COL761

MININGMAVERICKS

Assignment 3

Chinmay Mittal *(2020CS10336)*
Parth Shah *(2020CS10380)*
Shubh Goel *(2020EE10672)*

Date of submission of Report: September 25, 2023

Q1 Curse of Dimensionality

The results for the experiments are as follows:

Dim	Avg Dist of Closest Point	Avg Dist of Farthest Point	Avg of Ratio of Distance to Farthest Point and Distance to Closest Point
1	0.000	0.75873	6793357.31644
2	0.00058	1.47260	4000.75831
4	0.03207	2.93430	98.40759
8	0.33697	5.47099	16.56263
16	1.53953	9.46377	6.20268
32	4.98861	16.78722	3.37477
64	13.05646	30.11494	2.30820

Table 1: Results for L_1 Distance

Dim	Avg Dist of Closest Point	Avg Dist of Farthest Point	Avg of Ratio of Distance to Farthest Point and Distance to Closest Point
1	0.00000	0.75873	6793357.31644
2	0.00047	1.05261	3514.04796
4	0.01927	1.48837	83.12299
8	0.15137	1.99176	13.45969
16	0.50091	2.52737	5.09026
32	1.14842	3.27528	2.85846
64	2.11627	4.27480	2.02165

Table 2: Results for L_2 Distance

Dim	Avg Dist of Closest Point	Avg Dist of Farthest Point	Avg of Ratio of Distance to Farthest Point and Distance to Closest Point
1	0.00000	0.75873	6793357.31644
2	0.00041	0.82831	3098.69003
4	0.01470	0.89492	65.93399
8	0.09097	0.95124	10.74377
16	0.23884	0.96795	4.09504
32	0.40933	0.98562	2.41976
64	0.56611	0.99117	1.75485

Table 3: Results for L_∞ Distance

This experiment illustrates the effect of the *"Curse of Dimensionality"*. This phenomenon is related to the distances of uniformly sampled point arises in high-dimensional spaces and affects data-mining algorithms like that of clustering which rely on distance metrics.

In low-dimensional spaces, like two or three dimensions, our intuitive understanding of distance and volume works well. However, as we move to higher dimensions, this intuition falters.

In high-dimensional spaces, the volume of the space expands so rapidly that the data points become sparse. This sparsity means that the distance between any two randomly chosen points

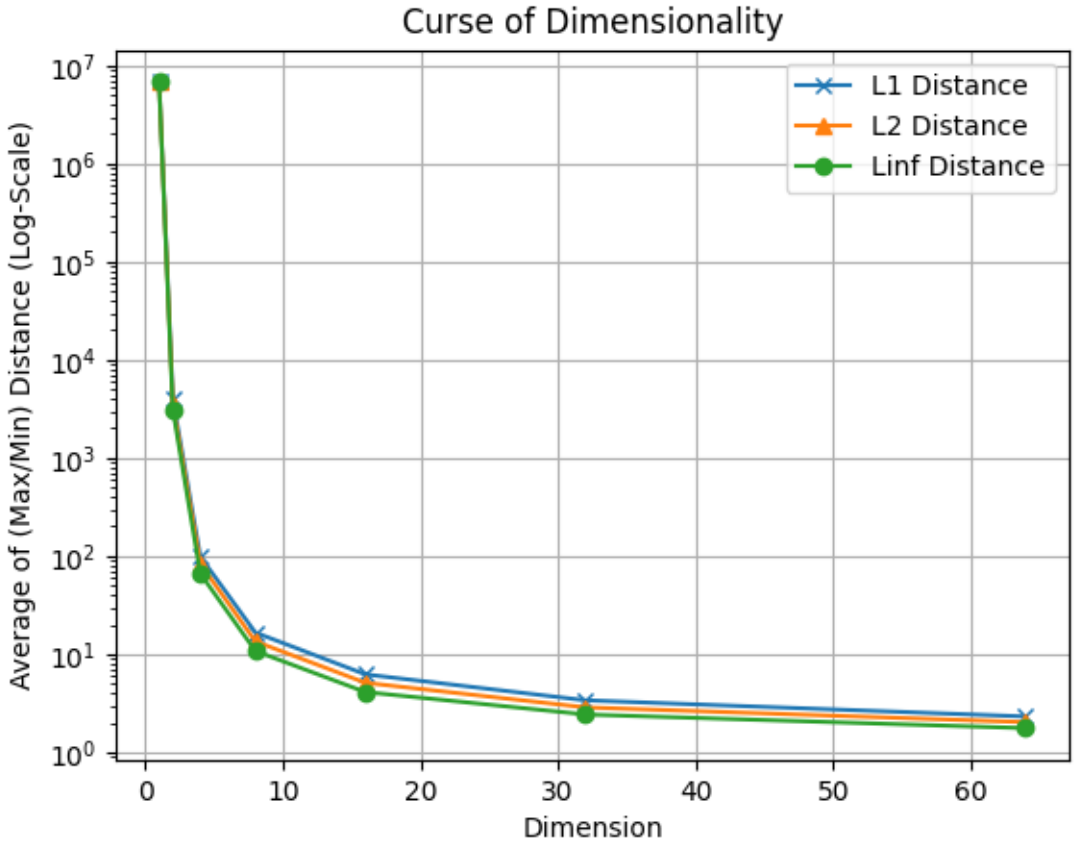

Figure 1: Illustrating Curse of Dimensionality

in a high-dimensional space is likely to be large, making it difficult to determine meaningful relationships between data points (this is known as space extension).

This is also means that there is no concept of near or far in higher dimensions (or the difference between far and near starts vanishing).

As the dimensionality of our data increases all points are spread out from each other, on average points are as far to a given point as they are close, meaning the concept of being near/far does not make sense, which is indicated by the average ratio falling rapidly in higher dimensional spaces.
Common distance metrics like Euclidean distance become less informative in high-dimensional spaces. In a high-dimensional cube, for instance, most of the volume is concentrated in the corners. As a result, if you randomly pick points in this space, they are more likely to be near a corner and equidistant from each other, making it challenging to differentiate between them based on distance alone.
As seen from the graph, all the three distance metrics are susceptible to the curse of dimensionality. However the L_1 seems to be less susceptible than L_2 which is less susceptible than L_∞.
This is explained by the nature of these distance functions. L_1 incorporates all dimensions without squaring, L_2 squares differences along each dimensions, hence dimensions which don't differ much are suppressed in the final distance calculation, this looses discriminative power which is essential for distinguishing nearby and farby points. L_∞ considers differences along a single dimension and hence a lot of information is lost out, which further reduces discriminative power making L_∞ the most susceptible to the curse of dimensionality.
By discriminative power we mean the ability to distinguish nearby and farby points in higher

dimensions. This shows that despite the fact the curse of dimensionality is an inherent problem in the mathematics of higher dimensions it can be supressed by using suitable distance metrics to some extent.

Graph Neural Networks

Visualization of Dataset

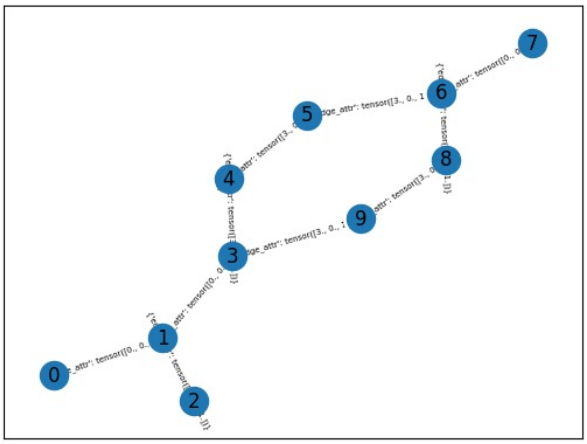

Figure 2: Label-1

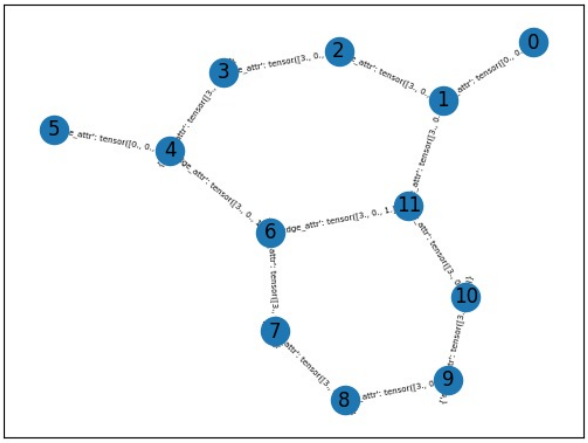

Figure 3: Label-1

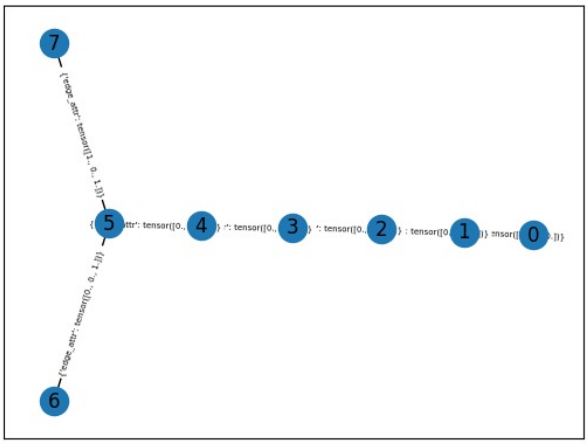

Figure 4: Label-0

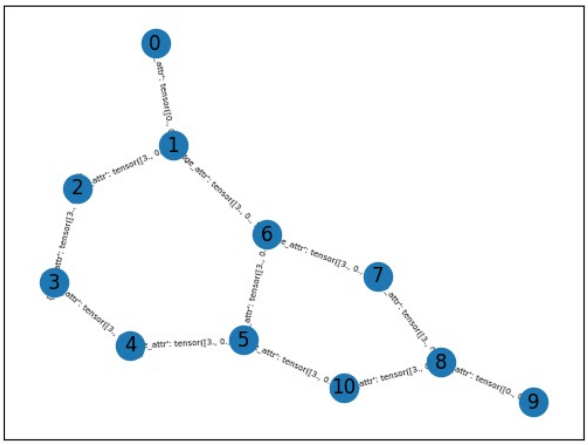

Figure 5: Label-0

Classification

We tried the following baselines, to get a preliminary idea of the kind of results to expect.

Baselines

For baselines we implement a random agent, simple logistic regression on pooled node embedding and a deep neural network on node embeddings. We can see that the baselines and GNNs are better than random guessing, and also that a simple Deep Neural Network is better than Logistic Regression. If GNNs are trained properly then they outperform simple deep neural networks indicating that they are able to take advantage of topological structure.

Effect of Layers

The number of layers in a neural network, particularly in Graph Neural Networks (GNNs), significantly influences both training and validation performance. In essence, adding more layers allows a neural network to learn more complex patterns and relationships in the data. In the context of Graph Neural Networks this means that each nodes embedding is influenced by far off neighbours as well which can increase expressivity of the network.

Although there is a tradeoff involved, increasing the number of layers can lead to overfitting, where the model performs exceptionally well on training data but fails to generalize to new,

unseen data. This is especially true for GNNs, as they often deal with complex and varied graph structures. Additionally, more layers can make the network harder to train due to issues like vanishing or exploding gradients, where the gradients become too small or too large to propagate useful learning back through the layers. In case of GNNs this can also lead to oversquashing and oversmoothing, which is a problem for our dataset given that most graphs don't have a larger diameter.

What we see that the performance of a single layer is often very poor with the the performance increasing as the layers increase uptill 4, after which the model starts overfitting. Hence we have decided to keep the number of layers in the neural network to 4 at max.

Effect of Type of Graph Neural Network

We tried the following GNNs

GCN

These are the most simple Message Passing Graph Neural Networks, our experiments show that their performance is very similar to Deep Neural Networks which means that they are not able to effectively make use of the Graph Structure in their learning.

GraphSage

This is a generalization to GCNs and provides more flexibility to the aggregation formulation along with other Deep Learning hacks. It does outperform GCNs on average but the performance remains relatively close to Deep Neural Networks indicating that the Graph Structure is again not being utilized effectively.

GAT

This network leverages the attention mechanism, allowing nodes to attend to neighbours with different weights which improves expressibility. Our experiments shows that the attention mechanism is effective for our dataset and it leads to improvements over other baselines and GNNs.

GIN

This network uses the summation as the aggregation function along with other hacks. The paper shows this to be effective and have better expressibility that other GNN architecutures. Our experimental results align with theory. GIN being the best GNN overall outperforming all other GNNs and baselines.

GIN > GAT > GraphSage $\sim$ Deep Neural Networks > GCN $\sim$ Logistic Regression > Random

Neural Network Training

We utilize the following techniques to improve Neural Network Training.

- **Adam Optimizer**
  - Adapts learning rates per parameter for efficiency.
  - Merges Momentum and RMSprop advantages.
  - Less sensitive to hyperparameter choices.

- **Dropout**
  - Reduces overfitting by randomly disabling neurons.
  - Forces network to learn redundant features for better generalization.
- **Batch Normalization**
  - Speeds up training by normalizing layer inputs.
  - Improves gradient flow, allowing higher learning rates.
- **Skip Connections**
  - Prevents vanishing gradients in deep networks.
  - Facilitates training of deeper networks by improving gradient flow.
- **Weight Decay**
  - Regularizes by penalizing large weights.
  - Helps in reducing overfitting.

0.1 Training Curves

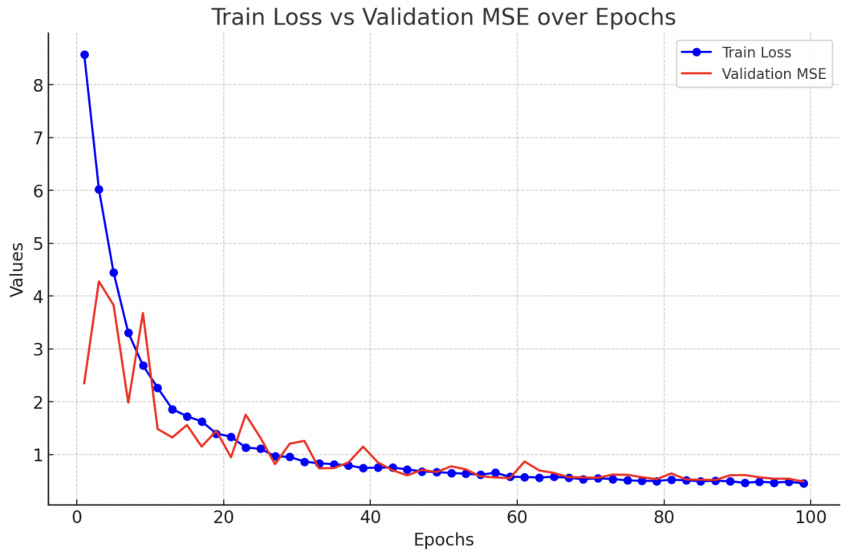

Figure 6: Regression Training Curves

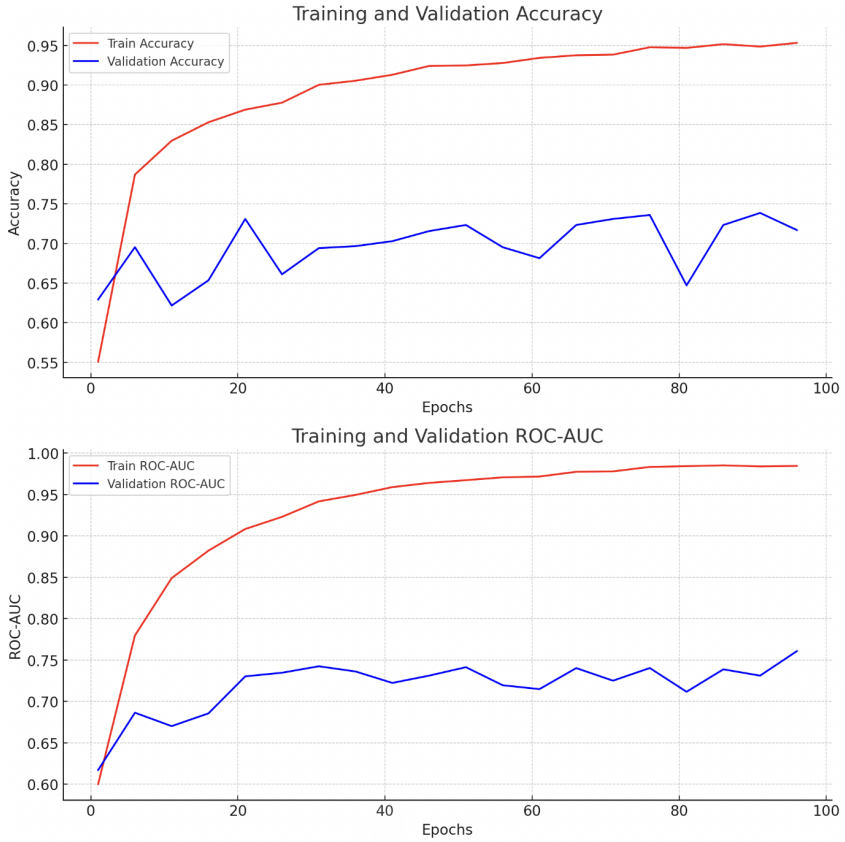

Figure 7: Classification ROC-Auc and Accuracy Curves

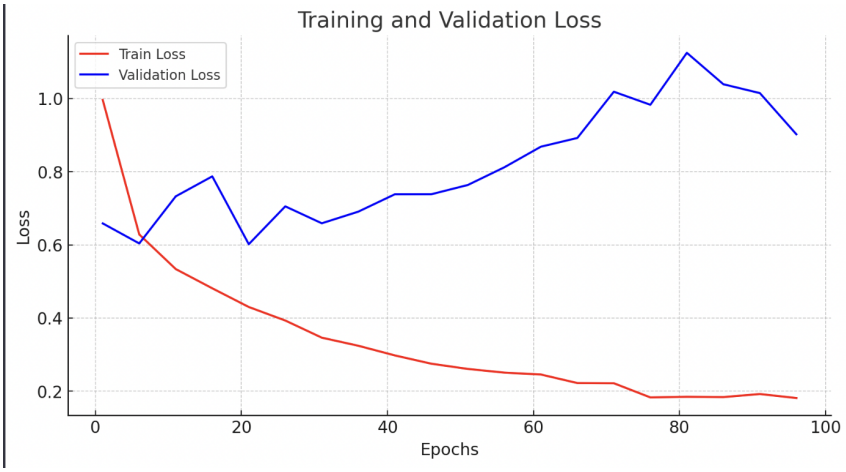

Figure 8: Classification Loss Curves

Error Analysis

The following are few graphs on which we make wrong predictions:

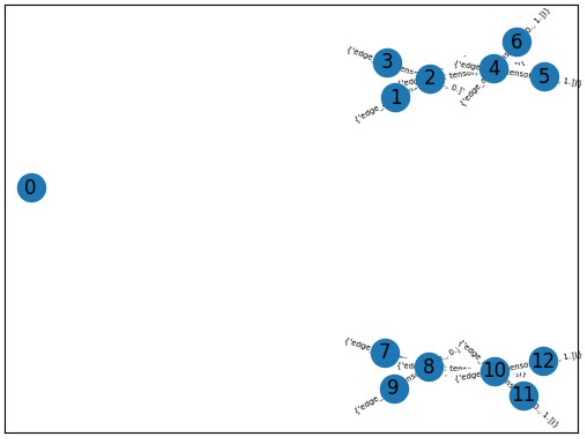

Figure 9: Label-0 Predicted Label-1

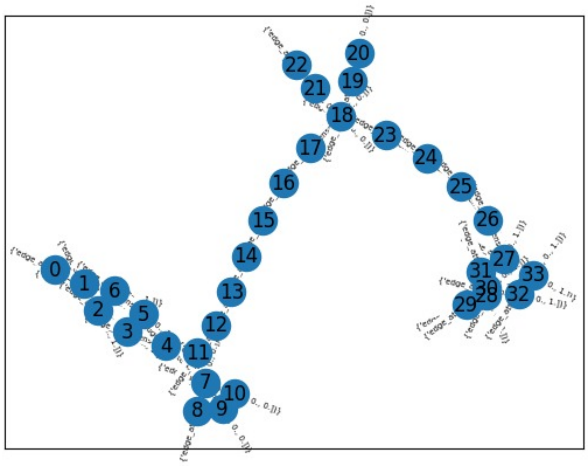

Figure 10: Label-1 Predicted Label-0

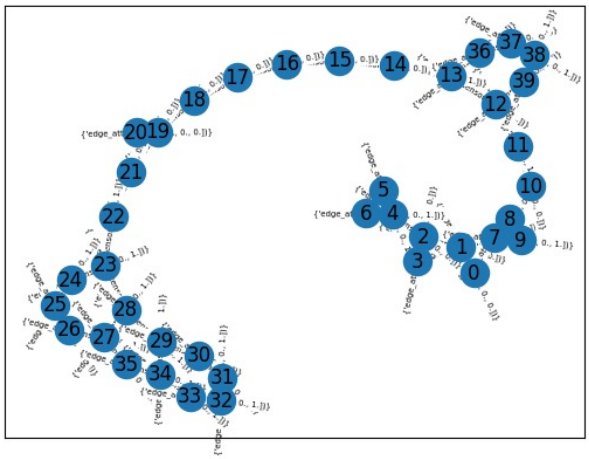

Figure 11: Label-1 Predicted Label-0

As we can see these are relatively difficult to model given most of these are disconnected making using GNNs not very useful, or they are extremely huge with uninteresting topologies again making modelling them by GNNs relatively hard.

Model	Embeddings Dim	Layers	Train Loss	Train Acc	Val ROC-AUC
Random	-	-	-	0.5	
Logistic Regression	32	1	0.6699	0.7919	0.6729
Logistic Regression	64	1	0.6649	0.7944	0.67878
Logistic Regression	128	1	0.6618	0.7952	0.6808
Deep Neural Network	32	4	0.6061	0.7972	0.704
Deep Neural Network	64	4	0.6062	0.7990	0.7033
Deep Neural Network	128	4	0.5878	0.7994	0.7035
GCN	32	1	0.6271	0.8116	0.600
GCN	32	2	0.5403	0.8350	0.6876
GCN	32	4	0.4759	0.8597	0.6853
GCN	64	1	0.6155	0.8054	0.6255
GCN	64	2	0.4845	0.8449	0.6689
GCN	64	4	0.4543	0.8566	0.6932
GCN	128	1	0.6032	0.8056	0.6450
GCN	128	2	0.4906	0.8484	0.6820
GCN	128	4	0.4412	0.8659	0.6785
GIN	32	1	0.5306	0.8339	0.6463
GIN	32	2	0.4390	0.8585	0.6827
GIN	32	4	0.3816	0.8684	0.7314
GIN	64	1	0.5151	0.8314	0.6590
GIN	64	2	0.4031	0.8700	0.6681
GIN	64	4	0.3660	0.8789	0.71193
GIN	128	1	0.5187	0.8174	0.6583
GIN	128	2	0.3711	0.8786	0.67127
GIN	128	4	0.3930	0.8736	0.7131
GAT	32	1	0.5822	0.8051	0.6500
GAT	32	2	0.5024	0.8397	0.7135
GAT	32	4	0.3731	0.8844	0.7146
GAT	64	1	0.5622	0.8075	0.6481
GAT	64	2	0.4269	0.8649	0.6839
GAT	64	4	0.3129	0.8956	0.70957
GAT	128	1	0.5683	0.8229	0.6472
GAT	128	2	0.4513	0.8495	0.6775
GAT	128	4	0.2841	0.9112	0.7273
GraphSage	32	1	0.5411	0.8238	0.6677
GraphSage	32	2	0.4520	0.8500	0.67458
GraphSage	32	4	0.3655	0.8766	0.6868
GraphSage	64	1	0.5332	0.8169	0.67528
GraphSage	64	2	0.4203	0.8547	0.6727
GraphSage	64	3	0.3027	0.9035	0.69249
GraphSage	128	1	0.5306	0.8215	0.6573
GraphSage	128	2	0.3818	0.8709	0.68474
GraphSage	128	4	0.3102	0.8936	0.66547

Table 4: Baseline Results for Classification