基于梯度下降的矩阵分解的推荐系统

摘要 [中文]:

推荐系统RS 收集用户过去对一组项目的行为并将其用于推荐。该信息可以是明确的信息（例如用户的评分）、隐含信息（例如浏览历史）、人口统计、社交（如朋友、标签、信任者或受托人）或上下文信息（如时间或地点）。然而，获得准确的推荐是一项艰巨的任务，因为大多数推荐系统面临数据稀疏和冷启动问题。此外，它们不能有效地处理高维数据数据并且可扩展性较差 。

有下面几个常用有效的方法来解决这些问题：协同的技术过滤（CF）方法，矩阵分解（MF）技术，它提供降维并适用于稀疏数据。在本文中，我们使用随机梯度下降法分解评分矩阵，可以在一个较短的时间内分解评分矩阵为P,Q，再用P*Q即可预测原来的评分矩阵中的缺失值，也可以粗略预测另一个数据集上的评分，基于得到的预测评分，便可向用户推荐预测评分较高的电影

一、项目概述（阐明项目的科学价值与相关研究工作，描述项目主要内容）

推荐系统RS 收集用户过去对一组项目的行为并将其用于推荐。该信息可以是明确的信息（例如用户的评分）、隐含信息（例如浏览历史）、人口统计、社交（如朋友、标签、信任者或受托人）或上下文信息（如时间或地点）。然而，获得准确的推荐是一项艰巨的任务，因为大多数推荐系统面临数据稀疏和冷启动问题。此外，它们不能有效地处理高维数据数据并且可扩展性较差 。

有下面几个常用有效的方法来解决这些问题：其中，基于协同的技术过滤（CF）方法，矩阵分解（MF）技术，它提供降维并适用于稀疏数据。在本文中，我们使用随机梯度下降法分解评分矩阵，可以在一个较短的时间内分解评分矩阵为P,Q，再用P*Q即可预测原来的评分矩阵中的缺失值，也可以粗略预测另一个数据集上的评分，基于得到的预测评分，便可向用户推荐预测评分较高的电影

二、问题定义（提供问题定义的语言描述与数学形式）

有用户userId，电影movieId，用户对电影评分rating的数据集，一个用户仅对其中一些电影进行评分，现在要根据已有的评分来预测缺失的评分，然后根据得到的预测评分，向用户推荐预测评分较高的电影。

预测的方法是将评分矩阵分解为P*Q，可以用梯度下降法通过最小化重构误差得到的更新公式来迭代得到近似的P、Q

重构误差选取RMSE

更新公式为:

$$g_1 \leftarrow \nabla_{p_{uj}} = -\sum_{i:(u,i)\in\mathbb{K}} e_{ui}q_{ji} + \lambda p_{uj};$$
$$g_2 \leftarrow \nabla_{q_{ji}} = -\sum_{u:(u,i)\in\mathbb{K}} e_{ui}p_{uj} + \lambda q_{ji};$$
参数更新：
$$p_{uj} \leftarrow p_{uj} - \epsilon g_1;$$
$$q_{ji} \leftarrow q_{ji} - \epsilon g_2.;$$

根据上面更新公式迭代多次，如果用新得到的P、Q进行预测的重构误差更小则更新P与Q的值，最后得到最优的P与Q

三、方法（问题解决步骤和实现细节）

数据预处理

原始数据：train_data

data > train.csv

```
userId,movieId,rating
12725,2716,3.0
12725,1256,3.0
12725,1281,3.0
12725,2011,2.0
12725,2989,3.0
12725,674,5.0
12725,1407,5.0
12725,3204,4.0
12725,2985,3.0
12725,1193,4.0
12725,2160,5.0
12725,1088,2.0
12725,3094,3.0
12725,1392,4.0
12725,902,5.0
12725,923,5.0
12725,2002,1.0
12725,32,4.0
12725,3701,3.0
```

```
train_dataset = pd.read_csv("data/train.csv", usecols=range(3),
dtype=dict(dtype))
def load_dataset(self, train_data, dev_data):
    self.train_data = pd.DataFrame(train_data)
    self.dev_data = pd.DataFrame(dev_data)

self.users_ratings = train_data.groupby(self.columns[0]).agg([list])
[[self.columns[1], self.columns[2]]]
self.items_ratings = train_data.groupby(self.columns[1]).agg([list])
[[self.columns[0], self.columns[2]]]
```

这样把训练数据变成了Dataframe形式：train_data，生成users_ratings，items_rating便于后面构造P与Q矩阵

矩阵分解

将矩阵分解为P*Q的形式

采用随机梯度下降的方法

初始化P与Q

```
def _init_matrix(self):
        '''
        ********************************
        用户矩阵P和物品矩阵Q的初始化也对算法优化有一定帮助，更好的初始化相当于先验信息。
        加分项：
        - 思考初始化的一些方法，正态分布等等；
        - 其他初始化方法？
        ********************************
        '''
        # User-LF
        P = dict(zip(
            self.users_ratings.index,
            np.random.rand(len(self.users_ratings),
self.hidden_size).astype(np.float32)
        ))
        # Item-LF
        Q = dict(zip(
            self.items_ratings.index,
            np.random.rand(len(self.items_ratings),
self.hidden_size).astype(np.float32)
        ))
        return P, Q
```

随机梯度下降sgd

用更新公式更新P与Q，此处仅考虑正则化

```
def sgd(self, P, Q):
        '''
        ********************************
        基本分：请实现【随机梯度下降】
        加分项：进一步优化如下
        - 考虑偏置项
        - 考虑正则化
        - 考虑协同过滤
        ********************************
        '''
        print('sgd')
        for uid,iid,R_ui in self.train_data.itertuples(index=False):
            e_ui=R_ui-np.dot(P[uid],Q[iid])
            P[uid]-=self.lr*(-e_ui*Q[iid]+self.reg_p*P[uid])
            Q[iid]-=self.lr*(-e_ui*P[uid]+self.reg_q*Q[iid])
        return P, Q
```

更新规则

一共要迭代10个Epoch，每次迭代后，如果预测的结果更加准确，则保存这个P,Q

而判断其是不是更准确的方法是在验证集dev_data上验证，即根据验证集进行预测，然后求验证集和预测之间的RMSE，如果当前的RMSE更低，则保存

```
def train(self, optimizer_type: str):
        '''
        训练模型
        :param dataset: uid, iid, rating
        :return:
        '''
        P, Q = self._init_matrix() # 初始化user、item矩阵
        best_metric_result = None
        best_P, best_Q = P, Q


        for i in range(self.epoch):
            print("Epoch: %d"%i)
            # 当前epoch，执行优化算法：
            if optimizer_type == "SGD": # 随机梯度下降
                P, Q = self.sgd(P, Q)
            elif optimizer_type == "BGD": # 批量梯度下降
                P, Q = self.bgd(P ,Q, batch_size=self.batch_size)
            else:
                raise NotImplementedError("Please choose one of SGD and BGD.")
            # 当前epoch优化后，在验证集上验证，并保存目前最好的P和Q
            metric_result = self.eval(P, Q)
            # 如果当前的RMSE更低，则保存
            print("Current dev metric result: {}".format(metric_result))
            if best_metric_result is None or metric_result <=
best_metric_result:
                best_metric_result = metric_result
                best_P, best_Q = P, Q
                print("Best dev metric result: {}".format(best_metric_result))

        # 最后保存最好的P和Q
        np.savez("best_pq.npz", P=best_P, Q=best_Q)
```

预测评分

```
    def predict_user_item_rating(self, uid, iid, P, Q):
        # 如果uid或iid不在，我们使用全剧平均分作为预测结果返回
        if uid not in self.users_ratings.index or iid not in
self.items_ratings.index:
            return self.globalMean

        p_u = P[uid]
        q_i = Q[iid]

        return np.dot(p_u, q_i)
```

预测评分$\hat{R} =< P[u], Q[i] >$

评估模型

以下是采用sgd迭代10次的RMSE，当第10次迭代结束后，RMSE下降到了0.88549

```
PS D:\学习资料\数据科学与工程算法\projects\project3> python main.py --train
Reading training data ...
Reading developing data ...
Starting training ...
Epoch: 0
Current dev metric result: 0.9731498611402187
Best dev metric result: 0.9731498611402187
Epoch: 1
Current dev metric result: 0.9318780937616398
Best dev metric result: 0.9318780937616398
Epoch: 2
Current dev metric result: 0.9153966503716564
Best dev metric result: 0.9153966503716564
Epoch: 3
Current dev metric result: 0.9042194531027072
Best dev metric result: 0.9042194531027072
Epoch: 4
Current dev metric result: 0.8966394529720655
Best dev metric result: 0.8966394529720655
Epoch: 5
Current dev metric result: 0.8917389047827565
Best dev metric result: 0.8917389047827565
Epoch: 6
Current dev metric result: 0.8886577851363265
Best dev metric result: 0.8886577851363265
Epoch: 7
Current dev metric result: 0.8868395949163173
Best dev metric result: 0.8868395949163173
Epoch: 8
Current dev metric result: 0.8858888995025499
Best dev metric result: 0.8858888995025499
Epoch: 9
Current dev metric result: 0.8854921490834647
Best dev metric result: 0.8854921490834647
Finish training.
```

Ln 122, Col 57 Spa

补全缺失评分

test.csv里面的userId与movieId没有实际评分，所以用我们训练得到的P与Q对其评分进行预测补全

data > test.csv

```
ID,userId,movieId
0,7314,553
1,18570,1485
2,10484,191
3,1841,2269
4,1234,1073
5,11514,5391
6,15364,539
7,6870,329
8,16730,2616
9,5467,2355
10,8631,799
11,3219,512
12,9273,1393
13,8666,5008
```

```
 def test(self, test_data):
        '''预测测试集榜单数据'''
        # 预测结果可以提交至：https://www.kaggle.com/competitions/dase-recsys/overview
        test_data = pd.DataFrame(test_data)
        # 加载训练好的P和Q
        best_pq = np.load("best_pq.npz", allow_pickle=True)
        P, Q = best_pq["P"][()], best_pq["Q"][()]

        save_results = list()
        for id,uid, iid in test_data.itertuples(index=False):
            pred_rating = self.predict_user_item_rating(uid, iid, P, Q)
            pred_tuple=[uid,iid,pred_rating]
            save_results.append(pred_tuple)

        log_path = "submit_results.csv"
        if os.path.exists(log_path):
            os.remove(log_path)
        file = open(log_path, 'a+', encoding='utf-8', newline='')
        csv_writer = csv.writer(file)
        csv_writer.writerow([f'userId', 'movieId','pred_rating'])
        for uid, iid,pred_rating in save_results:
            csv_writer.writerow([uid, iid,pred_rating])
        file.close()
```

得到的预测评分保存到submit_results.csv中

submit_results.csv

```
userId,movieId,pred_rating
7314,553,3.4791987
18570,1485,2.990498
10484,191,3.261292
1841,2269,3.1153936
1234,1073,3.1842918
11514,5391,4.143806
15364,539,3.647659
6870,329,3.5690036
16730,2616,2.4516537
5467,2355,3.3075528
8631,799,2.9918594
3219,512,1.8094159
9273,1393,3.8201132
8666,5008,4.301792
2950,2414,3.5026631
1841,17,3.3650227
3476,2469,2.5229352
7490,2450,1.7360973
```

应用推荐

输入一个用户，向其推荐该用户预测评分最高的movieId即可

```
def recommend(self,userid):
        # 加载训练好的P和Q
        best_pq = np.load("best_pq.npz", allow_pickle=True)
        P, Q = best_pq["P"][()], best_pq["Q"][()]
        rating_list=[]
        for movieid in self.items_ratings.index:
            pred_rating=np.dot(P[userid],Q[movieid])
            rating_sublist=[movieid,pred_rating]
            rating_list.append(rating_sublist)
        user_rating=pd.DataFrame(rating_list,columns=['movieId','pred_rating'])

 max_item=user_rating.loc[user_rating['pred_rating']==user_rating['pred_rating']
.max()]
        print("the recommended item for user %d is item %d, the predicted rating
is %f"%(userid,max_item['movieId'],max_item['pred_rating']))
```

例子如下：

向用户8推荐：

```
PS D:\学习资料\数据科学与工程算法\projects\project3> python main.py --recommend 8
Reading training data ...
Reading developing data ...
start recommending
the recommended item for user 8 is item 58559, the predicted rating is 5.022246
finish recommending
```

可以看到对用户8推荐了movie 58559，这是user 8 预测评分最高的电影，预测评分为5.022246，超过了实际的评分上限，说明预测与实际之间还有误差，可以把超过5.0的预测评分都记作5.0，这样预测模型会更准确些

四、实验结果（验证提出方法的有效性和高效性）

准确性：

sgd

```
PS D:\学习资料\数据科学与工程算法\projects\project3> python main.py --train
Reading training data ...
Reading developing data ...
Starting training ...
Epoch: 0
Current dev metric result: 0.9731498611402187
Best dev metric result: 0.9731498611402187
Epoch: 1
Current dev metric result: 0.9318780937616398
Best dev metric result: 0.9318780937616398
Epoch: 2
Current dev metric result: 0.9153966503716564
Best dev metric result: 0.9153966503716564
Epoch: 3
Current dev metric result: 0.9042194531027072
Best dev metric result: 0.9042194531027072
Epoch: 4
Current dev metric result: 0.8966394529720655
Best dev metric result: 0.8966394529720655
Epoch: 5
Current dev metric result: 0.8917389047827565
Best dev metric result: 0.8917389047827565
Epoch: 6
Current dev metric result: 0.8886577851363265
Best dev metric result: 0.8886577851363265
Epoch: 7
Current dev metric result: 0.8868395949163173
Best dev metric result: 0.8868395949163173
Epoch: 8
Current dev metric result: 0.8858888995025499
Best dev metric result: 0.8858888995025499
Epoch: 9
Current dev metric result: 0.8854921490834647
Best dev metric result: 0.8854921490834647
Finish training.
```

Ln 122, Col 57 Spa

最后的RMSE为0.88549

用时：

评分矩阵：20225*65133，训练P和Q一共用时331 s，可见用时较快，在可接受范围内

```
Finish training.
time cost 331.20567440986633 s
```

五、结论（对使用的方法可能存在的不足进行分析，以及未来可能的研究方向进行讨论）

使用随机梯度下降法分解评分矩阵，可以在一个较短的时间内分解评分矩阵为P,Q，再用P*Q即可预测原来的评分矩阵中的缺失值，也可以粗略预测另一个数据集上的评分，基于得到的预测评分，便可向用户推荐预测评分较高的电影

不足之处：预测的评分没有考虑评分只能在[0,5]内，应该把小于0的预测评分置为0，大于5的评分置为5

本次实验中只用了随机梯度下降法，还可以再尝试批量梯度下降法

并且只优化了正则化这一步，还可以考虑偏置项与协同过滤