《机器学习》
-推荐系统实验手册

华为技术有限公司

目录

1 实验介绍

推荐系统（RS）主要是指应用协同智能（collaborative intelligence）做推荐的技术。推荐系统的两大主流类型是基于内容的推荐系统和协同过滤（Collaborative Filtering）。另外还有基于知识的推荐系统（包括基于本体和基于案例的推荐系统）是一类特殊的推荐系统，这类系统更加注重知识表征和推理。

本章实验将会学习使用协同过滤技术分析用户对电影的评分数据，并基于这个数据建立一个推荐系统，根据用户输入的一部感兴趣的电影，为其推荐其他可能感兴趣的电影。此案例中，我们使用的数据集是用户对电影的评分数据，包含用户数据、评分数据、电影数据。

本章实验难度：中级。

中级实验：①电影推荐案例实验

1.1 实验目的

本案例将掌握如何使用机器学习算法全流程构建一个电影推荐系统的方案。通过本实验的学习，您将能够：

掌握如何使用华为云 ModelArts Notebook 上传数据、执行 Python 代码；

掌握如何载入、查阅、清洗、合并用户的数据，并计算物品相似度矩阵；

掌握如何使用物品的相似度矩阵，为客户进行推荐其他物品。

1.2 实验清单

实验	简述	难度	软件环境	开发环境
电影推荐案例	基于电影评分数据，使用推荐算法进行电影推荐	中级	Python3	ModelArts

1.3 开发平台介绍

ModelArts 是面向开发者的一站式 AI 开发平台，为机器学习与深度学习提供海量数据预处理及半自动化标注、大规模分布式 Training、自动化模型生成，及端-边-云模型按需部署能力，帮助用户快速创建和部署模型，管理全周期 AI 工作流。

具体内容请参考平台介绍 PPT。

1.4 背景知识

1.4.1 协同过滤算法介绍

协同过滤(Collaborative Filtering)作为推荐算法中最经典的类型，包括在线的协同和离线的过滤两部分。所谓在线协同，就是通过在线数据找到用户可能喜欢的物品，而离线过滤，则是过滤掉一些不值得推荐的数据，比如推荐值评分低的数据，或者虽然推荐值高但是用户已经购买的数据。

协同过滤的模型一般为 m 个物品，n 个用户的数据，只有部分用户和部分数据之间是有评分数据的，其它部分评分是空白，此时我们要用已有的部分稀疏数据来预测那些空白的物品和数据之间的评分关系，找到最高评分的物品推荐给用户。

一般来说，协同过滤推荐分为三种类型。

第一种是基于用户(user-based)的协同过滤

第二种是基于物品(item-based)的协同过滤

第三种是基于模型(model based)的协同过滤

基于用户(user-based)的协同过滤主要考虑的是用户和用户之间的相似度，只要找出相似用户喜欢的物品，并预测目标用户对对应物品的评分，就可以找到评分最高的若干个物品推荐给用户。

而基于物品(item-based)的协同过滤和基于用户的协同过滤类似，只不过这时我们转向找到物品和物品之间的相似度，只有找到了目标用户对某些物品的评分，那么我们就可以对相似度高的类似物品进行预测，将评分最高的若干个相似物品推荐给用户。比如你在网上买了一本机器学习相关的书，网站马上会推荐一堆机器学习，大数据相关的书给你，这里就明显用到了基于物品的协同过滤思想。

基于用户的协同过滤和基于物品的协同过滤对比：基于用户的协同过滤需要在线找用户和用户之间的相似度关系，计算复杂度肯定会比基于基于物品的协同过滤高。但是可以帮助用户找到新类别的有惊喜的物品。而基于物品的协同过滤，由于考虑的物品的相似性一段时间不

会改变，因此可以很容易的离线计算，准确度一般也可以接受，但是推荐的多样性来说，就很难带给用户惊喜了。

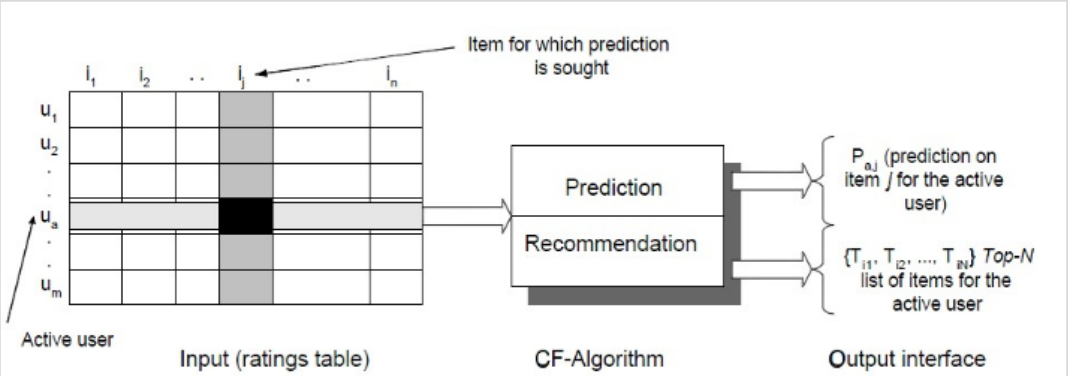


Figure 1: The Collaborative Filtering Process.

1.4.2 基于物品的协同过滤算法

假设某天你看了电影《蚁人》，那么影院的 APP 可能会给你推荐电影《复仇者联盟》。因为机器经过判断得出这两者相似度很高，你既然会喜欢《蚁人》那么喜欢《复仇者联盟》的概率会很大。因此，**基于物品的协同过滤算法就是给用户推荐那些和他们之前喜欢的物品相似的物品。**由于本实验采用的是基于物品的协同过滤算法，所以将着重介绍此算法。而基于用户的协同过滤与基于模型的协同过滤可自行查阅资料。

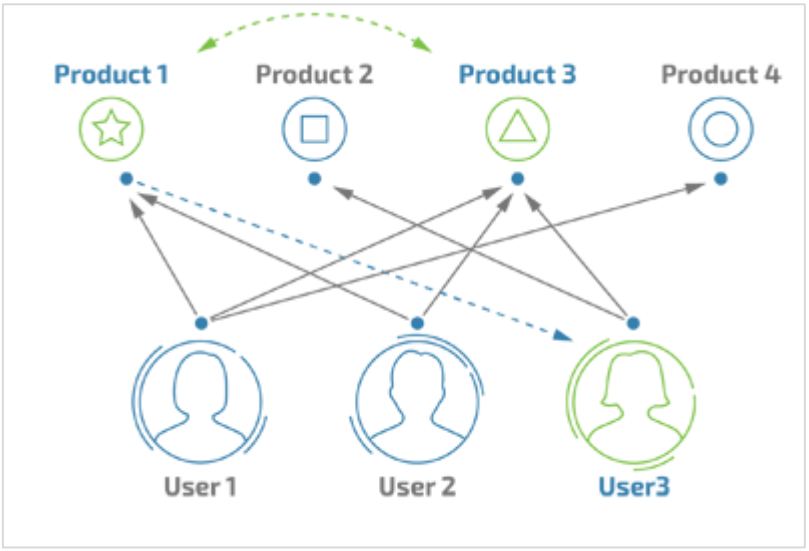


图1-1 基于物品的协同过滤

步骤 2 **计算物品间的相似度：**这里我们使用余弦相似度算法（Ochiai coefficient 落和系数）。

$$w_{ij} = \frac{|N(i) \cap N(j)|}{\sqrt{|N(i)||N(j)|}}$$

步骤 3 **倒排：过滤出完全没关系的物品i和j。**

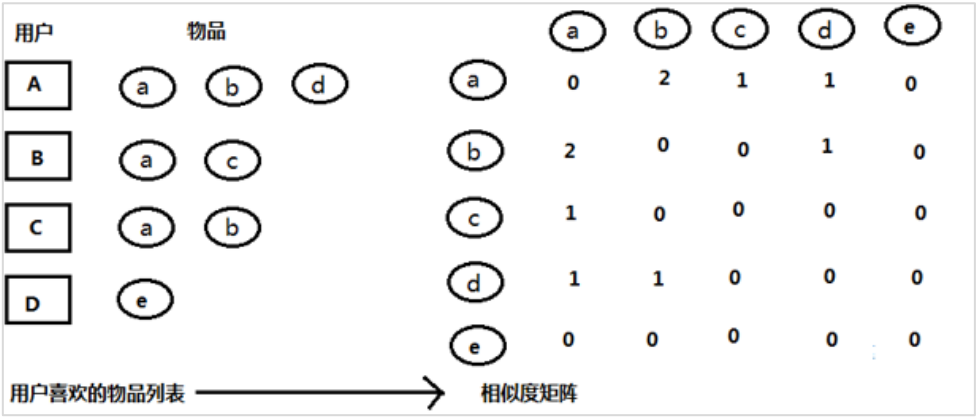


图1-2 倒排

通过计算得到w_{ab} =0.82，w_{ac}=0.58，w_{ad}=0.58

步骤 4 计算用户对物品的兴趣度

用户u对物品j的感兴趣程度计算公式：

$$p(u,j) = \sum_{i \in N(u) \cap S(j,k)} w_{ji} r_{ui}$$

此公式与上面基于用户的公式非常相像。$S(j,k)$表示与 j 物品最相似的k个物品的集合。$N(u)$表示用户喜欢的物品集合。w_{ji}表示物品之间的相似度。r_{ui}是用户u对物品i的兴趣。（对于隐反馈数据集，如果用户u对物品i有过行为，即可令r_{ui}=1。）

4. Top-N 分析

同上，计算出$p(u,j)$之后，对其排序，取前几名作为推荐物品推荐给用户。

2 电影推荐实验

2.1 实验介绍

2.1.1 简介

本实验用协同过滤技术分析用户对电影的评分数据，并基于这个数据建立一个推荐系统，根据用户输入的一部感兴趣的电影，为其推荐其他可能感兴趣的电影。此案例中，我们使用的数据集是用户对电影的评分数据，包含用户数据、评分数据、电影数据。

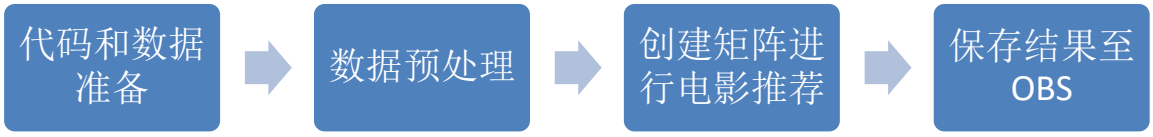
2.1.2 实验目的

- 掌握协同过滤算法的基本原理。
- 掌握推荐系统的整体流程。

2.2 实验环境要求

华为云 ModelArts

2.3 实验总体设计



2.4 实验过程

2.4.1 代码和数据准备

步骤 1　案例配置信息填写

案例中需要将运行结果上传至 OBS 中，因此，在实验之前需创建自己的 obs 桶，并且桶所在区域与所创建的 notebook 属于一个区域。我们需要设置以下相关参数（使用自己真实的桶名和唯一 ID 替换掉*号）：

BUCKET_NAME ： 自己的 OBS 桶名

UNIQUE_ID：唯一 ID，填写自己的学号或者 IAM 子账号名称相关代码、数据和模型都已准备好存放在 OBS 中，执行下面一段代码即可将其拷贝到 Notebook 中。

```
BUCKET_NAME = '**********'
#UNIQUE_ID = '**********'
#OBS_BASE_PATH = BUCKET_NAME + '/' + UNIQUE_ID
OBS_BASE_PATH = BUCKET_NAME
```

步骤 2 初始化 ModelArts SDK

```
from modelarts.session import Session
session = Session()
```

步骤 3 下载源代码和数据

这一步准备案例所需的源代码和数据，相关资源已经保存在 OBS 中，我们通过 ModelArts SDK 将资源下载到本地，并解压到当前目录下。解压后，当前目录包含 ml-100k 目录，存有数据集。

```
session.download_data(bucket_path="ai-course-common-20-bj4/movie_recommendation/movie_recommendation.tar.gz", path="./movie_recommendation.tar.gz")
# 使用 tar 命令解压资源包
!tar xf movie_recommendation.tar.gz
```

输出结果：

```
Successfully download file ai-course-common-20-bj4/movie_recommendation/movie_recommendation.tar.gz from OBS to local ./movie_recommendation.tar.gz
```

步骤 4 导入基本工具库

执行下面方框中的这段代码，可以导入本次实验中使用的 Python 开发基本工具库。numpy 是数据分处理工具,pandas 是文件读取和数据处理工具，scipy 是一个科学计算库，这里导入了 cosine, correlation 两种距离计算方法。此段代码只是引入 Python 包，无回显（代码执行输出）。

```
# import same usefull libraries
import numpy as np
import pandas as pd
from sklearn.metrics import pairwise_distances
from scipy.spatial.distance import cosine, correlation
```

2.4.2 查看数据

使用 pandas 库导入用户的个人信息

用户数据的字段描述如下：

user_id：用户 ID

age：用户年龄

sex：性别

occupation：职业

zip_code：邮编

评分数据的字段描述如下：

user_id：用户 ID

movide_id：电影 ID

rating：评分

unix_tiemstamp：评分时间

电影数据的字段描述如下：

user_id：用户 ID

movide_id：电影 ID

rating：评分

unix_tiemstamp：评分时间

步骤 1　查看用户信息

（1）定义用户信息

```
# 用户信息
users_cols = ['user_id', 'age', 'sex', 'occupation', 'zip_code']
users = pd.read_csv('./ml-100k/u.user', sep='|', names=users_cols, parse_dates=True)
```

（2）打印用户信息

打印前 5 个用户的个人信息，可以看到用户个人信息包含用户 ID（user_id）、年龄(age)、性别(sex)、职业(occupation)、邮编(zip_code)。

```
users.head()
```

输出结果：

	user_id	age	sex	occupation	zip_code
0	1	24	M	technician	85711
1	2	53	F	other	94043
2	3	23	M	writer	32067
3	4	24	M	technician	43537
4	5	33	F	other	15213

图2-1 前 5 个用户的个人信息

（3）打印数据表格的大小，可以看到这是一个 943x5 的矩阵， 其中 943 代表有 943 个用户，5 代表每个用户有 5 项信息。

```
users.shape
```

输出结果：

```
(943, 5)
```

步骤 2 查看用户的评分信息

参照查看用户信息的实现，请自行编码实现以下步骤：

（1）使用 Pandas 库导入用户的评分信息。

（2）打印前 5 个评分信息，可以看到评分信息包含用户 ID（user_id）、电影 ID(movide_id)、评分(rating)、评分时间(unix_tiemstamp)。

（3）打印数据表格的大小，可以看到这是一个 10000x4 的矩阵， 其中 10000 代表有 10000 条评论，4 代表每个评论有 5 项信息。

步骤 3 查看电影信息

参照查看用户信息的实现，请自行编码实现以下步骤：

（1）使用 pandas 库导入电影的信息。

（2）打印前 5 个电影信息，可以看到电影信息包含电影 ID(movide_id)、电影名称(title)、发布时间(release_date)、视频发布时间(video_release_date)、评论网站 URL 链接(imdb_url)。

（3）打印数据表格的大小，可以看到这是一个 1682x5 的矩阵， 其中 1682 代表有 1682 部电影，5 代表每部电影有 5 项信息。

2.4.3 数据预处理

步骤 1 数据合并

（1）把电影数据表、评论数据表、用户信息数据表进行合并，最后得到一张数据信息总表。

```
# Merging movie data with their ratings
movie_ratings = pd.merge(movies, ratings)
# merging movie_ratings data with the User's dataframe
df = pd.merge(movie_ratings, users)
```

（2）查看数据信息总表

```
df.head()
```

输出结果：

	movie_id	title	release_date	video_release_date	imdb_url	user_id	rating	unix_timestamp	age	sex	occupation	zip_code
0	1	Toy Story (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Toy%20Story%2...	308	4	887736532	60	M	retired	95076
1	4	Get Shorty (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Get%20Shorty%...	308	5	887737890	60	M	retired	95076
2	5	Copycat (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Copycat%20(1995)	308	4	887739608	60	M	retired	95076
3	7	Twelve Monkeys (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Twelve%20Monk...	308	4	887738847	60	M	retired	95076
4	8	Babe (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Babe%20(1995)	308	5	887736696	60	M	retired	95076

图2-2 前 5 个用户的数据信息总表

（3）打印数据总表的大小，可以看到这是一个 10000x12 的矩阵， 其中 10000 代表有 10000 条评论，12 代表每条评论有 12 项属性，包括电影 ID，电影信息，用户 ID，评分，用户信息等。

```
df.shape
```

输出结果：

```
(100000, 12)
```

步骤 2 数据清洗

（1）去除一些无效或不需要的信息，比如 video_release_date、imdb_url、unix_timestamp。

```
# pre-processing
# dropping colums that aren't needed
df.drop(df.columns[[3, 4, 7]], axis=1, inplace=True)
ratings.drop("unix_timestamp", inplace=True, axis=1)
movies.drop(movies.columns[[3, 4]], inplace=True, axis=1)
```

（2）查看新的数据信息总表。

```
df.head()
```

输出结果：

	movie_id	title	release_date	user_id	rating	age	sex	occupation	zip_code
0	1	Toy Story (1995)	01-Jan-1995	308	4	60	M	retired	95076
1	4	Get Shorty (1995)	01-Jan-1995	308	5	60	M	retired	95076
2	5	Copycat (1995)	01-Jan-1995	308	4	60	M	retired	95076
3	7	Twelve Monkeys (1995)	01-Jan-1995	308	4	60	M	retired	95076
4	8	Babe (1995)	01-Jan-1995	308	5	60	M	retired	95076

图2-3 前 5 个用户的新数据信息总表

2.4.4 创建相关矩阵进行电影推荐

步骤 1 创建用户-电影评分矩阵

（1）根据评分数据表(ratings),创建用户-电影评分矩阵。

```
# Pivot Table(This creates a matrix of users and movie_ratings)
ratings_matrix = ratings.pivot_table(index=['movie_id'], columns=['user_id'], values='rating').reset_index(drop=True)
ratings_matrix.fillna(0, inplace=True)
cmu = ratings_matrix
```

（2）查看用户-电影评分矩阵，其中每一行代表每部电影来自所有用户的评分；每一列代表每个用户对所有电影的评分。

```
cmu.head()
```

输出结果：

user_id	1	2	3	4	5	6	7	8	9	10	...	934	935	936	937	938	939	940	941	942	943
0	5.0	4.0	0.0	0.0	4.0	4.0	0.0	0.0	0.0	4.0	...	2.0	3.0	4.0	0.0	4.0	0.0	0.0	5.0	0.0	0.0
1	3.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	...	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.0
2	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	3.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	4.0	...	5.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0
4	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	0.0	0.0	0.0	0.0

5 rows × 943 columns

图2-4 用户-电影评分矩阵

打印数据总表的大小，可以看到这是一个 1682x943 的矩阵， 其中 1682 代表有 1682 部电影，943 代表有 943 个用户。

步骤 2　创建电影的相似矩阵

（1）根据 943 位用户对每部电影的评分，创建 1682 部电影的相似矩阵，矩阵大小为 1682x1682 的形状。

```
# Cosine Similarity(Creates a cosine matrix of similaraties ..... which is the pairwise distances
# between two items )

movie_similarity = 1 - pairwise_distances(ratings_matrix.values, metric="cosine")
np.fill_diagonal(movie_similarity, 0)
ratings_matrix = pd.DataFrame(movie_similarity)
```

（2）查看电影相似矩阵，以第三行，第二列为例，数值为 0.273069，这个值代表第二部电影与第三部电影的相似度。

```
ratings_matrix.head()
```

输出结果：

	0	1	2	3	4	5	6	7	8	9	...	1672	1673	1674	1675	1676	1677
0	0.000000	0.402382	0.330245	0.454938	0.286714	0.116344	0.620979	0.481114	0.496288	0.273935	...	0.035387	0.0	0.000000	0.000000	0.035387	0.0
1	0.402382	0.000000	0.273069	0.502571	0.318836	0.083563	0.383403	0.337002	0.255252	0.171082	...	0.000000	0.0	0.000000	0.000000	0.000000	0.0
2	0.330245	0.273069	0.000000	0.324866	0.212957	0.106722	0.372921	0.200794	0.273669	0.158104	...	0.000000	0.0	0.000000	0.000000	0.032292	0.0
3	0.454938	0.502571	0.324866	0.000000	0.334239	0.090308	0.489283	0.490236	0.419044	0.252561	...	0.000000	0.0	0.094022	0.094022	0.037609	0.0
4	0.286714	0.318836	0.212957	0.334239	0.000000	0.037299	0.334769	0.259161	0.272448	0.055453	...	0.000000	0.0	0.000000	0.000000	0.000000	0.0

5 rows × 1682 columns

图2-5 电影相似矩阵

（3）查看形状

```
ratings_matrix.shape
```

输出结果：

```
(1682, 1682)
```

步骤 3　根据电影的相似矩阵，推荐电影

（1）当用户查看了 Copycat (1995)，那么根据电影的相似矩阵，推荐与 Copycat (1995) 近似分数比较高的电影。具体如下：根据电影名 Copycat (1995)， 查询电影信息表 (movies)中的 index 序号。

```
# user_inp=input('Enter the reference movie title based on which recommendations are to be made: ')
user_inp = "Copycat (1995)"
inp = movies[movies['title'] == user_inp].index.tolist()
inp = inp[0]
```

（2）根据电影 index 序号，去查电影的相似矩阵，得到 1682 部电影的相似值，并打印表格中的 5 部电影的相似值。

```
movies['similarity'] = ratings_matrix.iloc[inp]
movies.columns = ['movie_id', 'title', 'release_date', 'similarity']
movies.head(5)
```

输出结果：

	movie_id	title	release_date	similarity
0	1	Toy Story (1995)	01-Jan-1995	0.286714
1	2	GoldenEye (1995)	01-Jan-1995	0.318836
2	3	Four Rooms (1995)	01-Jan-1995	0.212957
3	4	Get Shorty (1995)	01-Jan-1995	0.334239
4	5	Copycat (1995)	01-Jan-1995	0.000000

图2-6 5 部电影的相似值

（3）把相似值进行排序，并打印最相似的 5 部电影

```
recommended_movies = movies.sort_values(["similarity"], ascending=False)[1:6]
print("Recommended movies based on your choice of ", user_inp, ": \n", recommended_movies)
```

输出结果：

```
Recommended movies based on your choice of  Copycat (1995) :
      movie_id                           title release_date  similarity
218       219  Nightmare on Elm Street, A (1984)  01-Jan-1984    0.472725
53         54                  Outbreak (1995)  01-Jan-1995    0.472399
233       234                      Jaws (1975)  01-Jan-1975    0.450780
52         53       Natural Born Killers (1994)  01-Jan-1994    0.445242
97         98   Silence of the Lambs, The (1991)  01-Jan-1991    0.440996
```

图2-7 最相似的 5 部电影

2.4.5 保存结果至 OBS

我们将推荐的前 5 部电影的信息保存到文本文件中，并上传到 OBS，以便以后查看。

步骤 1　写入本地文件

将电影的信息写入到文本文件中。会打印成功保存的信息。

```
import os
if not os.path.exists('results'):
    os.mkdir('results') #  创建本地保存路径
with open('./results/recommended_movies.txt', 'w') as f:
    f.write(str(recommended_movies)) #  写入本地文本文件

print('Successfully saved!')
```

输出结果：

```
Successfully saved!
```

步骤 2　上传文件至 OBS

使用 ModelArts SDK 上传本地文件至 OBS。可以看到上传成功的日志。

```
session.upload_data(bucket_path=OBS_BASE_PATH + '/movie_recommendation/results/',
path='./results/recommended_movies.txt')
```

输出结果：

```
Successfully upload file ./results/recommend_movies.txt to OBS professional-
construction/movie_recommendtion/results
```

注意：该案例所在的 OBS 存储路径下，results 目录下，有模型文件 recommended_movies.txt。

2.5 实验小结

本实验基于 ModelArts 平台使用协同过滤算法中基于物品的协同过滤实现了电影推荐案例。

2.6 思考题参考答案

2.6.1 查看用户的评分信息

（1）使用 Pandas 库导入用户的评分信息

```
# Ratings
rating_cols = ['user_id', 'movie_id', 'rating', 'unix_timestamp']
```

```
ratings = pd.read_csv('./ml-100k/u.data', sep='\t', names=rating_cols)
```

（2）打印前 5 个评分信息，可以看到评分信息包含用户 ID（user_id）、电影 ID(movide_id)、评分(rating)、评分时间(unix_tiemstamp)

```
ratings.head()
```

输出结果：

	user_id	movie_id	rating	unix_timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596

图2-8 前 5 个用户的评分信息

（3）打印数据表格的大小，可以看到这是一个 10000x4 的矩阵， 其中 10000 代表有 10000 条评论，4 代表每个评论有 5 项信息

```
ratings.shape
```

输出结果：

```
(100000, 4)
```

2.6.2 查看电影信息

（1）使用 pandas 库导入电影的信息

```
# Movies
movie_cols = ['movie_id', 'title', 'release_date', 'video_release_date', 'imdb_url']
movies = pd.read_csv('./ml-100k/u.item', sep='|', names=movie_cols, usecols=range(5), encoding='latin-1')
```

（2）打印前 5 个电影信息，可以看到电影信息包含电影 ID(movide_id)、电影名称(title)、发布时间(release_date)、视频发布时间(video_release_date)、评论网站 URL 链接(imdb_url)

```
movies.head()
```

输出结果：

	movie_id	title	release_date	video_release_date	imdb_url
0	1	Toy Story (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Toy%20Story%2...
1	2	GoldenEye (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?GoldenEye%20(...
2	3	Four Rooms (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Four%20Rooms%...
3	4	Get Shorty (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Get%20Shorty%...
4	5	Copycat (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Copycat%20(1995)

图2-9 前 5 部电影信息

（3）打印数据表格的大小，可以看到这是一个 1682x5 的矩阵， 其中 1682 代表有 1682 部电影，5 代表每部电影有 5 项信息

```
movies.shape
```

输出结果：

```
(1682, 5)
```