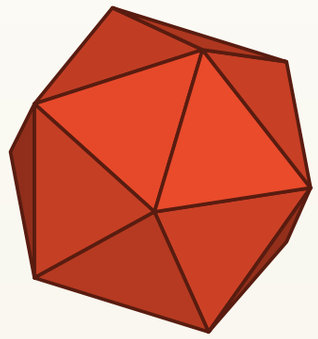
dortmund university

SFB 876 Providing Information
by Resource-Constrained Data Analysis

Updates on  PyTorch geometric

Matthias Fey

matthias.fey@udo.edu

/rusty1s/pytorch_geometric

Overview

GNNs

Cheby GCN SAGE PointNet MoNet MPNN GAT
SplineCNN AGNN EdgeCNN S-GCN R-GCN PointCNN
ARMA APPNP GIN GIN-E CG GatedGCN NMF TAG
Signed-GCN DNA PPFNet FeaST Hyper-GCN GravNet

Pooling

Set2Set SortPool DiffPool MinCUT Graclus
VoxelGrid TopK SAG EdgePool ASAP

Models

(V)GAE ARG(V)A DGI Node2Vec GraphUNet
GeniePath SchNet DimeNet MetaPath2Vec ReNet

Utilities

GNNExplainer SIGN GDC ClusterGCN DropEdge
GraphSAINT NeighborSampling GraphSizeNorm JK

Large-Scale Graph Support

```
def train(data):  # Full-batch training
    out = model(data.x, data.edge_index)
    loss = F.nll_loss(out, data.y)
```

RuntimeError: CUDA error: out of memory

↓ ↓ ↓

```
+ loader = ClusterLoader(  # Cluster-GCN
+     ClusterData(data, num_parts=128), batch_size=32)

+ loader = GraphSAINTRandomWalkSampler(  # GraphSAINT
+     data, batch_size=128, walk_length=3)

~ def train(loader):  # Mini-batch training
+     for data in loader:
          out = model(data.x, data.edge_index)
          loss = F.nll_loss(out, data.y)
```

Bipartite Graphs

Most GNNs can work on **bipartite graphs**, *e.g.*, for *neighbor sampling* or *heterogeneous graphs*:

```
SAGEConv(in_channels=(64, 128), out_channels=256)
```

Different feature sizes for source and destination nodes!

```
loader = NeighborSampler(   # GraphSAGE
    edge_index, sizes=[15, 10, 5], batch_size=1024)

for n_id, adjs in loader:
    x_src = x[n_id]
    for edge_index, size in adjs:
        x_dst = x_src[:size[1]]
        x_src = conv((x_src, x_dst), edge_index)
```

Select destination nodes from source nodes

Heterogeneous Graphs

```
Data(
  edge_index_dict={
    ('paper', 'written_by', 'author'): ...,
    ('paper', 'published_in', 'venue'): ...,
  },
  x_dict = { 'paper': ... },
  y_dict = { 'paper': ... },
)

# Pass messages between all relation types
for tuple, edge_index in edge_index_dict.items():
    x_src = x_dict[tuple[0]]
    x_dst = x_dict[tuple[2]]
    out = conv((x_src, x_dst), edge_index)
    out_dict[tuple[2]] += out
```

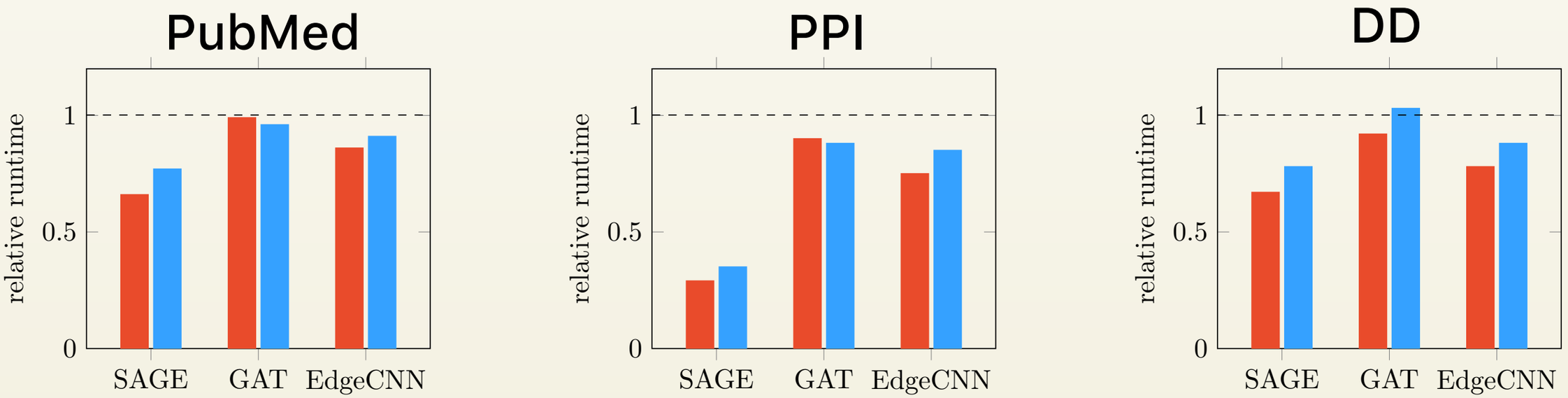
Memory-Efficient Aggregations

Additional graph storage format: `SparseTensor`

- ✓ *Fast* **`matmul`** via Yang et al. (2018)
- ✓ *Fast* **`transpose`** via caching

```
class MyConv(MessagePassing):
    def message_and_aggregate(self, adj, x):
        return adj.t() @ x
```

Supports multiple aggregations!



TorchScript

```
class Net(torch.nn.Module):
    def __init__(self, ...):
        super(Net, self).__init__()
        self.conv1 = GCNConv(...)
        self.conv2 = GCNConv(...)

    def forward(self, x, edge_index):
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = self.conv2(x, edge_index)
        return x.log_softmax(dim=-1)


model = Net(...)
```

TorchScript

```
  class Net(torch.nn.Module):
      def __init__(self, ...):
          super(Net, self).__init__()
~         self.conv1 = GCNConv(...).jittable()
~         self.conv2 = GCNConv(...).jittable()

      def forward(self, x, edge_index):
          x = self.conv1(x, edge_index)
          x = F.relu(x)
          x = self.conv2(x, edge_index)
          return x.log_softmax(dim=-1)

~ model = torch.jit.script(Net(...))
```

Creates jittable instances!

Create TorchScript program!

GNNExplainer

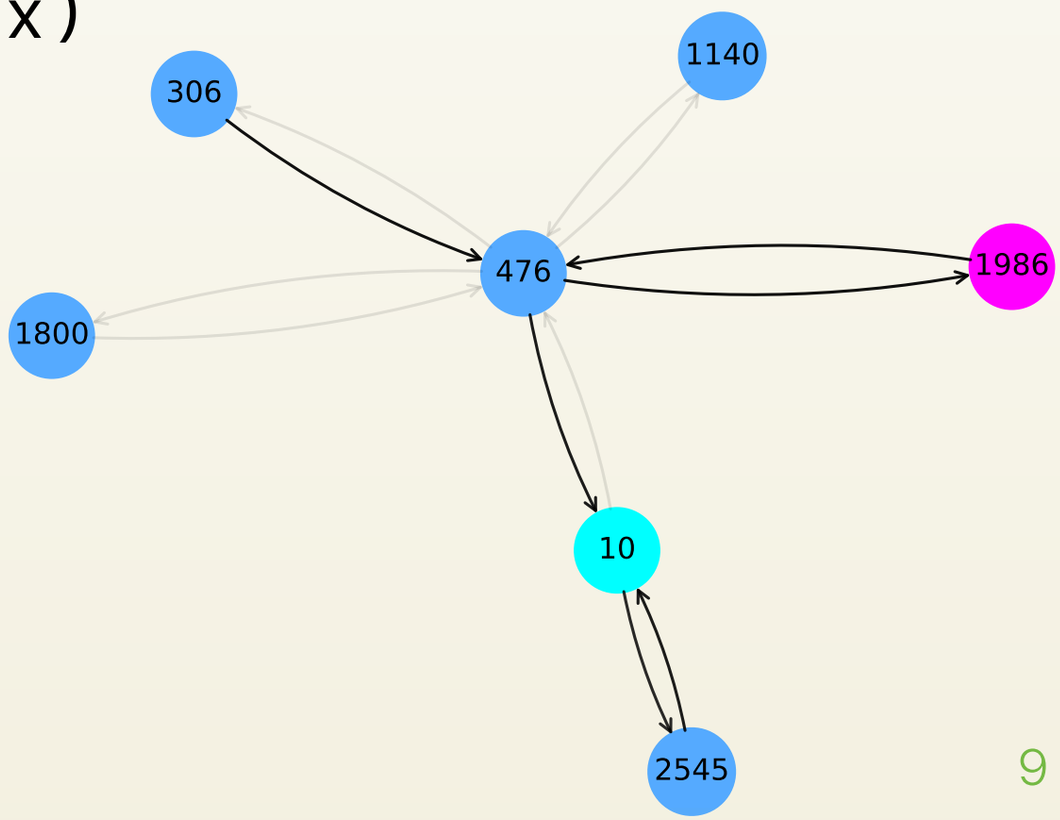
First step towards **interpreting** and **understanding GNN models**:

Works with any PyG model!

```
explainer = GNNExplainer(model, epochs=200)

node_feat_mask, edge_mask = explainer.explain_node(
    node_idx=10, x, edge_index)

explainer.visualize_subgraph(
    node_idx=10,
    edge_index,
    edge_mask)
```



Additional Features

GNNs can now also operate on **static graphs**:

```
x = torch.randn(batch_size, num_nodes, in_channels)
x = conv(x, edge_index)
>>> torch.Size([batch_size, num_nodes, out_channels])
```

Fast installation for *all* major OS/CUDA combinations:

```
$ pip install torch-geometric
$ pip install torch-scatter==latest+cu102 -f ...
$ pip install torch-sparse==latest+cu102 -f ...
```

- ✓ Dead-simple access to an ever-increasing set of datasets
- ✓ Deterministic GNN operators
- ✓ OGB examples

/rusty1s/pytorch_geometric