

Graph Neural Network

Lecture 2



- 1 Introduction to PyG
- 2 Basic Code Examples
- 3 Advanced Features
- 4 Conclusion

What is PyTorch Geometric?

- **PyG** ([PyTorch Geometric](#)) is a library for deep learning on graphs built on PyTorch.
- Provides tools for graph creation, transformations, and GNN layers.
- Efficient handling of large-scale graphs.
- Installation:
 - `pip install torch_geometric`
 - `conda install pyg c pyg`

- **Data Handling:** Easy-to-use data structures for graphs.
- **Predefined Layers:** GCN, GAT, GraphSAGE, etc.
- **Datasets:** Built-in datasets like Cora, Citeseer, and more.
- **Extensibility:** Custom layers and datasets.

- 1 Introduction to PyG
- 2 Basic Code Examples**
- 3 Advanced Features
- 4 Conclusion

Creating a simple graph

```
1 import torch
2 from torch_geometric.data import Data
3
4 # Define node features and edges
5 x = torch.tensor([[1], [2], [3], [4]],
6                  dtype=torch.float)
7 edge_index = torch.tensor([[0, 1, 2, 3],
8                             [1, 0, 3, 2]],
9                             dtype=torch.long)
10 # Create a graph data object
11 data = Data(x=x, edge_index=edge_index)
12 print(data)
```

Accessing Graph Properties

```
1 print(f'Number of nodes: {data.num_nodes}')
```

```
2 print(f'Number of edges: {data.num_edges}')
```

```
3 print(f'Has isolated nodes: {data.has_isolated_nodes()}')
```

```
4 print(f'Is undirected: {data.is_undirected()}')
```

Adding Edge Weights

```
1 | edge_attr = torch.tensor([[0.5], [0.8], [0.3], [1.0]],  
2 |                      dtype=torch.float)  
3 | data.edge_attr = edge_attr  
4 | print(data)
```


Convert to Undirected Graph

```
1 from torch_geometric.utils import to_undirected
2
3 edge_index = to_undirected(edge_index)
```

Using Built-in Datasets

```
1 from torch_geometric.datasets import Planetoid
2
3 # Load the Cora dataset
4 dataset = Planetoid(root='/tmp/Cora', name='Cora')
5 data = dataset[0] # Get the first graph
6
7 print(f'Number of nodes: {data.num_nodes}')
8 print(f'Number of edges: {data.num_edges}')
9 print(f'Number of classes: {dataset.num_classes}')
```

- ① Introduction to PyG
- ② Basic Code Examples
- ③ Advanced Features
- ④ Conclusion

- PyG supports batching multiple graphs into a single graph object.
- Useful for training on datasets with multiple small graphs.

```
1 | # -----  
2 | from torch_geometric.data import DataLoader  
3 |  
4 | loader = DataLoader(dataset,  
5 |                     batch_size=32,  
6 |                     shuffle=True)  
7 | for batch in loader:
```

- PyG provides utilities for graph transformations (e.g., normalization, augmentation).

```
1 # -----  
2 from torch_geometric.transforms import NormalizeFeatures  
3  
4 dataset = Planetoid(root='/tmp/Cora', name='Cora',  
5                     transform=NormalizeFeatures())  
6 data = dataset[0]
```

```
1 # -----
2 from torch_geometric.nn import GCNConv
3 import torch.nn.functional as F
4
5
6 class GCN(torch.nn.Module):
7     def __init__(self):
8         super().__init__()
9         self.conv1 = GCNConv(1, 16)
10        self.conv2 = GCNConv(16, 2)
11
12    def forward(self, x, edge_index):
13        x = self.conv1(x, edge_index)
14        x = F.relu(x)
```

Defining a Custom GNN Layer

```
1  
2  
3 # -----  
4 from torch_geometric.nn import MessagePassing  
5  
6  
7 class CustomGNN(MessagePassing):  
8     def __init__(self):
```

Training a GNN Model

```
1      def forward(self, x, edge_index):
2          return self.propagate(edge_index, x=x)
3
4
5      # -----
6      model = GCN()
7      optimizer = torch.optim.Adam(model.parameters(),
8                                     lr=0.01)
9
10     for epoch in range(100):
```


- 1 Introduction to PyG
- 2 Basic Code Examples
- 3 Advanced Features
- 4 Conclusion

- PyTorch Geometric simplifies working with GNNs.
- Supports various graph neural network architectures.
- Efficient for large-scale graph data processing.