基于pytorch框架实现FM对用户电影的评分预测

# 项目背景与挖掘目标

## 1.1项目背景

推荐系统是一种信息过滤技术，旨在根据用户的历史行为、偏好和其他相关数据，向用户推荐他们可能感兴趣的物品或服务。推荐系统广泛应用于电子商务、社交媒体、内容推荐等领域，极大地提升了用户体验和平台的用户黏性。

因子分解机（Factorization Machines，FM）是一种强大的推荐算法，能够有效地处理高维稀疏数据，并且能够捕捉到特征之间的二阶交互作用。FM 模型在推荐系统中表现出色，尤其适用于用户-物品评分预测任务。推荐系统中的数据通常是稀疏的，FM 模型通过分解的方式能够高效处理这种数据。相比于传统的线性模型，FM 模型能够捕捉到特征之间的二阶交互作用，从而提升推荐效果。FM 模型不仅适用于评分预测任务，还可以应用于点击率预估等其他预测任务。

## 1.2数据来源

本次项目使用的数据集来自国外网站（https://grouplens.org/datasets/movielens/）由GroupLens Research 从 MovieLens 网站 （https://movielens.org） 收集并提供了评级数据集。MovieLens 25M 电影评分。稳定的基准数据集。 162,000 名用户对 62,000 部电影应用了 2500 万次评分和 100 万次标签应用程序。包括标签基因组数据，在 1,129 个标签中具有 1500 万个相关性分数。2019 年 12 月发布。

## 1.3项目目标

本项目的主要目标是基于 PyTorch 框架实现因子分解机（FM）模型，并利用该模型对用户电影评分进行预测。具体目标包括：

**数据预处理**：清洗和处理原始数据集，生成训练集和测试集。

**模型实现**：使用 PyTorch 构建 FM 模型，包括模型的初始化、前向传播和参数更新。

**模型训练**：在训练集上训练 FM 模型，并在测试集上评估模型性能。

**模型评估**：使用均方根误差（RMSE）等指标评估模型的预测效果，并分析模型的性能和误差来源。

# 数据预处理

## 2.1数据清洗

首先打开原始的 ratings.csv 文件进行读取，同时创建一个新的 ratingsNoHead.csv 文件用于写入。创建 CSV 读取器和写入器对象。遍历原始文件的每一行，跳过第一行（表头），将剩余行写入新的文件。通过这种方式，去除了原始 CSV 文件中的表头，并将数据保存到新的文件中。

代码实现：

1. origin\_f = open('data/ratings.csv','rt',encoding='utf-8',errors="ignore")
2. new\_f = open('data/ratingsNoHead.csv','wt+',encoding='utf-8',errors="ignore",newline="")
3. reader = csv.reader(origin\_f)
4. writer = csv.writer(new\_f)
5. **for** i,row **in** enumerate(reader):
6. **if** i>0:
7. writer.writerow(row)
8. origin\_f.close()
9. new\_f.close()

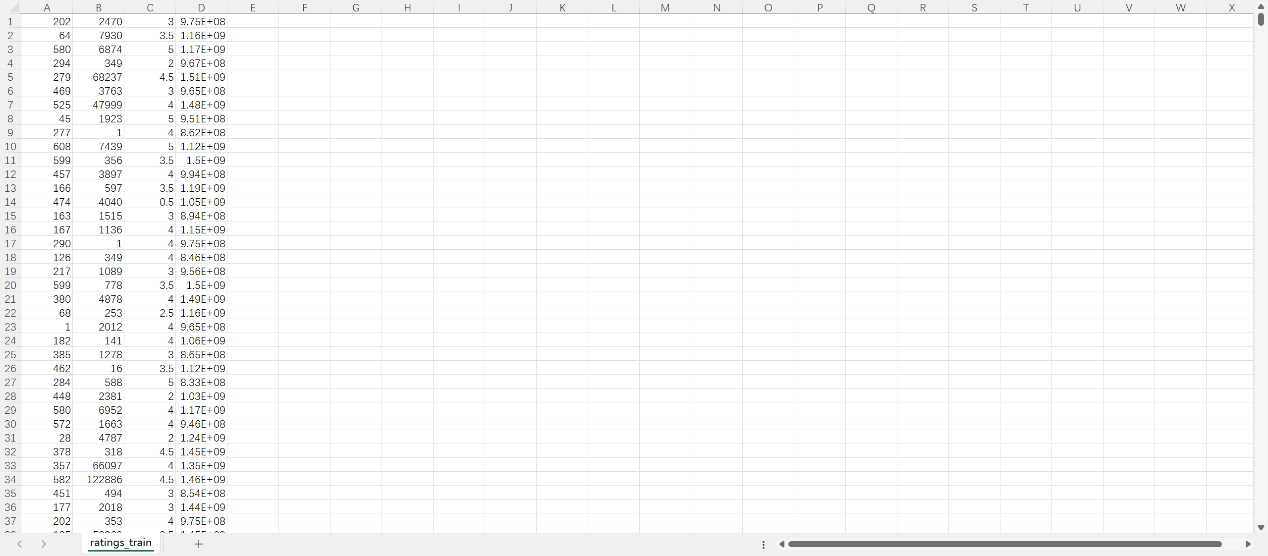
## 2.2划分数据集

第一步读取没有表头的数据文件 ratingsNoHead.csv。然后打乱数据并按照 8:2 的比例划分为训练集和测试集。将训练集保存到临时文件 ratings\_train\_tmp.csv，然后处理并保存到最终文件 ratings\_train.csv。将测试集保存到临时文件 ratings\_test\_tmp.csv，然后处理并保存到最终文件 ratings\_test.csv。数据被划分并存储为无表头的 CSV 文件，以便后续的模型训练和评估。

代码实现：

1. #将数据按照8:2的比例进行划分得到训练数据集与测试数据集
2. df = pd.read\_csv('data/ratingsNoHead.csv', encoding='utf-8')
3. # df.drop\_duplicates(keep='first', inplace=True)  # 去重，只保留第一次出现的样本
4. # print(df)
5. df = df.sample(frac=1.0)  # 全部打乱
6. cut\_idx = int(round(0.2 \* df.shape[0]))
7. df\_test, df\_train = df.iloc[:cut\_idx], df.iloc[cut\_idx:]
8. # 打印数据集中的数据记录数
9. **print**(df.shape,df\_test.shape,df\_train.shape)
10. # print(df\_train)
11. # 将数据记录存储到csv文件中
12. df\_train=pd.DataFrame(df\_train)
13. df\_train.to\_csv('data/ratings\_train\_tmp.csv',index=False)
14. # 由于一些不知道为什么的原因，使用pandas读取得到的数据多了一行，在存储时也会将这一行存储起来，所以应该删除这一行（如果有时间在查一查看能不能解决这个问题）
15. origin\_f = open('data/ratings\_train\_tmp.csv','rt',encoding='utf-8',errors="ignore")
16. new\_f = open('data/ratings\_train.csv','wt+',encoding='utf-8',errors="ignore",newline="")     #必须加上newline=""否则会多出空白行
17. reader = csv.reader(origin\_f)
18. writer = csv.writer(new\_f)
19. **for** i,row **in** enumerate(reader):
20. **if** i>0:
21. writer.writerow(row)
22. origin\_f.close()
23. new\_f.close()
24. os.remove('data/ratings\_train\_tmp.csv')
25. # 存储测试数据集
26. df\_test=pd.DataFrame(df\_test)
27. df\_test.to\_csv('data/ratings\_test\_tmp.csv')
28. origin\_f = open('data/ratings\_test\_tmp.csv','rt',encoding='utf-8',errors="ignore")
29. new\_f = open('data/ratings\_test.csv','wt+',encoding='utf-8',errors="ignore",newline="")
30. reader = csv.reader(origin\_f)
31. writer = csv.writer(new\_f)
32. **for** i,row **in** enumerate(reader):
33. **if** i>0:
34. writer.writerow(row)
35. origin\_f.close()
36. new\_f.close()
37. os.remove('data/ratings\_test\_tmp.csv')

结果：



# 3.因子分解机模型

## 3.1模型简介

推荐系统中常用的协同过滤方法通过用户-物品矩阵来预测用户对物品的偏好。然而，由于数据的稀疏性，直接使用矩阵分解模型（如 SVD）可能会导致模型效果不佳。因子分解机通过引入隐向量（latent vectors）来捕捉变量间的交互作用，能够在稀疏数据上表现出色。

因子分解机模型的公式如下：

FM 能够在数据非常稀疏的情况下有效地捕捉特征之间的交互作用。由于只需计算隐向量之间的内积，计算复杂度较低。适用于各种推荐系统、分类和回归任务。

## 3.2模型实现

1.定义因子分解机（FM）模型：

定义一个因子分解机模型类 FM\_model。这个类包含线性部分和交互部分。p 是特征数量，k 是因子数量。

1. **class** FM\_model(pt.nn.Module):
2. **def** \_\_init\_\_(self, p, k):
3. super(FM\_model, self).\_\_init\_\_()
4. self.p = p    # 特征数量
5. self.k = k    # 因子的数量
6. self.linear = pt.nn.Linear(self.p, 1, bias=True)   # 线性部分
7. self.v = pt.nn.Parameter(pt.rand(self.k, self.p))  # 交互作用部分的隐向量
9. **def** fm\_layer(self, x):
10. # 线性部分
11. linear\_part = self.linear(pt.tensor(x).float())
12. # 交互作用部分
13. inter\_part1 = pt.mm(pt.tensor(x).float(), self.v.t())
14. inter\_part2 = pt.mm(pt.pow(pt.tensor(x).float(), 2), pt.pow(self.v, 2).t())
15. pair\_interactions = pt.sum(pt.sub(pt.pow(inter\_part1, 2), inter\_part2), dim=1)
16. output = linear\_part.transpose(1, 0) + 0.5 \* pair\_interactions
17. **return** output
19. **def** forward(self, x):
20. output = self.fm\_layer(x)
21. **return** output

2.实例化模型：

1. k = 10   # 因子的数量
2. fm = FM\_model(p, k)

3.定义损失函数和优化器：

1. optimizer = pt.optim.SGD(fm.parameters(), lr=0.01)   # 学习率为0.01
2. loss\_func = pt.nn.MSELoss()  # 使用均方误差损失函数

# 6.总结

本项目的主要目标是基于PyTorch框架实现因子分解机（Factorization Machine, FM）模型，以预测用户对电影的评分。

为了将数据转换为模型可以处理的格式，我们使用了 `DictVectorizer` 对数据进行了特征向量化。特征向量化将类别特征转换为one-hot编码的形式，从而生成了高维稀疏特征矩阵。这个过程生成的特征矩阵是模型输入的重要部分。

我们在PyTorch中定义了因子分解机（FM）模型。FM模型包括线性部分和交互作用部分，通过隐向量捕捉特征之间的二阶交互作用。模型训练使用随机梯度下降法（SGD）进行优化。我们定义了均方误差损失函数（MSELoss）来评估预测评分与真实评分之间的误差，并加入L2正则化以防止过拟合。训练过程中，每个epoch都会更新模型参数以最小化损失函数。

训练完成后，我们使用均方根误差（RMSE）作为模型的评估指标。通过计算测试集上的RMSE，我们可以评估模型的预测性能。较低的RMSE值表明模型在评分预测方面具有较好的准确性。

在我们的实验中，模型的测试损失（RMSE）为1.5298，这表明模型能够较好地预测用户对电影的评分。尽管还有改进的空间，例如优化超参数或采用更复杂的模型结构，但当前模型已经展示了FM在推荐系统中的潜力。