model-sageconv

March 18, 2024

[1]:
```
!pip install torch-geometric
```

```
Collecting torch-geometric
  Downloading torch_geometric-2.5.1-py3-none-any.whl (1.1 MB)
                             1.1/1.1 MB
7.6 MB/s eta 0:00:00
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-
packages (from torch-geometric) (4.66.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages
(from torch-geometric) (1.25.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages
(from torch-geometric) (1.11.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages
(from torch-geometric) (2023.6.0)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages
(from torch-geometric) (3.1.3)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-
packages (from torch-geometric) (3.9.3)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from torch-geometric) (2.31.0)
Requirement already satisfied: pyparsing in /usr/local/lib/python3.10/dist-
packages (from torch-geometric) (3.1.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-
packages (from torch-geometric) (1.2.2)
Requirement already satisfied: psutil>=5.8.0 in /usr/local/lib/python3.10/dist-
packages (from torch-geometric) (5.9.5)
Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->torch-geometric) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-
packages (from aiohttp->torch-geometric) (23.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->torch-geometric) (1.4.1)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->torch-geometric) (6.0.5)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-
packages (from aiohttp->torch-geometric) (1.9.4)
Requirement already satisfied: async-timeout<5.0,>=4.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->torch-geometric) (4.0.3)
```

```
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from jinja2->torch-geometric) (2.1.5)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->torch-geometric) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests->torch-geometric) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->torch-geometric) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->torch-geometric)
(2024.2.2)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn->torch-geometric) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn->torch-geometric)
(3.3.0)
Installing collected packages: torch-geometric
Successfully installed torch-geometric-2.5.1
```

[2]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Loading chunks of the dataset

Note: To preserve RAM, several unused variables are deleted throughout the norebook

[3]:
```
import torch
chunk1 = torch.load('/content/drive/MyDrive/SCI_data/first.pt')
```

[4]:
```
chunk2 = torch.load('/content/drive/MyDrive/SCI_data/second (1).pt')
```

[9]:
```
from torch.utils.data import Dataset

class GraphDataset(Dataset):
  def __init__(self, data, preprocess):
    self.data = data
    self.preprocess = preprocess

  def __len__(self,):
    return len(self.data)

  def __getitem__(self, index):
    res = self.data[index]
    for p in self.preprocess:
      res = p(res)
    return res
```

```
chunk1+=chunk2
del chunk2
```

[10]:
```
from sklearn.model_selection import train_test_split
rand_seed = 42
X_train, X_test = train_test_split(chunk1, test_size=0.1, random_state =␣
 ↪rand_seed)
X_train, X_val = train_test_split(X_train, test_size=0.1, random_state =␣
 ↪rand_seed)
print(len(X_train), len(X_val), len(X_val))
```

```
14580 1620 1620
```

[12]:
```
device = 'cuda'
```

[13]:
```
from torch_geometric.loader import DataLoader

train_loader = DataLoader(X_train, batch_size=32, shuffle=True)
val_loader = DataLoader(X_val, batch_size=32, shuffle=False)
test_loader = DataLoader(X_test, batch_size=32, shuffle=False)
```

[14]:
```
!pip install torchmetrics
```

```
Collecting torchmetrics
  Downloading torchmetrics-1.3.1-py3-none-any.whl (840 kB)
                             840.4/840.4
kB 10.5 MB/s eta 0:00:00
Requirement already satisfied: numpy>1.20.0 in
/usr/local/lib/python3.10/dist-packages (from torchmetrics) (1.25.2)
Requirement already satisfied: packaging>17.1 in /usr/local/lib/python3.10/dist-
packages (from torchmetrics) (23.2)
Requirement already satisfied: torch>=1.10.0 in /usr/local/lib/python3.10/dist-
packages (from torchmetrics) (2.1.0+cu121)
Collecting lightning-utilities>=0.8.0 (from torchmetrics)
  Downloading lightning_utilities-0.10.1-py3-none-any.whl (24 kB)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-
packages (from lightning-utilities>=0.8.0->torchmetrics) (67.7.2)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.10/dist-packages (from lightning-
utilities>=0.8.0->torchmetrics) (4.10.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from torch>=1.10.0->torchmetrics) (3.13.1)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages
(from torch>=1.10.0->torchmetrics) (1.12)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-
packages (from torch>=1.10.0->torchmetrics) (3.2.1)
```

```
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages
(from torch>=1.10.0->torchmetrics) (3.1.3)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages
(from torch>=1.10.0->torchmetrics) (2023.6.0)
Requirement already satisfied: triton==2.1.0 in /usr/local/lib/python3.10/dist-
packages (from torch>=1.10.0->torchmetrics) (2.1.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from
jinja2->torch>=1.10.0->torchmetrics) (2.1.5)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-
packages (from sympy->torch>=1.10.0->torchmetrics) (1.3.0)
Installing collected packages: lightning-utilities, torchmetrics
Successfully installed lightning-utilities-0.10.1 torchmetrics-1.3.1
```

[17]:
```python
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torch_geometric.nn import GCNConv, global_mean_pool, GATConv, SAGEConv,␣
 ↪SAGPooling
from torch_geometric.nn.conv import GraphConv
from torch_geometric.utils import to_undirected
from torch_geometric.data import DataLoader
from torchmetrics.classification import BinaryAUROC
auroc = BinaryAUROC()


class Network(nn.Module):
    def __init__(self, c_in, c_hidden, c_out, p=0.3):
        super().__init__()
        torch.manual_seed(123)
        self.conv1 = SAGEConv(c_in, c_hidden, aggr='mean')
        self.conv2 = SAGEConv(c_hidden,3*c_hidden, aggr='mean')
        self.conv3 = SAGEConv(3*c_hidden, 2*c_hidden, aggr='mean')
        self.conv4 = SAGEConv(2*c_hidden, c_hidden, aggr='mean')

        # self.pool = SAGPooling(c_hidden)

        self.lin1 = nn.Linear(c_hidden, 4*c_out)
        self.lin2 = nn.Linear(4*c_out, c_out)
        self.p = p

    def forward(self, x, edge_index, batch, is_train):
        x = self.conv1(x, edge_index)
        x = x.relu()

        x = self.conv2(x, edge_index)
```

```
        x = x.relu()

        x = self.conv3(x, edge_index)
        x = x.relu()

        x = self.conv4(x, edge_index)



        x = global_mean_pool(x, batch)

        # classifier

        x = F.dropout(x, p=self.p, training=is_train)
        x = self.lin1(x)

        x = F.dropout(x, p=self.p, training=is_train)
        x = self.lin2(x)

        return x


def evaluate(loader):
    model.eval()
    total_loss = 0.0
    correct = 0
    total_samples = 0
    all_preds = []
    all_labels = []

    with torch.no_grad():
        for batch in loader:
            batch.to(device)
            # print(batch.edge_index)
            pred = model(batch.x.float(), batch.edge_index, batch.batch, False)
            target = F.one_hot(batch.y, 2).float()
            loss = criterion(pred, target)
            total_loss += loss.item()

            pred_labels = torch.softmax(pred, -1).argmax(dim=-1)
            correct += (pred_labels == batch.y).sum().item()
            total_samples += len(batch.y)
            all_labels.append(batch.y)
            all_preds.append(pred_labels)

    pred = all_preds[0]
    label = all_labels[0]
```

```
    for p, l in zip(all_preds[1:], all_labels[1:]):
      pred = torch.cat([pred, p])
      label = torch.cat([label, l])

    return total_loss / len(loader), correct / total_samples, auroc(pred, label)

# Training loop
num_epochs = 50
embedding_dim = 64
model = Network(c_in=5, c_hidden=embedding_dim, c_out=2).to(device)
optimizer = optim.Adam(model.parameters(), lr=3e-4)
criterion = nn.BCEWithLogitsLoss()

for epoch in range(num_epochs):
    model.train()
    epoch_loss = 0

    for idx, batch in enumerate(train_loader):

        batch = batch.to(device)

        pred = model(batch.x.float(), batch.edge_index, batch.batch, True)
        target = F.one_hot(batch.y, 2).float()
        loss = criterion(pred, target)
        epoch_loss += loss.item()

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

    avg_train_loss = epoch_loss / len(train_loader)
    avg_val_loss, val_accuracy, val_auroc = evaluate(val_loader)

    print(f'Epoch {epoch + 1}/{num_epochs}, Train Loss: {avg_train_loss:.4f},␣
 ↪Val Loss: {avg_val_loss:.4f}, Val Accuracy: {val_accuracy:.4f}, Val AUROC:␣
 ↪{val_auroc:.4f}')
```

```
Epoch 1/50, Train Loss: 0.6614, Val Loss: 0.6176, Val Accuracy: 0.6796, Val
AUROC: 0.6809
Epoch 2/50, Train Loss: 0.6218, Val Loss: 0.6058, Val Accuracy: 0.6716, Val
AUROC: 0.6749
Epoch 3/50, Train Loss: 0.6071, Val Loss: 0.5873, Val Accuracy: 0.7136, Val
AUROC: 0.7143
Epoch 4/50, Train Loss: 0.6035, Val Loss: 0.5835, Val Accuracy: 0.7099, Val
AUROC: 0.7081
Epoch 5/50, Train Loss: 0.5970, Val Loss: 0.5874, Val Accuracy: 0.7154, Val
```

```
AUROC: 0.7134
Epoch 6/50, Train Loss: 0.5982, Val Loss: 0.5797, Val Accuracy: 0.7136, Val
AUROC: 0.7123
Epoch 7/50, Train Loss: 0.5988, Val Loss: 0.5819, Val Accuracy: 0.7179, Val
AUROC: 0.7165
Epoch 8/50, Train Loss: 0.5972, Val Loss: 0.5818, Val Accuracy: 0.7154, Val
AUROC: 0.7150
Epoch 9/50, Train Loss: 0.5964, Val Loss: 0.5774, Val Accuracy: 0.7198, Val
AUROC: 0.7193
Epoch 10/50, Train Loss: 0.5908, Val Loss: 0.5764, Val Accuracy: 0.7185, Val
AUROC: 0.7179
Epoch 11/50, Train Loss: 0.5922, Val Loss: 0.5772, Val Accuracy: 0.7142, Val
AUROC: 0.7129
Epoch 12/50, Train Loss: 0.5911, Val Loss: 0.5788, Val Accuracy: 0.7185, Val
AUROC: 0.7165
Epoch 13/50, Train Loss: 0.5915, Val Loss: 0.5765, Val Accuracy: 0.7198, Val
AUROC: 0.7202
Epoch 14/50, Train Loss: 0.5902, Val Loss: 0.5848, Val Accuracy: 0.7136, Val
AUROC: 0.7152
Epoch 15/50, Train Loss: 0.5913, Val Loss: 0.5774, Val Accuracy: 0.7235, Val
AUROC: 0.7239
Epoch 16/50, Train Loss: 0.5914, Val Loss: 0.5857, Val Accuracy: 0.7117, Val
AUROC: 0.7134
Epoch 17/50, Train Loss: 0.5910, Val Loss: 0.5784, Val Accuracy: 0.7235, Val
AUROC: 0.7218
Epoch 18/50, Train Loss: 0.5891, Val Loss: 0.5737, Val Accuracy: 0.7228, Val
AUROC: 0.7229
Epoch 19/50, Train Loss: 0.5907, Val Loss: 0.5722, Val Accuracy: 0.7185, Val
AUROC: 0.7176
Epoch 20/50, Train Loss: 0.5881, Val Loss: 0.5729, Val Accuracy: 0.7191, Val
AUROC: 0.7179
Epoch 21/50, Train Loss: 0.5878, Val Loss: 0.5750, Val Accuracy: 0.7228, Val
AUROC: 0.7212
Epoch 22/50, Train Loss: 0.5874, Val Loss: 0.5737, Val Accuracy: 0.7290, Val
AUROC: 0.7285
Epoch 23/50, Train Loss: 0.5871, Val Loss: 0.5815, Val Accuracy: 0.7142, Val
AUROC: 0.7117
Epoch 24/50, Train Loss: 0.5862, Val Loss: 0.5714, Val Accuracy: 0.7204, Val
AUROC: 0.7201
Epoch 25/50, Train Loss: 0.5862, Val Loss: 0.5728, Val Accuracy: 0.7222, Val
AUROC: 0.7227
Epoch 26/50, Train Loss: 0.5853, Val Loss: 0.5700, Val Accuracy: 0.7191, Val
AUROC: 0.7189
Epoch 27/50, Train Loss: 0.5873, Val Loss: 0.5689, Val Accuracy: 0.7222, Val
AUROC: 0.7212
Epoch 28/50, Train Loss: 0.5859, Val Loss: 0.5709, Val Accuracy: 0.7247, Val
AUROC: 0.7235
Epoch 29/50, Train Loss: 0.5907, Val Loss: 0.5702, Val Accuracy: 0.7210, Val
```

```
AUROC: 0.7196
Epoch 30/50, Train Loss: 0.5871, Val Loss: 0.5690, Val Accuracy: 0.7191, Val
AUROC: 0.7186
Epoch 31/50, Train Loss: 0.5860, Val Loss: 0.5742, Val Accuracy: 0.7241, Val
AUROC: 0.7227
Epoch 32/50, Train Loss: 0.5864, Val Loss: 0.5702, Val Accuracy: 0.7272, Val
AUROC: 0.7266
Epoch 33/50, Train Loss: 0.5848, Val Loss: 0.5683, Val Accuracy: 0.7290, Val
AUROC: 0.7279
Epoch 34/50, Train Loss: 0.5867, Val Loss: 0.5708, Val Accuracy: 0.7247, Val
AUROC: 0.7233
Epoch 35/50, Train Loss: 0.5839, Val Loss: 0.5757, Val Accuracy: 0.7185, Val
AUROC: 0.7194
Epoch 36/50, Train Loss: 0.5846, Val Loss: 0.5700, Val Accuracy: 0.7198, Val
AUROC: 0.7189
Epoch 37/50, Train Loss: 0.5854, Val Loss: 0.5772, Val Accuracy: 0.7265, Val
AUROC: 0.7257
Epoch 38/50, Train Loss: 0.5834, Val Loss: 0.5700, Val Accuracy: 0.7204, Val
AUROC: 0.7204
Epoch 39/50, Train Loss: 0.5855, Val Loss: 0.5707, Val Accuracy: 0.7265, Val
AUROC: 0.7264
Epoch 40/50, Train Loss: 0.5846, Val Loss: 0.5674, Val Accuracy: 0.7265, Val
AUROC: 0.7256
Epoch 41/50, Train Loss: 0.5834, Val Loss: 0.5676, Val Accuracy: 0.7259, Val
AUROC: 0.7249
Epoch 42/50, Train Loss: 0.5872, Val Loss: 0.5739, Val Accuracy: 0.7210, Val
AUROC: 0.7204
Epoch 43/50, Train Loss: 0.5837, Val Loss: 0.5696, Val Accuracy: 0.7247, Val
AUROC: 0.7248
Epoch 44/50, Train Loss: 0.5855, Val Loss: 0.5688, Val Accuracy: 0.7259, Val
AUROC: 0.7261
Epoch 45/50, Train Loss: 0.5859, Val Loss: 0.5711, Val Accuracy: 0.7204, Val
AUROC: 0.7184
Epoch 46/50, Train Loss: 0.5841, Val Loss: 0.5697, Val Accuracy: 0.7259, Val
AUROC: 0.7264
Epoch 47/50, Train Loss: 0.5845, Val Loss: 0.5666, Val Accuracy: 0.7278, Val
AUROC: 0.7268
Epoch 48/50, Train Loss: 0.5830, Val Loss: 0.5729, Val Accuracy: 0.7210, Val
AUROC: 0.7205
Epoch 49/50, Train Loss: 0.5826, Val Loss: 0.5708, Val Accuracy: 0.7259, Val
AUROC: 0.7264
Epoch 50/50, Train Loss: 0.5831, Val Loss: 0.5683, Val Accuracy: 0.7222, Val
AUROC: 0.7222
```

[20]:
```
# Testing
test_loss, test_accuracy, test_auroc = evaluate(test_loader)
```

```
print(f'Test Loss: {test_loss:.4f}, Test Accuracy: {100*test_accuracy:.4f}%,␣
  ↪Test AUROC: {test_auroc:.4f}')
```

```
Test Loss: 0.5703, Test Accuracy: 72.3333%, Test AUROC: 0.7239
```

```
[ ]:
```