedding and estimation of the number of clusters. We first learn the deep embedding of the raw data based on a multilayer autoencoder, which effectively generates good and lower-dimensional representations for clustering. We then estimate the number of clusters by learning a softmax autoencoder from the deep embedding and integrate it into the proposed clustering method. The extensive experiments on image and text datasets demonstrate that our proposed approach can find a correct number of clusters and outperform other clustering methods, such as DEC, NMF-SC, SEC, SC and k-means. Another advantage of our approach is efficiency, which allows it to scale to large datasets.

Several topics need further investigation. First, we are to develop an embedding technique that can optimize the deep embedding, estimation of the number of clusters and clustering simultaneously to further improve the clustering performance. Second, we are to apply our proposed method for estimating the number of clusters to other clustering methods. Third, we are to scale up the proposed method to deal with large-scale datasets such as the ImageNet (http://www.image-net.org/index) dataset.

Acknowledgments

This work is supported in part by National Key R&D Program of China 2018YFB1700403 and NSFC U1636210&61421003. For any correspondence, please refer to Shuai Ma.

References

[1] C. C. Aggarwal and C. K. Reddy, *Data Clustering: Algorithms and Applications*. CRC Press, 2013.

[2] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques (Third Edition)*. Morgan Kaufmann, 2012.

[3] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[4] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[5] R. E. Bellman, *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[6] G. Hamerly and C. Elkan, "Learning the k in k-means," in *NIPS*, 2004, pp. 281–288.

[7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[8] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, pp. 504–507, 2006.

[9] Y. Bengio, L. Yao, G. Alain, and P. Vincent, "Generalized denoising auto-encoders as generative models," in *NIPS*, 2013, pp. 899–907.

[10] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *ICML*, 2016, pp. 478–487.

[11] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards k-means-friendly spaces: Simultaneous deep learning and clustering," in *ICML*, 2017, pp. 3861–3870.

[12] X. Guo, L. Gao, X. Liu, and J. Yin, "Improved deep embedded clustering with local structure preservation," in *IJCAI*, 2017, pp. 1753–1759.

[13] T. Caliński and J. Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics*, vol. 3, no. 1, pp. 1–27, 1974.

[14] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 224–227, 1979.

[15] P. J.Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, no. 2, pp. 53–65, 1987.

[16] D. Pelleg and A. Moore, "X-means: Extending k-means with efficient estimation of the number of clusters," in *ICML*, 2000, pp. 727–734.

[17] C. Dian and X. He, "K-means clustering via principal component analysis," in *ICML*, 2004, p. 29.

[18] D. Cai, X. He, and J. Han, "Locally consistent concept factorization for document clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 6, pp. 902–913, 2011.

[19] F. Shahnaz, M. W. Berry, V. P. Pauca, and R. J. Plemmons, "Document clustering using nonnegative matrix factorization," *Information Processing and Management*, vol. 42, pp. 373–386, 2006.

[20] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *NIPS*, 2002, pp. 849–856.

[21] P. Huang, Y. Huang, W. Wang, and L. Wang, "Deep embedding network for clustering," in *ICPR*, 2014, pp. 1532–1537.

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[24] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, "Rcv1: A new benchmark collection for text categorization research," *Journal of Machine Learning Research*, vol. 5, no. Apr, pp. 361–397, 2004.

[25] K. Y. Yeung and W. L. Ruzzo, "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*, vol. 17, no. 9, pp. 763–774, 2001.

[26] F. Nie, Z. Zeng, I. W. Tsang, D. Xu, and C. Zhang, "Spectral embedded clustering: A framework for in-sample and out-of-sample spectral clustering," *IEEE Transactions on Neural Networks*, vol. 22, no. 11, pp. 1796–1808, 2011.

[27] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, 1999.

[28] K. Chaudhuri, S. M. Kakade, K. Livescu, and K. Sridharan, "Multi-view clustering via canonical correlation analysis," in *ICML*, 2009, pp. 129–136.

[29] M. Zheng, J. Bu, C. Chen, C. Wang, L. Zhang, G. Qiu, and D. Cai, "Graph regularized sparse coding for image representation," *IEEE Transactions on Image Processing*, vol. 20, no. 5, pp. 1327–1336, 2011.

[30] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, "Learning deep representations for graph clustering," in *AAAI*, 2014.

[31] E. Aljalbout, V. Golkov, Y. Siddiqui, M. Strobel, and D. Cremers, "Clustering with deep learning: Taxonomy and new methods," *arXiv*, vol. 1801.07648v2, 2018.

[32] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou, "Variational deep embedding: An unsupervised and generative approach to clustering," in *IJCAI*, 2017, pp. 1965–1972.

[33] M. M.-T. Chiang and B. Mirkin, "Intelligent choice of the number of clusters in k-means clustering: An experimental study with different cluster spreads," *Journal of Classification*, vol. 27, no. 1, pp. 3–40, 2010.

[34] A. Kolesnikov, E. Trichina, and T. Kauranne, "Estimating the number of clusters in a numerical data set via quantization error modeling," *Pattern Recognition*, vol. 48, no. 3, pp. 941–952, 2015.