Knowledge and Information Systems
https://doi.org/10.1007/s10115-020-01530-8

REGULAR PAPER

Improving spectral clustering with deep embedding, cluster estimation and metric learning

Liang Duan[1] · Shuai Ma[2,3] · Charu Aggarwal[4] · Saket Sathe[4]

Received: 27 January 2020 / Revised: 26 October 2020 / Accepted: 1 November 2020

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 could be insufficient when dealing with most datasets having complex statistical properties, and it requires users to specify the number k of clusters and a good distance metric to construct the similarity graph. To address these problems, in this article, we propose an approach to extending spectral clustering with deep embedding, cluster estimation, and metric learning. First, we generate the deep embedding via learning a deep autoencoder, which transforms the raw data into their lower dimensional representations suitable for clustering. Second, we provide an effective method to estimate the number of clusters by learning a softmax autoencoder from the deep embedding. Third, we construct a more powerful similarity graph by learning a distance metric from the embedding using a Siamese network. Finally, we conduct an extensive experimental study on image and text datasets, which verifies the effectiveness and efficiency of our approach.

Keywords Spectral clustering · Deep embedding · Autoencoder · Cluster estimation · Metric learning

1 Introduction

Clustering is 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 [17]. Thus, a notion of dissimilarity or similarity is central to clustering algorithms and most of the existing methods focus on modeling the dissimilarity relationships among data points

✉ Shuai Ma
mashuai@buaa.edu.cn

1 School of Information Science and Engineering, Yunnan University, Kunming, China

2 SKLSDE Lab, Beihang University, Beijing, China

3 Beijing Advanced Innovation Center for Big Data and Brain Computing, Beijing, China

4 IBM T. J. Watson Research Center, New York, USA

based on the data representation in a feature space. For example, the traditional clustering method k-means [21] uses the Euclidean distances 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) [38] 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. 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 distances, which might make SC work poorly on high dimensional data due to the curse of dimensionality [3]. Furthermore, SC requires users 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 [16].

Recently, deep learning has achieved widespread success in numerous machine learning tasks [14], 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 [4,18] 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) [39] is proposed to simultaneously learn feature representations and cluster assignments. It pre-trains 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 cannot guarantee good embedding and might lead to corruption of embedded space [15,40]. 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, referred to as spectral clustering with deep embedding (SCDE).

Another limitation of spectral clustering is that the number of clusters needs to be provided. In the last decades, several methods [6,11,30–32] 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 [16]. They usually use splitting or merging rules for cluster centroids to increase or decrease k as the algorithms proceed and estimate the optimum number by applying different clustering evaluation criteria, such as Calinski-Harabasz Index [6], Davies-Bouldin Index [11], Silhouette Statistic [32], Bayesian Information Criterion [31]. However, these methods mostly 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 could 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 the number of clusters directly rather than run 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+.

Since SC clusters the data based on their pairwise similarities, how to construct a good similarity graph is crucial for SC [38]. However, traditional methods for similarity graph construction often use the Euclidean distances, which might be too simplistic [8,41]. Several methods have been proposed to learn a distance metric from the data, and the Siamese network is one of the most successful methods recently [19,33,34]. Inspired by the unsupervised training of Siamese network in SpectralNet [34], we adopt the Siamese network to learn a distance metric for constructing a better similarity graph to further improve the clustering

performance. Different from SpectralNet, we train the Siamese network on the deep embedding rather than on the original data. We also integrate this modification into SCDE+ and refer to it as SCDE+ with metric learning (SCDEM+).
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 do clustering.
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. Incorporating with the learned deep embedding is effective for cluster estimating.
3. We thirdly learn a distance metric from the deep embedding with a Siamese network to replace the Euclidean distance in similarity graph construction, which further improves the clustering performance.
4. 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 article is organized as follows. Section 2 reviews the related work. Section 3 provides the basic notations for clustering and describes spectral clustering with deep embedding. Section 4 discusses how to determine the optimal number of clusters. Section 5 presents a method to learn a distance metric from the deep embedding for similarity graph construction. Section 6 presents the experimental results followed by conclusions in Sect. 7.

2 Related work

This study extends our previous work [13] by adding (a) a metric learning method for constructing a better similarity graph (Sect. 5) and (b) a more detailed experimental study (Sect. 6).
Clustering methods Clustering is one of the most fundamental tasks in data mining and machine learning [1,22], and a large number of clustering algorithms have been developed and successfully applied in enormous real world applications [17]. These methods can be classified into feature-based clustering and similarity-based clustering. A feature-based method takes a $n \times d$ matrix as its input, where n is the number of samples and d is the dimension of features. One famous feature-based method is the k-means [21], which partitions the samples into k clusters so as to minimize the sum of the Euclidean distances between samples to the corresponding centroids. However, the Euclidean distance metric is limited to the raw data space and makes k-means ineffective when the input data are high dimensional [39]. Therefore, several variants of k-means have been proposed, including principal component analysis (PCA) [12], nonnegative matrix factorization (NMF) [5,35], canonical correlation analysis (CCA) [7] and sparse coding [43], to reduce the high dimensional data into a much lower dimensional data space that is more suitable for performing k-means [40]. However, most of these methods are linear embedding and not sufficient for complex data.

Different from feature-based methods, similarity-based methods construct a $n \times n$ similarity matrix on 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 [28,38]. Compared with k-means, SC has the advantage that kernel func-

tions or domain-specific similarity can be incorporated into the construction of the similarity matrix and generally performs better than k-means [28]. 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 [29]. Another improves SC to replace the eigenvalue decomposition with deep autoencoder [37], but it increases memory consumption. Different from these studies, we learn a deep autoencoder to generate better embedding to improve the clustering accuracy of SC and to automatically estimate the number of clusters.

Deep embedded clustering Recently, 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 nonlinear mappings that transform the raw data into more clustering-friendly representations [2]. Deep embedded clustering (DEC) is proposed to learn feature representations and cluster assignments simultaneously using DNNs [39]. DEC learns a mapping from data space to a lower-dimensional feature space in which it iteratively optimizes the 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 [40]. Variational deep embedding (VaDE) is a generative model for clustering by modeling the data generative procedure with a Gaussian mixture model and a DNN [23]. Most of these methods use a two-phased training procedure. The first phase pre-trains an autoencoder with the standard reconstruction loss. The second phase combines the autoencoder with a clustering method (e.g., k-means or agglomerative clustering), and fine-tunes the joint model with a loss function consisting of the reconstruction loss and a clustering loss iteratively [2]. 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 [15]. 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.

Estimating the number of clusters Estimating the optimal number of clusters is an important and yet unsolved problem in unsupervised clustering and attracts considerable interests [9]. An effective solution for this problem is to run k-means clustering on the input data for a range of values for k, and for each value of k, to calculate a cost function that incorporates the k and the error in clustering [25]. A number of measure criteria for such cost function have been proposed, such as Calinski-Harabasz Index [6], Davies-Bouldin Index [11], Silhouette Statistic [32]. X-means uses a splitting rule for k-means centroids to search the optimal k based on Bayesian Information Criterion [31]. 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 is Gaussian [16]. Different from these methods, our estimation method is efficient, which does not need to run multiple k-means for a range of k values. Further, by incorporating with the deep embedding, our method finds a cluster number close to the ground truth.

Metric learning for clustering Metric learning is the task of learning a distance metric from data, and the learned distance metric can then be used to perform various tasks, e.g., classification, clustering, and information retrieval. Metric learning algorithms can be classified into supervised and unsupervised methods. Most unsupervised metric learning methods aim to project data points into a low-dimensional space, where geometric relationships, such as the pairwise distances, are preserved [41]. These methods could be applied to traditional clustering methods, such as k-means, spectral clustering, and DBSCAN, which depend on the pairwise distances of data points. AML [41] is an unsupervised adaptive metric learning method that performs clustering and distance metric learning simultaneously. NAML [8] is another unsupervised adaptive metric learning method for clustering. SpectralNet [34]

provides a method to learn an effective distance metric for spectral clustering from unlabeled data using a Siamese network. Different from these methods using metric learning as a dimensionality reduction technique, we learn the distance metric from the low-dimensional embedding to construct a better similarity graph for spectral clustering.

3 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 [21] 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 is 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 [40]. In order to solve this problem, several methods have been proposed by using a dimensionality reduction technique, such as principal component analysis (PCA) [12] or nonnegative matrix factorization (NMF) [5,35], to reduce the original input data into a lower dimensional space and then apply 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, and its process is shown in Algorithm 1. It 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. Then 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 [28]. SC takes $O(n^2 d)$ time to compute A and $O(n^2 k)$ time to find eigenvectors. Thus, its total time complexity is $O(n^2(d + k))$.

SC often outperforms other traditional approaches due to the good representations generated by the top eigenvectors of the Laplacian matrix of the similarity matrix [28]. However, SC does not directly cluster on the raw data but on the similarity matrix A, which is usually constructed by a K-nearest neighbor graph based on Euclidean distances or a fully connected graph based on the Gaussian similarity function [38]. This may be ineffective when the input

Algorithm 1 SC Algorithm

Input: X, a data set; k, the number of clusters
Output: $\mathcal{C}$, a set of k clusters $\{C_1, ..., C_k\}$

1: 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 $\boldsymbol{x}_j$;
2: 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;
3: Find the k largest eigenvectors $\boldsymbol{u}_1, ..., \boldsymbol{u}_k$ of L_n and then form a new matrix $Y = [\boldsymbol{u}_1 ... \boldsymbol{u}_k]$;
4: Normalize the rows of Y by $y_{ij} = y_{ij} / \sqrt{\sum_{j=1}^{k} y_{ij}^2}$;
5: $\mathcal{C} = \{C_1, ..., C_k\} \leftarrow$ cluster Y with k-means.

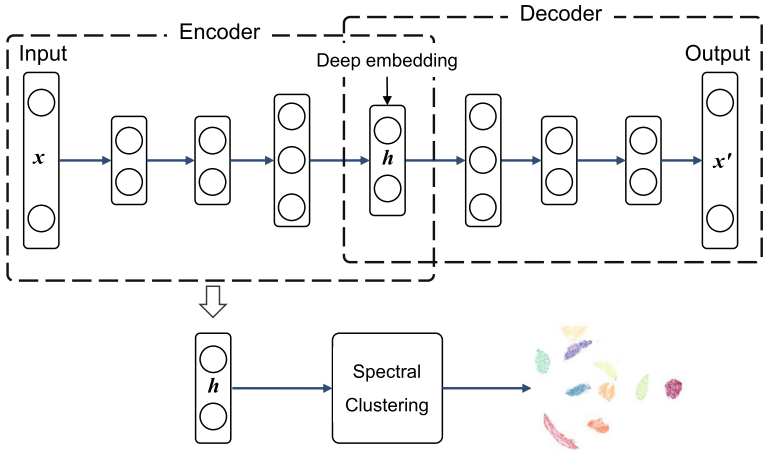


Fig. 1 Framework of SCDE. The top is the network architecture of autoencoder and the bottom is the clustering phase that SCDE applies spectral clustering on the deep embedding

dataset consists of complex statistical properties [20]. Therefore, we extract more effective representations 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 [14]. 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 producing 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 $\boldsymbol{x}$. When we constrain $\boldsymbol{h}$ to have a smaller dimension than $\boldsymbol{x}$, it forces the autoencoder to capture the most salient features 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 [18,20,39,40]. DEC is one of these successful methods and uses 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 (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 [15] improves DEC by taking the reconstruction loss into consideration, but can not solve this problem completely.

To this end, we adopt the deep autoencoder in DEC to obtain good embedding of the raw data. We then apply SC on the embedding to cluster the data. This method is referred to as spectral clustering with deep embedding (SCDE), shown in Fig. 1, and the complete procedure of SCDE is shown in Algorithm 2.

We first learn a deep autoencoder from X with the stochastic gradient descent (SGD) algorithm to minimize the data reconstruction loss L_r, defined below:

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

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 the 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

Algorithm 2 SCDE Algorithm

Input: X, a data set; k, the number of clusters
Output: $\mathcal{C}$, a set of k clusters $\{C_1, ..., C_k\}$

1: train a deep autoencoder: $\boldsymbol{x}_i \xrightarrow{f} \boldsymbol{h}_i \xrightarrow{g} \boldsymbol{x}'_i$ with data reconstruction loss in (1);
2: obtain deep embedding: $\boldsymbol{h}_i = f(\boldsymbol{x}_i)$;
3: $\mathcal{C} = \{C_1, ..., C_k\} \leftarrow$ cluster $\{\boldsymbol{h}_i\}$ with SC (Algorithm 1).

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 iterative clustering methods such as DEC, our approach cannot distort the deep embedding. However, SC requires the number of clusters, which is typically unclear for practical users. In order to handle this problem, we propose an effective and efficient method to estimate cluster numbers.

4 SCDE+: Improving SCDE with automatic cluster estimation

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.

Estimation of the number of clusters The basic idea of this estimation method is to use the softmax autoencoder as a clustering method and use 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 the data.

The softmax autoencoder contains a total of $2m + 1$ layers, including the input layer. The innermost layer, i.e., the $(m + 1)$-th layer, contains k_u units, and it represents the upper bound of the number of clusters into which we wish to partition. This layer uses the softmax activation function and is not truly a hidden layer, because its output is visible and is used to infer the probabilistic assignments of data points to clusters. The estimated number is the total number of clusters assigned to data 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 using different activation functions 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 ReLU activation function. The innermost layer uses the softmax activation function, and therefore,

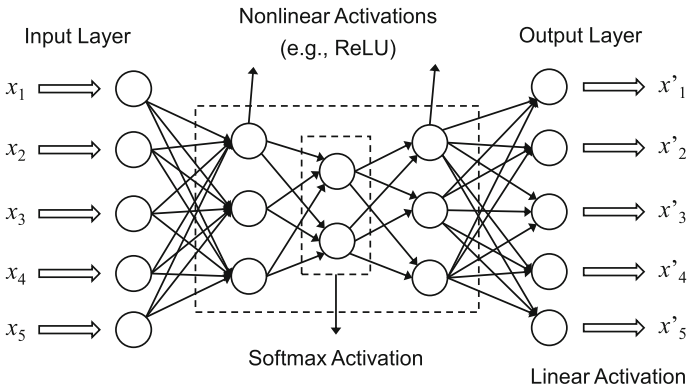


Fig. 2 The architecture of softmax autoencoder

its k_u outputs sum to 1. The overall architecture of softmax autoencoder is schematically shown in Fig. 2.

Loss function An important part of the softmax autoencoder is its loss function, ensuring the autoencoder to create good enough representations 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 a data point corresponds to the probabilities of its memberships to clusters, which is used for the estimation. Furthermore, a high-quality clustering assigns 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 representations can reconstruct the data. This is a standard squared loss and defined in (1). The second part 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 possible to convert it into a hard clustering by assigning each point to the cluster with the highest membership probability. In practice, however, it is natural to have overlapping

Algorithm 3 SCDE+ Algorithm

Input: X, a data set; k_u, the upper bound of the estimation of the number of clusters
Output: C, a set of k' clusters $\{C_1, ..., C_{k'}\}$

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

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.

4.1 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, good representations of the deep embedding can help the estimation method. 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+, shown in Algorithm 3.

We learn a deep autoencoder to obtain the deep embedding along the same lines 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'. 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, respectively.

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.

5 SCDEM+: improving SCDE+ with metric learning

In this section, we provide a method to learn a distance metric from the deep embedding for constructing a better similarity matrix, which could further improve the clustering accuracy of SCDE+.

As mentioned in Sect. 3, SC clusters data on the similarity matrix that usually constructed by K-nearest neighbor graph or Gaussian similarity function based on the Euclidean distance, not suitable for handling high dimensional and complex data. Several methods have been

proposed to learn a good distance metric for similarity graph construction [8,41]. Siamese networks [24,33] are trained to learn similarity relations between data points, and SpectralNet [34] shows that the unsupervised application of a Siamese network to determine the distance often improves the clustering performance. Thus, we adopt the Siamese network to learn a distance metric for the similarity matrix construction.

The main idea of Siamese networks is to find a function that maps input data into a target space such that a simple distance, such as Euclidean distances, in the target space approximates the "semantic" distance in the input space [10]. Specifically, given a neural network $\boldsymbol{s} = f_\psi(\boldsymbol{x})$ parameterized by ψ, Siamese networks aim to find the parameters ψ such that the distance metric $E_\psi(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|f_\psi(\boldsymbol{x}_i) - f_\psi(\boldsymbol{x}_j)\|$ is small if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are in the same category, and large if they are in different ones. The network is often trained on a set of similar (positive) and dissimilar (negative) pairs of data points and usually uses a contrastive loss function to minimize $E_\psi(\boldsymbol{x}_i, \boldsymbol{x}_j)$ when $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are from the same category and maximize $E_\psi(\boldsymbol{x}_i, \boldsymbol{x}_j)$ when $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are belong to different categories. Since we focus on the unsupervised clustering problem, we adopt the same training phase of Siamese networks as SpectralNet [34], which learns the distance metric from a K-nearest neighbor graph of input data. Since we learn the deep embedding $\boldsymbol{h}$ for each point, we train the Siamese network on the deep embedding rather than on the original data. Thus, we generate the training set by setting pair $(\boldsymbol{h}_i, \boldsymbol{h}_j)$ to positive if $\boldsymbol{h}_j$ is one of the nearest neighbors of $\boldsymbol{h}_i$, and negative if $\boldsymbol{h}_j$ is far from $\boldsymbol{h}_i$ (e.g., $\boldsymbol{h}_j$ is not in the nearest neighbor set of $\boldsymbol{h}_i$).

Once the Siamese network is trained, we apply it to construct the similarity matrix A of SC by replacing the distance $\|\boldsymbol{h}_i - \boldsymbol{h}_j\|$ with $\|\boldsymbol{s}_i - \boldsymbol{s}_j\|$. We integrate this modification into SCDE+ to enhance the clustering performance, referred to as SCDE+ with metric learning (SCDEM+). Training the Siamese network takes $O(nKw''t'')$ time, where K is the number of nearest neighbors in the training set, w'' and t'' are the number of weights and the number of epochs. Thus, the total time complexity of SCDEM+ is $O(n(wt+w't'+Kw''t'')+n^2(h+k'))$.
Discussions Siamese networks can capture more complex similarity relations and yield improved clustering quality compared with the Euclidean distance [34]. Different from traditional algorithms that learn the distance metric from the original data, we learn the metric from the low dimensional embedding in a more efficient way. Moreover, this metric learning method is not limited to SC and may be applied to other clustering methods, such as k-means and DBSCAN [17].

6 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 existing methods.

6.1 Experimental settings

We first present our experimental settings.
Datasets We performed experiments on four image datasets and two text datasets, widely used for evaluating the performance of clustering methods.

Table 1 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	4096	10
Reuters-8	10,000	2000	8
20Newsgroups	18,846	2000	20

- MNIST [26]: Consists of total 70,000 handwritten digits (0–9) of 28 × 28 pixel size. We reshaped each gray image to a 784 dimensional vector.
- Fashion-MNIST (https://github.com/zalandoresearch/fashion-mnist): Consists of total 70,000 data samples of Zalando's article images, which is often served as a replacement of MNIST 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 (https://cs.nyu.edu/%7Eroweis/data.html): Consists of total 11,000 gray-scale handwritten digits (0–9) with size of 16 × 16 pixels. We reshaped each gray image to a 256 dimensional vector.
- STL-10 (https://cs.stanford.edu/%7Eacoates/stl10): 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 [36] and the dimensionality of the extracted features is 4096.
- Reuters-8 [27]: 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 [39], we computed TF-IDF features on the 2000 most frequently occurring word stems for clustering.
- 20Newsgroups (http://qwone.com/%7Ejason/20Newsgroups): A collection of 18,846 text documents partitioned into 20 different newsgroups. We also extracted the TF-IDF features on the 2000 most frequently used words for clustering.

We summarize the important statistics about these datasets in Table 1.

Evaluation metrics As all datasets have ground truth clusters, we evaluate the performance of clustering algorithms with two standard metrics: Normalized Mutual Information (NMI) [5] and Adjusted Rand Index (ARI) [42]. NMI has a range of [0,1] with one being the best and zero the worst, and ARI has a range of $[-1, 1]$ with one being the best and minus one the worst.

Comparison algorithms We have carefully chosen methods for comparison. For clustering, we compared our clustering methods SCDE and SCDE+ with k-means, spectral clustering (SC) [28], spectral embedding clustering (SEC) that combines SC and linear embedding to improve the performance [29], the classic dimensionality reduction method NMF followed by SC (NMF+SC) and the autoencoder-based method DEC [39]. For estimating the number of clusters, we compared our softmax autoencoder method (denoted as SA) with two splitting-rule-based methods X-means [31] and G-means [16] and two methods that run multiple k-means in an increasing list of k to determine the best k using Davies-Bouldin Index (DB) [11] and Silhouette Statistic (SS) [32]. For metric learning, we compared our SCDE and

SCDE+ with metric learning (denoted as SCDEM and SCDEM+) with their counterparts SCDE and SCDE+, respectively.

Implementation We have implemented all algorithms based on Python and Keras (https://github.com/keras-team). For deep embedding, we adopt the same network architecture of the autoencoder in DEC by setting the network dimensions 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 activations function. All the autoencoders are trained for 50 epochs, and the mini-batch size is fixed to 256. For cluster estimation, we set the SA network architecture 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 metric learning, we set the Siamese network architecture to d_s-1000-500-10 for MNIST, Fashion-MNIST and USPS, and d_s-500-10 for the others. All hidden layers use the ReLU activation function. The number of epochs is 50 and the mini-batch size is 1024. We construct a 5-nearest neighbor graph to train the Siamese network for each dataset. For spectral clustering, we construct the similarity matrix by K-nearest neighbor graph, where K is fixed to [8, 10, 50, 20, 5] for USPS, STL-10, Reuters-8, 20Newsgroups and the others to make the graph is fully connected.

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

6.2 Experimental results

We next present our findings.

Exp-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 methods SCDEM and SCDE compared with k-means, SC, SEC and DEC. To better understand the contribution of deep embedding, we compared SCDEM and 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 methods. The number k was 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 on NMI and ARI are reported in Figs. 3 and 4, respectively.

The results tell us that (a) our methods SCDEM and SCDE outperform the 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) SCDEM works better than its counterpart SCDE on all datasets due to the Siamese network can learn a more useful distance metric than the simplistic Euclidean distance, which is insufficient to deal with the data with complex similarity relations, (c) the deep embedding significantly improves 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, (d) SC is the third best method on NMI while DEC is the third best method on ARI, and (e) NMF+SC works well on Fashion-MNIST, USPS and STL-10, but performs poorly on the other datasets, which means that NMF cannot generate good enough representations for clustering compared with deep embedding. This verifies the effectiveness of our approach.

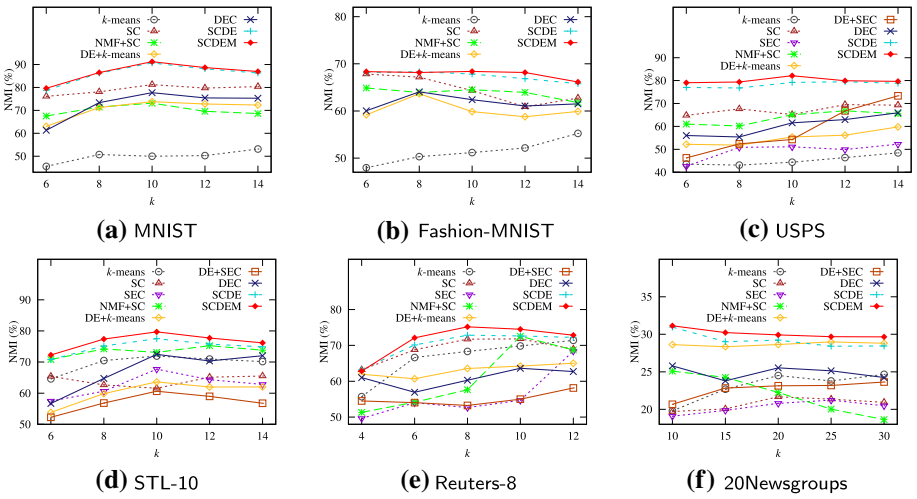


Fig. 3 Clustering comparison on NMI: with respect to the number k of clusters

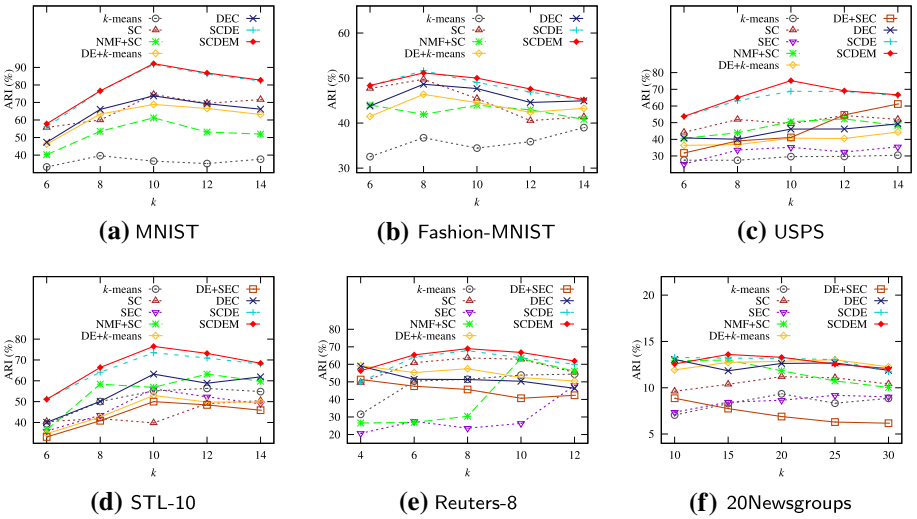
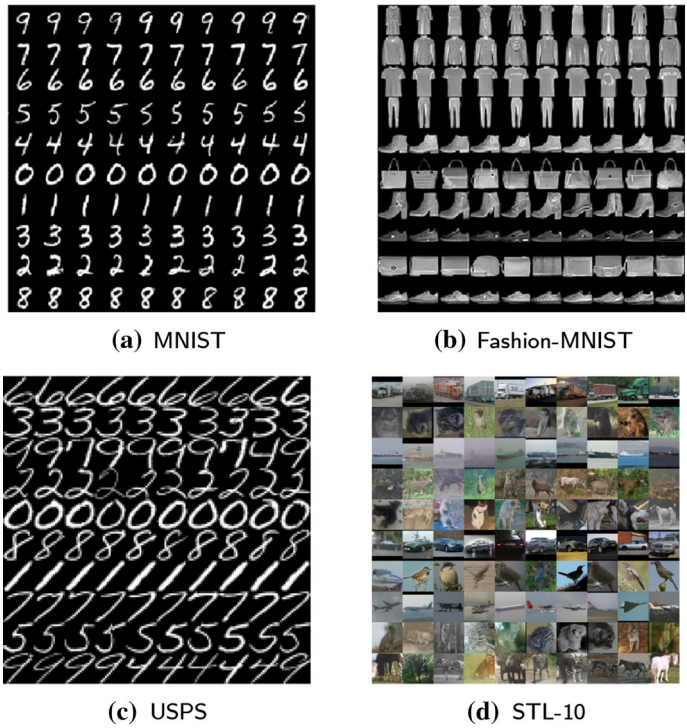


Fig. 4 Clustering comparison on ARI: with respect to the number k of clusters

To better illustrate the performance of SCDEM, we plot the top 10 scoring images of each cluster from its results 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 distances to the cluster centroid obtained by the k-means in SCDEM. We can see that (a) SCDEM clusters very well on MNIST and assigns each point to the correct cluster, (b) SCDEM also works well on Fashion-MNIST, which serves as a direct drop-in replacement for MNIST, and is more difficult to cluster, (c) the clustering results on USPS are as good as on MNIST, with the exception of confusing 4, 7 and 9, which also exists in DEC [39], and (d) for STL-10, SCDEM is mostly correct on truck, ship, car, bird and airplane categories and provides



(a) MNIST **(b)** Fashion-MNIST

(c) USPS **(d)** STL-10

Fig. 5 Results of SCDEM. Each row contains the top 10 scoring elements from one cluster

interesting clustering assignments on the other categories. For instance, all animals in the cluster of the 10-th row of Fig. 5d have four legs.

Exp-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 best values obtained by each method and the corresponding running time are reported in Tables 2 and 3, respectively.

The estimation results tell us that (a) our method DE+SA is the best at finding the correct k and outperforms the 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 the other methods on most of the datasets, and their estimation accuracy cannot be improved by the deep embedding. Specifically, DE+SA obtains the best 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. 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.

Table 2 Estimation of the number of clusters of each method on all datasets

Method	MNIST	Fashion-MNIST	USPS	STL-10	Reuters-8	20Newsgroups
Ground truth	10	10	10	10	8	20
X-means	20	20	20	20	16	40
G-means	15	16	20	20	16	40
DB	20	3	18.8	6.6	2	34.8
SS	2	2	5	3	15.4	2
SA (ours)	11.2	12	6.6	2.8	12.6	31.6
DE+X-means	20	20	20	20	16	40
DE+G-means	9	19.4	20	16	16	40
DE+DB	10.8	6.2	17.8	8.8	14.4	25
DE+SS	9	5.2	12.2	7.8	14.8	2
DE+SA (ours)	**10.2**	**9.2**	**9.6**	**9.8**	**8**	**20.8**

The k is a floating number since it is the average of found ks
The best estimations are highlighted in boldface

Table 3 The running time (sec.) of each method on all datasets

Method	MNIST	Fashion-MNIST	USPS	STL-10	Reuters-8	20Newsgroups
X-means	357	321	19	235	56	470
G-means	732	647	41	605	121	1892
DB	912	785	37	588	114	1840
SS	2972	2935	88	664	150	2232
SA (ours)	**85**	**88**	**10**	**56**	**24**	**59**
DE+X-means	1304	1328	169	474	164	508
DE+G-means	1301	1352	172	475	165	550
DE+DB	1313	1340	169	473	162	536
DE+SS	2475	2413	222	512	196	701
DE+SA (ours)	1301	1327	167	469	162	494

The best running time are highlighted in boldface

The running time results tell us that (a) our method SA outperforms the other methods on all datasets, (b) our method DE+SA is faster than the 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 distances 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 the 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.

Table 4 Comparison of clustering NMI (%) with cluster estimation on all datasets

Method	MNIST	Fashion-MNIST	USPS	STL-10	Reuters-8	20Newsgroups
SA+k-means	50.19	51.90	43.35	53.41	71.12	24.15
SA+SC	79.43	62.17	65.67	53.91	70.91	20.98
SA+SEC	N/A	N/A	45.23	51.65	67.48	20.80
SA+NMF+SC	70.67	63.21	61.00	56.59	69.96	18.50
DE+SA+k-means	74.81	61.12	53.00	62.38	62.82	28.41
DE+SA+SEC	N/A	N/A	54.69	58.54	54.31	23.13
DEC+SA	78.63	63.32	58.86	69.68	59.57	25.20
SCDE+ (ours)	89.02	67.45	77.04	75.91	71.61	28.92
SCDEM+ (ours)	**90.64**	**67.86**	**79.30**	**78.13**	**74.27**	**29.28**

N/A means no enough memory on the dataset
The best NMI are highlighted in boldface

Table 5 Comparison of clustering ARI (%) with cluster estimation on all datasets

Method	MNIST	Fashion-MNIST	USPS	STL-10	Reuters-8	20Newsgroups
SA+k-means	35.90	35.84	27.53	23.99	51.01	08.41
SA+SC	69.37	41.57	46.49	21.49	57.51	10.49
SA+SEC	N/A	N/A	27.96	24.02	45.90	08.91
SA+NMF+SC	54.81	42.60	42.13	21.74	58.12	09.72
DE+SA+k-means	70.01	44.95	38.10	49.74	55.62	12.79
DE+SA+SEC	N/A	N/A	41.31	45.89	45.97	06.58
DEC+SA	74.80	48.16	43.32	58.47	49.32	12.55
SCDE+ (ours)	86.31	49.29	63.97	68.61	64.56	13.44
SCDEM+ (ours)	**89.19**	**49.94**	**68.01**	**70.86**	**66.46**	**13.52**

N/A means no enough memory on the dataset
The best ARI are highlighted in boldface.

Exp-3: Clustering with estimation of k In the third set of tests, we evaluated the effectiveness of our clustering methods SCDEM+ and 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 tests, 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 on NMI and ARI are reported in Tables 4 and 5, respectively. Although the classic DBSCAN [17] does not need to specify k, it performs poorly on these datasets. Thus, we did not choose it for comparison.

The results tell us that (a) our proposed methods SCDEM+ and SCDE+ outperform the other methods by large margins on all datasets, (b) SCDEM+ is better than its counterpart SCDE+ on all datasets by using the Siamese network to learn a better distance that can construct a more accurate similarity matrix than the traditional Euclidean distance, (c) SCDE+ significantly improves the NMI and ARI over its counterpart SA+SC by incorporating with the deep embedding, which is the same to DE+SA+k-means and DE+SA+SEC on most of the datasets, (d) 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 (e) DEC+SA achieves the third best performance by jointly learning the deep embedding and cluster assignments in an iterative way. In fact, our method SCDEM+ improves NMI and ARI by (2%, 1%, 3%, 3%, 4%, 1%) and (3%, 1%, 6%, 3%, 3%, 1%), (15%, 7%, 35%, 12%, 25%, 16%) and (19%, 4%, 57%, 21%, 35%, 8%) over the second best method SCDE+, the third best method DEC+SA on MNIST, Fashion-MNIST, USPS, STL-10, Reuters-8 and 20Newsgroups, respectively. Note that SCDEM+ improves the clustering quality over SCDE+ and obtains the best NMI and ARI on all datasets, which implies that the Siamese network can learn a more powerful distance metric for clustering. This verifies the effectiveness of our approach.

Summary From these experimental results, we find the following.

1. Our clustering methods SCDEM and SCDE perform better than existing methods, including k-means, SC, SEC, NMF+SC and DEC, on a large range of the number of clusters. Actually, SCDEM 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, 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 of cluster numbers. Furthermore, our method has the efficiency advantage of a linear complexity in the number of data points, which makes it faster than DB and SS that need multiple runs of k-means to determine the number of clusters.
3. Incorporating with the estimation method SA, our method SCDE+ can estimate the number k automatically and cluster data effectively. Moreover, by constructing the similarity matrix of SC with the good distance metric learned from the deep embedding using the Siamese network, our method SCDEM+ improves NMI and ARI over the second best method SCDE+ and outperforms the other methods by large margins on all datasets. For instance, it improves NMI and ARI by (15%, 7%, 35%, 12%, 25%, 16%) and (19%, 4%, 57%, 21%, 35%, 8%) over the third best method DEC+SA on MNIST, Fashion-MNIST, USPS, STL-10, Reuters-8 and 20Newsgroups, respectively.

7 Conclusions and future work

In this article, we have proposed an approach to improving spectral clustering with deep embedding, cluster estimation and metric learning. We first learn a good and low dimensional deep embedding for the raw data based on a multilayer autoencoder. We then estimate the number of clusters by a softmax autoencoder, and learn a distance metric by a Siamese network from the deep embedding. An extensive experimental study on image and text datasets has been conducted, which shows that our proposed approach can find a good number of clusters and outperform existing clustering methods, and are scalable to large datasets.

A couple of topics need further investigation. First, we are to develop an embedding technique that can optimize the deep embedding, estimation of the number of clusters, distance metric learning and clustering simultaneously to further improve the clustering performance. Second, we are to apply our proposed method for estimating the number of clusters for other clustering methods.

Acknowledgements This work is supported in part by National Key R&D Program of China 2018YFB1700403, NSFC 61925203 & U1636210 & 61421003 & U1802271, Science Foundation for Distinguished Young Scholars of Yunnan Province 2019FJ011 and China Postdoctoral Science Foundation 2020M673310.

References

1. Aggarwal CC, Reddy CK (2013) Data clustering: algorithms and applications. CRC Press, Bca Raton
2. Aljalbout E, Golkov V, Siddiqui Y, Strobel M, Cremers D (2018) Clustering with deep learning: taxonomy and new methods. arXiv:1801.07648v2
3. Bellman RE (1961) Adaptive control processes: a guided tour. Princeton University Press, Princeton
4. Bengio Y, Yao L, Alain G, Vincent P (2013) Generalized denoising auto-encoders as generative models. In: NIPS
5. Cai D, He X, Han J (2011) Locally consistent concept factorization for document clustering. IEEE Trans Knowl Data Eng 23(6):902–913
6. Caliński T, Harabasz J (1974) A dendrite method for cluster analysis. Commun Stat 3(1):1–27
7. Chaudhuri K, Kakade SM, Livescu K, Sridharan K (2009) Multi-view clustering via canonical correlation analysis. In: ICML
8. Chen J, Zhao Z, Ye J, Liu H (2007) Nonlinear adaptive distance metric learning for clustering. In: SIGKDD
9. Chiang MMT, Mirkin B (2010) Intelligent choice of the number of clusters in k-means clustering: an experimental study with different cluster spreads. J Classif 27(1):3–40
10. Chopra S, Hadsell R, LeCun Y (2005) Learning a similarity metric discriminatively, with application to face verification. In: CVPR
11. Davies DL, Bouldin DW (1979) A cluster separation measure. IEEE Trans Pattern Anal Mach Intell 20:224–227
12. Dian C, He X (2004) K-means clustering via principal component analysis. In: ICML
13. Duan L, Aggarwal C, Ma S, Sathe S (2019) Improving specral clustering with deep embedding and cluster estimation. In: ICDM
14. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press, Cambridge
15. Guo X, Gao L, Liu X, Yin J (2017) Improved deep embedded clustering with local structure preservation. In: IJCAI
16. Hamerly G, Elkan C (2004) Learning the k in k-means. In: NIPS
17. Han J, Kamber M, Pei J (2012) Data mining: concepts and techniques, 3rd edn. Morgan Kaufmann, Burlington
18. Hinton GE, Salakhutdinov RR (2006) Reducing the dimensionality of data with neural networks. Science 313:504–507
19. Hsu YC, Lv Z, Kira Z (2018) Learning to cluster in order to transfer across domains and tasks. In: ICLR
20. Huang P, Huang Y, Wang W, Wang L (2014) Deep embedding network for clustering. In: ICPR
21. Jain AK (2010) Data clustering: 50 years beyond k-means. Pattern Recogn Lett 31(8):651–666
22. Jain AK, Murty MN, Flynn PJ (1999) Data clustering: a review. ACM Comput Surv 31(3):264–323
23. Jiang Z, Zheng Y, Tan H, Tang B, Zhou H (2017) Variational deep embedding: an unsupervised and generative approach to clustering. In: IJCAI
24. Koch G, Zemel R, Salakhutdinov R (2015) Siamese neural networks for one-shot image recognition. In: ICML deep learning workshop, vol 2
25. Kolesnikov A, Trichina E, Kauranne T (2015) Estimating the number of clusters in a numerical data set via quantization error modeling. Pattern Recogn 48(3):941–952
26. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
27. Lewis DD, Yang Y, Rose TG, Li F (2004) Rcv1: a new benchmark collection for text categorization research. J Mach Learn Res 5:361–397
28. Ng AY, Jordan MI, Weiss Y (2002) On spectral clustering: analysis and an algorithm. In: NIPS
29. Nie F, Zeng Z, Tsang IW, Xu D, Zhang C (2011) Spectral embedded clustering: a framework for in-sample and out-of-sample spectral clustering. IEEE Trans Neural Netw 22(11):1796–1808
30. Patil C, Baidari I (2019) Estimating the optimal number of clusters k in a dataset using data depth. Data Sci Eng 4(2):132–140
31. Pelleg D, Moore A (2000) X-means: Extending k-means with efficient estimation of the number of clusters. In: ICML
32. Rousseeuw PJ (1987) Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J Comput Appl Math 20:53–65

33. Shaham U, Lederman RR (2018) Learning by coincidence: Siamese networks and common variable learning. Pattern Recogn 74:52–63
34. Shaham U, Stanton KP, Li H, Basri R, Nadler B, Kluger Y (2018) Spectralnet: spectral clustering using deep neural networks. In: ICLR
35. Shahnaz F, Berry MW, Pauca VP, Plemmons RJ (2006) Document clustering using nonnegative matrix factorization. Inf Process Manag 42:373–386
36. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: ICLR
37. Tian F, Gao B, Cui Q, Chen E, Liu TY (2014) Learning deep representations for graph clustering. In: AAAI
38. von Luxburg U (2007) A tutorial on spectral clustering. Stat Comput 17(4):395–416
39. Xie J, Girshick R, Farhadi A (2016) Unsupervised deep embedding for clustering analysis. In: ICML, pp 478–487
40. Yang B, Fu X, Sidiropoulos ND, Hong M (2017) Towards k-means-friendly spaces: simultaneous deep learning and clustering. In: ICML
41. Ye J, Zhao Z, Liu H (2007) Adaptive distance metric learning for clustering. In: CVPR
42. Yeung KY, Ruzzo WL (2001) 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
43. Zheng M, Bu J, Chen C, Wang C, Zhang L, Qiu G, Cai D (2011) Graph regularized sparse coding for image representation. IEEE Trans Image Process 20(5):1327–1336

Liang Duan is currently a postdoctoral research fellow at the School of Information Science and Engineering, Yunnan University, China. He received his Ph.D. degree in computer software and theory from Beihang University in 2020, M.S. degree in computer software and theory from Yunnan University in 2014, and B.S. degree in computer science and technology from Beihang University in 2009. His research interests lie in the areas of social network analysis and unsupervised learning.

Shuai Ma is a professor at the School of Computer Science and Engineering, Beihang University, China. He obtained his Ph.D. degrees from University of Edinburgh in 2010, and from Peking University in 2004, respectively. He was a postdoctoral research fellow in the database group, University of Edinburgh, a summer intern at Bell labs, Murray Hill, USA and a visiting researcher of MSRA. He is a recipient of the best paper award for VLDB 2010. He is an Associate Editor of VLDB Journal since 2017 and IEEE Transactions On Big Data Since 2020. His current research interests include database theory and systems, and big data.

Charu Aggarwal is a Distinguished Research Staff Member (DRSM) at the IBM T. J. Watson Research Center in Yorktown Heights, New York. He completed his undergraduate degree in Computer Science from the Indian Institute of Technology at Kanpur in 1993 and his Ph.D. degree in Operations Research from the Massachusetts Institute of Technology in 1996. He has published more than 300 papers in refereed conferences and journals and has applied for or been granted more than 80 patents. He is author or editor of 16 books, including textbooks on data mining, recommender systems, and outlier analysis. Because of the commercial value of his patents, he has thrice been designated a Master Inventor at IBM. He has received several internal and external awards, including the EDBT Test-of-Time Award (2014) and the IEEE ICDM Research Contributions Award (2015). He has also served as program or general chair of many major conferences in data mining. He is a fellow of the SIAM, ACM, and the IEEE, for "contributions to knowledge discovery and data mining algorithms".

Saket Sathe is a Research Staff Member (RSM) at the IBM T. J. Watson Research Center in Yorktown Heights, New York. He has worked at IBM Research (Australia/United States) since 2013. Saket received a Ph.D. degree in Computer Science from EPFL (Lausanne) in 2013. Before that he received a Master's (M.Tech.) degree in Electrical Engineering from the Indian Institute of Technology at Bombay and also spent one year working for a startup. His primary areas of interest are data mining and data management. Saket has served on program committees of several top-ranked conferences and has been invited to review papers for prominent peer-reviewed journals. His research has led to more than 25 papers and 5 patents. His work on sensor data management received the runner-up best-paper award in IEEE CollaborateCom 2014. He is a member of the ACM, IEEE, and the SIAM.