DeepCluster: A General Clustering Framework based on Deep Learning

Kai Tian[1], Shuigeng Zhou[1⋆], and Jihong Guan[2]

[1] School of Computer Science, and
Shanghai Key Lab of Intelligent Information Processing
Fudan University, Shanghai 200433, China
{ktian14,sgzhou}@fudan.edu.cn
[2] Department of Computer Science and Technology
Tongji University, Shanghai 201804, China
jhguan@tongji.edu.cn

Abstract. In this paper, we propose a general framework DeepCluster to integrate traditional clustering methods into deep learning (DL) models and adopt Alternating Direction of Multiplier Method (ADMM) to optimize it. While most existing DL based clustering techniques have separate feature learning (via DL) and clustering (with traditional clustering methods), DeepCluster simultaneously learns feature representation and does cluster assignment under the same framework. Furthermore, it is a general and flexible framework that can employ different networks and clustering methods. We demonstrate the effectiveness of DeepCluster by integrating two popular clustering methods: K-means and Gaussian Mixture Model (GMM) into deep networks. The experimental results shown that our method can achieve state-of-the-art performance on learning representation for clustering analysis.

1 Introduction

Clustering is one of the most important techniques for analyzing data in an unsupervised manner, it has a wide range of applications including computer vision [11, 14, 23], natural language processing [1, 2, 26] and bioinformatics [22, 28]. In the past decades, a large number of algorithms have been proposed to handle clustering problems [6, 15]. However, there is no algorithm that fits all problems. Clustering method choosing depends on the data to handle and the specific task. Roughly, there are two sets of approaches, the feature-based clustering algorithms and the similarity-based clustering algorithms. Most of them try to find the intrinsic data structure from the original feature space or the underlying subspace.

Among the existing algorithms, K-means [12] and Gaussian Mixture Models (GMM) [4] are two popular feature-based methods. K-means makes hard clustering that assigns each sample to its nearest cluster center. GMM assumes that data are generated from several independent Gaussian distributions and tries to infer these distributions from the data. Thus, it makes soft assignments. However, they both do clustering in the original

⋆ correspondence author.

feature space. Spectral clustering [15] is a representative algorithm of similarity-based clustering or subspace clustering methods. Most of those approaches start with building an affinity matrix and project the original data to a linear subspace. Finally, clustering is done in the subspace.

One problem with most feature-based clustering methods is that they cannot scale well to high-dimensional data due to the curse of dimensionality. In high-dimensional data analysis, it is more reasonable to consider some compact and representative features instead of the whole feature space. Recently, deep learning (DL) has been developed and with a great success in many areas, such as image classification and speech recognition [19]. DL aims to learn a powerful representation from the raw data through high-level non-linear mapping [3]. Recently, how to use deep representation to improve clustering performance becomes a hot research topic.

Basically, there are mainly two ways to use deep features for clustering. One is clustering the hidden features that are extracted from a well-trained deep network [26, 21]. However, these approaches cannot fully exploit the power of deep neural network for clustering. The other is to embed an existing clustering method into DL models, which is an end-to-end approach. For example, [18] integrates K-means algorithm into deep autoencoders and does cluster assignment on the middle layers. It alternatively updates the network parameters and cluster centers. [25] proposes a clustering objective to learn non-linear deep features, which minimizes the KL divergence between the auxiliary target distribution and the model assignments.

In this paper, we propose a new and general framework to integrate traditional clustering methods into deep learning models and develop an algorithm to optimize the underlying non-convex and non-linear objective based on Alternating Direction of Multiplier Method (ADMM) [5]. Concretely, we can use a deep autoencoder to reconstruct the data, and associate deep features with clustering methods by introducing a dummy variable (say $\mathbf{Y}$). We combine deep models and clustering methods with the constraint $\mathbf{Y} = f_{\boldsymbol{\theta}_1}(\mathbf{X})$ where $\mathbf{X}$ is the data and $f_{\boldsymbol{\theta}_1}(\cdot)$ is the encoder of deep network. In the optimization process, we optimize each part of the model's parameters alternatively. Our experimental results shown that instead of directly clustering the hidden features, our framework works better.

The novelties and contributions of our work are as follows:

- A general clustering framework base on DL, where clustering parameters (except for the network parameters) can be represented in closed form.
- Both network and clustering method are configurable according to user requirements, which make it a flexible framework.
- Based on ADMM, relaxation is introduced to the model by doing clustering on dummy variable $\mathbf{Y}$.
- Experiments on real datasets show that the new method can achieve state-of-the-art clustering performance.

2 Related Work

Clustering is an extensively studied area, and up to now many clustering methods have been developed. Here, we review mainly on the clustering methods that employ DL

techniques, and briefly highlight the advantages/differences of our work over/from the most-related existing ones.

Among the popular clustering methods, K-means and GMM are widely used in many applications. However, they have two drawbacks: one is that they mainly work in the original feature space; the other is that they cannot handle large and high-dimensional data sets well.

Spectral clustering and its variants are extensively popular among subspace clustering methods. [13] develops a distributed framework to solve sparse subspace clustering via ADMM. However, it considers only linear subspaces. To address this problem, another approach [16] was proposed to incorporate nonlinearity into subspace clustering. The objective is to minimize the data reconstruction error and add a sparsity prior to the deep features.

To make use of deep learning features, some works first train a network and then cluster the hidden features. One of them is to learn a deep autoencoder on a graph and then run K-means algorithm on the non-linear embedding to get cluster assignments [21]. [26] introduces a novel method for short text clustering by first training a Convolutional Neural Network (CNN). The target to train CNN is spectral hashing code, and after network training it extracts deep features on which K-means is run. These approaches have separate feature learning and clustering.

To conduct end-to-end clustering in deep networks, [18] proposes a model to simultaneously learn the deep representations and the cluster centers. It makes hard assignment to each sample and directly does clustering on the hidden features of deep autoencoder. A recent attempt is the Deep Embedding Clustering (DEC) method [25], which achieves stat-of-the-art results on many datasets, but it may fails when closely related clusters exist.

Different from the above works, our DeepCluster is a general DL based clustering framework that can embrace different clustering methods and network structures such as DAEs, CNNs and RNNs. It provides a flexible mechanism to fit a clustering method to a deep network for a specific clustering task. Concretely, the most-related existing methods are DAEC [18] and DEC [25].

Though DAEC is the first work to explore deep feature learning and clustering simultaneously, it does clustering directly on the feature space, which is not flexible. DEC uses only the encoder part of DAE, and assumes that the model can correctly make high confidence predictions, which if not satisfied, it performs badly. DeepCluster introduces a copy of features $\mathbf{Y}$ and does cluster assignment on it, which makes feature learning and clustering independent from each other given this $\mathbf{Y}$. On the one hand, DeepCluster is able to fully take advantages of deep representation for clustering; On the other hand, recent studies have shown that ADMM can be used to train deep neural network without backpropagation, which means that our DeepCluster is parallelizable and can be used in asynchronous systems [20]. In summary, DeepCluster provides the feasibility of combining the most suitable network and clustering method for a specific clustering task.

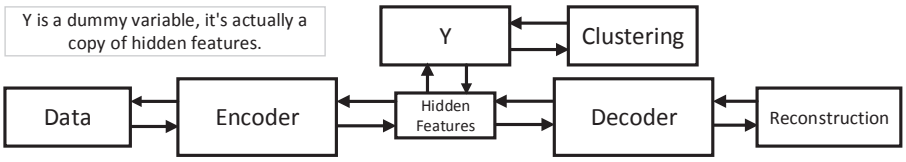


Fig. 1: The DeepCluster architecture.

3 Preliminaries

3.1 Deep Autoencoder (DAE)

A single autoencoder is a three layer neural network with one hidden layer, its output is to reconstruct the input $\boldsymbol{x}$. An autoencoder is composed of an encoder and a decoder. The encoder can be formalized as

$$\boldsymbol{a}_1 = f(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1) \tag{1}$$

where $\boldsymbol{W}_1$ is the weight and $\boldsymbol{b}_1$ is the bias of encoder, $\boldsymbol{a}_1$ means the hidden features of $\boldsymbol{x}$. The decoder is formulated as

$$\hat{\boldsymbol{x}} = g(\boldsymbol{W}_2\boldsymbol{a}_1 + \boldsymbol{b}_2) \tag{2}$$

where $\boldsymbol{W}_2$ is the weight and $\boldsymbol{b}_2$ is the bias of decoder, $\hat{\boldsymbol{x}}$ is the reconstruction of input.

Deep autoencoder is to stack several autoencoders to build a deep neural network, where the hidden features learned by a lower level autoencoder are fed to a higher level autoencoder as input. The first layer autoencoder takes raw data as input.

3.2 Alternating Directed Method of Multipliers (ADMM)

Let us consider a general optimization problem as suggested in [7], which can be formulated as follows:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} h(\boldsymbol{x}) + o(\boldsymbol{D}\boldsymbol{x}) \tag{3}$$

where $\boldsymbol{D}$ is an $m \times n$ matrix, which is often assumed to have full column rank. h and o are supposed to be convex functions on $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively. To solve Eq. (3), we can rewrite it by introducing an additional dummy variable $\boldsymbol{z} \in \mathbb{R}^m$:

$$\begin{aligned} &\min h(\boldsymbol{x}) + o(\boldsymbol{z}) \\ &s.t.\ \ \boldsymbol{D}\boldsymbol{x} = \boldsymbol{z}. \end{aligned} \tag{4}$$

This is a constrained convex problem that can be solved by the classical augmented Lagrangian algorithm (ALM). However, it is not decomposable due to the constraints, and the subproblems are unlikely to be easier to solve than the original one. The alternating directed method of multipliers (ADMM) is proposed to overcome the drawbacks of ALM. It is robust and supports decomposition.

To solve Eq. (3) and Eq. (4), ADMM uses the following forms with a scalar parameter $\rho > 0$:

$$\begin{aligned}
\boldsymbol{x}^{k+1} &\in \arg\min_{\boldsymbol{x}\in\mathbb{R}^n} \left\{ h(\boldsymbol{x}) + o(\boldsymbol{z}^k) + \left\langle \lambda^k, \boldsymbol{D}\boldsymbol{x} - \boldsymbol{z}^k \right\rangle + \frac{\rho}{2} \parallel \boldsymbol{D}\boldsymbol{x} - \boldsymbol{z}^k \parallel^2 \right\} \\
\boldsymbol{z}^{k+1} &\in \arg\min_{\boldsymbol{z}\in\mathbb{R}^m} \left\{ h(\boldsymbol{x}^{k+1}) + o(\boldsymbol{z}) + \left\langle \lambda^k, \boldsymbol{D}\boldsymbol{x}^{k+1} - \boldsymbol{z} \right\rangle + \frac{\rho}{2} \parallel \boldsymbol{D}\boldsymbol{x}^{k+1} - \boldsymbol{z} \parallel^2 \right\} \\
\boldsymbol{\lambda}^{k+1} &= \boldsymbol{\lambda}^k + \rho(\boldsymbol{D}\boldsymbol{x}^{k+1} - \boldsymbol{z}^{k+1})
\end{aligned} \tag{5}$$

From Eq. (5), we can see that ADMM essentially decouples the functions h and o, and makes it possible to exploit the individual structures of h and o. Thus, the optimization procedure can be efficient and parallelizable.

Although the ADMM method was proposed to solve convex problems, many studies have shown that this approach can be used in non-convex cases, such as nonnegative matrix factorization [5] and network lasso [8].

4 The DeepCluster Framework

In this section, we give the architecture, formal formulation and algorithm of our DeepCluster framework.

Fig. 1 is the architecture of DeepCluster. The encoder learns hidden feature representations and $\mathbf{Y}$ is a dummy variable that is required to be equal to the features. We do clustering on $\mathbf{Y}$ instead of the features. The constraint controls the interaction between autoencoder and the clustering method. During the clustering process, $\mathbf{Y}$ is adjusted to minimize the objective, meanwhile the constraint requires the encoder to learn better representations for clustering. This framework is flexible and robust, it can be seen as a multi-task learning model or the clustering part can be seen as a regularizer of deep autoencoder (DAE). By introducing a copy of deep features as $\mathbf{Y}$ and requiring it equal to the hidden features, we essentially introduce a kind of relaxation to the optimization procedure, which makes the framework decomposable to two components.

DeepCluster is formulated as follows:

$$\begin{aligned}
&\min : \|\mathbf{X} - \hat{\mathbf{X}}\|_F^2 + \lambda * \mathcal{G}_{\mathbf{w}}(\mathbf{Y}) \\
&\text{s.t.} \quad \mathbf{Y} = f_{\boldsymbol{\theta}_1}(\mathbf{X})
\end{aligned} \tag{6}$$

where $\mathbf{X}$ is the raw data to be clustered and $\hat{\mathbf{X}}$ is the reconstruction learned by the deep network. $\mathcal{G}_w(\mathbf{Y})$ can be any specific clustering objective function such as K-means. $\mathbf{Y}$ is the dummy variable in order to make parameters decomposable. λ defines a trade-off between the network objective and the clustering objective. When $\lambda = 0$, the problem degrades to deep network optimization. We add the constraint on $\mathbf{Y}$ to make it close to the features learned from $\mathbf{X}$.

The first part of the above objective is non-convex as deep neural networks contain multilayer non-linear transformations. It is hard to optimize this model by gradient-based optimization methods directly, because some clustering methods cannot be solved

Algorithm 1 The general learning algorithm of DeepCluster

Input: input data $\mathbf{X}$, hyperparameters ρ, λ, learning rate η
Output: a well-trained autoencoder, cluster assignments

Initialize parameters of DAE $\boldsymbol{\theta}$, parameters of clustering model $\mathbf{w}$
while not converged **do**
  update $\boldsymbol{\theta}$: $\boldsymbol{\theta} = \boldsymbol{\theta} - \eta * d\boldsymbol{\theta}$
  update $\mathbf{Y}$:
  $\mathbf{Y} = \text{argmin}_{\mathbf{Y}}\ \lambda * \mathcal{G}_{\mathbf{w}}(\mathbf{Y}) + \frac{\rho}{2}\|\mathbf{Y} - f_{\boldsymbol{\theta}_1}(\mathbf{X}) + \mathbf{U}\|_F^2$
  update $\mathbf{U}$: $\mathbf{U} = \mathbf{U} + \mathbf{Y} - f_{\boldsymbol{\theta}_1}(\mathbf{X})$
  update $\mathbf{w}$: $\mathbf{w} = \text{argmin}_{\mathbf{w}}\ \mathcal{G}_{\mathbf{w}}(\mathbf{Y})$
return $\boldsymbol{\theta}, \mathbf{w}$

by them. In this paper, we adopt ADMM to optimize it. First, we derive its augmented Lagrangian [9] as follows:

$$\begin{aligned}\mathcal{L}_\rho(\boldsymbol{\theta}, \mathbf{Y}, \mathbf{U}, \mathbf{w}) =& \|\mathbf{X} - \hat{\mathbf{X}}\|_F^2 + \lambda * \mathcal{G}_{\mathbf{w}}(\mathbf{Y}) + \\ & \frac{\rho}{2}\|\mathbf{Y} - f_{\boldsymbol{\theta}_1}(\mathbf{X}) + \mathbf{U}\|_F^2\end{aligned} \tag{7}$$

where $\mathbf{U}$ is the scaled dual variable (or the reciprocal of $\boldsymbol{\lambda}$ in Eq. (5)) and $\rho > 0$ is the penalty parameter. ρ is a very important parameter to control how close between $\mathbf{Y}$ and $f_{\boldsymbol{\theta}_1}(\mathbf{X})$. Then, we solve the equation by alternatively optimizing some parameters while keeping the others fixed. For a deep autoencoder that consists of an encoder and a decoder, we denote $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2\}$ as the total network parameters, where $\boldsymbol{\theta}_1$ indicates the encoder parameters and $\boldsymbol{\theta}_2$ means the decoder parameters. To optimize $\boldsymbol{\theta}$, we actually optimize $\|\mathbf{X} - \hat{\mathbf{X}}\|_F^2 + \alpha \cdot \frac{\rho}{2}\|\mathbf{Y} - f_{\boldsymbol{\theta}_1}(\mathbf{X}) + \mathbf{U}\|_F^2$. Here, α is to control the gradient influence of the constraint, which is set to 1 by default.

Alg. 1 outlines the general optimization process. Note that by introducing $\mathbf{Y}$, we can decompose the parameters into two parts and keep the parameter update formulations of clustering methods $\mathcal{G}_{\mathbf{w}}(\mathbf{Y})$ remain the same as before. The challenge is how to optimize $\mathbf{Y}$ that depends on the specific clustering model. This framework is flexible because we can select the most appropriate clustering algorithms for any specific clustering task. Moreover, the deep network is also configurable. There are many variants of deep autoencoders such as denoise autoenocders, convolutional autoencoders and variational autoenocders.

In what follows, we give a brief convergence and complexity analysis on DeepCluster. DeepCluster's objective function consists of three parts. The first part is the reconstruction error of DAE, the parameter update rules here are the same as stochastic gradient descent (SGD) or its variants. The second part is the clustering objective that is independent from DAE when $\mathbf{Y}$ is given. The last part is the constraint imposed on $\mathbf{Y}$, which is a convex function of $\mathbf{Y}$. Following the ADMM optimization procedure, as DAE is nonconvex function of $\boldsymbol{\theta}$, it is hard to prove DeepCluster's global convergence directly. Although there are some works to prove the convergence of ADMM on nonconvex problems under some specific assumptions, they do not suit for our case [10, 24]. However, our experiments have shown that DeepCluster can converge to a good local minima when the value of ρ is properly chosen.

The computational complexity of DeepCluster consists of three parts: the complexity of clustering algorithm, the complexity of DAE, and the computation of $\mathbf{Y}$. The complexity of $\mathbf{Y}$ optimization is related to the clustering algorithm. For example, it takes $O(TND)$ in DC-Kmeans where T, N and D are the number of iterations, the number of samples and the hidden feature dimensionality respectively. This term can often be omitted in the complexity of clustering algorithm. Thus, DeepCluster has similar time complexity to that of DAEC. However, DeepCluster needs additional $O(ND)$ space to store $\mathbf{Y}$ and $\mathbf{U}$.

5 Two Specific DeepCluster Implementations

Here we give two specific implementations of integrating commonly used clustering algorithms into deep autoencoder (DAE). We choose K-means and GMM as examples.

5.1 DC-Kmeans: Integrating K-means into DAE

K-means tries to find the nearest cluster center for each sample. That is, if the j-th cluster center is the closest to $\mathbf{x}_i$, it assigns $\mathbf{x}_i$ to cluster j with 100% confidence. K-means is a very simple and has no tunable parameter except *K*. In the following, we embed K-means into DeepCluster, and refer this method as DC-Kmeans for convenience.

Following Eq. (6), we have the objective function of DC-Kmeans as follows:

$$\begin{aligned} &\min : \frac{1}{N}\sum_{i=1}^{N}\|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2 + \frac{\lambda}{2} * \|\mathbf{y}_i - \mathbf{c}_i^*\|^2 \\ &s.t. \quad \mathbf{y}_i = f_{\boldsymbol{\theta}_1}(\mathbf{x}_i) \;\; i = 1, ..., N \end{aligned} \tag{8}$$

where $\mathbf{c}_i^* = \text{argmin}_{\mathbf{c}_j} \|\mathbf{y}_i - \mathbf{c}_j\|^2, j = 1, ..., K$ is the closest centroid to $\mathbf{y}_i$. Besides, N is the total number of samples and K is the number of clusters. And following Eq. (7), we have the corresponding augmented Lagrangian of Eq. (8):

$$\begin{aligned} \mathcal{L}_\rho(\boldsymbol{\theta}, \mathbf{Y}, \mathbf{U}, \mathbf{C}) = &\frac{1}{N}\sum_{i=1}^{N}\|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2 + \frac{\lambda}{2} * \|\mathbf{y}_i - \\ &\mathbf{c}_i^*\|^2 + \frac{\rho}{2}\|\mathbf{y}_i - f_{\boldsymbol{\theta}_1}(\mathbf{x}_i) + \mathbf{u}_i\|^2 \end{aligned} \tag{9}$$

To solve the above equation, we treat an autoencoder as a non-linear and non-convex function of $\boldsymbol{\theta}$. Inspired by ADMM, we alternatively optimize each part of the variables in this objective function. Concretely, we use gradient-based optimization algorithm to find a good candidate for the network parameters $\boldsymbol{\theta}$. As for $\mathbf{Y}$, $\mathbf{C}$ and $\mathbf{U}$, closed form solutions are available as follows (detailed derivations are omitted):

$$\begin{aligned} \mathbf{y}_i^{\text{new}} &= \frac{\lambda * \mathbf{c}_i^* + \rho * (f_{\boldsymbol{\theta}_1^{\text{new}}}(\mathbf{x}_i) - \mathbf{u}_i)}{\lambda + \rho} \\ \mathbf{u}_i^{\text{new}} &= \mathbf{u}_i + f_{\boldsymbol{\theta}_1^{\text{new}}}(\mathbf{x}_i) - \mathbf{y}_i^{\text{new}} \\ \mathbf{c}_j^{\text{new}} &= \frac{1}{N_j}\sum_{\mathbf{x}_j \in \mathcal{C}_j} \mathbf{y}_i^{\text{new}} \end{aligned} \tag{10}$$

Algorithm 2 The Learning Algorithm of DC-Kmeans

Input: input data $\mathbf{X}$, hyperparameters ρ, λ, learning rate η
Output: a well-trained autoencoder, cluster assignments

Initialize parameters of DAE $\boldsymbol{\theta}$, parameters of clustering model $\mathbf{C}$
while not converged **do**
  update $\boldsymbol{\theta}$: $\boldsymbol{\theta} = \boldsymbol{\theta} - \eta * d\boldsymbol{\theta}$
  update $\mathbf{Y}$: $\mathbf{y}_i = \frac{\lambda * \mathbf{c}_i^* + \rho * (f_{\boldsymbol{\theta_1}}(\mathbf{x}_i) - \mathbf{u}_i)}{\lambda + \rho}, \ \ i = 1, ..., N$
  update $\mathbf{U}$: $\mathbf{u}_i = \mathbf{u}_i + \mathbf{y}_i - f_{\boldsymbol{\theta_1}}(\mathbf{x}_i), \ \ i = 1, ..., N$
  update $\mathbf{C}$: $\mathbf{c}_j = \frac{1}{N_j} \sum_{\mathbf{x}_i \in \mathcal{C}_j} \mathbf{y}_i, \ \ j = 1, ..., K$
return $\boldsymbol{\theta}, \mathbf{C}$

where N_j is the size of the j-th cluster, $\mathcal{C}_j$ denotes the j-th cluster data set. We set $\lambda = 1$ in our experiments for simplicity. We outline the learning algorithm in Alg. 2.

5.2 DC-GMM: Integrating GMM into DAE

Another widely-used feature-based clustering method is Gaussian mixture model (GMM). GMM assumes that all data samples are generated from multiple independent Gaussian distributions. GMM can be seen as generalized K-means clustering by incorporating the covariance structure of each cluster. GMM is a probabilistic model and its objective function is to maximize likelihood $P(\mathbf{Y})$. Let π_i be the mixing coefficient of each Gaussian distribution component, $\mathbf{z}_i$ be a K-dimensional binary random variable with $\sum_k z_{ik} = 1$. We denote the posterior probability of z_{ik} as $p(z_{ik} = 1|\mathbf{y}_i) = \gamma(z_{ik})$. Each cluster is assumed a multivariate Gaussian distribution with mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$.

We can simply treat this model as adding mixture-of-Gaussian prior to the dummy variables $\mathbf{Y}$, rather than to deep features. The log likelihood of $\mathbf{Y}$ is defined as below:

$$\ln p(\mathbf{Y}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{i=1}^{N} \ln \left[\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{y}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right] \tag{11}$$

where $\mathcal{N}(\boldsymbol{y}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is a multivariate Gaussian distribution with $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ as its parameters. In this case, our objective function is:

$$\begin{aligned} &\min : \frac{1}{N} \sum_{i=1}^{N} \left\{ \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|^2 - \lambda * \ln \left[\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{y}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right] \right\} \\ &s.t. \quad \boldsymbol{y}_i = f_{\boldsymbol{\theta}_1}(\boldsymbol{x}_i) \ \ i = 1, ..., N \end{aligned} \tag{12}$$

And the augmented Lagrangian of this model is:

$$\begin{aligned}\mathcal{L}_\rho(\boldsymbol{\theta}, \mathbf{Y}, \mathbf{U}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \frac{1}{N}\sum_{i=1}^{N}\Bigg\{ \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|_2^2 \\ - \lambda * \ln\left[\sum_{k=1}^{K}\pi_k p(\boldsymbol{y}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right] \\ + \frac{\rho}{2}\|\boldsymbol{y}_i - f_{\boldsymbol{\theta}_1}(\boldsymbol{x}_i) + \boldsymbol{u}_i\|^2\Bigg\}\end{aligned} \tag{13}$$

We can derive the update equation of $\boldsymbol{y}_i$ in closed form as follows:

$$\begin{aligned}\boldsymbol{y}_i^{\text{new}} = \Big[\lambda\sum_{k=1}^{K}\gamma(z_{ik})\boldsymbol{\Sigma}_k^{-1} + \rho\mathbf{I}\Big]^{-1}\Big[\rho * (f_{\boldsymbol{\theta}_1}(\boldsymbol{x}_i) - \boldsymbol{u}_i) \\ + \lambda\sum_{k=1}^{K}\gamma(z_{ik})\boldsymbol{\Sigma}_k^{-1}\boldsymbol{\mu}_k\Big]\end{aligned} \tag{14}$$

where $\mathbf{I}$ is the identity matrix. The other parameters remains the same as in standard GMM algorithm:

$$\begin{aligned}\mathbf{u}_i^{\text{new}} &= \mathbf{u}_i + \mathbf{y}_i^{\text{new}} - f_{\boldsymbol{\theta}_1^{\text{new}}}(\mathbf{x}_i) \\ \boldsymbol{\mu}_k^{\text{new}} &= \frac{1}{N_k}\sum_{i=1}^{N}\gamma(z_{ik})\mathbf{y}_i^{\text{new}} \\ \boldsymbol{\Sigma}_k^{\text{new}} &= \frac{1}{N_k}\sum_{k=1}^{K}\gamma(z_{ik})(\mathbf{y}_i^{\text{new}} - \boldsymbol{\mu}_k^{\text{new}})(\mathbf{y}_i^{\text{new}} - \boldsymbol{\mu}_k^{\text{new}})^T \\ \pi_k^{\text{new}} &= \frac{N_k}{N}\end{aligned} \tag{15}$$

where $N_k = \sum_{i=1}^{N}\gamma(z_{ik})$.

As in DC-Kmeans, we set $\lambda = 1$ in this model too. The learning algorithm of DC-GMM is given in Alg. 3.

6 Performance Evaluation

To evaluate our framework, we use three real-world datasets and compare our methods against several existing clustering methods. As our aim is to demonstrate the effectiveness of our framework, instead of pursuing the best performance on each dataset, we choose the vanilla deep autoencoder for simplicity. There are many choices that can be exploited to achieve better results, including substituting the network or the clustering algorithm by a better one.

Algorithm 3 The Learning Algorithm of DC-GMM

Input: input data $\mathbf{X}$, hyperparameters ρ, λ, learning rate η
Output: a well-trained autoencoder, cluster assignments

Initialize parameters of DAE $\boldsymbol{\theta}$, parameters of GMM $\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}$
while not converged **do**
  update $\boldsymbol{\theta}$: $\boldsymbol{\theta} = \boldsymbol{\theta} - \eta * d\boldsymbol{\theta}$
  update Y:

$$\boldsymbol{y}_i = \Big[\lambda \sum_{k=1}^{K} \gamma(z_{ik}) \boldsymbol{\Sigma}_k^{-1} + \rho \mathbf{I}\Big]^{-1} \Big[\rho * (f_{\boldsymbol{\theta}_1}(\boldsymbol{x}_i) - \boldsymbol{u}_i) + \lambda \sum_{k=1}^{K} \gamma(z_{ik}) \boldsymbol{\Sigma}_k^{-1} \boldsymbol{\mu}_k\Big], \quad i = 1, ..., N$$

  update U: $\boldsymbol{u}_i = \boldsymbol{u}_i + \boldsymbol{y}_i - f_{\boldsymbol{\theta}_1}(\boldsymbol{x}_i), \quad i = 1, ..., N$
  update μ: $\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i=1}^{N} \gamma(z_{ik}) \boldsymbol{y}_i, \quad k = 1, ..., K$
  update Σ:

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{k=1}^{K} \gamma(z_{ik}) (\boldsymbol{y}_i - \boldsymbol{\mu}_k)(\boldsymbol{y}_i - \boldsymbol{\mu}_k)^T,$$

  $k = 1, ..., K$
  update $\boldsymbol{\pi}$: $\pi_k = \frac{N_k}{N}, k = 1, ..., K$
return $\boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}$

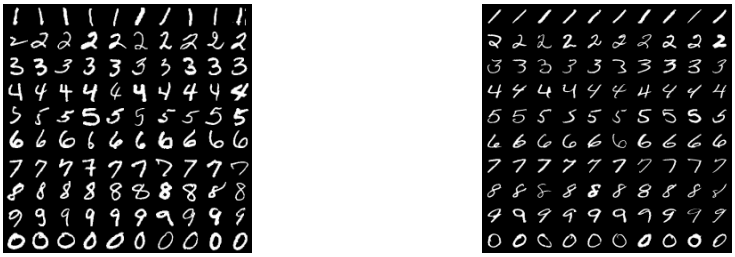


Fig. 2: Sample images of MNIST (left) and USPS (right).

6.1 Experimental Settings

Baseline algorithms: We compare DeepCluster with K-means and GMM method on the original data space as well as deep feature space of a fine-tuned autoencoder. We denote them as DAE+Kmeans and DAE+GMM. We also compare DeepCluster with DAEC [18] that is closely related to DC-Kmeans and DEC [25] that is a state-of-the-art unsupervised clustering model. To evaluate the effectiveness of DeepCluster, we simply set the number of clusters to the true number of classes for all experiments.

Datasets: Three real datasets are used to do evaluation and comparison, including two handwritten digits datasets and one text datasets[*].

– **MNIST:** a benchmark dataset for many machine learning tasks. It has 60,000 handwritten digit images in the training dataset and 10,000 images in the test dataset. Each image is of 28*28 pixel size. Some sample images are shown in Fig. 2(left).

[*] USPS can be downloaded from: `http://www.cs.nyu.edu/~roweis/data.html`

- **USPS:** also a handwritten digits (0-9) dataset and each class have 1100 samples. These images have been deslanted and size normalized. So the total number of images in this dataset is 11000, and each image is of 16*16 pixel size. Some sample images are shown in Fig. 2(right).
- **Reuters10k:** there are 810000 English news stories in the Reuters dataset. Here, following [25], we consider the four root categories: corprate/industrial, government/social, markets, economics and computer, and use TF-IDF features of the top-2000 frequently used word stems. Finally, we randomly select 10000 examples as in DEC because some methods do not scale well.

Table 3 summarizes the major statistics of the datatsets.

Table 1: Dataset Information

Dataset	#Classes	#Dims	#Samples
MNIST	10	784	70,000
USPS	10	256	11,000
Reuters10k	4	2000	10,000

Evaluation Measures: To measure clustering performance, we adopt three metrics, *i.e.*, *Accuracy*, Normalized Mutual Information (*NMI*), *Purity* and Adjusted Rand Index (*ARI*). High value of these metrics indicates better performance. These measures are defined as follows:

$$\begin{aligned} ACC &= \max_{m} \frac{\sum_{i=1}^{N} \mathbf{1}(r_i = m(c_i))}{N} \\ NMI &= \frac{I(\boldsymbol{r}, \hat{\boldsymbol{c}})}{(H(\boldsymbol{r}) + H(\hat{\boldsymbol{c}}))/2} \\ Purity &= \sum_{k=1}^{K} \frac{max_i(n_k^i)}{N} \\ ARI &= \frac{\sum_{ij} \binom{n_{ij}}{2} - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}] - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}]/\binom{n}{2}} \end{aligned} \tag{16}$$

where $\mathbf{1}(\cdot)$ is an indicator function, r_i is the ground-truth label, c_i is the cluster assignment and $m(\cdot)$ denotes all possible one-to-one mapping between clusters and labels. $\boldsymbol{r}$ denotes the ground truth labels and $\hat{\boldsymbol{c}}$ is the cluster assignments. $I(\cdot)$ is the mutual information metric and H is the entropy. n_k^i is the number of samples of class i but assigned to cluster k. ARI is quite complex, n_{ij}, a_i, b_j are values from the contingency table (see details in [17]).

6.2 Implementation Details

In order to conduct a fair comparison, we choose the same network structure as in [25]. In other words, we stack four autoencoders to build a deep autoencoder and the encoder structure is *d*-500-500-2000-10, where *d* is the dimension of original data. To obtain a good initialization for our methods, we also do layer-wise pre-training and fine-tuning.

Note that the features of Reuters10k are very sparse and feature cardinality ranges from 0 to 38, which is quite different from the other datasets. We conduct two different sets of initialization steps and activation functions. For Reuters10k, we use exactly the same initialization and the same activation functions as DEC. The pre-training of the two image datasets are done by layer-wise training of RBMs, and the fine-tuning are done as DAEC [18] with sigmoid activation functions. We optimize $\boldsymbol{\theta}$ by the AdaDelta algorithm, which is a variant of the stochastic gradient decent algorithm [27]. However, we find SGD is more suitable for the Reuters10k dataset as suggested in DEC.

In the training stage of our methods, we use warm start by initializing $\mathbf{Y}$ to $f_{\boldsymbol{\theta}_1}(\mathbf{X})$ and $\mathbf{U} = 0$. As mentioned before, we set $\lambda = 1$ for all experiments. We set $\alpha = 0.01$ for DC-GMM models on image datasets and DC-Kmeans on Reuters10K dataset. We carry out linear search to find the best ρ. The convergence threshold of DC-Kmeans is set to 0.1% and the max iteration is set to 200.

For K-means and GMM-based methods, we run each method 10 times and report the best performance. For DAEC, we vary λ in $\{0.2 * i\}, i = 1, ..., 5$ and report the best performance. We give the best results of DEC by running its code and do the hyper-parameter selection as suggested[*].

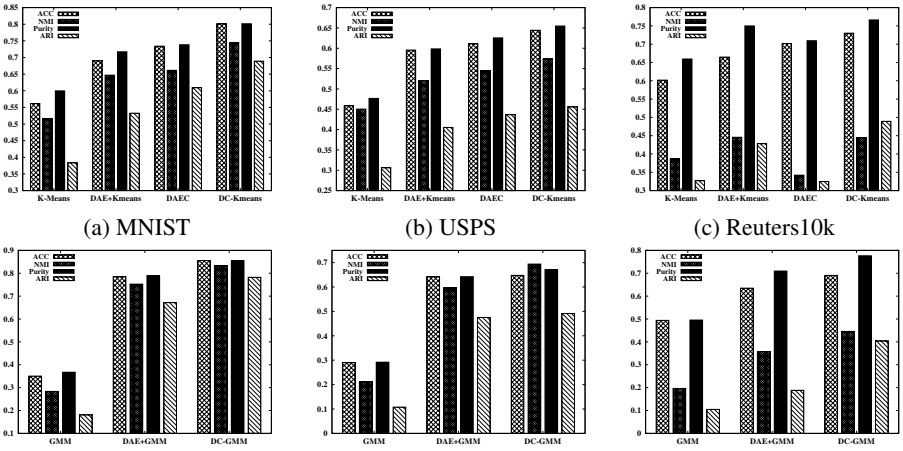
Table 2: Performance comparison on the three real datasets. The highest values of all metrics on the datasets are in bold.

Method	MNIST				USPS				Reuters10K			
	Accuracy	NMI	Purity	ARI	Accuracy	NMI	Purity	ARI	Accuracy	NMI	Purity	ARI
K-means	0.5618	0.5163	0.5997	0.3837	0.4585	0.4503	0.4767	0.3063	0.6018	0.3863	0.6595	0.3271
GMM	0.3505	0.2836	0.3672	0.1811	0.29	0.2107	0.2917	0.1077	0.494	0.1964	0.4954	0.1048
DAE+Kmeans	0.6903	0.6469	0.7171	0.5325	0.5955	0.5203	0.5985	0.4053	0.6648	0.4456	0.7499	0.4283
DAE+GMM	0.7853	0.7525	0.7896	0.6718	0.6422	0.5967	0.6422	0.475	0.6349	0.3576	0.7096	0.1884
DAEC	0.734	0.6615	0.7383	0.6093	0.6111	0.5449	0.6255	0.4368	0.7019	0.342143	0.7096	0.3247
DEC	0.8496	0.8273	0.8496	0.7721	0.6246	0.6191	0.651	0.4692	0.6945	**0.5124**	0.7726	**0.4963**
DC-Kmeans	0.8015	0.7448	0.8015	0.689	0.6442	0.5737	0.6546	0.4559	**0.7301**	0.4447	0.7663	0.4892
DC-GMM	**0.8555**	**0.8318**	**0.8555**	**0.7823**	**0.6476**	**0.6939**	**0.6713**	**0.4913**	0.6906	0.4458	**0.7765**	0.404

6.3 Experimental Results

Effect of Deep Features: We compare the performance of different K-means based algorithms and GMM based algorithms on all datasets, the results are shown in Fig. 3. Obviously, the performance on the raw data is worse than on learnt feature spaces. When features learnt by a fine-tuned deep autoencoder are used, better performance can be achieved. Moreover, if we embed K-means algorithm into deep models as in DAEC, the results are much better, as shown in Fig. 3(a),(b),(c). DC-Kmeans outperforms DAEC on all metrics because of the introduced relaxation. Fig. 3(d),(e),(f) also show that feature learning and clustering jointly is better than doing them separately. Thus, we can conclude that a more compact and representative feature space is important and effective for clustering, and deep learning can learn such features.

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



(a) MNIST (b) USPS (c) Reuters10k

(d) MNIST (e) USPS (f) Reuters10k

Fig. 3: Performance comparison in different feature spaces on all datasets. Kmeans-based methods (a)(b)(c). GMM-based methods (d)(e)(f).

Effectiveness of DeepCluster Framework: As we know, the digits images of USPS are more illegible than MNIST. We reconstruct the cluster centers by DAEC, DC-Kmeans and DC-GMM, and the results are shown in Fig. 4. We can see that the centers learned by our methods are more reasonable than DAEC as there exist duplicated digits in the reconstructed centroids by DAEC.

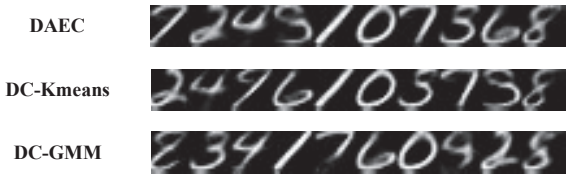


Fig. 4: The reconstruction of cluster centroids learned by different methods.

To demonstrate the effectiveness of our methods, we present the performance results of all methods on the USPS dataset in Fig. 5, and give the performance results on all datasets in Table 2. We can see that DC-GMM outperforms the other methods in all metrics on USPS and MNIST. Both DC-Kmeans and DC-GMM outperform DAEC, which indicates that clustering the dummy variable is better than clustering the deep features directly. Certainly, DEC outperforms the other methods in *NMI* and performs slightly better than DC-Kmeans in *ARI* on Reuters10K. But DC-Kmeans outperforms DEC in *ACC*. DC-GMM achieves comparable performance to DEC, however it underperforms DEC and DC-Kmeans in *ARI*.

As shown in Fig. 6, DeepCluster's convergence is fast and stable. It achieves state-of-the-art performance after 60 epcohs, and then its performance gets improved slowly till it converges.

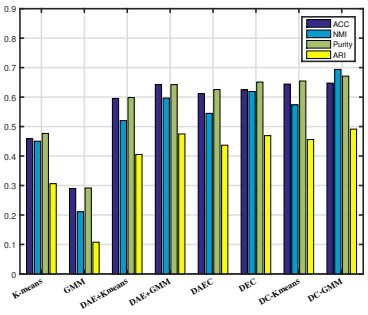


Fig. 5: Performance comparison on dataset USPS.

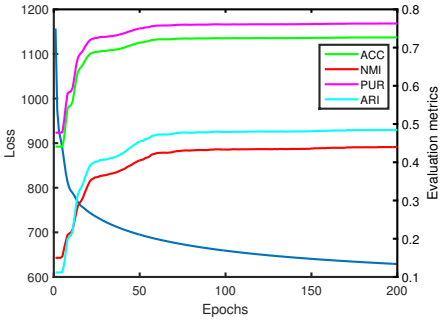


Fig. 6: Convergence analysis on Reuters10k dataset.

Effect of Hyperparameter Choice: Here we check our method's sensitivity to the hyperparameter ρ. We use different values of ρ to evaluate the model's sensitivity on all our experimental settings, the results are shown in Fig. 7. We can see that our method is robust to ρ. This is very important as it is not possible to do cross-validation in real-world applications [25].

Time Analysis: We present the time cost of our method in Table 3. DeepCluster algorithms spend approximately the same time as DAEC. DEC spend the least time because it works on GPU in Caffe, while DeepCluster and DAEC run on CPU in MATLAB code.

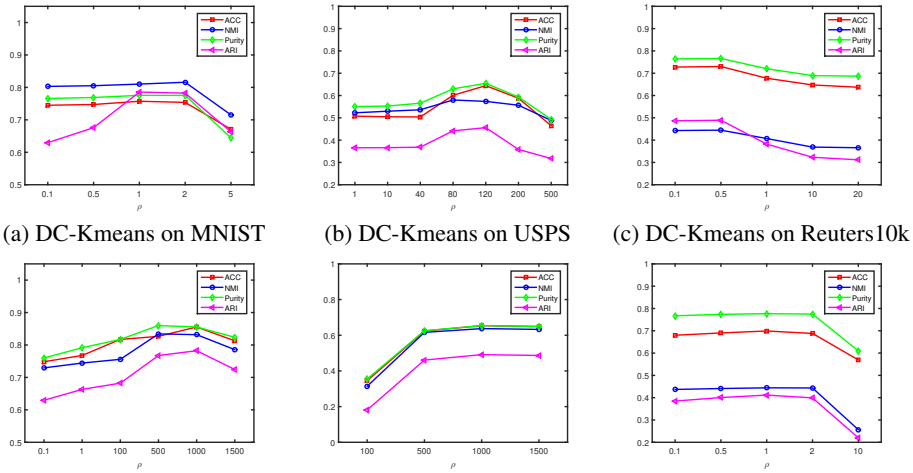
Table 3: Experimental Time Information.(In seconds)

Methods	**DAEC**	**DEC**	**DC-Kmeans**	**DC-GMM**
MNIST	18323	*1802*	20778	19234
USPS	2896	*1037*	2657	2715
Reuters10k	2025	*1994*	2527	2332

7 Conclusion

This paper presents a deep learning based clustering framework that simultaneously learns hidden features and does cluster assignment. Thanks to employing the ADMM algorithm, we can optimize our models in an end-to-end manner. We demonstrate the effectiveness of this framework by embedding K-means and GMM into DAE. Experimental results validate the effectiveness and advantage of our framework, which can achieve state-of-the-art performance on real-world datasets. Compared to DAEC, our framework runs clustering algorithms on the dummy variable while constraining the variable close to the learned features. By introducing relaxation and variable decomposition. We can optimize this framework by ADMM. Extensive network architectures and clustering methods will be exploited under this framework. For the proposed method's reproducible test, please go to this link* for the executable code and data.

* `https://github.com/JennyQQL/DeepClusterADMM-Release`



(a) DC-Kmeans on MNIST (b) DC-Kmeans on USPS (c) DC-Kmeans on Reuters10k

(d) DC-GMM on MNIST (e) DC-GMM on USPS (f) DC-GMM on Reuters10k

Fig. 7: Parameter sensitivity analysis of DeepCluster algorithms on different datasets.

Acknowledgments. This work was partially supported by the Key Projects of Fundamental Research Program of Shanghai Municipal Commission of Science and Technology (14JC1400300). Jihong Guan was supported by National Natural Science Foundation of China (NSFC) (61373036) and the Program of Shanghai Subject Chief Scientist (15XD1503600).

References

1. Aggarwal, C.C., Zhai, C.: A survey of text clustering algorithms. In: Mining text data, pp. 77–128. Springer (2012)
2. Beil, F., Ester, M., Xu, X.: Frequent term-based text clustering. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 436–442. ACM (2002)
3. Bengio, Y., Courville, A., Vincent, P.: Representation learning: A review and new perspectives. IEEE transactions on pattern analysis and machine intelligence 35(8), 1798–1828 (2013)
4. Bishop, C.M.: Pattern recognition. Machine Learning 128, 1–58 (2006)
5. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends® in Machine Learning 3(1), 1–122 (2011)
6. Dueck, D., Frey, B.J.: Non-metric affinity propagation for unsupervised image categorization. In: Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on. pp. 1–8. IEEE (2007)
7. Eckstein, J., Yao, W.: Understanding the convergence of the alternating direction method of multipliers: Theoretical and computational perspectives. Pac. J. Optim. To appear (2015)
8. Hallac, D., Leskovec, J., Boyd, S.: Network lasso: Clustering and optimization in large graphs. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 387–396. ACM (2015)
9. Hestenes, M.R.: Multiplier and gradient methods. Journal of Optimization Theory and Applications 4(5), 303–320 (1969)

10. Hong, M., Luo, Z.Q., Razaviyayn, M.: Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems. SIAM Journal on Optimization 26(1), 337–364 (2016)
11. Joulin, A., Bach, F., Ponce, J.: Discriminative clustering for image co-segmentation. In: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. pp. 1943–1950. IEEE (2010)
12. Kanungo, T., Mount, D.M., Netanyahu, N.S., Piatko, C.D., Silverman, R., Wu, A.Y.: An efficient k-means clustering algorithm: Analysis and implementation. IEEE transactions on pattern analysis and machine intelligence 24(7), 881–892 (2002)
13. Liu, B., Yuan, X.T., Yu, Y., Liu, Q., Metaxas, D.N.: Decentralized robust subspace clustering. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. pp. 3539–3545. AAAI Press (2016)
14. Liu, H., Shao, M., Li, S., Fu, Y.: Infinite ensemble for image clustering. In: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2016)
15. Ng, A.Y., Jordan, M.I., Weiss, Y., et al.: On spectral clustering: Analysis and an algorithm. Advances in neural information processing systems 2, 849–856 (2002)
16. Peng, X., Xiao, S., Feng, J., Yau, W.Y., Yi, Z.: Deep subspace clustering with sparsity prior. In: The 25th International Joint Conference on Artificial Intelligence (2016)
17. Santos, J., Embrechts, M.: On the use of the adjusted rand index as a metric for evaluating supervised classification. Artificial neural networks–ICANN 2009 pp. 175–184 (2009)
18. Song, C., Liu, F., Huang, Y., Wang, L., Tan, T.: Auto-encoder based data clustering. In: Iberoamerican Congress on Pattern Recognition. pp. 117–124. Springer (2013)
19. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–9 (2015)
20. Taylor, G., Burmeister, R., Xu, Z., Singh, B., Patel, A., Goldstein, T.: Training neural networks without gradients: A scalable admm approach. In: International Conference on Machine Learning (2016)
21. Tian, F., Gao, B., Cui, Q., Chen, E., Liu, T.Y.: Learning deep representations for graph clustering. In: AAAI. pp. 1293–1299 (2014)
22. Tian, K., Shao, M., Wang, Y., Guan, J., Zhou, S.: Boosting compound-protein interaction prediction by deep learning. Methods 110, 64–72 (2016)
23. Wang, R., Shan, S., Chen, X., Gao, W.: Manifold-manifold distance with application to face recognition based on image set. In: Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. pp. 1–8. IEEE (2008)
24. Wang, Y., Yin, W., Zeng, J.: Global convergence of admm in nonconvex nonsmooth optimization. arXiv preprint arXiv:1511.06324 (2015)
25. Xie, J., Girshick, R., Farhadi, A.: Unsupervised deep embedding for clustering analysis. In: International Conference on Machine Learning (ICML) (2016)
26. Xu, J., Wang, P., Tian, G., Xu, B., Zhao, J., Wang, F., Hao, H.: Short text clustering via convolutional neural networks. In: Proceedings of NAACL-HLT. pp. 62–69 (2015)
27. Zeiler, M.D.: Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701 (2012)
28. Zhang, R., Cheng, Z., Guan, J., Zhou, S.: Exploiting topic modeling to boost metagenomic reads binning. BMC Bioinformatics 16, S2 (2015)