Unsupervised Deep Embedding for Clustering Analysis

2020年1月15日 22:29

一、Abstract

本文提出了一种聚类算法——**DEC（Deep Embedded Clustering）**

DEC是一种使用**深度神经网络**同时进行**feature representation（特征表示）**和**cluster assignment（聚类划分）**的算法。

- **feature representation（特征表示）**：一个从**data space（数据原始空间）**到**low-dimensional feature space（低维特征空间）**的**mapping（映射函数）**
- **cluster assignment（聚类划分）**：在特征空间里**迭代地优化地进行聚类划分。**

二、Introduction

1. Why , What & How

当前热门的Clustering problem有：

- What defines a cluster?
- What is the right distance metric? （距离度量的选取）
- How to group instances into clusters?
- How to validate clusters?
- 。。。

However，很少有关于**特征空间（即Abstract提到的feature representation）的无监督学习。**

为此，该论文定义了从**data space（数据原始空间）*X*** 到 **low-dimensional feature space（低维特征空间）*Z*** 的 **非线性映射（即Abstract提到的mapping）**，在转换后的特征空间里进行**聚类划分。**

2. DEC的效果表现在

(1) good **accuracy**

(2) fast **running time**

(3) less sensitive to the choice of hyperparameters（**robustness**）

三、Related work

- k-means
- Spectral clustering（光谱聚类）
- t-SNE

四、Deep Embedded Clustering

1. 引言

(1) 为什么会想到用深度学习?

> 因为要将数据从一个空间映射到另一个空间，那么就需要一个mapping函数，这个函数的参数是需要学习的。。。这就很显然要用深度学习了。

(2) DEC的两个阶段:

① **parameter initialization** with **a deep autoencoder**;

② **parameter optimization** (i.e., clustering), where we iterate between computing **an auxiliary target distribution and minimizing the Kullback–Leibler (KL) divergence to it**.

首先，让我们来看看DEC的第二个阶段:

2. Clustering with KL Divergence

第二个阶段（即parameter optimization）有两个步骤:

① we compute a **soft assignment** between **the embedded points** and **the cluster centroids**;

② we update the deep mapping f_θ and refine the cluster centroids by **learning from current high confidence assignments using an auxiliary target distribution**. 一直到收敛到一定程度才停止**（没看太懂。。。）**

2.1 SOFT ASSIGNMENT

（其实soft assignment就是求每一个点（the embedded point） 属于每一个簇（the

cluster centroid）的概率大小)

本篇论文使用的是学生分布来作为soft assignment的：

Student’ s t-distribution as the kernel to measure **the similarity between embedded point z_i** and **centroid μ_j**:

$$q_{ij} = \frac{(1 + \|z_i - \mu_j\|^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{j'}(1 + \|z_i - \mu_{j'}\|^2/\alpha)^{-\frac{\alpha+1}{2}}}, \qquad (1)$$

where $z_i = f_\theta(x_i) \in Z$ corresponds to $x_i \in X$ after embedding, α are the degrees of freedom of the Student's t-distribution and q_{ij} can be interpreted as the probability of assigning sample i to cluster j (i.e., a soft assignment). Since we cannot cross-validate α on a validation set in the unsupervised setting, and learning it is superfluous (van der Maaten, 2009), we let $\alpha = 1$ for all experiments.

2.2 KL Divergence Minimization

（1）深度学习的Loss Function：

this model is trained by **matching the soft assignment to the target distribution**. To this end, we define **our objective（所谓目标，就是深度学习要优化的loss function）** as a KL divergence loss between the soft assignments q_i and the auxiliary distribution p_i as follows:

$$L = \mathrm{KL}(P\|Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}. \qquad (2)$$

（KL就是我们深度学习中常说的相对熵~~~，KL越小越好）

（2）如何选择那个所谓的auxiliary target distribution：

we would like our target distribution to have the **following properties**:

① strengthen prediction（即improve cluster purity）

② put more emphasis on data points assigned with high confidence（高置信度）；

③ normalize loss contribution of each centroid to prevent large clusters from distorting the hidden feature space.

（没看懂）

计算 p_i:

First raising qi to the second power and then normalizing by frequency per cluster，即下面的公式：

$$p_{ij} = \frac{q_{ij}^2/f_j}{\sum_{j'} q_{ij'}^2/f_{j'}}, \qquad (3)$$

where $f_j = \sum_i q_{ij}$ are soft cluster frequencies.

2.3 Optimization（优化）

策略概述：

使用**SGD with momentum**来优化cluster centers$\{\mu_j\}$ and DNN parameters θ。

梯度计算：

$$\frac{\partial L}{\partial z_i} = \frac{\alpha+1}{\alpha}\sum_j (1+\frac{\|z_i-\mu_j\|^2}{\alpha})^{-1} \quad (4)$$
$$\times(p_{ij}-q_{ij})(z_i-\mu_j),$$
$$\frac{\partial L}{\partial \mu_j} = -\frac{\alpha+1}{\alpha}\sum_i (1+\frac{\|z_i-\mu_j\|^2}{\alpha})^{-1} \quad (5)$$
$$\times(p_{ij}-q_{ij})(z_i-\mu_j).$$

The gradients $\partial L/\partial z_i$ are then passed down to the DNN and used in standard backpropagation to compute the DNN's parameter gradient $\partial L/\partial\theta$. For the purpose of discovering cluster assignments, we stop our procedure when less than $tol\%$ of points change cluster assignment between two consecutive iterations.

下面来回过头讨论DEC的第一步——parameter initialization

3. Parameter Initialization（参数初始化）

在上面中，我们讨论了如何在假定好的 θ 和 $\{\mu_j\}$ 的情况下进行训练，下面我们来讨论这些parameters和centroids是如何初始化的。（注意，对于传统的聚类方法来说，质心的初始化很重要）

初始化DEC的方法（即初始化θ）：

- stacked autoecoder **(SAE)**

  如下图可见：

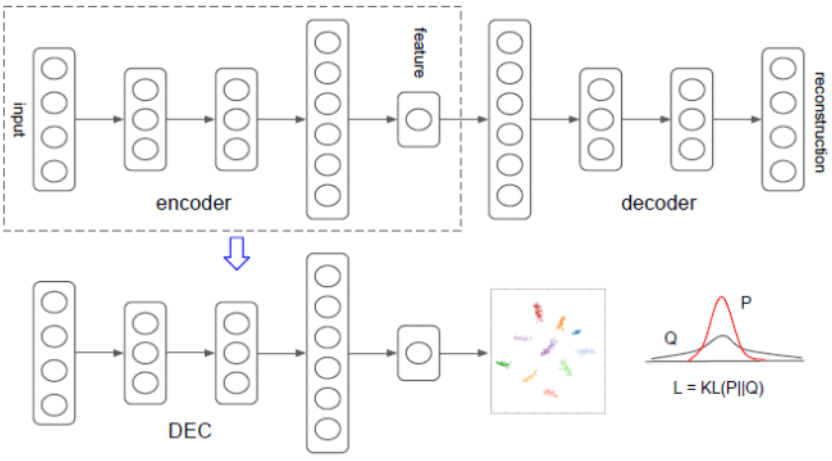

Figure 1. Network structure

初始化centroid的方法（即初始化μ）：

To initialize the cluster centers, we **pass the data through the initialized DNN to get embedded data points** and **then perform standard k-means clustering in the feature space Z to obtain k initial centroids** $\{\mu_j\}_{j=1}^k$。

五、Experiments

1. Datasets（实验数据集）

two image datasets and one text data set

- **MNIST**
- **STL-10**
- **REUTERS**

2. Evaluation Metric

对于所有算法，我们将簇的数量设置为**ground-truth**类别的数量，并在无监督的簇精度（ACC）下评估性能：

$$ACC = \max_m \frac{\sum_{i=1}^n \mathbf{1}\{l_i = m(c_i)\}}{n}, \qquad (10)$$

where l_i is the ground-truth label, c_i is the cluster assignment produced by the algorithm, and m ranges over all possible one-to-one mappings between clusters and labels.

3. Implementation（实现部分）（可不看，前面已经介绍了思想）

对于无监督学习而言，使用Validation Set在上的Cross Validation是不行的。该论文采用的是commonly used parameters for DNNs and avoid dataset specific tuning as much as possible。

Specifically,inspired by van der Maaten (2009), we set network dimensions to **d–500–500–2000–10** for all datasets, where d is the data-space dimension, which varies between datasets.All layers are **densely (fully) connected.（全连接网络）**

During greedy layer-wise pretraining we **initialize the weights** to random numbers drawn from **a zero-mean Gaussian distribution with a standard deviation of 0.01**.

Each layer is pretrained for 50000 iterations with a dropout rate of 20%.
The entire deep autoencoder is further finetuned for 100000 iterations without dropout.

For both **layer-wise pretraining** and **end-to-end finetuning** of the **autoencoder** the **minibatch size** is set to 256, starting **learning rate** is set to 0.1, which is divided by 10 every 20000 iterations, and **weight decay** is set to 0.

All of the above parameters are set to achieve a reasonably good **reconstruction loss** and are held constant across all datasets.

Dataset-specific settings of these parameters might improve performance on each dataset（所以，对于我们的论文可以使用特定的参数）, but we refrain from this type of unrealistic parameter tuning.

To initialize centroids, we run k-means with 20 restarts and select the best solution.

In the **KL divergence minimization phase**, we train with **a constant learning rate of 0.01**. **The convergence threshold is set to tol = 0.1%**. Our implementation is based on Python and Caffe (Jia et al., 2014) and is available at https://github.com/piiswrong/dec.

For all **baseline algorithms**, we perform **20 random restarts** when **initializing centroids** and pick the result with the best objective value.

For a fair comparison with previous work (Yang et al., 2010), we **vary one hyperparameter for each algorithm over 9 possible choices** and report the best accuracy in **Table 2** and the range of accuracies in **Fig. 2**.

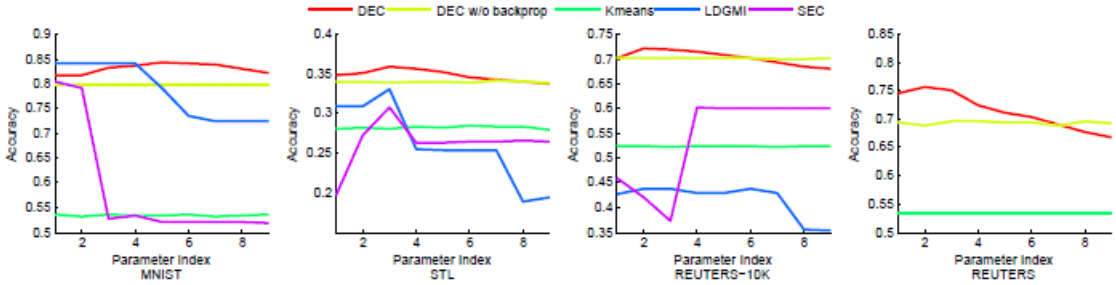

Figure 2. Clustering accuracy for different hyperparameter choices for each algorithm. DEC outperforms other methods and is more robust to hyperparameter changes compared to either LDGMI or SEC. Robustness is important because cross-validation is not possible in real-world applications of cluster analysis. This figure is best viewed in color.

When hyperparameters changes，DEC is much more robust。

Table 2. Comparison of clustering accuracy (Eq. 10) on four datasets.

Method	MNIST	STL-HOG	REUTERS-10k	REUTERS
k-means	53.49%	28.39%	52.42%	53.29%
LDMGI	84.09%	33.08%	43.84%	N/A
SEC	80.37%	30.75%	60.08%	N/A
DEC w/o backprop	79.82%	34.06%	70.05%	69.62%
DEC (ours)	**84.30%**	**35.90%**	**72.17%**	**75.63%**

For our proposed algorithm, we vary **λ**, the parameter that controls **annealing speed**, over 2i *10; i = 0,1,..., 8.

Since k-means does not have tunable hyperparameters (aside from k), we simply run them 9 times. GMMs（高斯混合模型） perform similarly to k-means so we only report k-means results. Traditional spectral clustering performs worse than LDGMI and SEC so we only report

the latter。

六、 Discussion（下面是关于DEC里面一些细节问题的讨论）

1. Assumption and Objective

The underlying assumption of DEC is that the **initial classifier’ s high confidence predictions are mostly correct**.

To verify that this assumption holds for our task and that **our choice of P has the desired properties**,

we plot the magnitude of the gradient of L with respect to each embedded point, $|\partial L/\partial z_i|$, against its soft assignment, q_{ij}, to a randomly chosen MNIST cluster j (Fig. 4).

We observe points that are closer to the cluster center (large q_{ij}) contribute more to the gradient.We also show the raw images of 10 data points at each 10 percentile sorted by q_{ij}。

Instances with higher similarity are more canonical examples of “5”. As confidence decreases, instances become more ambiguous and eventually turn into a mislabeled “8” **suggesting the soundness of our assumptions**.（即前面关于target distribution的一些假设）

2. Contribution of **Iterative Optimization**

In Fig. 5 we visualize the progression of the embedded representation of a random subset of MNIST during training. For visualization we use t-SNE (van der Maaten & Hinton,2008) applied to the embedded points zi. It is clear that the clusters are becoming increasingly well separated. Fig. 5 (f) shows how accuracy correspondingly improves over SGD epochs.

3. Contribution of **Autoencoder Initialization**

To better understand the contribution of each component,we show the performance of all algorithms with autoencoder features in Table 3. We observe that SEC and LDMGI’ s performance do not change significantly with autoencoder feature, while k-means improved but is still below DEC. This demonstrates the power of deep embedding and the benefit of fine-tuning with the proposed KL divergence objective.

4. Performance on **Imbalanced Data**

- In order to study the effect of imbalanced data, we sample subsets of MNIST with various retention rates. For minimum retention rate r_{min}, **data points of class 0 will be kept with probability r_{min}** and **class 9 with probability 1**, with the **other classes linearly in between**.
- As a result the largest cluster will be $1/r_{min}$ times as large as the smallest one. From Table 4 we can see that DEC is fairly robust against cluster size variation. We also observe that KL divergence minimization (DEC) consistently improves clustering accuracy after autoencoder and k-means initialization (shown as AE+k-means).

Table 4. Clustering accuracy (Eq. 10) on imbalanced subsample of MNIST.

Method \ r_{min}	0.1	0.3	0.5	0.7	0.9
k-means	47.14%	49.93%	53.65%	54.16%	54.39%
AE+k-means	66.82%	74.91%	77.93%	80.04%	81.31%
DEC	70.10%	80.92%	82.68%	84.69%	85.41%

5. Number of Clusters

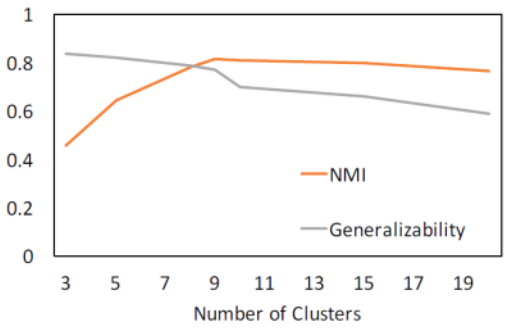

Figure 6. Selection of the centroid count, k. This is a plot of Normalized Mutual Information (NMI) and Generalizability vs. number of clusters. Note that there is a sharp drop of generalizability from 9 to 10 which means that 9 is the optimal number of clusters. Indeed, we observe that 9 gives the highest NMI.

在上面所有的讨论中，我们都是直接使用ground truth数量的簇数来比较不同的聚类算法，但是在实际应用中，我们是并不知道簇数的。因此，我们定义了两个指标：

1. **the standard metric**，**Normalized Mutual Information（NMI）** for evaluating clustering results with different cluster number:

$$NMI(l,c) = \frac{I(l,c)}{\frac{1}{2}[H(l) + H(c)]},$$

   where I is the mutual information metric and H is entropy。
2. **generalizability (G)（泛化能力）** which is defined as the ratio between training and validation loss:

$$G = \frac{L_{train}}{L_{validation}}.$$

G is small when training loss is lower than validation loss,which indicate a high degree of overfitting.