Deep Co-clustering and its improvisations

Kuan-Lun Tseng
Oregon State University
Corvallis, OR 97331
tsengku@oregonstate.edu

Nuttaree Busarapongpanich
Oregon State University
Corvallis, OR 97331
busarapn@oregonstate.edu

Saurabhkumar Makwana
Oregon State University
Corvallis, OR 97331
makwanas@oregonstate.edu

Abstract

Co-clustering can lead to enhanced performance in clustering algorithms by partitioning features and instances simultaneously. It is done by exploiting the duality between them in the co-cluster matrix. Recent research has shown how effective deep learning could be to extracting effective features. It utilizes the deep autoencoders for dimensionality reduction and employs a variant of Gaussian Mixture model (GMM) to infer the cluster assignments. We decided to explore further into this topic by incorporating sparse and convolutional autoencoders, instead of the traditional deep autoencoders based on the type of dataset. Our results show that sparse autoencoders could yield better results when compared to the baseline model with deep autoencoder architecture whereas convolutional autoencoders might require further improvements.

1. Introduction

In recent years, many co-clustering algorithms have been proposed with the objective to cluster both instances and features simultaneously to organize them into homogeneous blocks. Some of them are based on information theory whereas others are based on the matrix decomposition theory. Co-clustering has been used in several application scenarios related to bioinformatics and graph mining where the datasets exhibit clear duality between instances and features. Further development in learning deep representations proved effective for feature extraction and hence encouraged incorporating deep learning models for better results. Moreover, end-to-end learning for learning the embedding representations for instance/feature and the cluster assignment jointly is required. In this paper, we decided to implement the DeepCC model for co-clustering with further improvement and exploration of the autoencoders used in it. Its architecture is shown in figure 1. It utilizes the deep autoencoder to generate low-dimensional representations for instances and features and employ a GMM variant for inferring cluster assignments. The inference network is a fully-connected neural network to generate the initial cluster assignment probability distribution for the GMM variant.

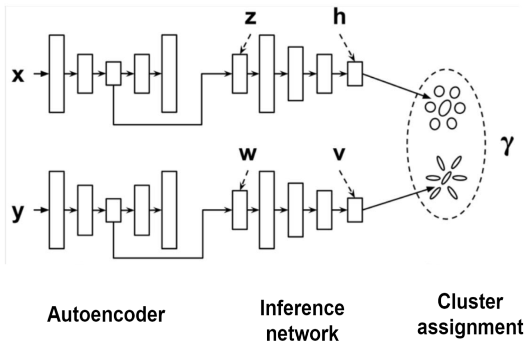

Figure 1: Overall architecture of DeepCC model

Figure 1. Shows the end-to-end DeepCC model, where x and y are the instances and features of the input data, z and w produced by deep autoencoder are low-dimensional representations, h and v are outputs of inference network and are utilized by a GMM variant to produce the co-cluster assignment, γ.

We further explored the autoencoder part and used other types of autoencoders like Sparse and Convolutional autoencoder based on the type of dataset. Sparse autoencoders are a type of neural network with sparseness constraints on hidden units. It prevents overfitting of the model as well by preventing the autoencoder from using all of its hidden nodes at once. Convolutional autoencoders have been known to work well with image data by removing noise from the picture and reconstructing missing parts. Our experimental results show the effectiveness of the above described approach on different datasets.

In the rest of the paper, Section 2 provides related background on our research. Section 3 describes the dataset used and section 4 talks about our detailed experimental approach. Section 5 talks about our results and their significance after analysis. Finally, section 6 concludes this paper with our reflection on the work done and further plans for improvement.

2. Related work and background

Although there has been previous work related to co-clustering, the research for leveraging deep representation learning for co-clustering is limited. Usually, feature learning and co-cluster assignment are two separate steps with inconsistent optimization goals. Dongkkuan et al. [1] proposed a novel DeepCC model for co-clustering with the aim to solve the above problem. It jointly minimizes the reconstruction error of the deep autoencoder and maximizes the variational lower bound of the log-likelihood in GMM. This indeed helps the deep autoencoder to escape from the local optima. Along with that, a mutual information loss is proposed to bridge the training of instances and features. Overall, the results proposed by this approach seem to be much better than the traditional approaches earlier.

Jun et al. [3] proposed feature transfer learning based on a sparse autoencoder method for discovering knowledge from small target data to improve overall performance when applying the knowledge to source data. The approach consists of two steps, with the first one being learning a representation using a single-layer autoencoder trained on class-specific instances from training data. The second step applies this representation to source data w.r.t the specific class for reconstructing those data and then use it for classification.

Mao et al. [6] proposed a deep fully convolutional autoencoder network for image restoration. It is based on the encoding-decoding framework with symmetric convolutional and deconvolutional networks. It learns end-to-end mappings from corrupted images to the original ones. The convolutional network captures the abstractions of the image contents while eliminating corruptions whereas the deconvolutional network is responsible for unsampling the feature maps and recovering the image details.

We build our approach on these prior work mentioned and extend our research on the DeepCC framework by exploring the use of sparse and convolutional autoencoders due to their respective advantages on benchmark datasets.

3. Dataset

We used a popular benchmark dataset for both image and text type input data. The Yale dataset was an image one which consisted of 165 grayscale images of 15 individuals with different facial expressions or configurations. Hence, there were a total of 11 images per subject. The WebKB4 text dataset had information about classified web pages about the computer science department of various universities. It was collected by the CMU text learning group and contained hypertext which consisted of pages and hyperlinks. More details about the dataset used could be found in Table 1.

Table 1: Description of the datasets

Dataset	Type	#instances	#features	#classes
Yale	Image	1,024	165	15
WebBK4	Text	1000	4199	4

4. Experiment Setup

4.1 Baseline Methods

We used a baseline DeepCC model provided by Dongkkuan et al. [1] and worked further on it. We finetuned the deep autoencoder part of the based on our scenario and the type of dataset by making adjustments to the activation function and the learning rate. All the layers are fully connected in this neural network and l2 loss is adopted for the reconstruction error of the deep autoencoder. Pre-training is adopted for the deep autoencoder to make the performance more stable. For the datasets, the number of instance clusters is set as the same as the number of feature clusters. DeepCC was implemented in Tensorflow without any dropout layer and trained by the Adam optimizer with a learning rate of 10^{-4}. The algorithm has been run multiple times and we noted down the best accuracy obtained by the model, which was 50.42% for the WebKB4 dataset and 44.1% for the Yale dataset.

Figure 2. Shows the difference in our approach from the DeepCC model architecture where the highlighted box indicates our change, where we are using sparse autoencoders instead of deep autoencoder for text dataset and convolutional one for image dataset

4.2 Evaluation Metric

We evaluate our accuracy and NMI in the same manner as Dongkkuan et al. [1] and it gives us a baseline approach to evaluate our modified model performance against theirs. A higher accuracy and NMI score indicates better clustering result.

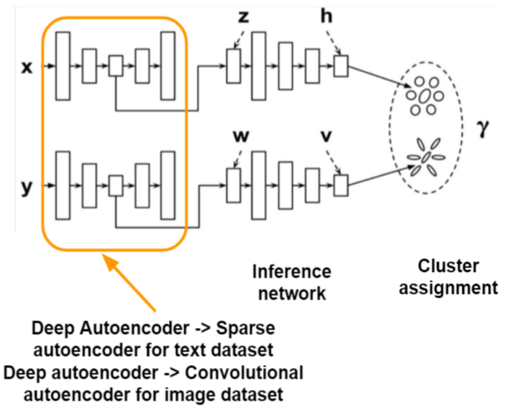
x
y
z
h
w
v
γ
Inference network
Cluster assignment
Deep Autoencoder -> Sparse autoencoder for text dataset
Deep autoencoder -> Convolutional autoencoder for image dataset

Figure 2: Modified approach of DeepCC

4.3 Convolutional Autoencoder

Then, we further modified the model by implementing convolutional autoencoders with an encoding-decoding framework. The encoder consists of a convolutional 2d layer, a maxpool and a dropout layer before it is connected to the fully connected layer. The decoder consists of a symmetrically inverse connection of the above mentioned layers for restoring the images. This is shown with the help of Figure 2 and 3. We again hypertuned our parameters for this approach and the learning rate used was 10^{-4} with a relu activation function. The dropout keep rate was 0.9 and the filter size used for this approach was 5x5.

Figure 3. Shows the model architecture of our implemented convolutional autoencoder

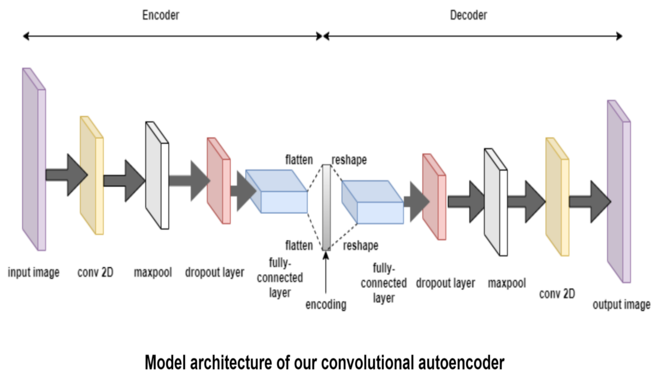

Figure 3: Convolutional autoencoder model architecture

4.4 Sparse Autoencoder

Finally, we implemented the sparse autoencoders for training text data as described in Figure 2. For this approach, we used two layers of encoders and decoders and applied KL Divergent to the cost function with the rho parameter of 0.01. Other hyperparameters were also tuned and the value of alpha and beta parameters for this approach were 0.0001 and 3. The learning rate was 10^-4 and the sigmoid activation function was used in our scenario.

5. Result

Our results obtained seem to indicate that sparse autoencoders perform slightly better than deep autoencoders for certain types of text datasets. This could be seen by the comparison of the both graphs for accuracy and NMI vs the number of iterations/epochs in Figure 4. Our baseline approach had slightly better accuracy of 53.67% with the WebKB4 text dataset whereas the deep autoencoder model had an accuracy of 50.42%. Similarly, our model had a higher NMI value of 0.22 compared to the deep autoencoder model which resulted in a value of 0.13.

Figure 4. Shows the accuracy and NMI comparison of optimized deep autoencoders vs sparse autoencoders. The top and bottom left ones are the accuracy obtained by the deep autoencoder with tuned learning rate of 10^{-4}, whereas the top and bottom right ones are sparse autoencoders with sigmoid activation function and the same learning rate

However, we did not obtain the desired results for convolutional autoencoder implementation. We obtained a best accuracy rate of 33.9 % and a NMI of 0.34 with the Yale image dataset. In comparison to that, the standard DeepCC model with deep autoencoder performed better and had a best accuracy of 44.1% and a NMI of 0.41.

This could be seen more clearly in Figure 5. which shows the comparison of both models. Also, table 2 summarizes these results for better concise understanding. Note that the above results obtained are for our baseline models without any further improvements.

Figure 5. Shows the accuracy and NMI comparison of optimized deep autoencoders vs our implemented convolutional autoencoders. The top and bottom left ones are the accuracy obtained by the deep autoencoder with tuned learning rate of 10^{-4}, whereas the top and bottom right ones are convolutional autoencoders with a learning rate of 10^{-4}.

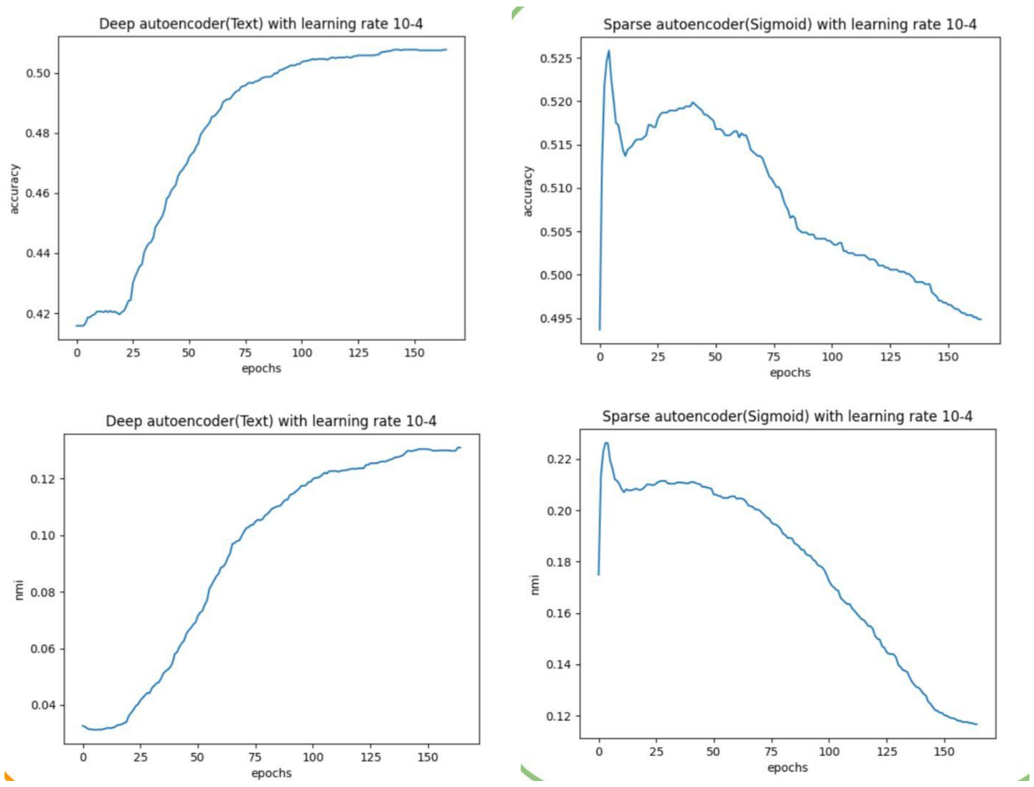

Figure 4: the accuracy and NMI of deep autoencoder and sparse autoencoder.

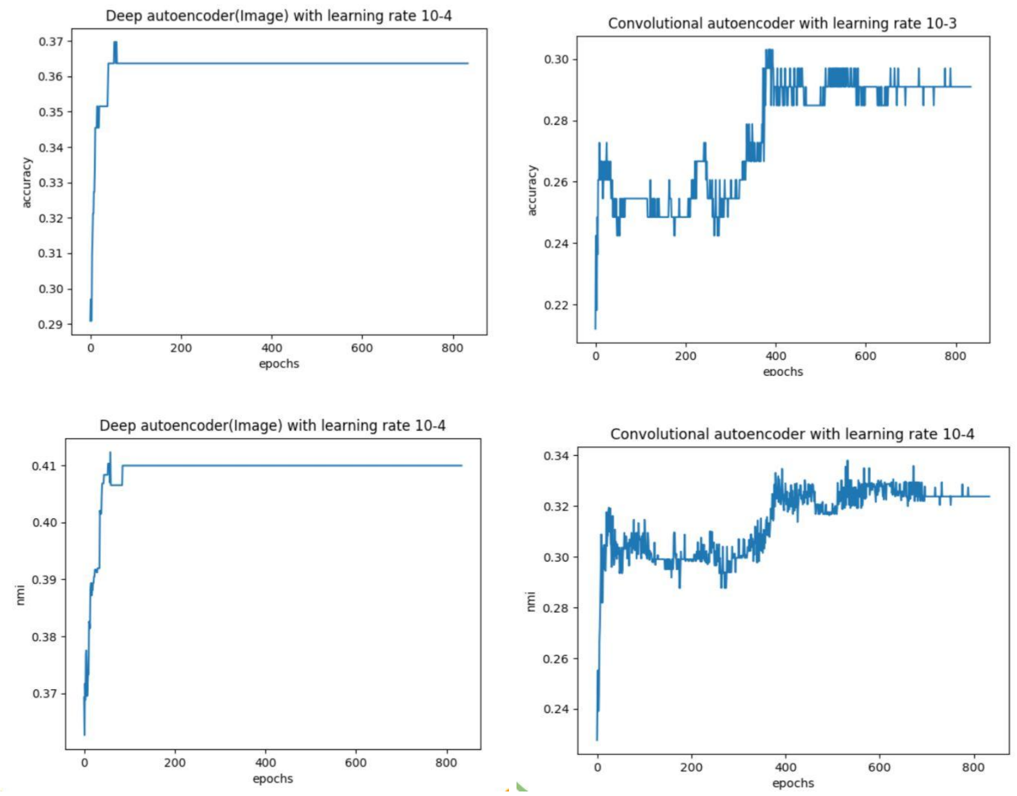

Figure 5: the accuracy and NMI of deep autoencoder and convolutional autoencoder.

Table 2: The accuracy and NMI results of different approaches

	Traditional Deep CC	Modified Deep CC with sparse autoencod er	Modified Deep CC with convolutiona l autoencoder
Accuracy	WebKB4 text dataset : 50.42%	53.67 %	33.9 %
	Yale image dataset: 44.1 %		
NMI	WebKB4 text dataset :013	0.22	0.34
	Yale image dataset: 0.41		

Table 2. Summarizes the accuracy and NMI experimental results for our implemented sparse and convolutional autoencoder vs the traditional deep autoencoders for the Yale image dataset and WebKB4 text dataset

6. Conclusion and Future Work

In this paper, we implemented the deep co-clustering model, DeepCC which utilizes the deep autoencoder to generate the low-dimensional representation for instances and features, which are further fed into the inference network in the GMM framework for cluster prediction. We then explored more into this deep autoencoder part, and implemented a couple of autoencoders. The objective of this work is to find out the best algorithm for the DeepCC model and whether different types of autoencoders can improve the overall performance of the model considering the type of input data and their respective advantages. Sparse autoencoders were chosen for the WebBK4 text dataset because it prevents overfitting by not allowing the autoencoders to use all of its hidden nodes at a time. For the Yale image dataset, Convolutional autoencoders were implemented which removes noise from the picture and better helps in reconstructing missing parts.

With our implemented approach and after finely hypertuning its parameters and comparing it to the baseline DeepCC model, we find out that our implemented Sparse autoencoder model does slightly better in terms of accuracy and NMI, where both of them indicate the clustering quality. Hence, our approach does look promising and further improvements to the model like adding more layers to the network or finely tuning the rho parameter of KL divergence could lead to significantly better results. Our implemented Convolutional autoencoder did not work as well as expected in comparison to the baseline DeepCC model. We still believe in our approach and feel that it might work better for a larger image dataset with realistically high dimensional images. We could also improve it by adding a pre-trained model with weights and changing the model architecture slightly. However, further research is required to dive deep into this topic. Overall, our approach does look promising and different other types of autoencoders could also be considered based on the input data and their advantages in future research.

References

[1] Xu, Dongkuan, Wei Cheng, Bo Zong, Jingchao Ni, Dongjin Song, Wenchao Yu, Yuncong Chen, Haifeng Chen, and Xiang Zhang. “Deep Co-Clustering.” Penn State. Society for Industrial and Applied Mathematics Publications, July 29, 2020.

[2] Ng, A. (n.d.). Sparse autoencoder - Stanford University. https://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf.

[3] J. Deng, Z. Zhang, E. Marchi and B. Schuller, "Sparse Autoencoder-Based Feature Transfer Learning for Speech Emotion Recognition," 2013 Humaine Association Conference on Affective Computing and Intelligent Interaction, 2013, pp. 511-516, doi: 10.1109/ACII.2013.90.

[4] Flores, Steven. “Variational Autoencoders Are Beautiful.” Variational Autoencoders are Beautiful | Blogs. Accessed June 12, 2021. https://www.compthree.com/blog/autoencoder/.

[5] Gu, Quanquan, and Jie Zhou. “Co-Clustering on Manifolds.” Co-clustering on manifolds | Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, June 1, 2009. https://dl.acm.org/doi/10.1145/1557019.1557063.

[6] Mao, Xiao-Jiao, Chunhua Shen, and Yu-Bin Yang. "Image Restoration Using Convolutional Auto-Encoders with Symmetric Skip Connections." NASA/ADS. Accessed June 12, 2021. https://ui.adsabs.harvard.edu/abs/2016arXiv160608921M/abstract.

[7] Prakash, Abhinav. "Different Types of Autoencoders."OpenGenus IQ: Computing Expertise &amp; Legacy, July 14, 2019. https://iq.opengenus.org/types-of-autoencoder/.