www.griddynamics.com

Grid Dynamics

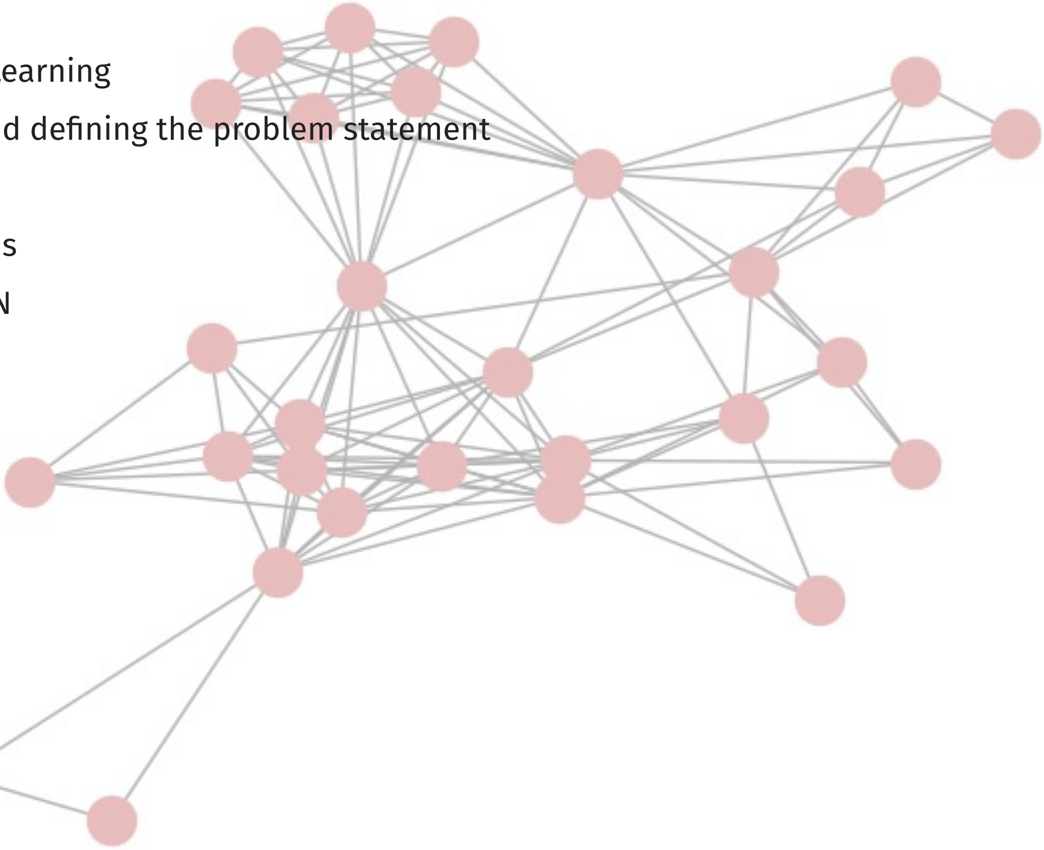
trusted engineering partner for digital transformation

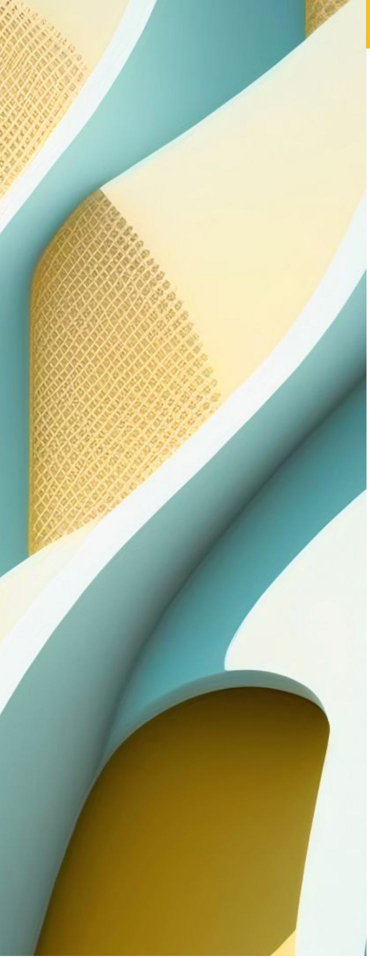
Deep Learning for Graphs

MARCH 2024

Agenda

1. Basics of Deep Learning
2. DBLP dataset and defining the problem statement
3. MLP
4. Analogy for GNNs
5. Building our GCN
6. GCN
7. Outcomes
8. Further scope





Deep Learning basics

Learn f(x)

- Complex function which maps inputs to outputs
- Goal is to learn the parameters for this function and use it to predict for unknown inputs
- e.g. slope, intercept in Linear Regression

Models

- Defines the structure of f(x)
- Use case dependent
- CNNs for images, RNNs for text generation etc.
- Analogous to selecting a cubic, quadratic or linear function to fit the data

Supervised/Unsupervised

- (input, desired output) available
- Unsupervised uses statistical measures
- e.g. clustering

Loss function

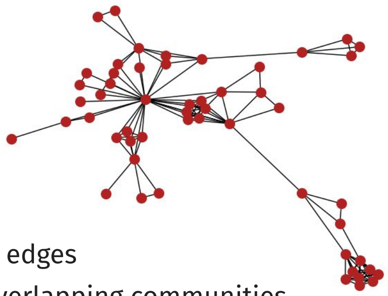
- Metric that measures the difference between the predicted and actual values
- Learning involves reducing the value of the loss function metric
- e.g. MSE

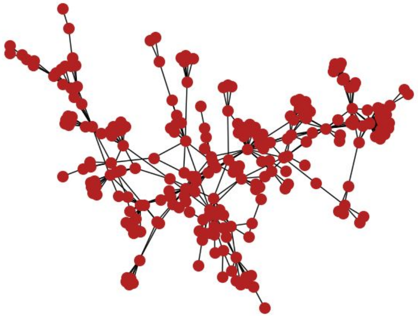
Gradient Descent

- Process of moving in a direction to reduce the loss

Testing and Evaluation

DBLP Dataset and problem formulation



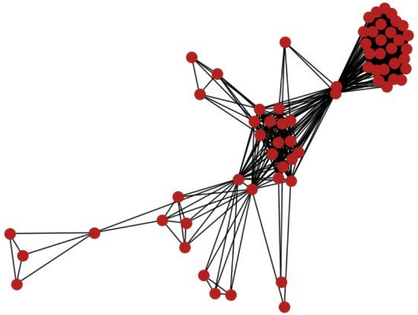


Dataset

- Graph data
- 317080 Nodes and 1049866 edges
- Nodes divided into 5000 overlapping communities

Use cases

- Market/Audience segmentation
- Recommender systems
- Fraud Detection

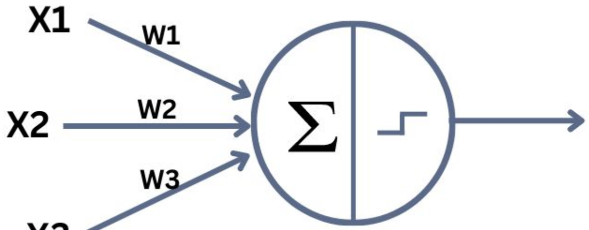


Defining the problem statement

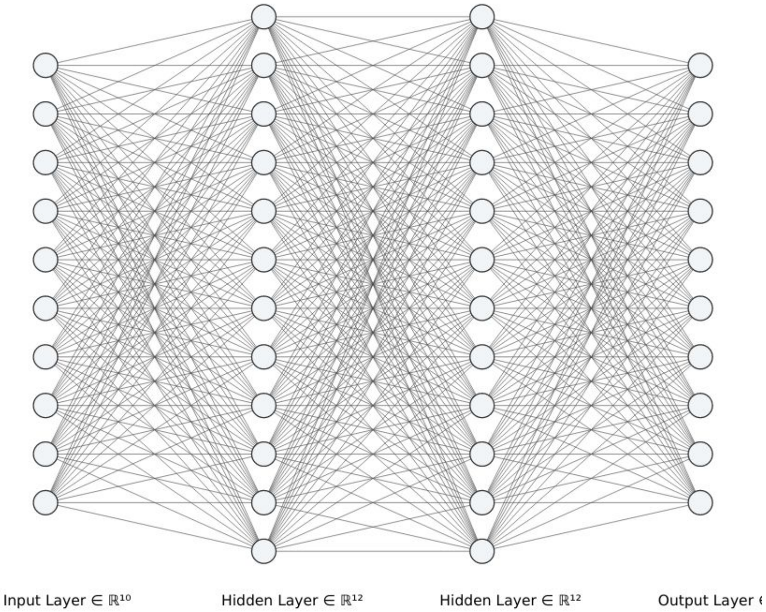
- Supervised Community Detection + Node classification
- For each node we need to predict the communities it belongs to

Approach

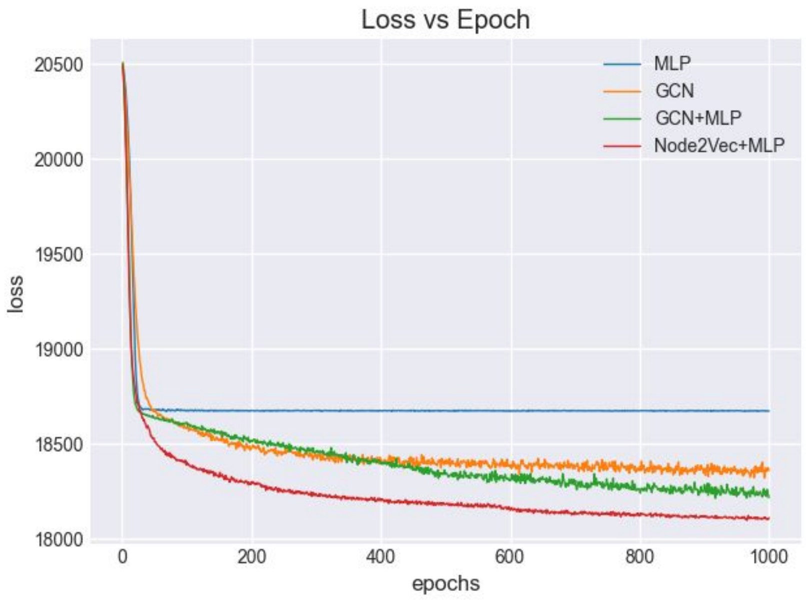
- Input: Graph (nodes and edges)
- Output: Matrix M, where each entry $m_{i,j}$ = 1 if node i is a member of community j else 0

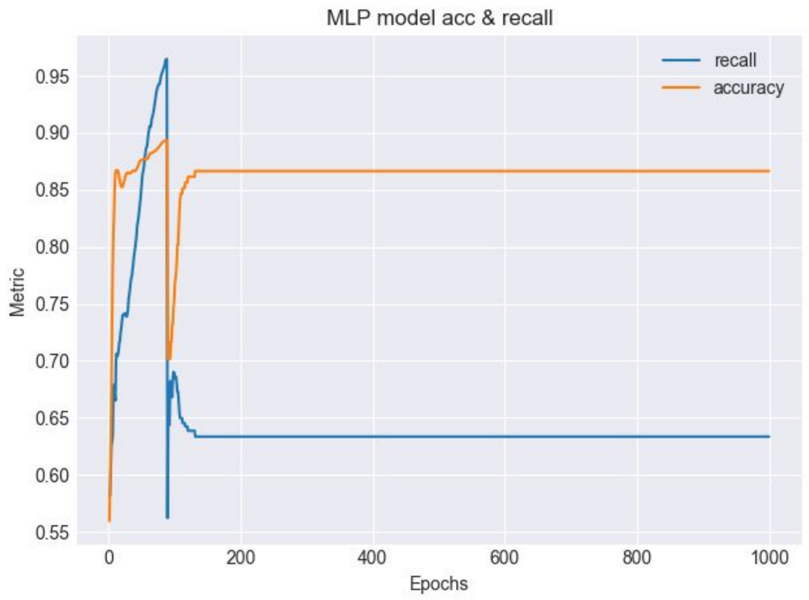


Single-layer perceptron

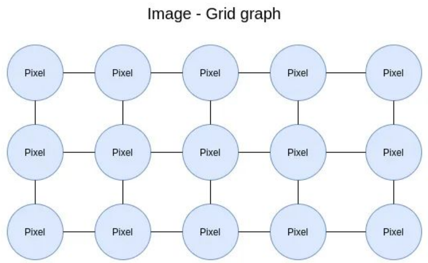


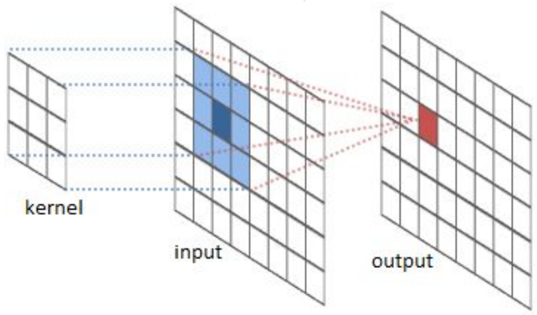
Multi-layer perceptron

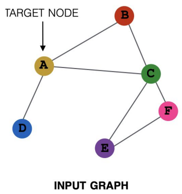

Loss vs Epoch
MLP
GCN
GCN+MLP
Node2Vec+MLP
loss
20500
20000
19500
19000
18500
18000
0
200
400
600
800
1000
epochs

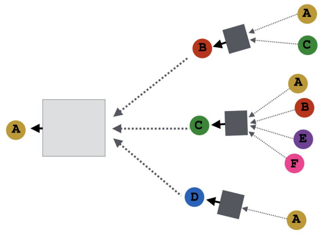

MLP model acc & recall
recall
accuracy
Metric
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0
200
400
600
800
1000
Epochs

GNNs are analogous to CNNs







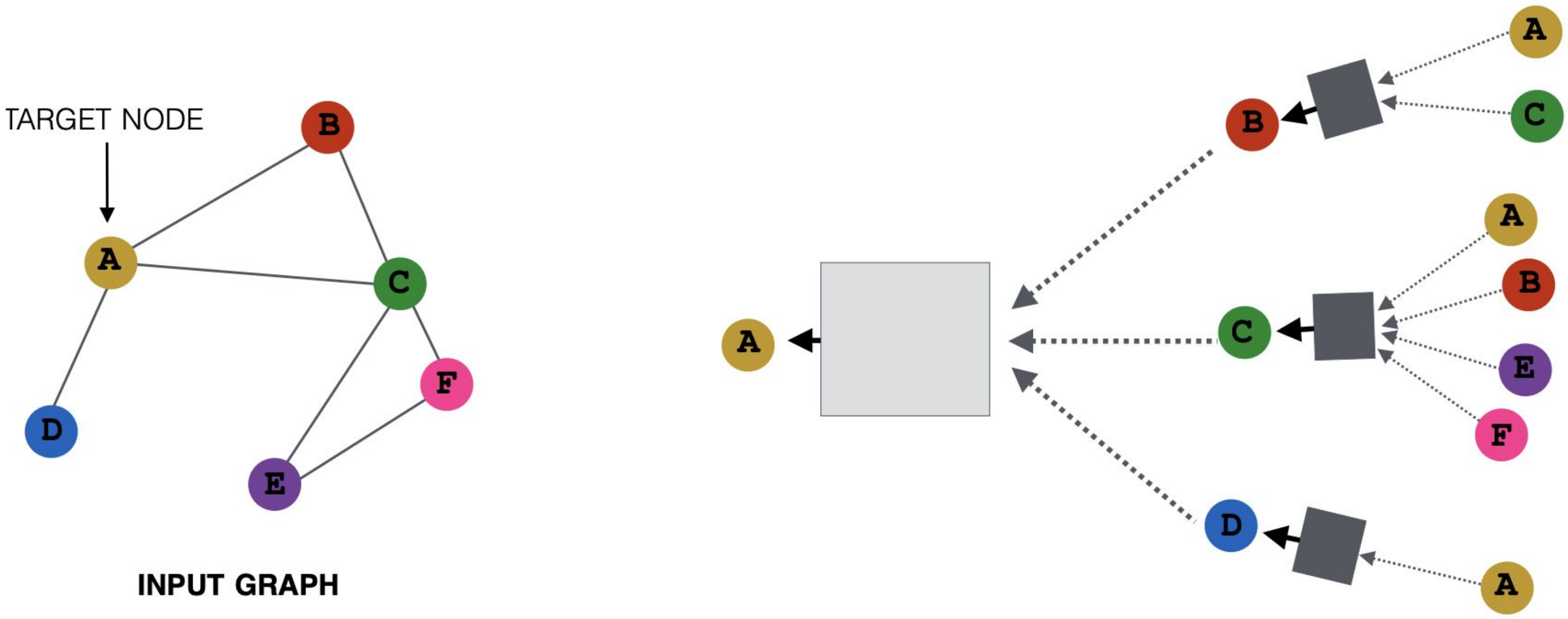


Images

- Images are a type of graph with a fixed structure.
- Kernels perform neighbourhood aggregation

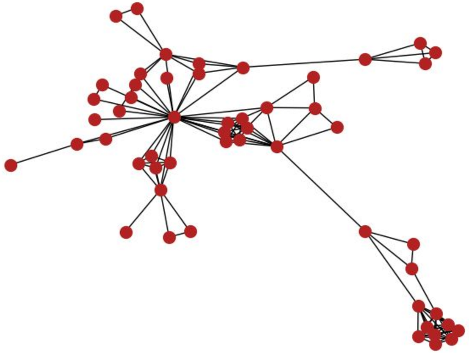
Graphs

- Inconsistent structure
- No inherent ordering
- Local neighbourhoods networks perform aggregation

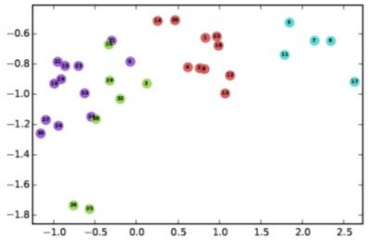


TARGET NODE
A
B
C
D
E
F
INPUT GRAPH

Feature Engineering



Input



Output

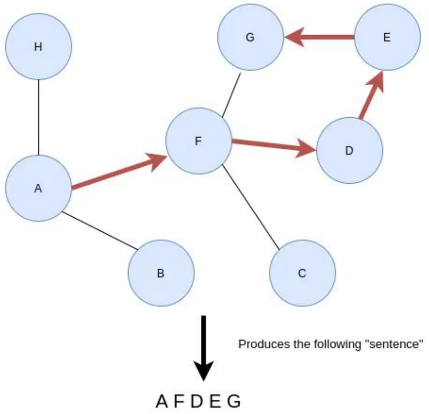
Graph Statistics

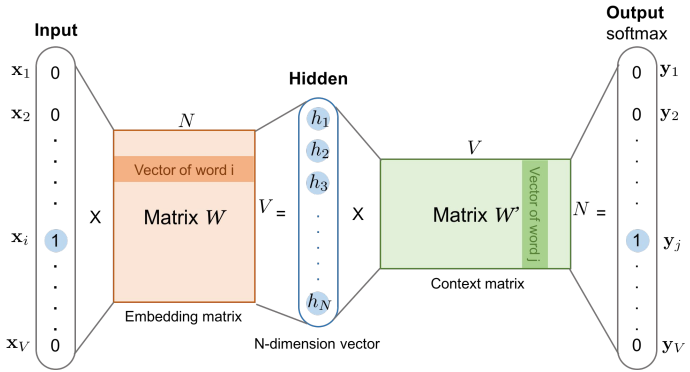
- Use node based statistical measures as features
- Often graph networks are unable to easily learn some of these
- E.g. degree, clustering coefficient, centrality

Embedding Technique

- Node2Vec is Word2Vec for graph nodes
- Embeds nodes as a vectors
- Closer nodes are embedded closer in the vector space

Node2Vec algorithm





Sentence generation based on walk length and walks per node

Training samples generated for each node based on window size (input,target)

The skip-gram model is trained on (input,target) training samples.

Finally the hidden dimension is used as the embedding

Loss Function

$$BCE = -\frac{1}{N}\sum_{i=0}^{N} y_i \cdot log(\hat{y}_i) + (1 - y_i) \cdot log(1 - \hat{y}_i)$$

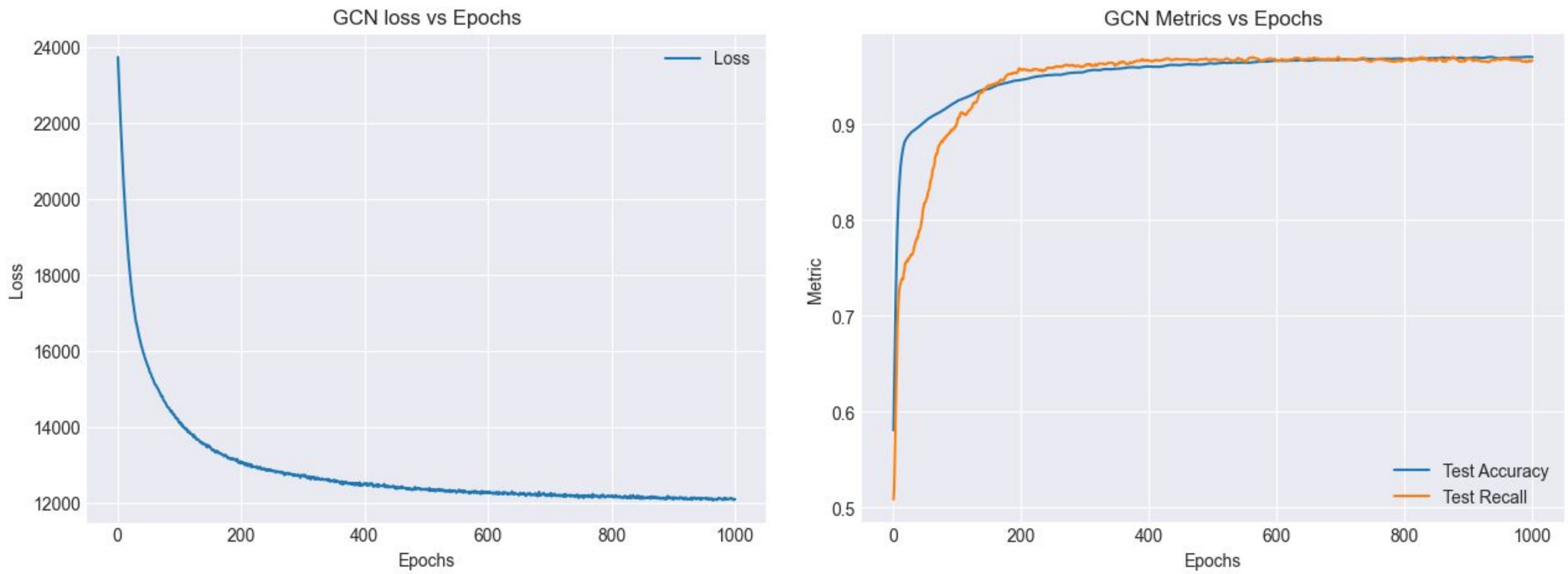
- The **Loss function** is important because gradient descents goal is to minimize the loss
- Output matrix is **sparse** (~2% 1s only), which causes **high accuracy and low recall** while using regular BCELoss
- i.e. predicts all zeroes

$$BCE_{sparse} = (-\Sigma\, w_{true} \cdot p_i \cdot \log(\widehat{p}_i) + w_{false} \cdot (1 - p_i) \cdot \log(1 - \widehat{p}_i))$$

- To solve this we use **weighted BCE loss** for sparse matrices.
- $\mathbf{W_{true}}$ is the relative cost of predicting a 1 as a 0
- $\mathbf{W_{false}}$ is the relative cost of predicting a 0 as a 1
- $\mathbf{W_{true} >>> W_{false}}$ makes the algorithm to increase recall as well as the accuracy

GCN training loss and evaluation metrics

200 communities and 50% randomly sampled edges



Limitations & Improvements

- GraphSAGE implementation for entire dataset
  - It is a model specifically designed for large graphs
- Testing different structures, activation functions and loss functions
- Adding skip connections to models
- Feature engineering
  - Learnable features
  - Removing dependency on Node2Vec algorithms

Edge Sampling

Edges retained	75%	56.25%	42.18%	31.64%
MLP	*87.32%*	*87.14%*	*87.19%*	*87.18%*
GCN	*89.2%*	*86.68%*	*86.95%*	*87.74%*
GCN + MLP processing	*87.8%*	*87.77%*	*87.45%*	*87.58%*
Node2Vec + MLP	*93.2%*	*90.58%*	*89.48%*	*88.57%*

Takeaways

- MLP model doesn't take edges as inputs so it is independent of edge sampling.
- Node2Vec + MLP model is the most reliable and accurate.
- Performance of GCN models decreases as the number of edges are reduced

Dataset Size

Number of communities	50	100	200	500
MLP	*76.8%*	*86.2%*	*86.73%*	*91.9%*
GCN	*83.83%*	*87.4%*	*89.69%*	*92.05%*
GCN + MLP processing	*86.67%*	*87.52%*	*88.91*	*92.94%*
Node2Vec + MLP	*91.28%*	*91.61%*	*92.94%*	*93.21%*

Takeaways

- Graph Networks perform better than the MLP model in all cases, as they leverage edge information.
- Increase in graph size corresponds to increase in the number of edges and nodes which graph networks leverage to predict more accurately.
- Node2Vec performs the best in all categories.

Graph Networks and their applications