Intro to GNN

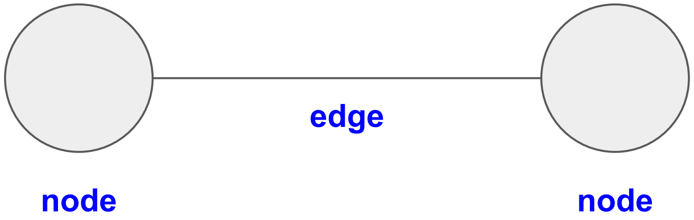
Part I

Introduction to Graph Data

Graphs are mathematical structures used to model pairwise relations between objects.

A graph is made up of **vertices** (or **nodes**) connected by **edges**.

Can represent a wide range of real world data: social networks, biological networks, transportation networks, etc.



Importance of Graph Data in Real-world Applications

Complex Systems Modeling: Graphs can model complex systems in a natural way, capturing the interconnections and relationships between different entities.

Insight and Decision Making: Understanding the structure and dynamics of graphs can lead to better decision-making in areas such as social media analysis, recommendation systems, and network security.

Data Interconnectivity: Unlike traditional data representations, graphs emphasize the relationships between data points, offering a more holistic view of the data set.

Challenges in Graph Data Analysis

Scalability: As graphs grow in size, analyzing them with traditional methods becomes computationally expensive or even infeasible.

Dynamic Nature: Many real-world graphs are dynamic, with nodes and edges changing over time, requiring flexible and adaptive analysis methods.

Heterogeneity: Graphs can contain various types of nodes and edges, making uniform analysis challenging.

Structural Complexity: The lack of a fixed structure in graphs (unlike images or text) complicates the application of machine learning models.

Graph Analysis Techniques

Traditional Methods: Earlier approaches include graph theory metrics (like centrality measures, clustering coefficient, etc.) and matrix factorization techniques.

Machine Learning on Graphs: Recent advancements involve applying machine learning to graphs, with graph neural networks (GNNs) being the forefront technology allowing for direct learning from graph-structured data.

Why Graph Data is Unique

Relational Information: Graphs inherently contain relational information, offering a rich source of data that is not easily captured by traditional tabular data.

Flexibility: Graphs are flexible in representing various types of data from different domains, allowing for the modeling of complex interactions and relationships.

Pytorch-geometric

Install

```
pip install torch_geometric
```

from torch_geometric.data import Data

```
import torch
from torch_geometric.data import Data


edge_index = torch.tensor([[0, 1, 1, 2],
                           [1, 0, 2, 1]], dtype=torch.long)
x = torch.tensor([[-1], [0], [1]], dtype=torch.float)


data = Data(x=x, edge_index=edge_index)
print(data)
```

Graph Validation

```
data.validate(raise_on_error=True)
```

Example dataset

```
from torch_geometric.datasets import TUDataset


dataset = TUDataset(root='/tmp/ENZYMES', name='ENZYMES')

print(dataset)

print(len(dataset))

print(dataset.num_classes)

print(dataset.num_node_features)


data = dataset[0]

print(data)

print(data.is_undirected())
```

```python
from torch_geometric.datasets import TUDataset

dataset = TUDataset(root='/tmp/ENZYMES', name='ENZYMES')
print(dataset)
print(len(dataset))
print(dataset.num_classes)
print(dataset.num_node_features)
```

```
Downloading https://www.chrsmrrs.com/graphkerneldatasets/ENZYMES.zip
Processing...
ENZYMES(600)
600
6
3
Done!
```

```python
data = dataset[0]
print(data)
print(data.is_undirected())
```

```
Data(edge_index=[2, 168], x=[37, 21], y=[1])
True
```

DataLoader

```
from torch_geometric.datasets import TUDataset
from torch_geometric.loader import DataLoader


dataset = TUDataset(root='/tmp/ENZYMES', name='ENZYMES', use_node_attr=True)
loader = DataLoader(dataset, batch_size=32, shuffle=True)


for batch in loader:
    print(batch)
    print(batch.num_graphs)
```

```
DataBatch(edge_index=[2, 3890], x=[1031, 21], y=[32], batch=[1031], ptr=[33])
32
DataBatch(edge_index=[2, 3478], x=[896, 21], y=[32], batch=[896], ptr=[33])
32
DataBatch(edge_index=[2, 3586], x=[1014, 21], y=[32], batch=[1014], ptr=[33])
32
DataBatch(edge_index=[2, 3808], x=[1014, 21], y=[32], batch=[1014], ptr=[33])
32
DataBatch(edge_index=[2, 4230], x=[1068, 21], y=[32], batch=[1068], ptr=[33])
32
DataBatch(edge_index=[2, 3566], x=[969, 21], y=[32], batch=[969], ptr=[33])
32
DataBatch(edge_index=[2, 4284], x=[1221, 21], y=[32], batch=[1221], ptr=[33])
32
DataBatch(edge_index=[2, 4128], x=[1047, 21], y=[32], batch=[1047], ptr=[33])
32
DataBatch(edge_index=[2, 3416], x=[885, 21], y=[32], batch=[885], ptr=[33])
32
DataBatch(edge_index=[2, 3464], x=[907, 21], y=[32], batch=[907], ptr=[33])
32
DataBatch(edge_index=[2, 4162], x=[1056, 21], y=[32], batch=[1056], ptr=[33])
32
DataBatch(edge_index=[2, 4070], x=[1045, 21], y=[32], batch=[1045], ptr=[33])
32
DataBatch(edge_index=[2, 4306], x=[1125, 21], y=[32], batch=[1125], ptr=[33])
32
DataBatch(edge_index=[2, 4412], x=[1158, 21], y=[32], batch=[1158], ptr=[33])
32
DataBatch(edge_index=[2, 3874], x=[1021, 21], y=[32], batch=[1021], ptr=[33])
32
DataBatch(edge_index=[2, 4568], x=[1147, 21], y=[32], batch=[1147], ptr=[33])
32
DataBatch(edge_index=[2, 4368], x=[1132, 21], y=[32], batch=[1132], ptr=[33])
32
DataBatch(edge_index=[2, 3860], x=[1003, 21], y=[32], batch=[1003], ptr=[33])
32
DataBatch(edge_index=[2, 3094], x=[841, 21], y=[24], batch=[841], ptr=[25])
24
```