

22 | YouTubeDNN：召回算法的后起之秀（下）

2023-06-05 黄鸿波 来自北京

《手把手带你搭建推荐系统》



你好，我是黄鸿波。

上节课我们讲了关于 YouTubeDNN 的召回模型，接下来，我们来看看如何用代码来实现它。

我们在做 YouTubeDNN 的时候，要把代码分成两个步骤，第一个步骤是对数据的清洗和处理，第二个步骤是搭建模型然后把数据放进去进行训练和预测。

数据的清洗和处理

先来讲数据部分。

按照 YouTubeDNN 论文来看，输入的数据是用户的信息、视频的 ID 序列、用户搜索的特征和一些地理信息等其他信息。到了基于文章内容的信息流产品中，就变成了用户 ID、年龄、

性别、城市、阅读的时间戳再加上视频的 ID。我们把这些内容可以组合成 YouTubeDNN 需要的内容，最后处理成需要的 Embedding。

由于前面没有太多的用户浏览数据，所以我先造了一批数据，数据集我会放到 GitHub 上（后续更新），数据的形式如下。

shikey.com 转载分享

```
user_id,item_id,age,gender,city,timestamp,rating
1,5fa359ed0574d6367aaf1f69,18,male,珠海,2020-09-01 00:03:09,5
1,5fa35b56ac0ea832d2bc1476,18,male,珠海,2020-09-01 00:43:24,2
1,5fa35b23ac0ea832d2bc12c3,18,male,珠海,2020-09-01 00:51:24,2
1,5fa359f10574d6367aaf1fab,18,male,珠海,2020-09-01 00:29:29,5
1,5fa35b4bac0ea832d2bc1400,18,male,珠海,2020-09-01 00:28:37,4
1,5fa35b5fac0ea832d2bc14f0,18,male,珠海,2020-09-01 00:26:40,5
1,5fa359fb0574d6367aaf2043,18,male,珠海,2020-09-01 02:15:55,1
1,5fa35b5fac0ea832d2bc14f2,18,male,珠海,2020-09-01 01:50:48,2
1,5fa3598a0574d6367aaf1acf,18,male,珠海,2020-09-01 01:06:36,3
1,5fa35afdac0ea832d2bc111e,18,male,珠海,2020-09-01 01:32:37,2
1,5fa359d90574d6367aaf1e40,18,male,珠海,2020-09-01 01:23:40,2
1,5fa359830574d6367aaf1a56,18,male,珠海,2020-09-01 01:43:43,1
1,5fa359940574d6367aaf1b77,18,male,珠海,2020-09-01 01:14:19,4
1,5fa35b03ac0ea832d2bc1164,18,male,珠海,2020-09-01 00:53:55,5
1,5fa35b5eac0ea832d2bc14e2,18,male,珠海,2020-09-01 02:29:07,2
1,5fa35afbac0ea832d2bc10fb,18,male,珠海,2020-09-01 00:32:48,5
1,5fa359f70574d6367aaf2002,18,male,珠海,2020-09-01 00:24:24,3
1,5fa35b76ac0ea832d2bc160d,18,male,珠海,2020-09-01 00:22:49,4
1,5fa35b89ac0ea832d2bc1705,18,male,珠海,2020-09-01 00:01:36,1
1,5fa35b8bac0ea832d2bc1718,18,male,珠海,2020-09-01 00:09:18,3
1,5fa359930574d6367aaf1b6f,18,male,珠海,2020-09-01 00:26:38,5
1,5fa35afcac0ea832d2bc1106,18,male,珠海,2020-09-01 01:00:45,1
1,5fa35af8ac0ea832d2bc10da,18,male,珠海,2020-09-01 02:45:03,2
1,5fa359bc0574d6367aaf1d74,18,male,珠海,2020-09-01 00:45:09,4
1,5fa35b5eac0ea832d2bc14df,18,male,珠海,2020-09-01 00:18:56,4
1,5fa35b89ac0ea832d2bc1707,18,male,珠海,2020-09-01 01:57:48,4
1,5fa35b57ac0ea832d2bc1484,18,male,珠海,2020-09-01 01:35:59,2
1,5fa35b36ac0ea832d2bc13ac,18,male,珠海,2020-09-01 01:20:28,4
```

接下来我们就把这批数据处理成 YouTubeDNN 需要的形式。首先在 recommendation-class 项目中的 utils 目录下建立一个 preprocess.py 文件，作为处理数据的文件。

我们要处理这一批数据，需要下面五个步骤。

1. 加载数据集。
2. 处理数据特征。
3. 特征转化为模型输入。
4. 模型的搭建和训练。
5. 模型评估。

shikey.com转载分享

在正式写代码之前，需要安装几个库，如下。

复制代码

```
1 deepctr
2 deepmatch
3 tensorflow==2.2
4 pandas
```

我们可以使用 pip install 加上库名来安装它们，也可以把它们放在一个叫 requirements.txt 的文件中，使用 pip install -r 进行安装。

安装完成之后，我们来写 preprocess.py 的代码。为了能够让你看得更明白，我在函数里加了一些注释，先上代码。

复制代码

```
1 from tqdm import tqdm
2 import numpy as np
3 import random
4 from tensorflow.python.keras.preprocessing.sequence import pad_sequences
5
6 def gen_data_set(data, negsample=0):
7     data.sort_values("timestamp", inplace=True) #是否用排序后的数据集替换原来的数据，这
8     item_ids = data['item_id'].unique() #item需要进行去重
```

```

9
10 train_set = list()
11 test_set = list()
12 for reviewrID, hist in tqdm(data.groupby('user_id')): #评价过, 历史记录
13     pos_list = hist['item_id'].tolist()
14     rating_list = hist['rating'].tolist()
15
16     if negsample > 0: #负样本
17         candidate_set = list(set(item_ids) - set(pos_list)) #去掉用户看过的it
18         neg_list = np.random.choice(candidate_set, size=len(pos_list) * negsa
19     for i in range(1, len(pos_list)):
20         if i != len(pos_list) - 1:
21             train_set.append((reviewrID, hist[::-1], pos_list[i], 1, len(hist
22                 for negi in range(negsample):
23                     train_set.append((reviewrID, hist[::-1], neg_list[i * negsamp
24         else:
25             test_set.append((reviewrID, hist[::-1], pos_list[i], 1, len(hist[
26
27     random.shuffle(train_set) #打乱数据集
28     random.shuffle(test_set)
29     return train_set, test_set
30
31 def gen_model_input(train_set, user_profile, seq_max_len):
32     train_uid = np.array([line[0] for line in train_set])
33     train_seq = [line[1] for line in train_set]
34     train_iid = np.array([line[2] for line in train_set])
35     train_label = np.array([line[3] for line in train_set])
36     train_hist_len = np.array([line[4] for line in train_set])
37
38     """
39     pad_sequences数据预处理
40     sequences: 浮点数或整数构成的两层嵌套列表
41     maxlen: None或整数, 为序列的最大长度。大于此长度的序列将被截短, 小于此长度的序列将在后部填
42     dtype: 返回的numpy array的数据类型
43     padding: 'pre'或'post', 确定当需要补0时, 在序列的起始还是结尾补`
44     truncating: 'pre'或'post', 确定当需要截断序列时, 从起始还是结尾截断
45     value: 浮点数, 此值将在填充时代替默认的填充值0
46     """
47     train_seq_pad = pad_sequences(train_seq, maxlen=seq_max_len, padding='post',
48     train_model_input = {"user_id": train_uid, "item_id": train_iid, "hist_item_i
49         "hist_len": train_hist_len}
50     for key in {"gender", "age", "city"}:
51         train_model_input[key] = user_profile.loc[train_model_input['user_id']][k
52
53     return train_model_input, train_label

```

这段代码主要用于生成训练集和测试集以及模型的输入。它看起来有点长，我来分别解释一下。

`gen_data_set()` 函数接受一个数据集 (`data`) 和一个负采样 (`negsample`) 参数，返回一个训练集列表和一个测试集列表。该函数首先将数据集根据时间戳排序，然后从每一个用户的历史记录中选取正样本和负样本，并将它们保存到训练集和测试集中。

`gen_model_input()` 函数接受一个训练集列表、用户画像信息和序列最大长度参数，返回训练模型的输入和标签。该函数将训练集列表拆分成 `train_uid`、`train_seq`、`train_iid`、`train_label` 和 `train_hist_len` 五部分。

`train_uid` 和 `train_iid` 为用户 ID 和物品 ID。

`train_seq` 为历史交互序列。

`train_label` 为正负样本标签。

`train_hist_len` 为历史交互序列的长度。

此外，它对历史交互序列进行了填充处理 (`pad_sequences`)，并且将用户画像信息加入到训练模型的关键字中。最终，该函数返回训练模型的输入和标签。

在 `gen_data_set()` 函数中，首先使用 `data.sort_values("timestamp", inplace=True)` 函数将数据集按照时间戳排序，这是为了保证数据按照时间顺序排列，便于后续处理。接下来使用 `data['item_id'].unique()` 函数获取数据集中所有不重复的物品 ID。因为后续需要筛选出用户未曾购买过的物品，要先获取数据集中所有的物品 ID 以便后续处理。

接下来使用 `groupby()` 函数将用户 ID (`user_id`) 相同的数据分组。对于每一组数据，将其分成正样本和负样本。其中正样本为用户已经购买过的物品，负样本为用户未购买过的其他物品。如果 `negsample` 参数大于 0，则需要进行负采样。随机选取一些未曾购买过的物品作为负样本，并将它们保存到训练集列表中。最后，将正负样本数据以及其他信息（如历史交互序列、用户 ID 和历史交互序列的长度）保存到训练集列表和测试集列表中。

在 `gen_model_input()` 函数中，首先将训练集列表拆分成 5 个列表，分别保存用户 ID、物品 ID、历史交互序列、正负样本标签和历史交互序列长度。然后使用 `pad_sequences()` 函数对历史交互序列进行填充处理，将其变成长度相同的序列。最后，将用户画像信息加入到训练模型的关键字中，返回训练模型的输入和标签。

搭建模型进行训练和预测

shikkey.com 转载分享

当数据处理完成后，接下来就可以来做 YouTubeDNN 的模型部分了，我们在 `recall` 目录下新建一个文件，名字叫 `YouTubeDNN`，然后编写如下代码。

 复制代码

```
1 import pandas as pd
2 from sklearn.preprocessing import LabelEncoder
3 from models.recall.preprocess import gen_data_set, gen_model_input
4 from deepctr.feature_column import SparseFeat, VarLenSparseFeat
5 from tensorflow.python.keras import backend as K
6 from tensorflow.python.keras.models import Model
7 import tensorflow as tf
8 from deepmatch.models import *
9 from deepmatch.utils import recall_N
10 from deepmatch.utils import sampledsoftmaxloss
11 import numpy as np
12 from tqdm import tqdm
13
14
15 class YoutubeModel(object):
16     def __init__(self, embedding_dim=32):
17         self.SEQ_LEN = 50
18         self.embedding_dim = embedding_dim
19         self.user_feature_columns = None
20         self.item_feature_columns = None
21
22     def training_set_construct(self):
23         # 加载数据
24         data = pd.read_csv('../data/read_history.csv')
25         # 负采样个数
26         negsample = 0
27         # 特征编码
28         features = ["user_id", "item_id", "gender", "age", "city"]
29         features_max_idx = {}
30         for feature in features:
31             lbe = LabelEncoder()
32             data[feature] = lbe.fit_transform(data[feature]) + 1
33             features_max_idx[feature] = data[feature].max() + 1
```

```

34
35 # 抽取用户、物品特征
36 user_info = data[["user_id", "gender", "age", "city"]].drop_duplicates(
37 item_info = data[["item_id"]].drop_duplicates('item_id')
38 user_info.set_index("user_id", inplace=True)
39
40 # 构建输入数据
41 train_set, test_set = gen_data_set(data, negsample)
42 # 转化为模型的输入
43 train_model_input, train_label = gen_model_input(train_set, user_info,
44 test_model_input, test_label = gen_model_input(test_set, user_info, sel
45 # 用户端特征输入
46 self.user_feature_columns = [SparseFeat('user_id', features_max_idx['us
47 SparseFeat('gender', features_max_idx['gen
48 SparseFeat('age', features_max_idx['age'],
49 SparseFeat('city', features_max_idx['city'
50 VarLenSparseFeat(SparseFeat('hist_item_id'
51 self.embedding
52 self.SEQ_LEN, 'mean', 'hi
53 ]
54 # 物品端的特征输入
55 self.item_feature_columns = [SparseFeat('item_id', features_max_idx['it
56
57 return train_model_input, train_label, test_model_input, test_label, tr
58
59 def training_model(self, train_model_input, train_label):
60 K.set_learning_phase(True)
61 if tf.__version__ >= '2.0.0':
62     tf.compat.v1.disable_eager_execution()
63 # 定义模型
64 model = YouTubeDNN(self.user_feature_columns, self.item_feature_columns
65 user_dnn_hidden_units=(128, 64, self.embedding_dim))
66 model.compile(optimizer="adam", loss=sampledsoftmaxloss)
67 # 保存训练过程中的数据
68 model.fit(train_model_input, train_label, batch_size=512, epochs=20, ve
69 return model
70
71 def extract_embedding_layer(self, model, test_model_input, item_info):
72 all_item_model_input = {"item_id": item_info['item_id'].values, }
73 # 获取用户、item的embedding_layer
74 user_embedding_model = Model(inputs=model.user_input, outputs=model.use
75 item_embedding_model = Model(inputs=model.item_input, outputs=model.ite
76
77 user_embs = user_embedding_model.predict(test_model_input, batch_size=2
78 item_embs = item_embedding_model.predict(all_item_model_input, batch_si
79 print(user_embs.shape)
80 print(item_embs.shape)
81 return user_embs, item_embs
82

```

```

83     def eval(self, user_embs, item_embs, test_model_input, item_info, test_set)
84         test_true_label = {line[0]: line[2] for line in test_set}
85         index = faiss.IndexFlagIP(self.embedding_dim)
86         index.add(item_embs)
87         D, I = index.search(np.ascontiguousarray(user_embs), 50)
88         s = []
89         hit = 0
90
91         # 统计预测结果
92         for i, uid in tqdm(enumerate(test_model_input['user_id'])):
93             try:
94                 pred = [item_info['item_id'].value[x] for x in I[i]]
95                 recall_score = recall_N(test_true_label[uid], pred, N=50)
96                 s.append(recall_score)
97                 if test_true_label[uid] in pred:
98                     hit += 1
99             except:
100                 print(i)
101
102         # 计算召回率和命中率
103         recall = np.mean(s)
104         hit_rate = hit / len(test_model_input['user_id'])
105
106         return recall, hit_rate
107
108     def scheduler(self):
109         # 构建训练集、测试集
110         train_model_input, train_label, test_model_input, test_label, \
111         train_set, test_set, user_info, item_info = self.training_set_construct
112         #
113         self.training_model(train_model_input, train_label)
114
115         # 获取用户、item的layer
116         # user_embs, item_embs = self.extract_embedding_layer(model, test_model
117         # # 评估模型
118         # recall, hit_rate = self.eval(user_embs, item_embs, test_model_input,
119         # print(recall, hit_rate)
120
121
122 if __name__ == '__main__':
123     model = YoutubeModel()
124     model.scheduler()

```

我来详细地解释下这段代码。首先根据导入的模块，可以看出这段代码主要使用了下面表格里的几个工具和库。

工具和库名称	作用
Pandas	用于读取和处理CSV格式的数据文件
sklearn.preprocessing.LabelEncoder	用于对数据进行特征编码
models.recall.preprocess.gen_data_set 和 models.recall.preprocess.gen_model_input	用于将原始数据转化为模型的输入格式
deepctr.feature_column.SparseFeat 和 deepctr.feature_column.VarLenSparseFeat	用于构建用户和物品的特征输入
DeepMatch	用于构建和训练推荐模型
Faiss	用于高效搜索特征空间



首先我们使用下面的代码加载数据。

复制代码

```
1 data = pd.read_csv('../data/read_history.csv')
```

这行代码使用 Pandas 库来读取 CSV 格式的历史阅读记录数据文件，将其存储到 data 这个 DataFrame 对象中。

然后我们对数据进行特征编码。

复制代码


```
1 features = ["user_id", "item_id", "gender", "age", "city"]
2 features_max_idx = {}
3 for feature in features:
4     lbe = LabelEncoder()
```

```
5 data[feature] = lbe.fit_transform(data[feature]) + 1
6 features_max_idx[feature] = data[feature].max() + 1
```

这段代码使用 `sklearn.preprocessing.LabelEncoder` 对原始数据的几个特征进行编码，将连续或离散的特征转化为整数类型。这里编码的特征包括 `user_id`、`item_id`、`gender`、`age`、`city`。将特征编码后，将最大索引值保存到 `features_max_idx` 字典中。

接下来，我们使用下面的代码来构建了数据集。

```
1 train_set, test_set = gen_data_set(data, negsample)
```

 复制代码

这行代码使用 `gen_data_set` 函数将原始数据划分为训练集和测试集，同时进行负采样操作。该函数的输入参数为原始数据和负采样个数。输出结果为经过负采样后的训练集和测试集。


然后我们就可以调用之前的 `gen_model_input` 函数将训练集和测试集转化为模型的输入格式，包括训练集 / 测试集的用户 ID、历史物品 ID 序列、历史物品 ID 序列的长度和待预测物品 ID。这些数据会作为训练模型的输入。

```
1 train_model_input, train_label = gen_model_input(train_set, user_info, self.SEQ_L
2 test_model_input, test_label = gen_model_input(test_set, user_info, self.SEQ_LEN)
```

 复制代码

接着，我们使用 `deepctr` 库中的 `SparseFeat` 和 `VarLenSparseFeat` 函数，分别构建了用户和物品的特征输入。其中 `SparseFeat` 表示离散特征，`VarLenSparseFeat` 表示变长特征。具体地，用户特征输入由 4 个离散特征和一个变长特征（历史物品 ID 序列）组成，物品特征输入只有一个离散特征（物品 ID）。

```
1 # 用户端特征输入
2 self.user_feature_columns = [SparseFeat('user_id', features_max_idx['user_id'], 1
3                               SparseFeat('gender', features_max_idx['gender'], 16)
```

 复制代码


```

4         SparseFeat('age', features_max_idx['age'], 16),
5         SparseFeat('city', features_max_idx['city'], 16),
6         VarLenSparseFeat(SparseFeat('hist_item_id', features
7                                 self.embedding_dim, embe
8                                 self.SEQ_LEN, 'mean', 'hist_len')
9     ]
10 # 物品端的特征输入
11 self.item_feature_columns = [SparseFeat('item_id', features_max_idx['item_id'], e

```

shickey.com转载分享

然后我们使用 deepmatch 库构建了含有 DNN 的 YouTube 推荐模型。该模型的输入由上一步定义的用户和物品特征输入组成，其中 num_sampled 表示分类器使用的采样点的数目。在模型构建和编译后，使用 fit 函数进行训练。

 复制代码

```

1 # 定义模型
2 model = YouTubeDNN(self.user_feature_columns, self.item_feature_columns, num_samp
3                 user_dnn_hidden_units=(128, 64, self.embedding_dim))
4 model.compile(optimizer="adam", loss=sampledsoftmaxloss)
5 # 保存训练过程中的数据
6 model.fit(train_model_input, train_label, batch_size=512, epochs=20, verbose=1, v

```

最后，利用训练好的模型提取用户和物品的 Embedding Layer，以便后续计算召回率和命中率。具体地，使用 Model 函数将模型的输入和它的用户 / 物品 Embedding 层关联起来，然后调用 predict 函数计算得到预测结果。

 复制代码

```

1 user_embs = user_embedding_model.predict(test_model_input, batch_size=2 ** 12)
2 item_embs = item_embedding_model.predict(all_item_model_input, batch_size=2 ** 12

```

实际上，到这里整个数据处理和训练部分的代码就已经结束了，接下来，就是要做召回率和命中率的计算。在这个部分，我们利用 Faiss 库计算用户和物品 Embedding Layer 之间的近邻关系，并根据预测的物品列表计算召回率和命中率。具体来说就是根据用户 ID 索引到对应的 Embedding 向量，然后在物品 Embedding 向量集合中搜索近邻，得到预测的物品列表。最后，根据预测的物品列表和真实的物品 ID，计算召回率和命中率。

```

1 def eval(self, user_embs, item_embs, test_model_input, item_info, test_set):
2     test_true_label = {line[0]: line[2] for line in test_set}
3     index = faiss.IndexFlagIP(self.embedding_dim)
4     index.add(item_embs)
5     D, I = index.search(np.ascontiguousarray(user_embs), 50)
6     s = []
7     hit = 0
8
9     # 统计预测结果
10    for i, uid in tqdm(enumerate(test_model_input['user_id'])):
11        try:
12            pred = [item_info['item_id'].value[x] for x in I[i]]
13            recall_score = recall_N(test_true_label[uid], pred, N=50)
14            s.append(recall_score)
15            if test_true_label[uid] in pred:
16                hit += 1
17        except:
18            print(i)
19
20    # 计算召回率和命中率
21    recall = np.mean(s)
22    hit_rate = hit / len(test_model_input['user_id'])
23
24    return recall, hit_rate
25    整个流程实际上到这里就结束了，那么最后，我们使用一个scheduler函数将它们串起来：
26    def scheduler(self):
27        # 构建训练集、测试集
28        train_model_input, train_label, test_model_input, test_label, \
29        train_set, test_set, user_info, item_info = self.training_set_construct()
30        #
31        self.training_model(train_model_input, train_label)
32
33        # 获取用户、item的layer
34        # user_embs, item_embs = self.extract_embedding_layer(model, test_model_input,
35        # # 评估模型
36        # recall, hit_rate = self.eval(user_embs, item_embs, test_model_input, item_in
37        # print(recall, hit_rate)
38

```

这里有一点需要注意，Faiss 库目前在 Windows 上无法使用，必须在 Linux 上才行。因此，在最后的 Schedule 阶段，我将这段代码进行了注释。

整个 YouTubeDNN 的召回层训练和预测到这里就结束了。

总结

到这里，今天的课程就讲完了，接下来我们来对今天的课程做一个简单的总结，学完本节课你应该知道下面三大要点。

1. 在 YouTubeDNN 中，数据处理会经过加载数据集、处理数据特征、特征转化为模型输入、模型的搭建和训练、模型评估这 5 个部分。
2. YouTubeDNN 模型通过将用户历史行为序列嵌入到低维向量空间中，来学习用户和物品之间的关系。它的输入包括用户历史行为序列以及物品 ID，输出包括用户和物品的嵌入向量以及它们之间的相似度得分。
3. 熟悉使用 Python 来搭建一整套 YouTubeDNN 模型代码。

课后题

本节课学完了，我来给你留两道课后题。

1. 实现本节课的代码。
2. 根据我们前面的知识，自动生成数据集。

欢迎你在留言区与我交流讨论，如果这节课对你有帮助，也欢迎你推荐给朋友一起学习。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

精选留言 (3)



alexliu

2023-06-07 来自上海

老师，faiss.IndexFlagIP这个应该是faiss.IndexFlatIP吧？另外index.add(n,x)有两个参数，为什么在代码里只有item_embs一个参数？

ps: add源码如下：

```
def add(self, n, x):
```

```
r""" default add uses sa_encode"""  
return _swigfaiss.IndexFlatCodes_add(self, n, x)
```



peter

2023-06-06 来自北京

YoutubeDNN是拿来就能用吗？类似于工具软件那种，不需要开发。



爱极客

2023-06-06 来自广东

老师，后面会出一篇课后答疑的文章吗？

作者回复: 如果有需要的话，可以的，到时候我和极客时间的工作人员商量一下，看看是以文字的形式还是直播的形式比较好。

共 2 条评论 >

