Московский государственный технический университет им. Н.Э. Баумана
Кафедра «Системы обработки информации и управления»

Лабораторная работа №6
по дисциплине
«Методы машинного обучения»

Выполнил:
студент группы ИУ5-22М
ЧжаоЛян

Москва — 2022 г.

Установка библиотек, выгрузка исходных датасетов

In [ ]:
```
# Slow method of installing pytorch geometric
# !pip install torch_geometric
# !pip install torch_sparse
# !pip install torch_scatter

# Install pytorch geometric
!pip install torch-sparse -f https://pytorch-geometric.com/whl/torch-1.11.0%2Bcu113.html
!pip install torch-cluster -f https://pytorch-geometric.com/whl/torch-1.11.0%2Bcu113.html
!pip install torch-spline-conv -f https://pytorch-geometric.com/whl/torch-1.11.0%2Bcu113.html
!pip install torch-geometric -f https://pytorch-geometric.com/whl/torch-1.11.0%2Bcu113.html
!pip install torch-scatter==2.0.8 -f https://data.pyg.org/whl/torch-1.11.0%2Bcu113.html
```

```
Looking in links: https://pytorch-geometric.com/whl/torch-1.11.0%2Bcu113.html
Requirement already satisfied: torch-sparse in /usr/local/lib/python3.7/dist-packages (0.6.13)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from torch-sparse) (1.4.1)
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib/python3.7/dist-packages (from scipy->torch-spar
se) (1.21.6)
Looking in links: https://pytorch-geometric.com/whl/torch-1.11.0%2Bcu113.html
Requirement already satisfied: torch-cluster in /usr/local/lib/python3.7/dist-packages (1.6.0)
Looking in links: https://pytorch-geometric.com/whl/torch-1.11.0%2Bcu113.html
Requirement already satisfied: torch-spline-conv in /usr/local/lib/python3.7/dist-packages (1.2.1)
Looking in links: https://pytorch-geometric.com/whl/torch-1.11.0%2Bcu113.html
Requirement already satisfied: torch-geometric in /usr/local/lib/python3.7/dist-packages (2.0.4)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from torch-geometric) (4.64.0)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from torch-geometric) (2.2
3.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from torch-geometric) (1.21.6)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from torch-geometric) (1.4.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from torch-geometric) (1.3.5)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (from torch-geometric)
(1.0.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.7/dist-packages (from torch-geometric) (2.11.
3)
Requirement already satisfied: pyparsing in /usr/local/lib/python3.7/dist-packages (from torch-geometric) (3.
0.9)
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/dist-packages (from jinja2->torch-
geometric) (2.0.1)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas->torch-geom
etric) (2022.1)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas->
torch-geometric) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3
->pandas->torch-geometric) (1.15.0)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->tor
ch-geometric) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->torch-ge
ometric) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests->to
rch-geometric) (2021.10.8)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packag
es (from requests->torch-geometric) (1.24.3)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-learn->torc
h-geometric) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-lea
rn->torch-geometric) (3.1.0)
```

In [ ]:
```
import numpy as np
import pandas as pd
import pickle
import csv
import os

from sklearn.preprocessing import LabelEncoder

import torch

# PyG - PyTorch Geometric
from torch_geometric.data import Data, DataLoader, InMemoryDataset

from tqdm import tqdm


RANDOM_SEED =  30#@param { type: "integer" }
BASE_DIR = '/content/'  #@param { type: "string" }
np.random.seed(RANDOM_SEED)
```

In [ ]:
```
# Check if CUDA is available for colab
torch.cuda.is_available
```

Out[3]:
```
<function torch.cuda.is_available>
```

In [ ]:
```
# Unpack files from zip-file
import zipfile
with zipfile.ZipFile(BASE_DIR + 'yoochoose-data-lite.zip', 'r') as zip_ref:
    zip_ref.extractall(BASE_DIR)
```

Анализ исходных данных

In [ ]:
```
# Read dataset of items in store
df = pd.read_csv(BASE_DIR + 'yoochoose-clicks-1ite.dat')
df.columns = ['session_id', 'timestamp', 'item_id', 'category']
df.head()
```

Out[32]:

	session_id	timestamp	item_id	category
0	3153234	2014-05-25T18:44:29.915Z	214839607.0	0.0
1	3153234	2014-05-25T18:44:31.938Z	214839607.0	0.0
2	3153234	2014-05-25T18:46:12.851Z	214836530.0	0.0
3	3153234	2014-05-25T18:46:14.412Z	214836530.0	0.0
4	3153232	2014-05-20T13:33:59.293Z	214800264.0	0.0

In [ ]:
```
# Read dataset of purchases
buy_df = pd.read_csv(BASE_DIR + 'yoochoose-buys-1ite.dat')
buy_df.columns = ['session_id', 'timestamp', 'item_id', 'price', 'quantity']
buy_df.head()
```

Out[6]:

	session_id	timestamp	item_id	price	quantity
0	420374	2014-04-06T18:44:58.314Z	214537888	12462	1
1	420374	2014-04-06T18:44:58.325Z	214537850	10471	1
2	489758	2014-04-06T09:59:52.422Z	214826955	1360	2
3	489758	2014-04-06T09:59:52.476Z	214826715	732	2
4	489758	2014-04-06T09:59:52.578Z	214827026	1046	1

In [ ]:
```
# Filter out item session with length < 2
df['valid_session'] = df.session_id.map(df.groupby('session_id')['item_id'].size() > 2)
df = df.loc[df.valid_session].drop('valid_session',axis=1)
df.nunique()
```

Out[7]:
```
session_id     4262
timestamp     22792
item_id        5084
category          1
dtype: int64
```

In [ ]:
```
# Randomly sample a couple of them
NUM_SESSIONS =  4000#@param { type: "integer" }
sampled_session_id = np.random.choice(df.session_id.unique(), NUM_SESSIONS, replace=False)
df = df.loc[df.session_id.isin(sampled_session_id)]
df.nunique()
```

Out[8]:
```
session_id     4000
timestamp     21300
item_id        4873
category          1
dtype: int64
```

In [ ]:
```
# Average length of session
df.groupby('session_id')['item_id'].size().mean()
```

Out[9]: 5.3255

In [ ]:
```
# Encode item and category id in item dataset so that ids will be in range (0,len(df.item.unique()))
item_encoder = LabelEncoder()
category_encoder = LabelEncoder()
df['item_id'] = item_encoder.fit_transform(df.item_id)
df['category']= category_encoder.fit_transform(df.category.apply(str))
df.head()
```

Out[37]:

	session_id	timestamp	item_id	category
0	3153234	2014-05-25T18:44:29.915Z	4567	0
1	3153234	2014-05-25T18:44:31.938Z	4567	0
2	3153234	2014-05-25T18:46:12.851Z	4316	0
3	3153234	2014-05-25T18:46:14.412Z	4316	0
4	3153232	2014-05-20T13:33:59.293Z	3661	0

In [ ]:
```
# Encode item and category id in purchase dataset
buy_df = buy_df.loc[buy_df.session_id.isin(df.session_id)]
buy_df['item_id'] = item_encoder.transform(buy_df.item_id)
buy_df.head()
```

Out[11]:

	session_id	timestamp	item_id	price	quantity
59565	3542963	2014-05-22T18:25:58.856Z	4131	0	0
59571	3542969	2014-05-23T19:54:15.427Z	2598	0	0
59580	3206154	2014-05-25T14:54:13.961Z	1329	0	0
59603	3206236	2014-05-23T10:54:30.684Z	1698	0	0
59617	3542791	2014-05-24T15:59:27.184Z	4775	0	0

In [ ]:
```
# Get item dictionary with grouping by session
buy_item_dict = dict(buy_df.groupby('session_id')['item_id'].apply(list))
buy_item_dict
```

Out[12]:
```
{3178534: [4773, 1502],
 3178853: [1240],
 3179241: [4709, 724, 3202],
 3179404: [3006],
 3179513: [2242],
 3179877: [3053, 3963, 3688, 4249, 3667, 4131, 4376, 3902, 964],
 3180527: [3582, 1577, 3574],
 3180596: [3915, 3916],
 3180682: [854],
 3180734: [3884, 3887, 3899],
 3180929: [2446],
 3181216: [4860],
 3181307: [3640, 3798],
 3181372: [4031, 4031, 4136],
 3181542: [3816],
 3181674: [4023, 2589],
 3181748: [3006],
 3182771: [4080, 4111, 4028, 4249],
 3182816: [4136, 3941, 3573],
```

Сборка выборки для обучения

In [ ]:
```
# Transform df into tensor data
def transform_dataset(df, buy_item_dict):
    data_list = []

    # Group by session
    grouped = df.groupby('session_id')
    for session_id, group in tqdm(grouped):
        le = LabelEncoder()
        sess_item_id = le.fit_transform(group.item_id)
        group = group.reset_index(drop=True)
        group['sess_item_id'] = sess_item_id

        #get input features
        node_features = group.loc[group.session_id==session_id,
                                  ['sess_item_id','item_id','category']].sort_values('sess_item_id')[['item_id','category']].drop_duplicates(
        node_features = torch.LongTensor(node_features).unsqueeze(1)
        target_nodes = group.sess_item_id.values[1:]
        source_nodes = group.sess_item_id.values[:-1]

        edge_index = torch.tensor([source_nodes,
                                   target_nodes], dtype=torch.long)
        x = node_features

        #get result
        if session_id in buy_item_dict:
            positive_indices = le.transform(buy_item_dict[session_id])
            label = np.zeros(len(node_features))
            label[positive_indices] = 1
        else:
            label = [0] * len(node_features)

        y = torch.FloatTensor(label)

        data = Data(x=x, edge_index=edge_index, y=y)

        data_list.append(data)

    return data_list

# Pytorch class for creating datasets
class YooChooseDataset(InMemoryDataset):
    def __init__(self, root, transform=None, pre_transform=None):
        super(YooChooseDataset, self).__init__(root, transform, pre_transform)
        self.data, self.slices = torch.load(self.processed_paths[0])

    @property
    def raw_file_names(self):
        return []

    @property
    def processed_file_names(self):
        return [BASE_DIR+'yoochoose_click_binary_100000_sess.dataset']

    def download(self):
        pass

    def process(self):
        data_list = transform_dataset(df, buy_item_dict)

        data, slices = self.collate(data_list)
        torch.save((data, slices), self.processed_paths[0])
```

In [ ]:
```
# Prepare dataset
dataset = YooChooseDataset('./')
```

Разделение выборки

In [ ]:
```
# train_test_split
dataset = dataset.shuffle()
one_tenth_length = int(len(dataset) * 0.1)
train_dataset = dataset[:one_tenth_length * 8]
val_dataset = dataset[one_tenth_length*8:one_tenth_length * 9]
test_dataset = dataset[one_tenth_length*9:]
len(train_dataset), len(val_dataset), len(test_dataset)
```

Out[15]:
```
(1600, 200, 200)
```

In [ ]:
```
# Load dataset into PyG loaders
batch_size= 512
train_loader = DataLoader(train_dataset, batch_size=batch_size)
val_loader = DataLoader(val_dataset, batch_size=batch_size)
test_loader = DataLoader(test_dataset, batch_size=batch_size)
```

```
/usr/local/lib/python3.7/dist-packages/torch_geometric/deprecation.py:12: UserWarning:
'data.DataLoader' is deprecated, use 'loader.DataLoader' instead
  warnings.warn(out)
```

In [ ]:
```
# Load dataset into PyG loaders
num_items = df.item_id.max() +1
num_categories = df.category.max()+1
num_items , num_categories
```

Out[17]:
```
(4873, 1)
```

Настройка модели для обучения

In [ ]:

```
embed_dim = 128
from torch_geometric.nn import GraphConv, TopKPooling, GatedGraphConv, SAGEConv, SGConv
from torch_geometric.nn import global_mean_pool as gap, global_max_pool as gmp
import torch.nn.functional as F

class Net(torch.nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        # Model Structure
        self.conv1 = GraphConv(embed_dim * 2, 128)
        self.pool1 = TopKPooling(128, ratio=0.9)
        self.conv2 = GraphConv(128, 128)
        self.pool2 = TopKPooling(128, ratio=0.9)
        self.conv3 = GraphConv(128, 128)
        self.pool3 = TopKPooling(128, ratio=0.9)
        self.item_embedding = torch.nn.Embedding(num_embeddings=num_items, embedding_dim=embed_dim)
        self.category_embedding = torch.nn.Embedding(num_embeddings=num_categories, embedding_dim=embed_dim)
        self.lin1 = torch.nn.Linear(256, 256)
        self.lin2 = torch.nn.Linear(256, 128)
        self.bn1 = torch.nn.BatchNorm1d(128)
        self.bn2 = torch.nn.BatchNorm1d(64)
        self.act1 = torch.nn.ReLU()
        self.act2 = torch.nn.ReLU()

    # Forward step of a model
    def forward(self, data):
        x, edge_index, batch = data.x, data.edge_index, data.batch

        item_id = x[:, :, 0]
        category = x[:, :, 1]


        emb_item = self.item_embedding(item_id).squeeze(1)
        emb_category = self.category_embedding(category).squeeze(1)

        x = torch.cat([emb_item, emb_category], dim=1)
        # print(x.shape)
        x = F.relu(self.conv1(x, edge_index))
        # print(x.shape)
        r = self.pool1(x, edge_index, None, batch)
        # print(r)
        x, edge_index, _, batch, _, _ = self.pool1(x, edge_index, None, batch)
        x1 = torch.cat([gmp(x, batch), gap(x, batch)], dim=1)

        x = F.relu(self.conv2(x, edge_index))

        x, edge_index, _, batch, _, _ = self.pool2(x, edge_index, None, batch)
        x2 = torch.cat([gmp(x, batch), gap(x, batch)], dim=1)

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

        x, edge_index, _, batch, _, _ = self.pool3(x, edge_index, None, batch)
        x3 = torch.cat([gmp(x, batch), gap(x, batch)], dim=1)

        x = x1 + x2 + x3

        x = self.lin1(x)
        x = self.act1(x)
        x = self.lin2(x)
        x = F.dropout(x, p=0.5, training=self.training)
        x = self.act2(x)

        outputs = []
        for i in range(x.size(0)):
            output = torch.matmul(emb_item[data.batch == i], x[i,:])

            outputs.append(output)

        x = torch.cat(outputs, dim=0)
        x = torch.sigmoid(x)

        return x
```

Обучение нейронной сверточной сети

In [ ]:
```
# Enable CUDA computing
device = torch.device('cuda')
model = Net().to(device)
# Choose optimizer and criterion for learning
optimizer = torch.optim.Adam(model.parameters(), lr=0.002)
crit = torch.nn.BCELoss()
```

In [ ]:
```
# Train function
def train():
    model.train()

    loss_all = 0
    for data in train_loader:
        data = data.to(device)
        optimizer.zero_grad()
        output = model(data)

        label = data.y.to(device)
        loss = crit(output, label)
        loss.backward()
        loss_all += data.num_graphs * loss.item()
        optimizer.step()
    return loss_all / len(train_dataset)
```

In [ ]:
```
# Evaluate result of a model
from sklearn.metrics import roc_auc_score
def evaluate(loader):
    model.eval()

    predictions = []
    labels = []

    with torch.no_grad():
        for data in loader:

            data = data.to(device)
            pred = model(data).detach().cpu().numpy()

            label = data.y.detach().cpu().numpy()
            predictions.append(pred)
            labels.append(label)

    predictions = np.hstack(predictions)
    labels = np.hstack(labels)

    return roc_auc_score(labels, predictions)
```

In [ ]:
```
# Train a model
NUM_EPOCHS =   10#@param { type: "integer" }
for epoch in tqdm(range(NUM_EPOCHS)):
    loss = train()
    train_acc = evaluate(train_loader)
    val_acc = evaluate(val_loader)
    test_acc = evaluate(test_loader)
    print('Epoch: {:03d}, Loss: {:.5f}, Train Auc: {:.5f}, Val Auc: {:.5f}, Test Auc: {:.5f}'.
          format(epoch, loss, train_acc, val_acc, test_acc))
```

```
 10%|█         | 1/10 [00:03<00:29,  3.30s/it]
Epoch: 000, Loss: 0.74019, Train Auc: 0.50109, Val Auc: 0.50688, Test Auc: 0.53295
 20%|██        | 2/10 [00:07<00:28,  3.57s/it]
Epoch: 001, Loss: 0.77293, Train Auc: 0.51751, Val Auc: 0.41170, Test Auc: 0.46785
 30%|███       | 3/10 [00:10<00:25,  3.62s/it]
Epoch: 002, Loss: 0.71779, Train Auc: 0.52556, Val Auc: 0.49923, Test Auc: 0.54498
 40%|████      | 4/10 [00:14<00:21,  3.55s/it]
Epoch: 003, Loss: 0.66214, Train Auc: 0.52141, Val Auc: 0.53117, Test Auc: 0.52894
 50%|█████     | 5/10 [00:15<00:14,  2.86s/it]
Epoch: 004, Loss: 0.64963, Train Auc: 0.55996, Val Auc: 0.47408, Test Auc: 0.50235
 60%|██████    | 6/10 [00:17<00:09,  2.40s/it]
```

Проверка результата с помощью примеров

In [ ]:
```
# Подход №1 - из датасета
evaluate(DataLoader(test_dataset[25:50], batch_size=10))
```

```
/usr/local/lib/python3.7/dist-packages/torch_geometric/deprecation.py:12: UserWarning: 'data.DataLoader' is d
eprecated, use 'loader.DataLoader' instead
  warnings.warn(out)
```

Out[24]: 0.704861111111111

In [ ]:
```
# Подход №2 - через создание сессии покупок
test_df = pd.DataFrame([
      [-1, 15219, 0],
      [-1, 15431, 0],
      [-1, 14371, 0],
      [-1, 15745, 0],
      [-2, 14594, 0],
      [-2, 16972, 11],
      [-2, 16943, 0],
      [-3, 17284, 0]
], columns=['session_id', 'item_id', 'category'])

test_data = transform_dataset(test_df, buy_item_dict)
test_data = DataLoader(test_data, batch_size=1)

with torch.no_grad():
    model.eval()
    for data in test_data:
        data = data.to(device)
        pred = model(data).detach().cpu().numpy()

        print(data, pred)
```

Список литературы

[1] Гапанюк Ю. Е. Лабораторная работа «Подготовка обучающей и тестовой выборки,
кросс-валидация и подбор гиперпараметров на примере метода ближайших соседей»
[Электронный ресурс] // GitHub. — 2019. — Режим доступа:
https://github.com/
ugapanyuk/ml_course/wiki/LAB_KNN (дата обращения: 05.04.2019).
[2] Team The IPython Development. IPython 7.3.0 Documentation [Electronic resource] //
Read the Docs. — 2019. — Access mode: https://ipython.readthedocs.io/en/
stable/ (online; accessed: 20.02.2019).

[3] Waskom M. seaborn 0.9.0 documentation [Electronic resource] // PyData. — 2018. —

Access mode: https://seaborn.pydata.org/ (online; accessed: 20.02.2019).
[4] pandas 0.24.1 documentation [Electronic resource] // PyData. — 2019. — Access mode:
http://pandas.pydata.org/pandas-docs/stable/ (online; accessed: 20.02.2019).
[5] dronio. Solar Radiation Prediction [Electronic resource] // Kaggle. — 2017. — Access
mode: https://www.kaggle.com/dronio/SolarEnergy (online; accessed: 18.02.2019).
[6] Chrétien M. Convert datetime.time to seconds [Electronic resource] // Stack Overflow.
— 2017. — Access mode: https://stackoverflow.com/a/44823381 (online; accessed:
20.02.2019).
[7] scikit-learn 0.20.3 documentation [Electronic resource]. — 2019. — Access mode: https:
//scikit-learn.org/ (online; accessed: 05.04.2019).