#Отчет по лабораторной работе №6 по ММО Кудрявцев С.Д. ИУ5-22М

Лабораторная работа №6:

"Разработка системы предсказания поведения на основании графовых моделей"

Цель: обучение работе с графовым типом данных и графовыми нейронными сетями.

Задача: подготовить графовый датасет из базы данных о покупках и построить модель предсказания совершения покупки.

Графовые нейронные сети

Графовые нейронные сети - тип нейронной сети, которая напрямую работает со структурой графа. Типичным применениями GNN являются:

- Классификация узлов;
- Предсказание связей;
- Графовая классификация;
- Распознавание движений;
- Рекомендательные системы.

В данной лабораторной работе будет происходить работа над **графовыми сверточными сетями**. Отличаются они от сверточных нейронных сетей нефиксированной структурой, функция свертки не является .

Подробнее можно прочитать тут: https://towardsdatascience.com/understanding-graph-convolutional-networks-for-node-classification-a2bfdb7aba7b

Тут можно почитать современные подходы к использованию графовых сверточных сетей https://paperswithcode.com/method/gcn

---

Датасет

В качестве базы данных предлагаем использовать датасет о покупках пользователей в одном магазине товаров RecSys Challenge 2015 (https://www.kaggle.com/datasets/chadgostopp/recsys-challenge-2015).

Скачать датасет можно отсюда: https://drive.google.com/drive/folders/1gtAeXPTj-c0RwVOKreMrZ3bfSmCwl2y-?usp=sharing (lite-версия является облегченной версией исходного датасета, рекомендуем использовать её)

Также рекомендуем загружать данные в виде архива и распаковывать через пакет zipfile или/и скачивать датасет в собственный Google Drive и примонтировать его в колаб.

Установка библиотек, выгрузка исходных датасетов

```
# 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 indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Looking in links: https://pytorch-geometric.com/whl/torch-1.11.0%2Bcu113.html
Collecting torch-sparse
  Downloading https://data.pyg.org/whl/torch-1.11.0%2Bcu113/torch_sparse-0.6.13-cp37-cp37m-linux_x86_64.whl (3.5 MB)
ent 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-sparse) (1.21.6)
Installing collected packages: torch-sparse
Successfully installed torch-sparse-0.6.13
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Looking in links: https://pytorch-geometric.com/whl/torch-1.11.0%2Bcu113.html
Collecting torch-cluster
  Downloading
```

```
https://data.pyg.org/whl/torch-1.11.0%2Bcu113/torch_cluster-1.6.0-
cp37-cp37m-linux_x86_64.whl (2.5 MB)
ple, https://us-python.pkg.dev/colab-wheels/public/simple/
Looking in links: https://pytorch-geometric.com/whl/torch-
1.11.0%2Bcu113.html
Collecting torch-spline-conv
  Downloading
https://data.pyg.org/whl/torch-1.11.0%2Bcu113/torch_spline_conv-1.2.1-
cp37-cp37m-linux_x86_64.whl (750 kB)
ple, https://us-python.pkg.dev/colab-wheels/public/simple/
Looking in links: https://pytorch-geometric.com/whl/torch-
1.11.0%2Bcu113.html
Collecting torch-geometric
  Downloading torch_geometric-2.0.4.tar.gz (407 kB)
ent already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages
(from torch-geometric) (4.64.0)
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: jinja2 in
/usr/local/lib/python3.7/dist-packages (from torch-geometric) (2.11.3)
Requirement already satisfied: requests in
/usr/local/lib/python3.7/dist-packages (from torch-geometric) (2.23.0)
Requirement already satisfied: pyparsing in
/usr/local/lib/python3.7/dist-packages (from torch-geometric) (3.0.9)
Requirement already satisfied: scikit-learn in
/usr/local/lib/python3.7/dist-packages (from torch-geometric) (1.0.2)
Requirement already satisfied: MarkupSafe>=0.23 in
/usr/local/lib/python3.7/dist-packages (from jinja2->torch-geometric)
(2.0.1)
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: pytz>=2017.3 in
/usr/local/lib/python3.7/dist-packages (from pandas->torch-geometric)
(2022.1)
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: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1
in /usr/local/lib/python3.7/dist-packages (from requests->torch-
geometric) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in
/usr/local/lib/python3.7/dist-packages (from requests->torch-
geometric) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from requests->torch-
```

```
geometric) (2022.5.18.1)
Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from requests->torch-
geometric) (3.0.4)
Requirement already satisfied: joblib>=0.11 in
/usr/local/lib/python3.7/dist-packages (from scikit-learn->torch-
geometric) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from scikit-learn->torch-
geometric) (3.1.0)
Building wheels for collected packages: torch-geometric
  Building wheel for torch-geometric (setup.py) ... etric:
filename=torch_geometric-2.0.4-py3-none-any.whl size=616603
sha256=b63dbf8ff281ac0c516c8201c85775b8a305716259b811f8e13b5967582a40f
f
  Stored in directory:
/root/.cache/pip/wheels/18/a6/a4/ca18c3051fcead866fe7b85700ee2240d8835
62a1bc70ce421
Successfully built torch-geometric
Installing collected packages: torch-geometric
Successfully installed torch-geometric-2.0.4
Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/
Looking in links: https://data.pyg.org/whl/torch-1.11.0%2Bcu113.html
Collecting torch-scatter==2.0.8
  Downloading torch_scatter-2.0.8.tar.gz (21 kB)
Building wheels for collected packages: torch-scatter
  Building wheel for torch-scatter (setup.py) ... e=torch_scatter-
2.0.8-cp37-cp37m-linux_x86_64.whl size=3222016
sha256=4728b436f34cf2a456ff7aab33942f5f2ea5a0f061cf397cd619d5f9921fcfc
4
  Stored in directory:
/root/.cache/pip/wheels/96/e4/4e/2bcc6de6a801960aedbca43f7106d268f766c
3f9f8ab49b3a5
Successfully built torch-scatter
Installing collected packages: torch-scatter
Successfully installed torch-scatter-2.0.8
```

```
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 = 42 #@param { type: "integer" }
BASE_DIR = '/content/' #@param { type: "string" }
np.random.seed(RANDOM_SEED)

# Check if CUDA is available for colab
torch.cuda.is_available

<function torch.cuda.is_available>

# 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

```
# Read dataset of items in store
df = pd.read_csv(BASE_DIR + 'yoochoose-clicks-lite.dat')
df.head()

/usr/local/lib/python3.7/dist-packages/IPython/core/
interactiveshell.py:2882: DtypeWarning: Columns (3) have mixed
types.Specify dtype option on import or set low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

	session_id	timestamp	item_id	category
0	9	2014-04-06T11:26:24.127Z	214576500	0
1	9	2014-04-06T11:28:54.654Z	214576500	0
2	9	2014-04-06T11:29:13.479Z	214576500	0
3	19	2014-04-01T20:52:12.357Z	214561790	0
4	19	2014-04-01T20:52:13.758Z	214561790	0

```
# Read dataset of purchases
buy_df = pd.read_csv(BASE_DIR + 'yoochoose-buys-lite.dat')
buy_df.head()
```

	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

```
# 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()
```

```
session_id    1000000
timestamp     5557758
item_id         37644
category          275
dtype: int64
```

```
# Randomly sample a couple of them
NUM_SESSIONS = 50000 #@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()
```

```
session_id     50000
timestamp     278442
item_id        18461
category         110
dtype: int64
```

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

```
5.56902
```

```
# 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()
```

```
     session_id                 timestamp  item_id  category
0             9  2014-04-06T11:26:24.127Z     3496         0
1             9  2014-04-06T11:28:54.654Z     3496         0
2             9  2014-04-06T11:29:13.479Z     3496         0
102         171  2014-04-03T17:45:25.575Z    10049         0
103         171  2014-04-03T17:45:33.177Z    10137         0
```

```
# 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()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
```

```
  This is separate from the ipykernel package so we can avoid doing
imports until
```

	session_id	timestamp	item_id	price	quantity
46	489491	2014-04-06T12:41:34.047Z	12633	1046	4
47	489491	2014-04-06T12:41:34.091Z	12634	627	2
61	70353	2014-04-06T10:55:06.086Z	14345	41783	1
62	489671	2014-04-03T15:48:37.392Z	12489	4188	1
63	489671	2014-04-03T15:59:35.495Z	12489	4188	1

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

```
{714: [14720, 14915, 14917, 3089],
 6016: [15154],
 9797: [12459, 11831],
 9862: [13621],
 10457: [10079, 2951],
 10587: [11764],
 10678: [6310, 3914],
 13476: [13631, 12881, 12878, 12880, 12852],
 16953: [2883, 7739],
 19029: [8276, 2171, 10385, 11419],
 19958: [10059, 10059],
 23548: [11236],
 24439: [12506, 12497],
 28709: [4037],
 29647: [12830, 12827],
 33907: [2480, 6012],
 34541: [627, 12827],
 36548: [14720, 14916],
 38019: [11222, 11227],
 38261: [3447],
 41333: [11839, 12826, 11839],
 41598: [10712, 10711, 2034, 10713],
 43834: [11203, 11203],
 44153: [15075, 15088],
 44813: [12819, 12634],
 48974: [7428, 12774],
 49886: [2373, 11207, 11221, 10625],
 54961: [1965, 11360, 12812, 7803],
 55877: [4804],
 62553: [12793],
 64802: [11317],
 69277: [7933],
 70353: [14345],
 71832: [11839, 12362],
 73271: [11236, 11236, 11237],
 74083: [12864],
```

```
79937: [11884, 11825, 11825, 11884],
87673: [12608],
88198: [11201, 11201],
88723: [11236, 11236],
89393: [11236],
91903: [11200],
92224: [255],
93282: [14842,
 5737,
 5733,
 421,
 4275,
 4489,
 7043,
 7047,
 5739,
 4498,
 7048,
 4508,
 7140,
 2065,
 2071,
 2066],
100952: [9849, 9849],
103133: [11228],
104659: [11424, 4839, 8471],
108344: [13479, 11221, 11208],
110788: [1187],
114461: [2376, 12826],
116004: [9730],
116373: [9868],
122179: [316],
125163: [3548, 5766],
126769: [3318, 12770, 12769],
127311: [12831, 12552, 12846, 12342, 12772, 12773],
128359: [4483],
128562: [11476, 12787, 12812, 12552],
132646: [5654],
136192: [12880, 12852, 12878],
138344: [4827],
142132: [627, 12767],
142902: [11226, 11226],
143246: [4791],
144241: [12632],
150664: [12825, 11333, 13046, 12839, 10484],
150723: [12621, 13444, 12619, 4145],
150926: [12812, 12787, 12813],
152571: [3943],
157247: [8285],
160843: [11792],
```

```
 164104: [12839, 12825, 12616, 9932],
 169179: [10715, 10714],
 171209: [12789],
 171554: [15058],
 172101: [11469, 8711, 8721, 12629],
 172324: [14720, 14915, 14917, 14916],
 175289: [12634],
 175897: [11223],
 177247: [12634, 12632],
 177641: [2248],
 182148: [13793, 12367],
 185409: [11200],
 187838: [3409],
 188339: [12630, 12788],
 188726: [11605],
 189138: [12812, 12787],
 192869: [12766],
 193483: [6333],
 197056: [13481, 13481, 13481],
 197889: [12557],
 198147: [11221],
 198939: [12557],
 201316: [12360],
 202037: [13479, 11227, 11202],
 203623: [8253],
 204676: [11236, 11236],
 205377: [12496, 12497, 12506, 12497, 12506, 12496],
 212573: [4254],
 214007: [12632],
 218862: [15058],
 219736: [12633],
 221904: [6044, 7972, 12619],
 222178: [9269],
 222293: [12881, 12852, 12880, 13631, 11658, 12878],
 223497: [12571, 12549, 7670, 12674, 7697, 12548, 12550, 8303, 12625,
8306],
 224769: [3914],
 224788: [4483],
 227657: [11226, 10534, 11249],
 229858: [14618, 576],
 232349: [179, 4804],
 232521: [12321, 12321],
 239123: [12677, 12632],
 239224: [4901, 4901],
 241053: [12813, 15582, 15554, 15560, 15249, 15446],
 242477: [12789, 12634, 1032, 12632],
 242636: [390],
 243838: [14989, 13981],
 245392: [7934, 10052, 12793],
 245427: [4484],
```

```
 245733: [12417, 12770, 12630],
 247706: [3718],
 248291: [11203, 11222],
 249438: [15015],
 251596: [12374],
 251777: [1835],
 256753: [11206],
 257992: [446, 11372, 12489, 9358],
 259584: [15304, 15303, 15064],
 260656: [16],
 270696: [7436, 1105, 7436, 1105],
 272839: [10292],
 276266: [9791],
 277534: [11521, 12832, 337],
 283099: [15058],
 284831: [11225],
 285344: [12506, 12503, 12497, 12557, 12506, 12497, 12503, 12557],
 287659: [8278],
 291433: [13666, 12620, 12618, 2971],
 294559: [8190],
 295229: [8293, 7924, 7908, 6489, 12548, 11740, 12550, 12832, 7919],
 295338: [12601],
 295684: [12766, 10480, 12443, 627, 12830],
 297264: [12789, 12811, 12788],
 298599: [12793, 7576],
 299953: [8681, 8820, 8785, 8791, 15101, 8820, 8681, 8785, 8791,
15101],
 300253: [1187],
 301426: [191, 12788],
 302057: [11202, 11202],
 304968: [3702, 1158, 231],
 305804: [11222],
 308539: [12619, 12388, 12677, 12393],
 314311: [9716, 1746, 11236, 11236],
 318261: [15058],
 319517: [12500, 12825, 12986, 8303],
 320068: [14174],
 321967: [12888, 10038],
 333733: [12864, 12776, 10998],
 336051: [14946, 14944],
 337309: [12864, 12766],
 341142: [2373, 12767, 12819],
 345242: [10049, 10059],
 349007: [12497, 12506],
 350491: [12633, 11839],
 357583: [12789],
 358303: [12843, 2370, 12456],
 363567: [3911, 3914],
 368121: [12505],
 374406: [12640, 12813],
```

379616: [10358, 8533],
381399: [13711],
383213: [340],
385363: [1392],
387298: [11205, 11205],
388957: [12825],
390967: [904],
391946: [8791, 15594, 15595],
394362: [12547],
397214: [14720],
400079: [12878, 13631, 12881, 12880, 10971, 12986, 8301, 12890],
400631: [11227],
404878: [11841],
406213: [14486],
407394: [12623],
408041: [12776, 12776],
408093: [3942],
410391: [5557],
411638: [14364, 14614, 14362, 14361],
413432: [14720],
416564: [974, 973],
417204: [12787, 12770, 12773, 12293, 12772, 12775, 12606],
417343: [13479],
419113: [13480],
424841: [12864, 12776],
425214: [14968],
427086: [11449, 11445, 12618, 13045, 12616],
430931: [12819, 12630],
435346: [11226],
435912: [11826, 11885],
438957: [14720],
440729: [12487, 4208, 12853, 12677, 12854, 3002],
441832: [13442, 14362],
445559: [2167, 12633, 1433, 12774],
446676: [4519],
446739: [3914, 11841],
448812: [3942],
458939: [11885, 11826],
460686: [4804, 4805],
463179: [11820, 9883],
463412: [11723, 12825],
467661: [12362, 4839, 12827, 11740],
470672: [11236],
472279: [11471],
473307: [2498, 3959, 7125],
473606: [11445],
476967: [12787, 12812, 12767, 2701, 12789],
476981: [8545],
477773: [306],
479172: [11824, 11885, 11883],

480299: [3548, 402, 3299],
483244: [8190],
483777: [8190],
484567: [9927, 9927],
487561: [11237, 11222],
488141: [3914],
488911: [3097, 7967],
489491: [12633, 12634],
489671: [12489, 12489, 12489],
491736: [12887, 12628],
492363: [1913],
492819: [12668],
497436: [12884, 12884],
499877: [15087, 15079],
501527: [12884, 5266, 14364, 14362, 12772],
502918: [12500],
505106: [12630, 12630],
506496: [12883, 12883, 12884, 2207],
510144: [904],
511682: [15058],
516839: [12655, 12641],
523179: [11713],
523616: [8999],
523856: [14720],
525857: [13052],
528754: [8284],
529859: [12853, 12854],
534599: [12621, 11445, 12846, 11349, 12599],
537107: [8394, 12771, 12769, 12557],
537548: [13895],
540604: [14363],
541089: [4951],
545092: [8721, 15511, 15079],
546013: [1854],
549136: [10292, 10306, 12890],
553203: [11221, 11221, 11227],
553306: [8664, 8253, 8253, 8664],
558932: [12014],
560902: [13051, 12882, 12890],
564282: [12887, 12888],
566017: [12495, 12495],
567827: [12833, 12890],
571197: [12883],
572453: [2157, 2173, 3157],
572511: [15017, 15164],
574467: [10982, 10982],
577249: [12645, 5680],
579243: [10607, 10477, 708],
582102: [338],
585099: [3690],

586406: [10755, 10753],
587411: [11227],
587832: [11222],
588458: [11229],
588787: [9068],
590151: [12828, 11756, 6660, 11349],
590697: [10152, 10152],
591052: [12507, 12507],
594087: [12290, 12657],
594661: [12046, 10511],
595793: [11201],
596653: [12678, 12831, 12833, 12828],
597733: [12630, 12633, 12632],
597994: [11182, 14357],
599287: [12773, 12771],
605941: [15094],
607649: [13445, 12617],
608719: [12641, 11349, 11740, 12488, 11477, 11662],
611642: [12210],
616894: [12828, 12548],
618656: [13631, 12878, 12852, 12880, 12881, 8301, 12848, 12986],
621736: [12633, 12633],
622232: [6164],
623548: [12990, 12630],
623654: [12640, 12640],
625397: [12644, 12639, 12639, 12644],
626218: [12771, 13723],
626573: [12828, 12846, 12831, 12678, 12831, 12828],
628053: [14878],
631077: [12645],
631366: [14363, 2884, 14356, 14364, 14362],
634152: [13444],
635213: [12986, 12986, 10971],
636986: [14750, 14748, 12655],
640533: [14720, 14916],
641813: [12645],
644397: [2121, 12805, 2121, 12805],
645702: [14943],
647687: [5967, 12381],
649002: [1956],
649628: [54],
654654: [5298, 3302],
655734: [12888, 1993, 12888, 1993],
664182: [12828, 5889, 4208],
664846: [12787, 13652],
667066: [10635],
671599: [12483, 12487],
672041: [12617, 12394, 11831],
677022: [2484, 904, 5936, 2373],
677912: [973, 4181, 2647],

678981: [4805],
679949: [4022, 4022],
681248: [12773, 12769, 13054, 12770, 14842, 13045],
684068: [12641, 12644, 15726, 12656, 12641, 12644, 12656, 15726],
684558: [12483, 12839, 11122, 12825],
685528: [8253, 8253],
688677: [1640, 4786],
694313: [11840],
694761: [12883, 12672, 12884],
695776: [8288],
696617: [12372, 2152],
697709: [13119],
699141: [9717, 8830],
699508: [10668],
705074: [337],
706698: [11089, 12846, 11349, 11839],
706699: [8518],
707079: [10944, 12879, 11729],
708779: [6175, 8474],
708928: [12633],
709489: [12889, 12557],
710244: [2106],
713189: [11223],
720422: [12986, 12848, 12655],
721932: [12446, 1119, 12789, 14663],
725051: [4856],
727601: [12496, 12496],
729226: [14720],
730308: [11236],
731793: [8195, 5950],
732911: [12640],
732971: [12639, 12640],
733843: [11515],
734488: [2395],
736806: [12645, 15726],
737843: [10059, 3570],
739826: [12675, 12675, 12597, 9936, 10039, 11616, 11451],
742357: [10477, 11467],
746527: [11517],
748854: [12640],
752348: [6923, 2211, 1348],
757116: [1022],
765684: [13631],
768353: [11236],
770326: [8732, 15315],
771664: [12640],
772446: [2371],
773194: [12630],
773778: [11204],
774249: [12668],

```
 775031: [8253, 15257, 15469, 15257, 15469, 8253],
 777389: [1308],
 778536: [12641],
 779433: [12816],
 779547: [2134, 2019, 2130],
 780062: [12640, 12640],
 780808: [12450, 12553, 9210, 12450, 12540],
 787042: [12634, 13117],
 787052: [11400, 12884],
 787246: [11201, 11201],
 792402: [11662],
 792918: [13060, 13010, 2378, 11772, 7219, 13060, 13010, 11772, 2378,
7219],
 793508: [5583],
 794318: [9903],
 795076: [12767, 12789, 11226],
 796793: [14364, 14362, 14614, 12678, 5889],
 798067: [4856],
 802356: [12640],
 809958: [193],
 811707: [12882, 12882],
 813496: [12877],
 813637: [12630, 12630, 12633, 12634],
 831707: [11263],
 831984: [12210],
 834009: [12630],
 837024: [12816],
 837333: [1429, 13052],
 837678: [12880, 3221],
 840161: [14720, 14917, 14918, 14916, 14720, 14917, 14918, 14916],
 841118: [8190, 12642],
 845652: [6639],
 846917: [12770, 12884],
 848304: [12639, 12640, 13051, 12990],
 849249: [12839, 12825, 12637],
 849736: [4196, 9979],
 851413: [12639, 12853],
 851564: [12678, 14778, 12983, 12548, 12877],
 857202: [12640, 12641],
 859203: [4942, 4942],
 861731: [6012],
 861928: [14297],
 864679: [11298, 12678, 11194],
 864879: [12645],
 865703: [12645, 12639, 12640],
 870926: [12640, 12639],
 872079: [12644],
 872332: [12852],
 873289: [12656, 12642, 13009],
 873586: [3640, 12391],
```

```
874572: [1187, 1187],
883819: [14356, 8627, 14362, 14614, 14364],
883909: [2165],
885063: [12839, 12825],
889477: [12882, 12801, 12801, 12882],
890253: [9192, 9192],
890273: [12770, 12773],
890716: [13052, 12887],
891871: [12848, 12986, 14362, 11165, 15726, 2341],
894636: [13051, 5266, 13051, 5266, 13051, 5266],
898714: [8347],
899332: [11808, 12775, 12546, 11332, 7454, 11322],
899853: [10606, 7227, 8421],
908233: [14597, 8637, 13018],
920332: [12326],
925381: [11173, 11172],
928356: [14720],
933423: [8608],
934768: [12451],
938391: [12640, 12642, 12655],
939578: [12831,
 15304,
 12599,
 12675,
 12362,
 3031,
 13012,
 12994,
 12863,
 11839,
 12775,
 12553,
 11658,
 2436],
940757: [3911],
941648: [11756, 13012, 12877, 10106, 13054, 11083, 12451],
943237: [13122, 13121],
944863: [12599, 12597, 12597, 12571, 4809],
947906: [12672, 8131],
949897: [12633, 12630],
952257: [13679],
952307: [10982],
957212: [12640, 12640],
957621: [13009, 13004],
961234: [12762, 13070, 4133],
964362: [3914],
964403: [12256, 12388, 11662, 12677, 11839, 13637, 13613, 13638],
964658: [1854, 1854],
966716: [7933],
968111: [14778, 4133, 11839, 4212],
```

968713: [13004, 13009, 12992],
974867: [12882],
975791: [1186],
975833: [7114],
977776: [13055, 13058],
979497: [4897],
981536: [7746],
984226: [13054, 14778, 12598, 12597, 13061],
988842: [12642, 12655, 12640, 12655, 12642, 12640],
991389: [12851, 8573, 12762],
992574: [12639, 12644, 12642, 12880],
993379: [4872],
995523: [12993, 13022],
997234: [13512, 241],
998687: [2883, 12678, 12846],
1002271: [13443],
1006869: [1515],
1012718: [12882, 1302],
1014966: [15056],
1018319: [2798, 2612, 2991, 8303, 11178],
1020822: [2104],
1021156: [12642, 12640, 12642, 12640],
1026056: [12640, 8197],
1028951: [12886, 13512],
1028974: [15836, 12891, 13355],
1033294: [14635],
1034343: [12875, 12851, 12851, 12755],
1037224: [12599, 14778, 12815, 8820, 13055, 13058],
1038979: [2395],
1039261: [12645, 12656],
1039916: [4184],
1040711: [14720],
1042072: [13122, 12990],
1042573: [6870],
1043108: [9886],
1043173: [15362],
1044326: [12428],
1053833: [12645, 12656],
1054062: [10480],
1056812: [2447, 11236],
1056966: [15094, 15071],
1058127: [3546, 12664],
1064341: [4254],
1066359: [3718],
1066943: [5890, 7222],
1067896: [4799, 10312],
1068996: [3747, 12884, 5266],
1070789: [11826],
1073364: [13637],
1078069: [14720],

```
1079146: [12497, 11173],
1082877: [15664],
1084508: [9280],
1092017: [2103],
1092039: [10038, 11772],
1094614: [11771, 3682, 10982],
1095243: [12378, 6044],
1097381: [4827],
1104232: [6111, 6111],
1105228: [12675, 12596, 12599],
1108214: [12656, 12645],
1112761: [10358],
1115834: [10982],
1123012: [1416, 1419, 1418],
1125239: [2378],
1126969: [12372],
1129263: [7395],
1131164: [11173, 11172],
1135644: [12642, 12639, 12640],
1135699: [9705],
1137587: [12882],
1138786: [11884],
1142834: [10631],
1145729: [9993],
1147053: [4133, 1187],
1147932: [12639],
1148139: [1933, 10048],
1150236: [12640],
1151242: [3803],
1151344: [12640, 12642, 12506],
1153151: [15734, 11178, 15129],
1157127: [2883],
1157667: [8545],
1161201: [4075, 4075],
1161621: [12642, 12640, 15154],
1162583: [13018],
1165452: [254],
1165966: [12630, 12630],
1173374: [8746, 11467],
1175294: [1746, 8785, 8774, 15494],
1181527: [12443],
1182557: [7634],
1182829: [12555],
1182831: [12411, 722],
1185874: [13054],
1191853: [4804, 4805],
1192516: [12654, 11656],
1192853: [7213, 13012, 2714, 4839, 4208],
1199632: [5784, 13051],
1200769: [1392],
```

```
1203011: [2166],
1209098: [16036],
1210184: [5266],
1215198: [13510, 14778, 13054, 12599, 12597],
1215351: [12540],
1222663: [13431,
 13382,
 13380,
 13388,
 13381,
 13430,
 13390,
 13389,
 13393,
 13394],
1223009: [13054, 12553],
1225102: [12768, 13637, 12646],
1226343: [11173],
1226948: [10998],
1227208: [12640, 12645],
1229892: [12637],
1230674: [12641, 15726],
1232798: [12958, 5230],
1233014: [11223],
1234837: [11828],
1236477: [358],
1238229: [12630],
1238528: [10982, 3682, 3686],
1238796: [12506, 12497, 12762, 12682],
1245438: [806],
1246737: [12417, 3788, 3735, 12321, 2719],
1247423: [9731, 12832, 12677, 11455],
1249612: [11317],
1252699: [13009, 13004],
1255306: [7934],
1256256: [15391],
1257036: [12641, 12655],
1257698: [1162, 1168, 933, 218],
1257958: [4951],
1260919: [2919],
1260956: [13373, 12815, 13426, 13369, 13396],
1261446: [12642, 12645, 12644, 12640, 12644, 12645, 12640, 12642],
1262673: [13122],
1262926: [578],
1266216: [4025, 12640],
1266597: [3747],
1266993: [10982, 3682],
1271356: [12640],
1271834: [8253],
1275363: [6193],
```

```
1285503: [8190],
1286224: [13060, 15691],
1286562: [13055, 12993],
1291759: [1854, 10298],
1293786: [11767],
1294016: [13071, 13054, 13072, 13074],
1299544: [7436, 7436],
1302189: [2166],
1303591: [2763],
1303604: [4133],
1305627: [13074, 13074],
1306142: [12987],
1307053: [5256, 12655, 17],
1313566: [13071, 6022],
1315506: [13071],
1319851: [4681],
1327284: [12981, 12981],
1329241: [12598, 12599, 14778],
1330633: [15750],
1332321: [4941],
1332924: [10607, 13638],
1338418: [15837],
1341883: [3441, 3441],
1347142: [15154],
1347982: [8732, 8693, 15489],
1354228: [15758],
1354498: [15015],
1354757: [12790, 13638],
1355912: [12882],
1356403: [15598, 15557, 15256],
1358583: [15064,
 15584,
 15087,
 7953,
 14674,
 7953,
 15589,
 15064,
 15087,
 14674,
 15589,
 15584,
 15735,
 15735,
 15064,
 15589,
 7953,
 15087,
 14674,
 15584,
```

 15735],
1358612: [13072, 12756, 13908],
1361927: [2683, 9938],
1365166: [14778, 5247],
1366611: [13161],
1369281: [8764, 15136, 8764, 15136],
1372538: [9],
1372827: [11658,
 13158,
 9483,
 12352,
 2321,
 12390,
 12292,
 11658,
 13158,
 9483,
 12352,
 2321,
 12292,
 12390],
1375507: [12978],
1381233: [12992, 9577],
1381846: [13638, 12768, 12682, 4023],
1382166: [13572],
1382517: [3850],
1387523: [13161, 12785],
1391357: [13074, 9067, 4119, 13496, 13005, 13442, 13444, 13090],
1391957: [11357],
1393129: [13638, 13637, 11349, 13638, 13637, 11349],
1393968: [12976, 1888, 15051, 12647],
1394442: [11616, 3002, 11616, 3002],
1397054: [14107],
1406226: [11466, 12981, 13638, 3199, 3198],
1411286: [12976],
1412437: [3984],
1414509: [3321],
1418609: [13059, 1431],
1420249: [13613, 13638],
1422411: [3911],
1423833: [13813],
1424019: [15297, 8347],
1424069: [12768, 13638, 13637],
1426397: [12768, 13638],
1427984: [12653],
1428369: [424, 424],
1431576: [10038, 12378],
1432276: [13636],
1435339: [13638, 12768],
1437233: [7330],

```
1438313: [12789],
1439748: [10367, 10477],
1440946: [8444, 1153],
1442774: [13635],
1449601: [15016],
1450532: [3816, 1107],
1452693: [4119, 12653, 13638, 12756],
1456618: [4133, 11130],
1459927: [12681, 14778],
1460198: [13637, 13638, 13638, 13637],
1460323: [5243,
 11654,
 12462,
 12825,
 12864,
 6774,
 8283,
 11012,
 6596,
 12390,
 6595,
 11011,
 11010,
 167],
1463706: [12990, 12990, 12990, 12990],
1464933: [9265],
1468993: [13637, 12768, 13638],
1475487: [14946, 14944],
1477591: [13064, 13613, 12762, 13638, 13064, 13613, 12762, 13638],
1479551: [12639],
1482757: [1998],
1489531: [12768, 13638],
1492547: [11825, 10606],
1495173: [13638, 12646, 12289],
1506466: [12988],
1510023: [13483],
1521264: [13637, 13638],
1521999: [12203,
 14914,
 2657,
 10534,
 1663,
 10035,
 13739,
 12356,
 2903,
 10970,
 2108],
1527541: [13637],
1530887: [11525],
```

1532511: [12634],
1537016: [10605],
1537743: [12784, 13162, 3526, 12451],
1538123: [11419, 3981],
1539006: [13638, 13064, 13637, 12981, 12768, 13613],
1540439: [13156, 5259, 13163],
1541301: [12383],
1541442: [11238],
1541681: [12653],
1543501: [8776, 13060, 9716],
1544581: [15490, 8774, 15468],
1544644: [12815],
1546431: [7503],
1546463: [13638, 12768],
1550739: [12378],
1552973: [13095],
1553521: [13064, 11661, 3459],
1554429: [4804, 4805],
1557062: [1403],
1557743: [11227, 11228],
1563959: [12642],
1566226: [12768, 13638, 13064, 12876, 12660, 12680, 13637, 13613],
1567239: [13056, 13056, 13056],
1573058: [8314, 2672],
1577606: [12981, 13635],
1584162: [4133, 13054],
1585873: [13638, 12768, 13638],
1586239: [10054],
1598006: [13998],
1599287: [11356],
1602886: [13698],
1602924: [4872],
1608737: [11501],
1609401: [12768, 12681, 13638, 12682, 12762, 13231],
1610992: [5329],
1611562: [12768, 13089, 13088, 13638],
1617777: [5769],
1618073: [16, 8190, 16, 8190],
1618726: [9220],
1620507: [13074, 13692],
1629947: [13064, 13638, 6628, 10607, 13637],
1632929: [3943, 4041, 12452, 10038],
1634784: [13156, 13162, 13162, 13156],
1635439: [5639],
1638611: [13637, 13638],
1641471: [6228],
1642447: [15748],
1648427: [13637, 12768, 13064, 13638, 11467, 10000],
1649499: [4786],
1650542: [12564],

1650811: [13124],
1651739: [12653, 13692],
1655121: [12630],
1657604: [13161, 13156, 13163, 13494, 13106],
1663779: [11375],
1664362: [12210, 13010, 13060],
1665372: [12768, 13638],
1670197: [13124],
1670831: [13064, 3948, 3948, 13064, 3948, 13064],
1671752: [13638, 12768, 13064, 12979],
1673383: [15058],
1675762: [10982, 12987],
1677874: [13445, 13064, 12620],
1678726: [14968, 14618],
1680706: [12768, 10000, 13070, 13637],
1683503: [13074, 12681, 12755, 13074],
1683519: [12768, 13638, 12762],
1684999: [13637, 12768, 13613, 13637, 12768, 13613],
1690711: [10059, 10059],
1691882: [13018, 13007],
1692087: [12613, 13022],
1694197: [13638, 12768],
1694476: [12653, 12818],
1698484: [13638, 12768, 6175, 13069],
1700783: [13510],
1700801: [13637, 13638, 12756],
1702029: [13638, 12768],
1706378: [8253, 4143],
1706896: [12506, 12497],
1710366: [2435, 2825, 12882],
1711881: [12656, 12775],
1712399: [1928],
1717318: [11349, 13637, 12833, 12985, 12550, 12768],
1718407: [12378, 4059],
1725314: [10607, 10607, 11467],
1728649: [12680, 12660],
1729571: [13004, 13009],
1734879: [15739],
1736678: [10542],
1746348: [355, 355],
1749913: [12640],
1753684: [13895, 13895],
1757311: [12785],
1757508: [4029],
1759453: [12756],
1762533: [8513],
1763361: [13010, 2378],
1764317: [10035, 8195, 3645, 3586, 3192, 9716, 11772, 2451],
1767864: [13160, 12980, 8708, 13159],
1775386: [12770, 12617],

1776397: [13638, 12635, 12818],
1780132: [13494],
1782183: [13072, 12756, 13637, 13158],
1783711: [3747],
1787382: [12762, 12681, 13637, 13637, 13638, 12768, 13064],
1789289: [12642, 13035, 3244],
1792339: [12762, 12762],
1794516: [12210],
1798914: [13447, 6679],
1799584: [8190],
1802618: [12378],
1803716: [13107, 13107],
1807291: [8176, 12978],
1807804: [13908, 13908],
1808092: [13056],
1809431: [4146, 13163, 13156, 13495],
1815429: [7276, 13644],
1819732: [13683],
1822064: [244, 8596],
1824159: [12653, 13121],
1826954: [12496],
1828084: [12880, 12878, 12881],
1829906: [12646],
1830611: [12507],
1831466: [11885, 11827, 13286, 11827, 8721],
1832139: [2311, 174],
1832963: [12976, 13683, 13496],
1835453: [13156],
1841093: [54],
1845774: [13164, 13069],
1847923: [13168, 13725],
1848376: [13166, 13164],
1851472: [3489, 1548],
1853764: [2120],
1856377: [1982, 11140, 1982, 11140],
1857674: [13157, 13158],
1858184: [3548, 10709, 6071, 420],
1859963: [10681, 10681, 10684, 3703, 10681],
1861751: [12630],
1862619: [12975, 11268, 13635],
1866224: [13495, 13088, 12791, 13065, 13066],
1869789: [12552, 12638, 13638],
1879983: [3654],
1880134: [15750],
1880239: [4703, 4714, 4714, 4703, 4714, 4703],
1882932: [13908, 12497, 14946],
1886472: [12645, 11525],
1888038: [5535],
1889778: [11528],
1890634: [13089, 12548, 14063],

1894668: [3822, 8352],
1897011: [13168, 5535],
1902694: [16],
1905582: [12998],
1908416: [5654],
1909574: [12444, 12870, 11653],
1911957: [9067, 9423, 1053],
1912998: [12791, 2475],
1913513: [9274, 13715, 13548],
1914727: [13908],
1917411: [11839, 13851, 13168, 13293, 12786, 13158],
1919047: [13010],
1921883: [13163, 13156, 13494, 13107, 13108, 13161, 13106],
1934562: [1392],
1935179: [13161],
1936644: [13635, 11689, 12632],
1936668: [13908, 13908],
1937761: [10365, 13148],
1941524: [13494],
1944674: [13495],
1945979: [12976, 1121],
1947186: [6140],
1964796: [304, 4704],
1964882: [13093, 13093],
1974939: [11449, 3735, 12548, 3774],
1978564: [5950],
1979014: [7954],
1979961: [12210, 13161],
1980292: [7058],
1984938: [13635],
2102581: [13156, 13158],
2111661: [13163, 13161, 13162],
2117836: [2351, 6006, 2351, 6006],
2121399: [15475],
2127257: [13445, 13442],
2127808: [1762, 1761, 1761, 1762],
2128477: [13124],
2129007: [4804, 4805],
2130942: [424, 13395, 13419],
2132376: [5090],
2133006: [11611, 1000],
2143481: [12790, 12786],
2143756: [11840],
2145847: [7938],
2146833: [12782, 12785, 12782, 12785],
2148627: [15739],
2150008: [13164],
2151423: [1121],
2151616: [108],
2155544: [10024],

2158569: [13071],
2159779: [1909],
2168694: [10124],
2179317: [10671],
2186254: [14064],
2186279: [1815],
2189388: [13088, 13089, 12980],
2195717: [12782],
2195959: [13494],
2205593: [11203, 12645],
2208297: [12506, 12497, 12505],
2210113: [2459],
2215774: [5235],
2219067: [9688, 2378],
2220744: [13166, 13164],
2226053: [13165, 14113, 13168],
2226289: [3718],
2229884: [13164, 13286, 14111],
2232111: [11613],
2232961: [7045],
2234868: [13164],
2236762: [12609],
2242719: [13168, 13286, 12870, 8393, 13290],
2243511: [3747, 694],
2248027: [15683],
2249276: [13164, 13166, 13286],
2249868: [7467, 1883],
2251514: [12838, 11524, 12655, 8290],
2257128: [11823],
2257633: [13165, 13164, 13167, 13168],
2264676: [6071],
2267673: [13286, 13164, 13285, 9999],
2268638: [13164, 13290, 13166, 13166, 13285, 13168],
2270702: [11885, 11826],
2273539: [10364],
2275046: [12676, 12875, 13165, 12851],
2279656: [13165],
2279838: [13494],
2280454: [10605, 10461, 6650, 14944, 10680],
2282024: [13164, 13166],
2284373: [1478, 12978],
2284789: [320, 1412, 13285, 13287],
2289552: [13164, 13168],
2290431: [13498],
2290698: [11827, 11824],
2291707: [15758, 15758],
2292108: [18460],
2292426: [11208],
2292499: [13156, 13108],
2293224: [7964, 7965],

2294903: [13494],
2295932: [13110],
2300511: [13167, 13164, 9691, 13286],
2300966: [14062, 1121],
2301182: [12974, 10465, 1819],
2301826: [13164, 13167, 13286, 13166],
2302889: [2125],
2303773: [13166, 13164, 13290, 13288, 15921],
2305544: [13164, 13168],
2306951: [13164, 13164, 13286, 13168, 13166, 13167, 13285],
2307524: [13164, 13286, 13167],
2310501: [13168, 8393, 13168],
2311333: [13285, 13165, 13168],
2312773: [13164, 13286, 9699],
2316132: [7877],
2317759: [14951, 7439, 7439, 14951],
2318429: [18460],
2318877: [11221, 12355],
2319826: [13279, 13285, 13164],
2320679: [13108],
2322056: [12869, 13510],
2322739: [3718, 8278, 10962],
2323802: [18460],
2324881: [13279, 10606, 10607],
2325937: [18460],
2326769: [12826, 13832],
2330123: [13603, 8531],
2330138: [13286, 13164],
2331598: [13165, 13285, 13066],
2332823: [12791, 11662, 12362, 11839, 13712],
2339363: [4025],
2343879: [13164, 13285, 13166],
2343936: [9615],
2343972: [13289, 13167, 13290, 13166, 13289, 13166, 13167, 13290],
2344106: [13165, 12871, 10626, 13168],
2344962: [18460],
2346906: [18460],
2349727: [13613, 13637, 263],
2349987: [12768, 13638, 8573, 13637],
2352556: [12205],
2356753: [3790, 13065],
2358446: [13712, 12868, 13293, 12655],
2362019: [15297, 15009],
2362443: [13712, 12362, 14111],
2369971: [13165, 13164, 13166, 13168],
2373271: [2106],
2374763: [6824],
2379251: [13167],
2381423: [11564],
2389753: [13285, 13164, 13166, 13168],

```
 2390563: [372],
 2390959: [832],
 2391268: [18460, 18460, 18460, 18460, 18460, 18460],
 2391993: [13164, 13285],
 2397941: [15683],
 2399516: [18460, 18460],
 2400744: [6813],
 2401798: [12630, 13499, 12630, 13499],
 2402286: [12815],
 2411157: [8253],
 ...}
```

Сборка выборки для обучения

```
# 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().values
        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])

# Prepare dataset
dataset = YooChooseDataset('./')
```

```
Processing...
  0%|          | 0/50000 [00:00<?,
?it/s]/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:21:
UserWarning: Creating a tensor from a list of numpy.ndarrays is
extremely slow. Please consider converting the list to a single
numpy.ndarray with numpy.array() before converting to a tensor.
(Triggered internally at  ../torch/csrc/utils/tensor_new.cpp:210.)
100%|██████████| 50000/50000 [03:20<00:00, 249.08it/s]
Done!
```

Разделение выборки

```
# 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)
```

```
(40000, 5000, 5000)
```

```
# 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)
```

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

```
(18461, 109)
```

Настройка модели для обучения

```
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

```
# Enable CUDA computing
device = torch.device('cuda')
model = Net().to(device)
# Choose optimizer and criterion for learning
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
crit = torch.nn.BCELoss()

# 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)

# 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)
```

```
# Train a model
NUM_EPOCHS =  5 #@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))
```

```
 20%|██        | 1/5 [00:44<02:56, 44.20s/it]

Epoch: 000, Loss: 0.69027, Train Auc: 0.51714, Val Auc: 0.50629, Test
Auc: 0.52047

  40%|████      | 2/5 [01:27<02:10, 43.46s/it]

Epoch: 001, Loss: 0.50524, Train Auc: 0.55385, Val Auc: 0.52314, Test
Auc: 0.53985

  60%|██████    | 3/5 [02:08<01:25, 42.68s/it]

Epoch: 002, Loss: 0.41198, Train Auc: 0.59141, Val Auc: 0.54215, Test
Auc: 0.55625

  80%|████████  | 4/5 [02:50<00:42, 42.19s/it]

Epoch: 003, Loss: 0.37019, Train Auc: 0.62583, Val Auc: 0.56084, Test
Auc: 0.56741

100%|██████████| 5/5 [03:31<00:00, 42.35s/it]

Epoch: 004, Loss: 0.36173, Train Auc: 0.63303, Val Auc: 0.55783, Test
Auc: 0.56239
```

Проверка результата с помощью примеров

```
# Подход №1 - из датасета
evaluate(DataLoader(test_dataset[40:60], batch_size=10))
```

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

0.6071055381400209
```

```
# 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)
```

```
100%|██████████| 3/3 [00:00<00:00, 164.06it/s]

DataBatch(x=[1, 1, 2], edge_index=[2, 0], y=[1], batch=[1], ptr=[2])
[0.00017722]
DataBatch(x=[3, 1, 2], edge_index=[2, 2], y=[3], batch=[3], ptr=[2])
[0.03716549 0.03555349 0.12272909]
DataBatch(x=[4, 1, 2], edge_index=[2, 3], y=[4], batch=[4], ptr=[2])
[0.05185128 0.04105826 0.00932805 0.12313381]


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