Deep Embedded Clustering

Narutatsu Ri

June 13, 2022

1 Introduction

Deep Embedded Clustering (DEC) [XGF16] is a methodology that concurrently learns cluster centroids utilizing Lloyd's Algorithm for the k-means problem and a non-linear re-representation of the data within a reduced dimension. This approach effectively combines the tasks of dimensionality reduction and clustering, which are typically performed sequentially.

This report arises from an independent endeavor to re-implement the model entirely from scratch. The objective is to provide an overview of DEC's operational principles, as well as to present and compare the obtained results against those reported in the original research publication [XGF16].[1]

2 Motivation for Deep Embedded Clustering

The motivation for integrating dimensionality reduction and clustering tasks stems from the fundamental requirements of clustering analysis. In any clustering algorithm, the notion of dissimilarity is essential, and the data representation employed must possess favorable characteristics. Ideally, data points belonging to the same cluster should exhibit proximity in terms of distance, while those from different clusters should display distinctiveness.

However, clustering analysis is inherently unsupervised, lacking explicit labels for evaluating the quality of a representation. While visual inspection can be employed for small datasets, it becomes impractical and challenging with a large number of data points. Consequently, a need arises for a methodology that can address this problem in an unsupervised setting.

Deep Embedded Clustering (DEC) presents a solution to this challenge, drawing inspiration from unsupervised metric learning principles. The central idea behind DEC lies in seamlessly integrating the unsupervised clustering task with the dimensionality reduction task, which resembles metric learning. Individually, these tasks are not conceptually complex, but the novelty lies in their synergistic connection within a cohesive model.

3 Formulation

In essence, DEC operates according to the following steps:

[1] Full code accessible at the following repository: `https://github.com/narutatsuri/deep_embedded_clustering`.

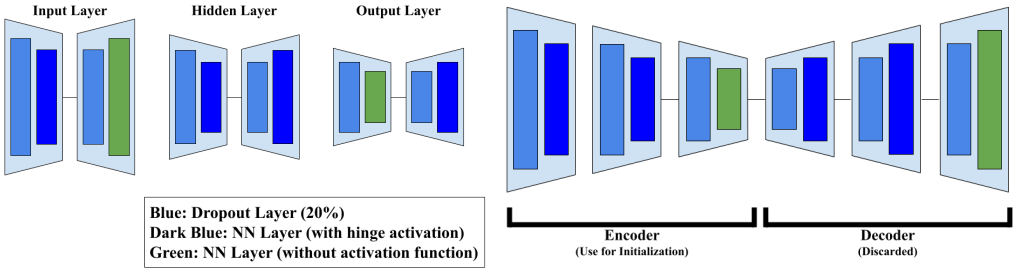

Figure 1: Diagram of the DEC model.

Algorithm 1: Deep Embedded Clustering Algorithm

Input : Original data
Output: Final re-representation and cluster centroids

1. Generate initial re-representation using a DNN;
2. Apply clustering algorithm (e.g., Lloyd's Algorithm) to the re-represented data, obtaining cluster centroids and an updated re-representation;
3. Evaluate quality of re-representations and cluster assignments using a unique approach;
4. Adjust re-representation and update the DNN and cluster centroids;
5. Repeat steps 1 to 4 until desired level of satisfaction is achieved;

Notably, DEC exhibits distinct characteristics primarily in two aspects: the generation of initial embeddings (a notable concern arises due to the non-random initialization of the embeddings, which will be addressed later), and the methodology employed to assess the quality of the embeddings and cluster assignments. The remaining steps either adhere to the aforementioned aspects or incorporate traditional methodologies.

Step 1. The authors employ a Stacked Autoencoder (SAE) model, consisting of multiple autoencoders arranged in a stacked manner, which is motivated by prior research [VLL+10]. While there is no formal mathematical proof supporting this approach, it is based on empirical evidence. Figure 1 provides a generalized diagram depicting the model's architecture.

The construction and training of individual autoencoders are performed to achieve data reconstruction. Two designated autoencoders are identified as the input and output layers, reflecting their position within the stacked configuration. The autoencoders' encoder and decoder components are separated, stacked, and combined to form the SAE structure. Subsequently, the stacked model is retrained using the dataset, discarding the decoder portion and retaining only the encoder. This encoder component serves as the Deep Neural Network (DNN) responsible for re-representing the original data in the embedded space.

Step 2. The data is passed through the DNN (i.e., the encoder of the SAE) to obtain initial embeddings. Subsequently, clustering is applied to the re-represented data using a standard clustering algorithm, such as k-means. The clustering process leads to the identification of cluster centroids, while the determination of the number of clusters, denoted as K, follows traditional approaches associated with the k-means problem.

Step 3. In this step, the authors draw inspiration from t-SNE, a method primarily utilized for data visualization. The fundamental idea of t-SNE is to learn a distance converted to a probability distribution around each point. The authors aim to define a soft cluster assignment, denoted as q_{ij}, representing the likelihood of data point i being assigned to cluster j based on the current representation. Additionally, a hard cluster assignment, denoted as p_{ij}, is introduced for each point. The equations governing these assignments are as follows:

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

$$p_{ij} = \frac{q_{ij}^2/f_j}{\sum_{j'} q_{ij'}^2/f_{j'}}$$

Here, z_i represents the re-represented data point in the embedding space, μ_j denotes the cluster centroid, and α serves as a hyperparameter controlling the shape of the distribution. The Euclidean distance between the data point and cluster centroid is quantified by $\|z_i - \mu_j\|^2$. The expression for q_{ij} corresponds to the Student's t-distribution, while p_{ij} represents a normalized version of q_{ij}.

The aim is to achieve concordance between the distributions q_{ij} and p_{ij}, effectively refining the initialization through iterative steps.

Step 4. The authors employ the Kullback-Leibler (KL) divergence to compare p_{ij} and q_{ij}. By computing the partial derivatives with respect to z_i and μ_j, the gradients are obtained, allowing for the update of the embeddings and cluster centroids. The DNN is then retrained using the updated embeddings, while the adjustments to the cluster centroids involve modifying their values without altering the overall structure.

An alternative perspective on DEC is to view it as a specialized form of multiclass classification. In traditional multiclass classification using a Deep Neural Network (DNN), the output typically consists of a multinomial probability distribution. The selected class for a given input is determined by choosing the element with the highest value in the distribution.

In the case of DEC, the desired output takes the form of a one-hot vector, where the index corresponding to the attributed class is assigned a value of 1, while the remaining indices are set to 0. The responsibility of learning this highly specific multinomial distribution lies with the DNN.

DEC alleviates this burden by allowing for more flexible outputs from the DNN. Instead of strictly mapping to a harsh one-hot vector, the outputs can gradually shift towards the correct answer, reducing the complexity imposed on the DNN.

4 Results

Figure 2 displays the obtained results. The left plot showcases the initial cluster embeddings, representing a subset of the MNIST dataset, and the right plot represents the final result.

Upon observing the left plot, a question arose regarding its quality, as it already appeared quite satisfactory. In fact, it exhibited a clustering accuracy ranging from approximately 70% to 80%. Although the authors acknowledge that SAE is known for yielding favorable results, the obtained accuracy seemed remarkably high, possibly encroaching on the role of DEC itself.

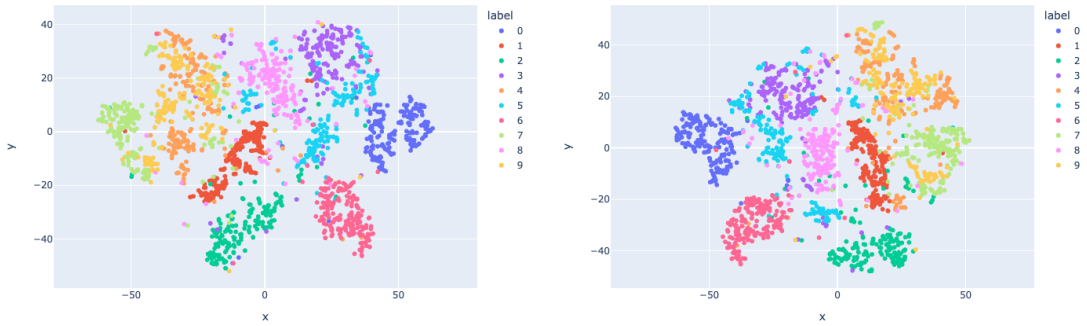

Figure 2: DEC applied on test data. Left is output of the stacked autoencoder, and right is the output of DEC.

The paper includes a figure, namely Figure 5(f), which illustrates the relationship between KL Divergence and cluster accuracy. Despite the unsupervised nature of the approach, the authors employed the MNIST dataset, assigning cluster labels through majority voting to evaluate accuracy. The figure reveals that even with k-means initialization, an accuracy of 80% was achieved, further increasing to approximately 85% as the KL Divergence loss "converged."

4.1 Evaluation of Random Initialization

Subsequently, an intuitive step to take was to explore the possibility of using random initialization and observe the outcomes. However, this approach proved to be wholly ineffective.

Figure 3 presents the outcomes of this attempt. The left plot illustrates the initial embeddings subjected to t-SNE, while the right plot showcases the final embeddings processed using t-SNE. Although the KL Divergence loss exhibited a decreasing trend, the clustering accuracy stagnated at around 20% with minimal improvement.

Furthermore, the authors assert the enhanced distinctiveness of clusters obtained through DEC compared to the initial embeddings. To support this claim, they employed a diagram featuring a subsample of the dataset on which DEC was trained. Specifically, DEC was trained on the complete MNIST dataset, and a subset of points was randomly selected from each cluster with equal probability.

It is important to note that inferring properties of the data solely from a t-SNE visualization is generally disapproved, as discussed earlier. However, the implicit justification in their paper rests on the assumption that if the clusters exhibit clear separation with discernible margins, t-SNE, which preserves local structure, would also reflect the same property in the original embeddings.

4.2 Alternate Implementations

Numerous alternative implementations of DEC have been developed by various individuals. Upon assessing their accuracy, it became apparent that these implementations were yielding results similar to those reported in the original paper. Consequently, doubts arose regarding the accuracy of my own code.

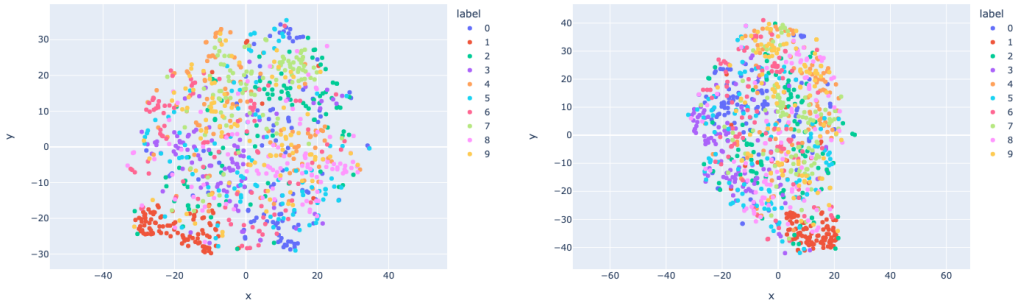

Figure 3: DEC applied on random initializations. Left shows the initial data (with no stacked autoencoder applied), and right is DEC's output.

Upon closer examination of these alternate implementations, it was discovered that they employed a somewhat deceptive approach. While the original paper stipulates updating z and μ through partial derivatives and similar procedures, these implementations opted to update z and μ explicitly to minimize the discrepancy between p and q. This approach deviates from the implicit convergence of p and q that occurs through the prescribed updating method outlined in the paper. While both methods should theoretically yield equivalent outcomes, the justification for this approximate approach remains uncertain. However, the authors of these implementations argue that minimizing KL Divergence reinforces the differentiation of clusters, and in this regard, their approximation technique can be deemed acceptable.

Based on this observation, it can be surmised that the discrepancies in my own results may stem from the differing update methods employed for z and μ, as these alternate implementations achieve favorable outcomes through a distinct methodology.

5 Conclusion & Future Directions

In conclusion, it is not uncommon for implementation-focused papers to appear somewhat superficial, particularly when examining marginal increases in accuracy. However, the true essence of papers such as DEC lies in the novel ideas they propose regarding the combination of dimensionality reduction and clustering, demonstrating the feasibility of such approaches. Furthermore, the authors' modeling assumptions, including the update mechanism utilizing p and q, provide avenues for further investigation. These assumptions can be evaluated through empirical validation, showcasing favorable results on specific datasets, or through rigorous mathematical justifications. While skepticism may initially pervade when approaching implementation papers, a broader and deeper understanding of their significance can be attained.

Further avenues of research can be explored to expand upon the concepts introduced in DEC. An additional paper, authored by an individual who also developed a customized implementation of DEC, delves into the preservation of local structure within clusters prior to embedding. While DEC aims to retain the implicit clusters present in the original data representation, thereby preserving global structure, it does not guarantee the preservation of local structure within each cluster. The aforementioned paper addresses this concern by proposing potential solutions.

References

[VLL+10] Pascal Vincent, H. Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, 2010.

[XGF16] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis, 2016.