Improved Deep Embedded Clustering with Local Structure Preservation

Xifeng Guo, Long Gao, Xinwang Liu, Jianping Yin

College of Computer, National University of Defense Technology, Changsha, China
guoxifeng1990@163.com, 1017730430@qq.com, 1022xinwang.liu@gmail.com, jpyin@nudt.edu.cn

Abstract

Deep clustering learns deep feature representations that favor clustering task using neural networks. Some pioneering work proposes to simultaneously learn embedded features and perform clustering by explicitly defining a clustering oriented loss. Though promising performance has been demonstrated in various applications, we observe that a vital ingredient has been overlooked by these work that the defined clustering loss may corrupt feature space, which leads to non-representative meaningless features and this in turn hurts clustering performance. To address this issue, in this paper, we propose the Improved Deep Embedded Clustering (IDEC) algorithm to take care of data structure preservation. Specifically, we manipulate feature space to scatter data points using a clustering loss as guidance. To constrain the manipulation and maintain the local structure of data generating distribution, an under-complete autoencoder is applied. By integrating the clustering loss and autoencoder’s reconstruction loss, IDEC can jointly optimize cluster labels assignment and learn features that are suitable for clustering with local structure preservation. The resultant optimization problem can be effectively solved by mini-batch stochastic gradient descent and backpropagation. Experiments on image and text datasets empirically validate the importance of local structure preservation and the effectiveness of our algorithm.

1 Introduction

Unsupervised clustering is a vital research topic in data science and machine learning. Traditional clustering algorithms like $k$-means [MacQueen, 1967], gaussian mixture model [Bishop, 2006] and spectral clustering [Von Luxburg, 2007] group data on handcrafted features according to intrinsic characteristics or similarity. However, when the dimension of input feature space (data space) is very high, the clustering becomes ineffective due to unreliable similarity metrics. Transforming data from high dimensional feature space to lower dimensional space in which to perform clustering is an intuitive solution and has been widely studied. This can be done by applying dimension reduction techniques like Principle Component Analysis (PCA), but the representation ability of these shallow models is limited. Thanks to the development of deep learning, such feature transformation can be achieved by using Deep Neural Networks (DNN). We refer to this kind of clustering as deep clustering.

Deep clustering is most recently proposed and leaves a lot of problems unsolved. For example, what types of neural networks are proper? How to provide guidance information i.e. to define clustering oriented loss function? Which properties of data should be preserved during transformation? The primitive work in deep clustering focuses on learning features that preserve some properties of data by adding priori knowledge to the subjective [Tian et al., 2014; Peng et al., 2016]. They are two-stage algorithms: feature transformation and then clustering. Later, algorithms that jointly accomplish feature transformation and clustering come into being [Yang et al., 2016; Xie et al., 2016]. The Deep Embedded Clustering (DEC) [Xie et al., 2016] algorithm defines an effective objective in a self-learning manner. The defined clustering loss is used to update parameters of transforming network and cluster centers simultaneously. The cluster assignment is implicitly integrated to soft labels. However, the local structure preservation can not be guaranteed by the clustering loss. Thus the feature transformation may be misguided, leading to corruption of embedded space.

To deal with this problem, in this paper, we assume that both clustering oriented loss guidance and local structure preservation mechanism are essential for deep clustering. Inspired by [Peng et al., 2016], we use under-complete autoencoder to learn embedded features and to preserve local structure of data generating distribution. We propose to incorporate autoencoder into DEC framework. In this way, the proposed framework can jointly perform clustering and learn representative features with local structure preservation. We refer to our algorithm as Improved Deep Embedded Clustering (IDEC). The optimization of IDEC can directly perform mini-batch stochastic gradient descent and backpropagation. At last, some experiments are carefully designed and conducted. The results validate our assumption and the effectiveness of our IDEC.

The contributions of this work are summarized as below:

- We propose a deep clustering algorithm that can jointly perform clustering and learn representative features with
local structure preservation.
• We empirically prove the importance of local structure preservation in deep clustering.
• The proposed IDEC outperforms the newest opponent in a large margin.

2 Related Work

2.1 Deep Clustering

Existing deep clustering algorithms broadly fall into two categories: (i) two-stage work that applies clustering after having learned a representation, and (ii) approaches that jointly optimize the feature learning and clustering.

The former category of algorithms directly take advantage of existing unsupervised deep learning frameworks and techniques. For example, [Tian et al., 2014] uses autoencoder to learn low dimensional features of original graph, and then runs k-means algorithm to get clustering results. [Chen, 2015] layer-wisely trains a Deep Belief Network (DBN) and then applies non-parametric maximum-margin clustering to learned intermediate representation. [Peng et al., 2016] uses autoencoder with sparsity prior to learn representations in nonlinear latent space that are adaptive to local and global subspace structure simultaneously, and then traditional clustering algorithms are employed to get label assignment.

The other category of algorithms try to explicitly define a clustering loss, simulating classification error in supervised deep learning. [Yang et al., 2016] proposes a recurrent framework in deep representations and image clusters, which integrates two processes into a single model with a unified weighted triplet loss and optimizes it end-to-end. DEC [Xie et al., 2016] learns a mapping from the observed space to a low-dimensional latent space with deep neural networks, which can obtain feature representations and cluster assignments simultaneously.

The proposed algorithm intrinsically is a modified version of DEC with incorporating an under-complete autoencoder to preserve local structure. It excels [Yang et al., 2016] by simplicity without recurrent and outperforms DEC in terms of clustering accuracy and feature’s representativeness. Since IDEC mainly depends on autoencoder and DEC, we will introduce them in more detail in the following sections.

2.2 Autoencoder

An autoencoder is a neural network that is trained to attempt to copy its input to its output. Internally, it has a hidden layer $z$ that describes a code used to represent the input. The network consists of two parts: an encoder function $z = f_W(x)$ and a decoder $x' = g_W'(z)$ that produces a reconstruction. There are two widely used types of autoencoders.

Under-complete autoencoder. It controls the dimension of latent code $z$ lower than input data $x$. Learning such under-complete representations force the autoencoder to capture the most salient features of the data.

Denoising autoencoder. Instead of reconstructing $x$ given $x$, denoising autoencoder minimizes the following objective:

$$L = \|x - g_W'(f_W(\tilde{x}))\|^2_2$$  \hspace{1cm} (1)

where $\tilde{x}$ is a copy of $x$ that is corrupted by some form of noise. Therefore, denoising autoencoder has to recover $x$ from this corruption rather than simply copying their input. In this way, denoising autoencoder can force encoder $f_W$ and decoder $g_W'$ to implicitly capture the structure of data generating distribution.

In our algorithm, the denoising autoencoder is used for pretraining and under-complete autoencoder is added to DEC framework after initialization.

2.3 Deep Embedded Clustering

Deep Embedded Clustering (DEC) [Xie et al., 2016] starts with pretraining an autoencoder and then removes the decoder. The remaining encoder is finetuned by optimizing the following objective:

$$L = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$  \hspace{1cm} (2)

where $q_{ij}$ is the similarity between embedded point $z_i$ and cluster center $\mu_j$ measured by Student’s $t$-distribution [Maaten and Hinton, 2008]:

$$q_{ij} = \frac{(1 + \|z_i - \mu_j\|^2)^{-1}}{\sum_j (1 + \|z_i - \mu_j\|^2)^{-1}}$$  \hspace{1cm} (3)

And $p_{ij}$ in (2) is the target distribution defined as

$$p_{ij} = \frac{q_{ij}^2 / \sum_j q_{ij}}{\sum_j (q_{ij}^2 / \sum_i q_{ij})}$$  \hspace{1cm} (4)

As we can see, the target distribution $P$ is defined by $Q$, so minimizing $L$ is a form of self-training [Nigam and Ghani, 2000].

Let $f_W$ be the encoder mapping, i.e. $z_i = f_W(x_i)$ where $x_i$ is input example from dataset $X$. After pretraining, all embedded points $\{z_i\}$ can be extracted using $f_W$. Then employ $k$-means on $\{z_i\}$ to get initial cluster centers $\{\mu_j\}$. Afterwards, $L$ can be computed according to (2), (3) and (4). And the predicted label of sample $x_i$ is $\text{arg max}_j q_{ij}$.

During backpropagation, $\frac{\partial L}{\partial z_i}$ and $\frac{\partial L}{\partial \mu_j}$ can be easily computed. Then $\frac{\partial L}{\partial z_i}$ is passed down to update $f_W$ and $\frac{\partial L}{\partial \mu_j}$ is used to update cluster center $\mu_j$:

$$\mu_j = \mu_j - \lambda \frac{\partial L}{\partial \mu_j}$$  \hspace{1cm} (5)

The biggest contribution of DEC is the clustering loss (or target distribution $P$, to be specific). It works by using high confidential samples as supervision and then making samples in each cluster distribute more densely. However, there is no guarantee of pulling samples near margins towards the correct cluster. We deal with this problem by explicitly preserving the local structure of data. Under this condition, the supervision information of high confidential samples can help the marginal samples walk to the correct cluster.

3 Improved Deep Embedded Clustering

Consider a dataset $X$ with $n$ samples and each sample $x_i \in \mathbb{R}^d$ where $d$ is the dimension. The number of clusters $K$ is
3.1 Clustering loss and Initialization

The clustering loss is proposed by [Xie et al., 2016]. It is defined as KL divergence between distributions \( P \) and \( Q \), where \( Q \) is the distribution of soft labels measured by Student’s t-distribution and \( P \) is the target distribution derived from \( Q \). That is to say, the clustering loss is defined as

\[
L_c = KL(P\|Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}
\]

where \( KL \) is Kullback-Leibler divergence that measures the non-symmetric difference between two probability distributions, \( P \) and \( Q \) are defined by (4) and (3). Details can be found in Section 2.3 and [Xie et al., 2016].

Follow suggestions in [Xie et al., 2016], we also pretrain a stacked denoising autoencoder before performing clustering. After pretraining, embedded points are valid feature representations for input samples. Then cluster centers \( \{\mu_j\}_{j=1}^K \) can be initialized by employing k-means on \( \{z_i = f_W(x_i)\}_{i=1}^n \)

3.2 Local structure preservation

The embedded points obtained in Section 3.1 are not necessarily suitable for clustering task. To this end, DEC [Xie et al., 2016] abandons the decoder and finetunes the encoder using clustering loss \( L_c \). However, we suppose that this kind of finetuning could distort the embedded space, weaken the representativeness of embedded features and thereby hurt clustering performance. Therefore, we propose to keep the decoder untouched and directly attach the clustering loss to embedded space.

To ensure the effectiveness of clustering loss, the stacked denoising autoencoder used in pretraining is not appropriate any more. Because the clustering should be performed on features of clean data, instead of noised data that used in denoising autoencoder. So we directly remove the noise. Then the stacked denoising autoencoder degenerates into an under-complete autoencoder (See Section 2.2). The reconstruction loss is measured by Mean Squared Error (MSE):

\[
L_r = \sum_{i=1}^n \|x_i - g_{W'}(z_i)\|_2^2
\]

where \( z_i = f_W(x_i) \) and \( f_W \) and \( g_W \) are encoder and decoder mappings respectively. As shown in [Peng et al., 2016] and [Goodfellow et al., 2016], autoencoders can preserve local structure of data generating distribution. Under this condition, manipulating embedded space slightly using clustering loss will not cause corruption. So the coefficient \( \gamma \) is better to be less than 1, which will be empirically demonstrated in Section 4.3.

3.3 Optimization

We optimize (6) using mini-batch stochastic gradient decent (SGD) and backpropagation. To be specific, there are three kinds of parameters to optimize or update: autoencoder’s weights, cluster centers and target distribution \( P \).

**Update autoencoder’s weights and cluster centers.** Fix target distribution \( P \), then the gradients of \( L_c \) with respect to embedded point \( z_i \) and cluster center \( \mu_j \) can be computed as:

\[
\frac{\partial L_c}{\partial z_i} = 2 \sum_{j=1}^K (1 + \|z_i - \mu_j\|^2)^{-1} (p_{ij} - q_{ij})(z_i - \mu_j)
\]

\[
\frac{\partial L_c}{\partial \mu_j} = 2 \sum_{i=1}^n (1 + \|z_i - \mu_j\|^2)^{-1} (q_{ij} - p_{ij})(z_i - \mu_j)
\]

Note that the above derivations are from [Xie et al., 2016]. Then given a mini batch with \( m \) samples and learning rate \( \lambda \), \( \mu_j \) is updated by

\[
\mu_j = \mu_j - \frac{\lambda}{m} \sum_{i=1}^m \frac{\partial L_c}{\partial \mu_j}
\]

The decoder’s weights are updated by

\[
W' = W' - \frac{\lambda}{m} \sum_{i=1}^m \frac{\partial L_r}{\partial W'}
\]
The encoder’s weights are updated by

\[ W = W - \frac{\lambda}{m} \sum_{i=1}^{m} \left( \frac{\partial L_r}{\partial W} + \gamma \frac{\partial L_c}{\partial W} \right) \]  

(13)

**Update target distribution.** The target distribution \( P \) serves as “groundtruth” soft label but also depends on predicted soft label. Therefore, to avoid instability, \( P \) should not be updated at each iteration (one update for autoencoder’s weights using a mini-batch of samples is called an iteration) using only a batch of data. In practice, we update target distribution using all embedded points every \( T \) iterations. See (3) and (4) for the update rules. When update target distribution, the label assigned to \( x_i \) is obtained by

\[ s_i = \arg \max_j q_{ij} \]  

(14)

where \( q_{ij} \) is computed by (3). We will stop training if label assignment change (in percentage) between two consecutive updates for target distribution is less than a threshold \( \delta \).

The whole algorithm is summarized in Algorithm 1.

---

**Algorithm 1: Improved Deep Embedded Clustering**

**Input:** Input data: \( X \); Number of clusters: \( K \); Target distribution update interval: \( T \); Stopping threshold: \( \delta \); Maximum iterations: \( MaxIter \).

**Output:** Autoencoder’s weights \( W \) and \( W' \); Cluster centers \( \mu \) and labels \( s \).

1. Initialize \( \mu, W' \) and \( W \) according to Section 3.1.
2. for \( iter \in \{0, 1, \ldots, MaxIter\} \) do
   3. if \( iter \% T == 0 \) then
      4. Compute all embedded points \( \{z_i = f_W(x_i)\}_{i=1}^{n} \)
      5. Update \( P \) using (3), (4) and \( \{z_i\}_{i=1}^{n} \).
      6. Save last label assignment: \( s_{old} = s \).
      7. Compute new label assignments \( s \) via (14).
      8. if \( \text{sum}(s_{old} \neq s)/n < \delta \) then
         9. Stop training.
   10. Choose a batch of samples \( S \in X \).
   11. Update \( \mu, W' \) and \( W \) via (11), (12) and (13) on \( S \).

---

It is not difficult to see that the time complexity of IDEC algorithm is \( O(nD^2 + ndK) \), where \( D, d \) and \( K \) are maximum number of neurons in hidden layers, dimension of embedding layer and number of clusters. Generally \( K \leq d \leq D \) holds, so the time complexity is \( O(nD^2) \).

## 4 Experiments

### 4.1 DataSets

The proposed IDEC method is evaluated on two image datasets and one text dataset:

- **MNIST**: The MNIST dataset [LeCun et al., 1998] consists of total 70000 handwritten digits of 28x28 pixel size. We reshaped each gray image to a 784 dimensional vector.

- **USPS**: The USPS dataset contains 9298 gray-scale handwritten digit images with size of 16x16 pixels. The features are floating point in \([0, 2]\).

- **Reuters-10K**: Reuters contains around 810000 English news stories labeled with a category tree [Lewis et al., 2004]. Following DEC [Xie et al., 2016], we used 4 root categories: corporate/industrial, government/social, markets and economics as labels and excluded all documents with multiple labels. Restricted by computational resources, we randomly sampled a subset of 10000 examples and computed tf-idf features on the 2000 most frequent words. The sampled dataset is referred to as **Reuters-10K**.

For all algorithms, we preprocessed datasets as same as DEC, i.e. normalizing each example \( x_i \in X \) to \( \frac{1}{2} \|x_i\|_2^2 \approx 1 \).

### 4.2 Experiment Setup

**Comparing methods.** We demonstrate the effectiveness of our IDEC algorithm mainly by comparing with DEC [Xie et al., 2016] which can be viewed as a special case of IDEC when the reconstruction term is set to zero. we use the pub-
Figure 3: Visualization of clustering results on subset of MNIST during training. Different colors mark different clusters. The first row is ours and second row corresponds to DEC. The proposed IDEC converges slower since it optimizes reconstruction loss as well. Both methods separate clusters well but the data structure in the first row is preserved better than DEC. Note points with red and blue color, they are totally mixed together in DEC while still somehow separable in our IDEC.

Table 2: Comparison of clustering performance in terms of accuracy (%) and NMI (% in bracket).

<table>
<thead>
<tr>
<th>Methods</th>
<th>MNIST</th>
<th>USPS</th>
<th>REUTERS-10K</th>
</tr>
</thead>
<tbody>
<tr>
<td>k-means</td>
<td>53.24</td>
<td>66.82</td>
<td>51.62</td>
</tr>
<tr>
<td>SEC</td>
<td>80.37</td>
<td>N/A</td>
<td>60.08</td>
</tr>
<tr>
<td>AE+k-means</td>
<td>81.82(74.73)</td>
<td>69.31(66.20)</td>
<td>70.52(39.79)</td>
</tr>
<tr>
<td>DEC</td>
<td>86.55(83.72)</td>
<td>74.08(75.29)</td>
<td>73.68(49.76)</td>
</tr>
<tr>
<td>IDEC</td>
<td>88.06(86.72)</td>
<td>76.05(78.46)</td>
<td>75.64(49.81)</td>
</tr>
</tbody>
</table>

Figure 2 illustrates the behavior of DEC and IDEC during training on MNIST. We observe the following phenomena. First, the final accuracies comply with results in Table 2. As it shows, deep clustering algorithms AE+k-means, DEC and IDEC outperform traditional clustering algorithms k-means and Spectral Embedded Clustering (SEC) [Nie et al., 2011] with a large margin, which indicates the fascinating potentials of deep learning in unsupervised clustering field. The performance gap between AE+k-means and DEC reflects the effect of clustering loss. And the outperformance of IDEC over DEC demonstrates that the autoencoder can help improve clustering performance.
2, i.e. IDEC outperforms DEC. Second, IDEC converges slower than DEC because of the fluctuation of reconstruction loss. Third, IDEC has larger clustering loss and higher clustering accuracy than DEC. This implies that the objective of DEC may mislead the clustering procedure by distorting the embedded feature space and breaking the intrinsic structure of data. Finally, the reconstruction losses at last few iterations approximately equal the loss at beginning. It implies that the performance improvement from DEC to IDEC is not likely due to the clustering ability of autoencoder. Actually, we did conduct an experiment that finetunes the autoencoder only using reconstruction loss $L_r$ (by setting coefficient $\gamma$ in (6) to 0) via various optimizers, and no improvement in terms of clustering accuracy was observed. So we assume that the autoencoder plays the role of preserving local structure of data, and under this condition clustering loss can manipulate embedded space to get better clustering accuracy.

We further prove our assumption about the role autoencoder acts by visualizing the embedded feature space during training. The t-SNE [Maaten and Hinton, 2008] visualization on a random subset of MNIST with 1000 samples is shown in Fig. 3. From left to right in the top row, the training process of IDEC, the “shape” of each cluster is almost maintained. On the contrary, the “shape” in the bottom is changed a lot with training proceeding. Furthermore, when you focus on clusters colored by red and blue (digits 4 and 9), in the first column they are still separable but become distinguishable in the last column. This is a loophole of DEC’s objective (clustering loss). Our IDEC doesn’t overcome this problem, but does go further than DEC. To validate this, see the figures in the last column: blue and red clusters of IDEC are still somehow separable while in DEC they are totally mixed up. This problem was not observed from Figure 5 in [Xie et al., 2016], but it indeed happens by using their released code. This is also pointed out by [Zheng et al., 2016]. It can be concluded that the autoencoder can preserve the intrinsic structure of data generating distribution and hence help clustering loss to manipulate the embedded feature space appropriately.

To see how the coefficient $\gamma$ of clustering loss in (6) affects the performance of IDEC algorithm, we conduct experiment on MNIST dataset by sampling $\gamma$ in range $[10^{-2}, 10^2]$. The optimizer is set as SGD with momentum $\beta = 0.9$, as same as DEC’s default setting, for fair comparison. The learning rate $\lambda$ is set as 0.1, 0.01, 0.001, 0.0001 successively. As shown in Figure 4, there are following observations.

For the best learning rate, IDEC ($\lambda = 0.1$) outperforms DEC ($\lambda = 0.01$) when $\gamma \in [0.05, 1.0]$. Because $\gamma$ with too small value eliminates the positive effect of clustering loss and large value tends to distort latent feature space. When $\gamma \rightarrow 0$, the clustering result approaches that of AE+k-means.

Learning rate $\lambda$ and clustering coefficient $\gamma$ are coupling. For larger $\gamma$, it requires smaller $\lambda$ to maintain performance. But the combination of small $\gamma$ and large $\lambda$ leads to higher performance. So we recommend $\gamma = 0.1$.

5 Conclusion

This paper proposes Improved Deep Embedded Clustering (IDEC) algorithm, which jointly performs clustering and learns embedded features that are suitable for clustering and preserve local structure of data generating distribution. IDEC manipulates feature space to scatter data by optimizing a KL divergence based clustering loss. And it maintains the local structure by incorporating an autoencoder. Empirical experiments demonstrate that structure preservation is vital to deep clustering algorithm and can favor clustering performance. Future work includes: adding more prior knowledge (e.g. sparsity) in IDEC framework, and incorporating convolutional layers for image datasets.

Acknowledgments

This work was financially supported by the National Natural Science Foundation of China (Project no. 60970034, 61170287, 61232016 and 61672528).
References


