Рекомендательные системы

Урок 5. Поиск похожих товаров и пользователей. Гибридные рекомендательные системы

Домашнее задание

1) Прочитать статьи про BPR, WARP loss

2) Сделать грид серч текущей модели, смотрите на метрику precision@5, считаем на тесте нашей функцией

Выполнил Соковнин ИЛ

In [1]:
```
!pip install lightfm
```

```
Requirement already satisfied: lightfm in /home/sil/anaconda3/lib/python3.7/site-packages (1.16)
Requirement already satisfied: numpy in /home/sil/anaconda3/lib/python3.7/site-packages (from lightfm) (1.
18.1)
Requirement already satisfied: requests in /home/sil/anaconda3/lib/python3.7/site-packages (from lightfm)
(2.22.0)
Requirement already satisfied: scipy>=0.17.0 in /home/sil/anaconda3/lib/python3.7/site-packages (from ligh
tfm) (1.4.1)
Requirement already satisfied: scikit-learn in /home/sil/anaconda3/lib/python3.7/site-packages (from light
fm) (0.22.1)
Requirement already satisfied: certifi>=2017.4.17 in /home/sil/anaconda3/lib/python3.7/site-packages (from
requests->lightfm) (2019.11.28)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /home/sil/anaconda3/lib/python3.
7/site-packages (from requests->lightfm) (1.25.8)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /home/sil/anaconda3/lib/python3.7/site-packages (f
rom requests->lightfm) (3.0.4)
Requirement already satisfied: idna<2.9,>=2.5 in /home/sil/anaconda3/lib/python3.7/site-packages (from req
uests->lightfm) (2.8)
Requirement already satisfied: joblib>=0.11 in /home/sil/anaconda3/lib/python3.7/site-packages (from sciki
t-learn->lightfm) (0.14.1)
```

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

# Для работы с матрицами
from scipy.sparse import csr_matrix, coo_matrix

# Матричная факторизация
from implicit.als import AlternatingLeastSquares
from implicit.nearest_neighbours import bm25_weight, tfidf_weight

from lightfm import LightFM

# Функции из 1-ого вебинара
import os, sys

module_path = os.path.abspath(os.path.join(os.pardir))
if module_path not in sys.path:
    sys.path.append(module_path)
```

In [3]:
```
from lightfm.evaluation import precision_at_k, recall_at_k

from metrics import precision_at_k as custom_precision, recall_at_k
from utils import prefilter_items
```

In [4]:
```
data = pd.read_csv('../data/retail_train.csv')

item_features = pd.read_csv('../data/product.csv')
user_features = pd.read_csv('../data/hh_demographic.csv')

# column processing
item_features.columns = [col.lower() for col in item_features.columns]
user_features.columns = [col.lower() for col in user_features.columns]

item_features.rename(columns={'product_id': 'item_id'}, inplace=True)
user_features.rename(columns={'household_key': 'user_id'}, inplace=True)

# train test split
test_size_weeks = 3  # 3 weeks

data_train = data[data['week_no'] < data['week_no'].max() - test_size_weeks]
data_test = data[data['week_no'] >= data['week_no'].max() - test_size_weeks]

data_train.head(2)
```

Out[4]:

	user_id	basket_id	day	item_id	quantity	sales_value	store_id	retail_disc	trans_time	week_no	coupon_disc	coupon_match_disc
0	2375	26984851472	1	1004906	1	1.39	364	-0.6	1631	1	0.0	0.0
1	2375	26984851472	1	1033142	1	0.82	364	0.0	1631	1	0.0	0.0

In [5]:
```
pwd
```
Out[5]:
```
'/home/sil/ML/RS/lesson_5/HW'
```

In [6]:
```
ls
```
```
lesson_5_hw.ipynb  metrics.py*  __pycache__/  utils.py*
```

In [7]:
```
ls ../data/
```
```
hh_demographic.csv*  product.csv*  retail_train.csv*  transaction_data.csv*
```

In [8]:
```
result = data_test.groupby('user_id')['item_id'].unique().reset_index()
result.columns=['user_id', 'actual']
result.head(2)
```
Out[8]:

	user_id	actual
0	1	[821867, 834484, 856942, 865456, 889248, 90795...
1	3	[835476, 851057, 872021, 878302, 879948, 90963...

In [9]:
```
item_features.head(2)
```
Out[9]:

	item_id	manufacturer	department	brand	commodity_desc	sub_commodity_desc	curr_size_of_product
0	25671	2	GROCERY	National	FRZN ICE	ICE - CRUSHED/CUBED	22 LB
1	26081	2	MISC. TRANS.	National	NO COMMODITY DESCRIPTION	NO SUBCOMMODITY DESCRIPTION	

In [10]:
```
user_features.head(2)
```
Out[10]:

	age_desc	marital_status_code	income_desc	homeowner_desc	hh_comp_desc	household_size_desc	kid_category_desc	user_id
0	65+	A	35-49K	Homeowner	2 Adults No Kids	2	None/Unknown	1
1	45-54	A	50-74K	Homeowner	2 Adults No Kids	2	None/Unknown	7

In [11]:
```
user_features['age_desc'].unique()
```
Out[11]:
```
array(['65+', '45-54', '25-34', '35-44', '19-24', '55-64'], dtype=object)
```

In [12]:
```
user_features['marital_status_code'].unique()
```
Out[12]:
```
array(['A', 'U', 'B'], dtype=object)
```

In [13]:
```
user_features['household_size_desc'].unique()
```
Out[13]:
```
array(['2', '3', '4', '1', '5+'], dtype=object)
```

1. Filter items

Фильтрация товара. Оставляем 5000 самых популярных товаров.

In [14]:
```
data_train.head(3)
```
Out[14]:

	user_id	basket_id	day	item_id	quantity	sales_value	store_id	retail_disc	trans_time	week_no	coupon_disc	coupon_match_disc
0	2375	26984851472	1	1004906	1	1.39	364	-0.6	1631	1	0.0	0.0
1	2375	26984851472	1	1033142	1	0.82	364	0.0	1631	1	0.0	0.0
2	2375	26984851472	1	1036325	1	0.99	364	-0.3	1631	1	0.0	0.0

In [15]:
```
n_items_before = data_train['item_id'].nunique()

data_train_filtered = prefilter_items(data_train, take_n_popular=5000, item_features=item_features)

n_items_after = data_train_filtered['item_id'].nunique()
print('Decreased # items from {} to {}'.format(n_items_before, n_items_after))
```

```
/home/sil/ML/RS/lesson_5/HW/utils.py:20: 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.htm
l#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#re
turning-a-view-versus-a-copy)
  data['price'] = data['sales_value'] / (np.maximum(data['quantity'], 1))

Decreased # items from 86865 to 5001
```

2. Prepare data set

2.1 Prepare csr train matrix

In [16]:
```

user_item_matrix = pd.pivot_table(data_train_filtered,
                                  index='user_id', columns='item_id',
                                  values='quantity', # Можно пробоват ьдругие варианты
                                  aggfunc='count',
                                  fill_value=0
                                 )

user_item_matrix = user_item_matrix.astype(float) # необходимый тип матрицы для implicit

# переведем в формат sparse matrix
sparse_user_item = csr_matrix(user_item_matrix).tocsr()

user_item_matrix.head(2)
```

Out[16]:

item_id	117847	818981	819255	819308	819400	819487	819590	819594	819840	819845	...	15926775	15926844	15926886	15972074	159722
user_id																
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	1.0	0.0	0.0	0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0

2 rows × 5001 columns

2.2 Prepare CSR test matrix

In [17]:
```
data_test = data_test[data_test['item_id'].isin(data_train['item_id'].unique())]

test_user_item_matrix = pd.pivot_table(data_test,
                                  index='user_id', columns='item_id',
                                  values='quantity', # Можно пробоват ьдругие варианты
                                  aggfunc='count',
                                  fill_value=0
                                 )

test_user_item_matrix = test_user_item_matrix.astype(float) # необходимый тип матрицы для implicit
```

In [18]:
```
test_user_item_matrix.head(2)
```

Out[18]:

item_id	32392	34873	42852	43094	44522	45507	49812	53516	58612	62804	...	17284297	17284423	17291184	17291665	17320698	173209
user_id																	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0

2 rows × 22143 columns

In [19]:
```python
userids = user_item_matrix.index.values
itemids = user_item_matrix.columns.values

matrix_userids = np.arange(len(userids))
matrix_itemids = np.arange(len(itemids))

# Строим матрицу преобразования
id_to_itemid = dict(zip(matrix_itemids, itemids))
id_to_userid = dict(zip(matrix_userids, userids))

itemid_to_id = dict(zip(itemids, matrix_itemids))
userid_to_id = dict(zip(userids, matrix_userids))
```

3. Prepare user and item features

In [20]:
```python
user_feat = pd.DataFrame(user_item_matrix.index)
user_feat
```

Out[20]:

	user_id
0	1
1	2
2	3
3	4
4	5
...	...
2492	2496
2493	2497
2494	2498
2495	2499
2496	2500

2497 rows × 1 columns

In [21]:
```python
# Объединение — функция pandas. merge() соединяет строки в Dataframe на основе одного или нескольких ключей
user_feat = user_feat.merge(user_features, on='user_id', how='left')
user_feat.set_index('user_id', inplace=True)
user_feat.head(100)
```

Out[21]:

	age_desc	marital_status_code	income_desc	homeowner_desc	hh_comp_desc	household_size_desc	kid_category_desc
user_id							
1	65+	A	35-49K	Homeowner	2 Adults No Kids	2	None/Unknown
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...
96	NaN	NaN	NaN	NaN	NaN	NaN	NaN
97	45-54	U	75-99K	Unknown	Single Female	1	None/Unknown
98	35-44	U	35-49K	Unknown	1 Adult Kids	2	1
99	NaN	NaN	NaN	NaN	NaN	NaN	NaN
100	NaN	NaN	NaN	NaN	NaN	NaN	NaN

100 rows × 7 columns

In [22]:
```python
user_feat.shape
```

Out[22]: (2497, 7)

In [23]:
```python
item_feat = pd.DataFrame(user_item_matrix.columns)
item_feat = item_feat.merge(item_features, on='item_id', how='left')
item_feat.set_index('item_id', inplace=True)

item_feat.head(2)
```

Out[23]:

	manufacturer	department	brand	commodity_desc	sub_commodity_desc	curr_size_of_product
item_id						
117847	450.0	NUTRITION	National	REFRIGERATED	SOY/RICE MILK	64 OZ
818981	194.0	GROCERY	National	COLD CEREAL	ALL FAMILY CEREAL	10.4 OZ

In [24]:
```
item_feat.shape
```
Out[24]:
```
(5001, 6)
```

Encoding features

In [25]:
```
# Кодирование в бинарном виде
user_feat_lightfm = pd.get_dummies(user_feat, columns=user_feat.columns.tolist())
item_feat_lightfm = pd.get_dummies(item_feat, columns=item_feat.columns.tolist())
```

In [26]:
```
user_feat_lightfm.head(2)
```
Out[26]:

	age_desc_19-24	age_desc_25-34	age_desc_35-44	age_desc_45-54	age_desc_55-64	age_desc_65+	marital_status_code_A	marital_status_
user_id								
1	0	0	0	0	0	1	1	
2	0	0	0	0	0	0	0	

2 rows × 41 columns

Init model

In [27]:
```
model = LightFM(no_components=40,
                loss='bpr', # "logistic","bpr"
                learning_rate=0.01,
                item_alpha=0.4,
                user_alpha=0.1,
                random_state=42,
                k=5,
                n=15,
                max_sampled=100)
```

Train

In [28]:
```
model.fit((sparse_user_item > 0) * 1,  # user-item matrix из 0 и 1
          sample_weight=coo_matrix(user_item_matrix),
          user_features=csr_matrix(user_feat_lightfm.values).tocsr(),
          item_features=csr_matrix(item_feat_lightfm.values).tocsr(),
          epochs=20,
          num_threads=20,
          verbose=True)
```
```
Epoch: 100%|██████████| 20/20 [01:24<00:00,  4.23s/it]
```
Out[28]:
```
<lightfm.lightfm.LightFM at 0x7f9576463d50>
```

In [ ]:

Getting embeddings

Векторизация

вектора по пользователям

In [29]:
```
user_emb = model.get_user_representations(features=csr_matrix(user_feat_lightfm.values).tocsr())
```

In [30]:
```
user_emb[0].shape # biases
```
Out[30]:
```
(2497,)
```

In [31]:
```
user_emb[1].shape # users vectors
```
Out[31]:
```
(2497, 40)
```

In [32]:
```
# user_emb
```

вектора по товарам

In [33]:
```
item_emb = model.get_item_representations(features=csr_matrix(item_feat_lightfm.values).tocsr())
```

In [34]:
```
item_emb[0].shape # biases
```
Out[34]:
```
(5001,)
```

In [35]:
```python
item_emb[1].shape # items vectors
```
Out[35]:
```
(5001, 40)
```

In [36]:
```python
# item_emb
```

Evaluation -> Train precision

Оценка -> Тренировка точности

In [37]:
```python
# мы можем использовать встроенные метрики lightFM
train_precision = precision_at_k(model, sparse_user_item,
                                  user_features=csr_matrix(user_feat_lightfm.values).tocsr(),
                                  item_features=csr_matrix(item_feat_lightfm.values).tocsr(),
                                  k=5).mean()

print(f"Train precision {train_precision}")
```
```
Train precision 0.4385262429714203
```

Predict

In [38]:
```python
# подготавливаемм id для юзеров и товаров в порядке пар user-item
users_ids_row = data_train_filtered['user_id'].apply(lambda x: userid_to_id[x]).values.astype(int)
items_ids_row = data_train_filtered['item_id'].apply(lambda x: itemid_to_id[x]).values.astype(int)
```

In [39]:
```python
users_ids_row[:10]
```
Out[39]:
```
array([2371, 1363, 1363, 1363, 1363, 1171, 1171, 1171, 1171, 1171])
```

In [40]:
```python
items_ids_row[:10]
```
Out[40]:
```
array([2959, 2040, 2040, 2040, 1325, 2040, 2040, 2049, 2040, 2840])
```

In [41]:
```python
# модель возвращает меру/скор похожести между соответствующим пользователем и товаром
predictions = model.predict(user_ids=users_ids_row,
                            item_ids=items_ids_row,
                            user_features=csr_matrix(user_feat_lightfm.values).tocsr(),
                            item_features=csr_matrix(item_feat_lightfm.values).tocsr(),
                            num_threads=10)
```

In [42]:
```python
# добавляем наш полученный скор в трейн датафрейм
data_train_filtered['score'] = predictions
```

In [43]:
```python
data_train_filtered.head()
```
Out[43]:

	user_id	basket_id	day	item_id	quantity	sales_value	store_id	retail_disc	trans_time	week_no	coupon_disc	coupon_match_disc	price
7	2375	26984851516	1	1085983	1	2.99	364	-0.40	1642	1	0.0	0.0	2.9
11	1364	26984896261	1	999999	1	2.19	31742	0.00	1520	1	0.0	0.0	2.1
12	1364	26984896261	1	999999	1	2.99	31742	-0.40	1520	1	0.0	0.0	2.9
13	1364	26984896261	1	999999	1	3.09	31742	0.00	1520	1	0.0	0.0	3.0
14	1364	26984896261	1	937406	1	2.50	31742	-0.99	1520	1	0.0	0.0	2.5

In [44]:
```python
# создаем предикт датафрейм в формате списка товаров
predict_result = data_train_filtered[['user_id','item_id','score']][data_train_filtered.item_id != 999999].
            unique().reset_index()
```

In [45]:
```python
predict_result.head()
```
Out[45]:

	user_id	item_id
0	1	[1029743, 877391, 986912, 6034857, 909497, 556...
1	2	[1106523, 916122, 945901, 1075368, 904236, 112...
2	3	[1106523, 983584, 1127831, 854261, 5585510, 10...
3	4	[1029743, 1075368, 1052294, 1044078, 970760, 5...
4	5	[1126899, 1029743, 916122, 6034991, 1010259, 9...

In [46]:
```python
# объединяем предикт и тест датасет для подсчета precision
df_result_for_metrics = result.merge(predict_result, on='user_id', how='inner')
```

In [47]:
```python
df_result_for_metrics.head()
```
Out[47]:

user_id	actual	item_id

	user_id	actual	item_id
0	1	[821867, 834484, 856942, 865456, 889248, 90795...	[1029743, 877391, 986912, 6034857, 909497, 556...
1	3	[835476, 851057, 872021, 878302, 879948, 90963...	[1106523, 983584, 1127831, 854261, 5585510, 10...
2	6	[920308, 926804, 946489, 1006718, 1017061, 107...	[1070820, 1029743, 1126899, 1121393, 9524291, ...

Test with custom precision func

In [48]:
```
precision = df_result_for_metrics.apply(lambda row: custom_precision(row['item_id'], row['actual'],k=5), ax:
print(f"Precision: {precision}")
```

```
Precision: 0.1426929392446615
```

Links

Neural networks for RS: [http://d2l.ai/chapter_recommender-systems/mf.html (http://d2l.ai/chapter_recommender-systems/mf.html)](http://d2l.ai/chapter_recommender-systems/mf.html)

LigthFM -> [https://arxiv.org/pdf/1507.08439.pdf (https://arxiv.org/pdf/1507.08439.pdf)](https://arxiv.org/pdf/1507.08439.pdf)

[https://making.lyst.com/lightfm/docs/home.html (https://making.lyst.com/lightfm/docs/home.html)](https://making.lyst.com/lightfm/docs/home.html)

Домашнее задание

1. Прочитать статьи про BPR, WARP loss
2. Сделать грид серч текущей модели, смотрите на метрику precision@5, считаем на тесте нашей функцией

Ищем оптимальные параметры (grid-search)

In [49]:
```
import itertools
import copy
```

In [ ]:

In [50]:
```
def get_predicts(model, users_ids_row, items_ids_row):

    # модель возвращает меру/скор похожести между соответствующим пользователем и товаром
    predictions = model.predict(user_ids=users_ids_row,
                                item_ids=items_ids_row,
                                user_features=csr_matrix(user_feat_lightfm.values).tocsr(),
                                item_features=csr_matrix(item_feat_lightfm.values).tocsr(),
                                num_threads=10)
    # добавляем наш полученный скор в трейн датафрейм
    data_train_filtered['score'] = predictions
    # создаем предикт датафрейм в формате списка товаров
    predict_result = data_train_filtered[['user_id','item_id','score']][data_train_filtered.item_id != 99999
            drop_duplicates().sort_values(by=['user_id','score'], ascending=False).groupby('user_id')['item_
            unique().reset_index()

    # объединяем предикт и тест датасет для подсчета precision
    # df_result_for_metrics
    res = result.merge(predict_result, on='user_id', how='inner')

    return res
```

In [51]:
```
def print_log(row, header=False, spacing=12):
    top = ''
    middle = ''
    bottom = ''
    for r in row:
        top += '+{}'.format('-'*spacing)
        if isinstance(r, str):
            middle += '| {0:^{1}} '.format(r, spacing-2)
        elif isinstance(r, int):
            middle += '| {0:^{1}} '.format(r, spacing-2)
        elif isinstance(r, float):
            middle += '| {0:^{1}.5f} '.format(r, spacing-2)
        bottom += '+{}'.format('='*spacing)
    top += '+'
    middle += '|'
    bottom += '+'
    if header:
        print(top)
        print(middle)
        print(bottom)
    else:
        print(middle)
```

```
        print(top)
```

In [52]:
```python
def learning_curve(model, sparse_user_item, user_item_matrix, user_feat_lightfm, item_feat_lightfm, epochs)
    user_item_precision = []

    headers = ['epochs', 'p@k user_item_matrix']

    for epoch in epochs:
        model.fit((sparse_user_item > 0) * 1,  # user-item matrix из 0 и 1
                  sample_weight=coo_matrix(user_item_matrix),
                  user_features=csr_matrix(user_feat_lightfm.values).tocsr(),
                  item_features=csr_matrix(item_feat_lightfm.values).tocsr(),
                  epochs=epoch,
                  num_threads=20,
                  verbose=False)

        # подготавливаемм id для юзеров и товаров в порядке пар user-item
        users_ids_row = data_train_filtered['user_id'].apply(lambda x: userid_to_id[x]).values.astype(int)
        items_ids_row = data_train_filtered['item_id'].apply(lambda x: itemid_to_id[x]).values.astype(int)

        df_result_for_metrics = get_predicts(model, users_ids_row, items_ids_row)
        precision = df_result_for_metrics.apply(lambda row: custom_precision(row['item_id'], \
                                                                  row['actual'],k=5), axis=1).mea
        user_item_precision.append(precision)
        row = [precision]
        print_log(row)
        print(row)

    return model, user_item_precision
```

In [53]:
```python
# def grid_search_learning_curve(base_model, user_item_matrix, param_grid,
#                                  user_index=None, patk=5, epochs=range(2, 40, 2)):
def grid_search_learning_curve(base_model, sparse_user_item, user_item_matrix, user_feat_lightfm, item_feat_
    """
    "Inspired" (stolen) from sklearn gridsearch
    https://github.com/scikit-learn/scikit-learn/blob/master/sklearn/model_selection/_search.py
    """

    curves = []
    keys, values = zip(*param_grid.items())
    for v in itertools.product(*values):
        params = dict(zip(keys, v))
        this_model = copy.deepcopy(base_model)
        print_line = []
        for k, v in params.items():
            setattr(this_model, k, v)
            print_line.append((k, v))

        print(' | '.join('{}: {}'.format(k, v) for (k, v) in print_line))

        _, user_item_patk = learning_curve(this_model,
                                           sparse_user_item,
                                           user_item_matrix,
                                           user_feat_lightfm,
                                           item_feat_lightfm,
                                           epochs=[20])

        curves.append({'params': params,
                       'patk': {'user_item_matrix': user_item_patk}})
    return curves
```

In [54]:
```python
param_grid = {'no_components': [20, 40],
#               'loss': ['bpr'],
               'learning_rate': [0.01, 0.1],
#               'item_alpha': [0.4],
#               'user_alpha': [0.1],
#               'random_state': [42],
#               'k': [5],
#               'n': [15],
              'max_sampled': [30, 50],
             }

user_index = range(user_item_matrix.shape[0])
```

In [55]:
```python
#base_model = LightFM()
base_model = LightFM(# no_components=40,
                loss='bpr', # "logistic","bpr"
#               learning_rate=0.01,
                item_alpha=0.4,
                user_alpha=0.1,
                random_state=42,
                k=5,
                n=15,
#               max_sampled=100
                )


base_model
```

Out[55]: `<lightfm.lightfm.LightFM at 0x7f9568248b90>`

In [56]:
```python
curves = grid_search_learning_curve(base_model,
                                    sparse_user_item,
                                    user_item_matrix,
                                    user_feat_lightfm,
                                    item_feat_lightfm,
                                    param_grid)
```

...

In [57]:
```python
type(curves)
```

Out[57]: list

In [58]:
```python
curves
```

Out[58]:
```
[{'params': {'no_components': 20, 'learning_rate': 0.01, 'max_sampled': 30},
  'patk': {'user_item_matrix': [0.14013136288998188]}},
 {'params': {'no_components': 20, 'learning_rate': 0.01, 'max_sampled': 50},
  'patk': {'user_item_matrix': [0.1393431855500804]}},
 {'params': {'no_components': 20, 'learning_rate': 0.1, 'max_sampled': 30},
  'patk': {'user_item_matrix': [0.09264367816091906]}},
 {'params': {'no_components': 20, 'learning_rate': 0.1, 'max_sampled': 50},
  'patk': {'user_item_matrix': [0.08949096880131333]}},
 {'params': {'no_components': 40, 'learning_rate': 0.01, 'max_sampled': 30},
  'patk': {'user_item_matrix': [0.1424958949096863]}},
 {'params': {'no_components': 40, 'learning_rate': 0.01, 'max_sampled': 50},
  'patk': {'user_item_matrix': [0.14200328407224783]}},
 {'params': {'no_components': 40, 'learning_rate': 0.1, 'max_sampled': 30},
  'patk': {'user_item_matrix': [0.05707717569786581]}},
 {'params': {'no_components': 40, 'learning_rate': 0.1, 'max_sampled': 50},
  'patk': {'user_item_matrix': [0.10515599343185475]}}]
```

ВЫВОД

Лучшее значение метрики Precision=0.1424958949096863 имеем при следующих параметрах
'params': {'no_components': 40, 'learning_rate': 0.01, 'max_sampled': 30}

In [64]:
```python
%%time
base_model = LightFM(no_components=40,
                loss='bpr', # "logistic","bpr"
                learning_rate=0.01,
                item_alpha=0.4,
                user_alpha=0.1,
                random_state=42,
                k=5,
                n=15,
                max_sampled=100
                )
```

```
CPU times: user 1.07 ms, sys: 0 ns, total: 1.07 ms
Wall time: 2.64 ms
```

In [65]:
```python
model.fit((sparse_user_item > 0) * 1,  # user-item matrix из 0 и 1
          sample_weight=coo_matrix(user_item_matrix),
          user_features=csr_matrix(user_feat_lightfm.values).tocsr(),
          item_features=csr_matrix(item_feat_lightfm.values).tocsr(),
          epochs=20,
          num_threads=20,
          verbose=True)
```

```
Epoch: 100%|██████████| 20/20 [00:29<00:00,  1.45s/it]
```

Out[65]: `<lightfm.lightfm.LightFM at 0x7f9576463d50>`

In [ ]:
```python
# модель возвращает меру/скор похожести между соответствующим пользователем и товаром
predictions = model.predict(user_ids=users_ids_row,
                            item_ids=items_ids_row,
                            user_features=csr_matrix(user_feat_lightfm.values).tocsr(),
                            item_features=csr_matrix(item_feat_lightfm.values).tocsr(),
                            num_threads=10)
```

In [70]:
```
df_result_for_metrics = get_predicts(model, users_ids_row, items_ids_row)
df_result_for_metrics.apply(lambda row: custom_precision(row['item_id'], row['actual'],k=5), axis=1).mean()
```
Out[70]: 0.14111658456485865

In [ ]: