Alma mater Studiorum – University of Bologna

Dimensionality Reduction + Clustering Techniques

Report of the project on Deep Embedded Clustering

Artificial Intelligence in Industry

Professor: Michele Lombardi

Ganesh Pavan Kartikeya Bharadwaj Kolluri    -   [ganeshpavan.kolluri@studio.unibo.it](mailto:ganeshpavan.kolluri@studio.unibo.it)

Matricola: 0000976948

GitHub Repository: <https://github.com/karthikbharadhwajKB/Deep_Embedded_Clustering-on-HPC-dataset>

**Introduction**

Clustering is difficult to do in higher dimensional space, because the distance between two points will become less precise in the higher dimensional space and multiple dimensions are hard to think in, impossible to visualize and due to exponential growth of the number of possible values with each dimension. Complete enumeration of all subspaces becomes intractable with increasing dimensionality. This is problem is known as “Curse of Dimensionality”.

Solution for this problem is to reduce the dimensionality of input data. Here I am going to use Auto-encoders to reduce the dimensions without losing valuable information. By using autoencoders I am going to map x input space to z lower dimension embedded space and then perform clustering on the embedded space.

**Dataset**

HPC dataset was provided dataset to perform Deep clustering. Data was collected from “Marconi-100” a supercomputer hosted at Cineca. Data was collected by using “examon”. The dataset is composed of several files. each file is a node. As I tried to load all files and concatenate them using pandas read\_parquet() method because files are in parquet format. But the problem is dataset is huge and it crashed the ram of the google colab environment. As a solution for this problem. I tried with pyspark to load files and then convert them into pandas dataframe. But unfortunately, this solution is not working. Then I contacted Andrea regarding this and now I just loading one file for lack of computational resources. The information collected from Marconi-100 is varied, ranging from the load of the different cores, to the temperature of room where the nodes are located, the speed of fans and memory accessing in writing or reading and etc.

**Data Preparation**

After loading the data, I have dropped the ‘label’ and ‘timestamp’ columns from the dataset. I have checked for any missing data to treat the missing values. There are no missing values in the dataset. We have to split the dataset into X and y for further model training and evaluation of the model. Here we can observe that there are two values in y. 0 means normal state of the node, 2 means anomalous state. So, we are going to map all 2 values with 1 value for our convenience in evaluation part. Here we can see that class 0 has 14,932 entries and class 1 has only 379. There is lot of imbalance ness in the dataset. But these class 1 values are Anomalous state. We can expect that there are very rare events. K-means is sensitive to the scale of feature values because it uses Euclidean distance as similarity metrics. So, we have to scale these features with using Minmax scaler.

**Methodology**

The methodology is to firstly reduce the higher dimensional data to lower dimensional data by apply dimensionality reduction to input data using autoencoders and then apply traditional clustering algorithm on the embedded space. Because traditional algorithms like K-means, Gaussian Mixture Model and Agglomerative clustering models are sensitive to higher dimensional data.

**Models**

1. **Baseline Model-1 K-Means**

I have applied K-means clustering on original input data and evaluated with clustering accuracy and silhouette score to compare their results. We can see that their results are some what better when compared to Deep embedded clustering model.

1. **Baseline Model-2 Gaussian Mixture Model**

I have applied Gaussian Mixture Model soft clustering on original input data and evaluated with clustering accuracy and silhouette score to compare their result. We can observe that their results are not that great when compared Deep embedded clustering model.

1. **Baseline Model-3 Agglomerative Clustering**

I have applied Agglomerative Clustering on original input data and evaluated with clustering accuracy and silhouette score to compare their result. We can observe that their results are not that great when compared Deep embedded clustering model.

**Auto-encoder Model**

Auto-encoder models are special kind of neural network where it consists of encoder part and decoder part. Encoder part encodes inputs into lower feature representation and decoder then decodes it from lower feature representation to original input space. Generally, we will train this model using reconstruction loss as objective. In this case I have used specific dimension for encoder and decoder parts d-500-500-2000-20. Here d is the input dimensions, and all layers are dense connected. In our case, input shape is 460. I have trained auto-encoder model for 100 epochs with SGD optimizer with 0.1 learning rate and 0.9 momentum. Our objective loss function is mean squared error and batch size is 128. After pre-training I have saved the weights of auto-encoders for further utilization.

Diagram

Description automatically generated with medium confidence

**Figure: Autoencoder model**

1. **K-Means on Embedded space**

After Training Auto-encoder model, encoder model encodes X input space into z embedded space. Then we will apply K-means algorithm and obtained some better results when compared to K-means on original input.

1. **GMM on Embedded space**

After Training Auto-encoder model, encoder model encodes X input space into z embedded space. Then we will apply Gaussian Mixture model algorithm and obtained far better results when compared to GMM on original input.

1. **Agglomerative Clustering on Embedded space**

After Training Auto-encoder model, encoder model encodes X input space into z embedded space. Then we will apply Agglomerative Clustering algorithm and this algorithm outperforms every model with 76% accuracy.

**Evaluation of model**

|  |  |
| --- | --- |
| **Models** | **Cluster Accuracy** |
| Baseline Model-1: K-Means on original input space | 0.72 % |
| Baseline Model-2: GMM on original input space | 0.52 % |
| Baseline Model-3: Agglomerative Clustering on original input space | 0.68 % |
| K-Means on embedded space | 0.73 % |
| GMM on embedded space | 0.74 % |
| **Agglomerative Clustering on embedded space** | **0.76 %** |

**Conclusion**

As want to conclude that Models on embedded space got far better results than models on original input space and Agglomerative Clustering algorithm on embedded space outperforms remaining all models.

**References:**

[**https://colab.research.google.com/drive/19ka\_DBqJvcCJIu1QNZozFR\_4wQapCUQS#scrollTo=qdJkBoNCYCZw**](https://colab.research.google.com/drive/19ka_DBqJvcCJIu1QNZozFR_4wQapCUQS#scrollTo=qdJkBoNCYCZw)

[**https://ieeexplore.ieee.org/document/5453745**](https://ieeexplore.ieee.org/document/5453745)