E-Commerce Product Clustering

Vanshita Garg Etiksha Jain Deepanshu Rohilla

November 14, 2023

Abstract

Recommendation systems have become the backbone of the e-commerce platforms like Myntra, Amazon etc. In this work, we work with a dataset of products and users from an e-commerce site and try different clustering techniques to see which works the best. The work focuses on the use of Graph Neural Networks to create hidden representations for products and see how these embeddings can be used in addition/in replacement of the existing raw features.

1 Introduction

Clustering e-commerce products can help us gain useful insights about product demand, which users prefer which type of products and so on. Sometimes only object features may not be sufficient to describe a product or its relations with other entities. We therefore explore the idea of using graph neural networks to create embeddings/ meaningful representaions for every product which can incorporate in its itself the product's interaction with different users , other products etc. Such embeddings can be obtained by modeling the problem as either a link prediction task or a node classification task. Further on, we use clustering methods like K-means , DBScan and compare the clusters obtained by using raw features and embeddings obtained from various graph learning algorithms.

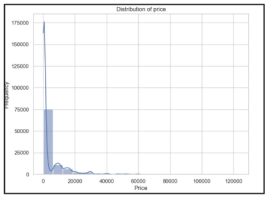
2 Data Analysis

Two csv files have been provided : one containes product ids along with the price and vertical of every product. The second file containes the user id, the product id the user has clicked on and the count of the number of times the product has been clicked upon. We do an inner join on the two tables to make one bigger table which can be used for further computation.Initially the data is highly positively skewed . We therefore perform data cleaning to remove outliers and also perform frequency encoding for the categorical variable- product vertical. The following figures display the data statistics:

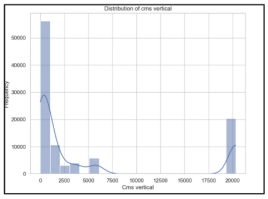
Unique Product Ids	**~ 16 million**
Unique Account Ids	**~ 2 million**
No. of Verticals	**~ 3K**

Mean Count of Users	**1.5 times**
Minimum count by user	**1**
Maximum count by user	**~ 2K**

Before Processing

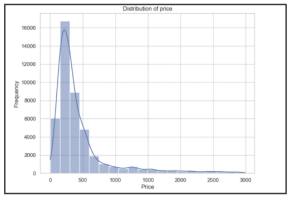


The data is highly positively skewed. As observed, very few nodes with prices > 2000. Can be considered as outliers.

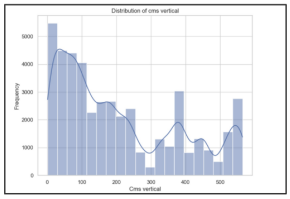


As observed , the distribution of products of different verticals is **asymmetric**.

After Processing



After outlier removal, data is now less skewed compared to before.



Since the frequency of occurrence of verticals for most products laid in the range 0-5000, we consider this to downsample our data

3 Problem Modeling

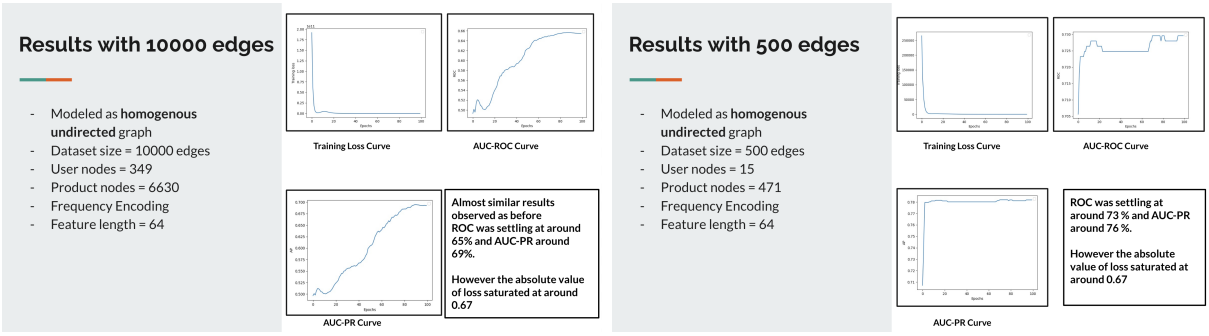
The problem is modeled as a graph which has two types of nodes, one representing users and others representing products. We create a bipartite graph such that an edge exists between a user node and a product node if the user has clicked on the product atleast once. Every node has its own features as well. The graph is created using the Deep Graph library (DGL) and visualised using networkx.
After the graph is created, we train the graph in two ways, one is the autoencoder approach and the other is Link Prediction approach. The embeddings obtained from both approaches are then compared with raw features by means of clustering algorithms.

4 Methods

4.1 AutoEncoder

The traditional autoencoder is a neural network that contains an encoder and a decoder. The encoder takes a data point X as input and converts it to a lower-dimensional representation (embedding) Z. The decoder takes the lower-dimensional representation Z and returns a reconstruction of the original input X-hat that looks like the input X. The quality of the embedding determines the quality of the output X-hat.
We implement the code of the reference paper [ ] wherein the autoencoder architecture contains : 2 Graph Convolution layer and then an InnerProduct Decoder. This model

treated the graph as **homogeneous**, hence the results obtained are just satisfactory with an AUC-ROC of 73% and AUC-PR pf 76%. The dataset used is also very less with only 10000 edges, due to computational limitations.



Raw features used as initial node embeddings

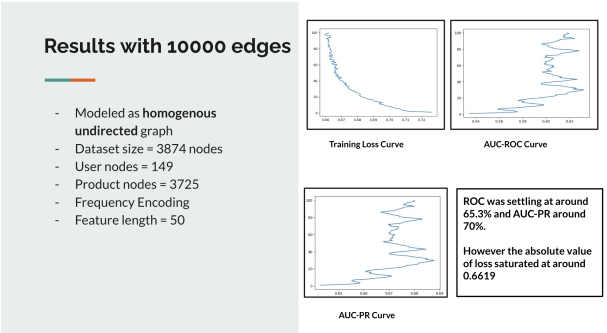
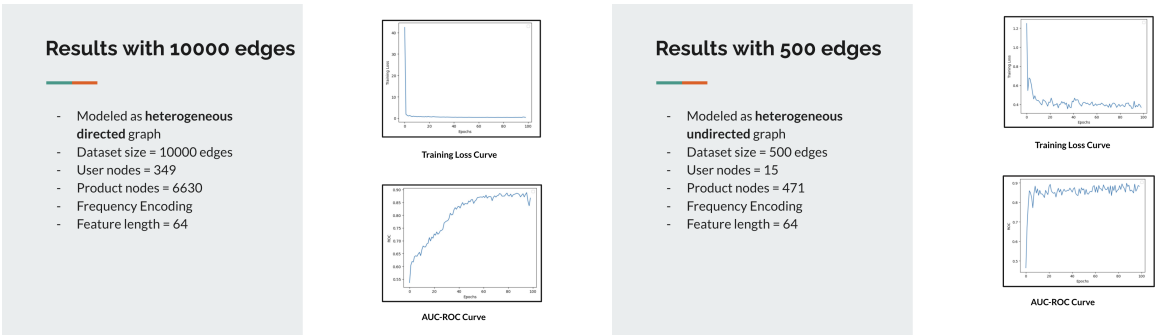


Figure 1
Collaborative filtering features used as initial node embeddings

4.2 Link Prediction using Graph Attention Network

In our next approach, we model the problem as a link prediction task, where the loss function takes the original adjacency matrix for computation. We use graph attention networks for this purpose. The graph is modeled as heterogeneous in this case and hence we get better results as compared to our previous approach, with an AUC-ROC of around 85%.The results obtained are as follows:



Raw Features used as initial node embeddings

4.3 Result Analysis

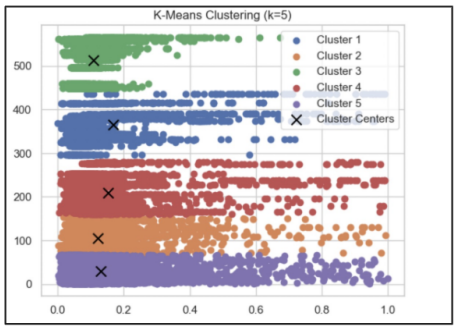
The above results show that modeling the graph as a heterogenous graph is more intutive and beneficial since we have two different types of nodes- users and products. Link Prediction is thus giving better results. The above results were computed on very little dataset and the same process was repeated after the data was cleaned. This improved our results but only marginally, taking ROC to around 78.9% in case of the autoencoder approach. At the same time the results also suggest that taking the features obtained from collaborative filtering approach as initial node embedding did not prove to be very useful as the Alternating Least Squares approach used to create them only takes ino account the interaction between a user and product based on the count of clicks, and does not take into account any other explicitly given product / user features.

5 Clustering

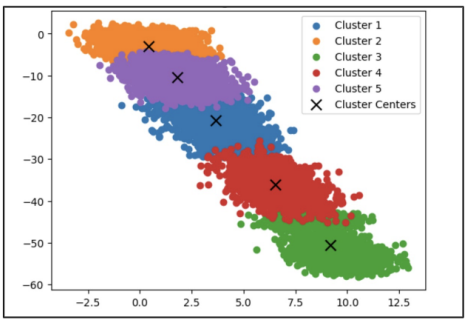
Clustering is employed to find out the hidden features of the data distribution and how we can group the data based on several aspects. In our case we perform 2 clusetering techniques and compare both of them in three scenarios - one, when we use the raw features of products for clustering, second when we use the mebeddings obtained from Autoencoder network as our product features and third when we use embeddings obtained from GAT as features of our products for clustering.

5.1 K-means Clustering

K-means is a centroid-based or partition-based clustering algorithm. This algorithm partitions all the points in the sample space into K groups of similarity. The similarity is usually measured using Euclidean Distance.



Using raw node features
X-axis: Price ; Y-axis : Vertical



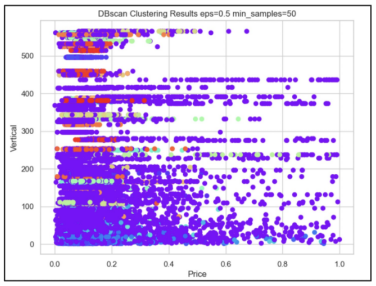
Using embeddings obtained from Graph Attention Network

5.2 DBScan Clustering

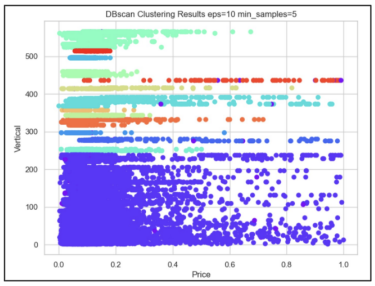
DBScan is a density-based clustering algorithm. The key fact of this algorithm is that the neighbourhood of each point in a cluster which is within a given radius (R) must have a minimum number of points (M).

The results of DBScan clustering depending largely on the initialization of parameters- epsilon, min samples. We therefore experiment with different set of parameters.

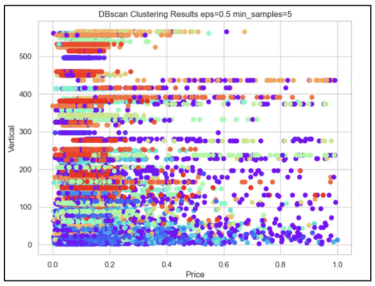
5.3 DBScan Clustering with Raw Features



Poor Clustering

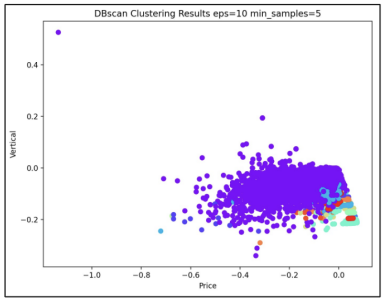


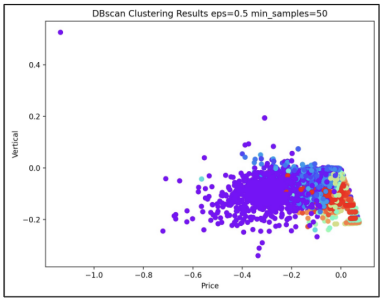
Epsilon changed; better results

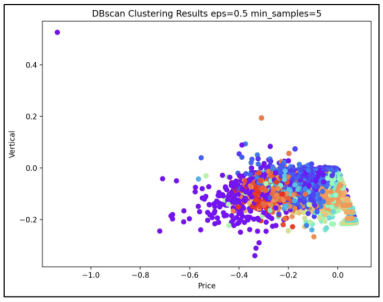


Epsilon reduced; discrete clusters

5.4 DBScan Clustering of Embeddings







5.5 Clustering Results Analysis

- Both raw features as well as embeddings give rise to about 5 clusters of products
- In case of raw features, irrespective of price, products are clustered on the basis of their vertical
- In case of embeddings, DBScan not able to do clear clustering due to high dimensionality of embeddings
- It is not clear whether clusters are based on price or vertical It might be the case that user features are affecting these clusters, but that is unclear here

6 Conclusion

Based on the analysis done so far, we reach the following conclusions:

- Inherently the products are clustered on the basis of the vertical to which they belong
- K-means clustering proves to be a good technique here , giving consistent results with raw features as well as embeddings
- The product embeddings obtained from link prediction modeling have similar cluster structure and centroid coordinates suggesting that the vertical of a product has a strong influence in its final embedding and is not much dependent on the user features we had considered
- DBScan doesn't work well for our data due to 'curse of high dimensionality ' along with problematic parameter tuning
- This might also mean that there might be a lack of good features in the data, pointing to the need for data enrichment

6.1 Future work

- The work can be extended to fit a larger dataset by utilizing HPC resources
- More data can be scraped/acquired along with better features to improve training accuracy
- Techniques like t-SNE visualization etc can be used further for better analysis

References

[1] Alternating Least Square for Implicit Dataset with code — by Himanshu Kriplani — Towards Data Science
[2] ALS Implicit Collaborative Filtering — by Victor — Rn Engineering — Medium
[3] [1611.07308] Variational Graph Auto-Encoders (arxiv.org)
[4] GitHub - tkipf/gae: Implementation of Graph Auto-Encoders in TensorFlow
[5] Medium - PyG: Link Prediction on Heterogeneous Graphs
[6] Automatic feature extraction with t-SNE — by Gonçalo Rodrigues — Jungle Book — Medium
[7] How to determine epsilon and MinPts parameters of DBSCAN clustering - Amir Masoud Sefidian - Sefidian Academy
[8] K-means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks — by Imad Dabbura — Towards Data Science