【Experiment name】

Open source software project recommendation system

【Purpose】

Understand the background and significance of open source software project recommendation

Master the principle of memory-based collaborative filtering algorithm (Memory-Based Collaborative Filtering)

Master the principle of model-based collaborative filtering algorithm (Model -Based Collaborative Filtering)

Implementation of open source software project recommendation system based on Spark

【Experimental content】

Topic : GitHub provides the Pull Request function for collaboration among developers in the open source software community. Usually, a software developer forks a software project, develops new features or fixes bugs, and then pulls this part of the code The request method is submitted to the original software project, and the core team members of the software project are responsible for reviewing the code submitted by the developer and deciding whether to merge it into the software project. Generally, a pull request has 5 states: o pened, closed, merged, reopened, syncrhonize . The merged status indicates that the code has been successfully received and merged into the software project. Please design a collaborative filtering recommendation algorithm based on the given data set and other self-collected auxiliary data sets, and recommend for developers who can successfully participate (that is, their pull request can be accepted).

Requirements :

1. Can be based on Spark MLlib or other tools;

2. Before building the model, it is necessary to analyze the data set and display the analysis results in the form of graphs;

3. Perform hyperparameter tuning;

4. Take RMSE and Precision@5 as evaluation indicators;

5. The prediction results are written into a file and submitted.

【Experimental principle】

In this experiment, I used the recommendation model A LS in pyspark.ml . The main principles are as follows:

2.4 推荐模型(pyspark.ml.recommendation)

（1）**ALS**：交替最小二乘(Alternating Least Squares)，是一种基于协同过滤原理的推荐算法。在已知用户（User）数量m和物品（Item）数量n的前提下，可以通过奇异值分解（Singular Value Decomposition，SVD）的方法，将用户—商品矩阵A_mxn分解为矩阵U和V，如下所示：

$$A_{m\times n} = U_{m\times k} * V_{k\times n}$$

然后，通过ALS方法对其进行优化（先固定其中一个变量，优化另一个；然后再固定另一个，优化前一个，交替迭代进行），最后可以据此得出用户对特定物品的偏好（评分或偏好值）。

Experimental results (experimental steps and related core codes) :

first part:

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

# 导入Spark相关包
from pyspark.sql import SparkSession
from pyspark import SparkConf
import pyspark.sql.types as T
import pyspark.sql.functions as F

# spark配置信息
config = SparkConf().setAll([
    ("spark.app.name", "lesson4_prediction"),
    ("spark.master", "local[*]"),
    ("spark.executor.memory", "8g"),
    ("spark.executor.cores", "2"),
    ("spark.diver.memory", "8g"),
    ("spark.diver.cores", "4")
])
#创建SparkSession
spark = SparkSession.builder.config(conf=config).getOrCreate()

samplesFilePath = "/home/jovyan/work/E11714076/csv/csv/user_pr_detail_train.csv"
# samplesFilePath = "./user_pr_detail.csv"

samplesSchema = T.StructType([
    T.StructField("uid", T.IntegerType(), False),  # 用户id
    T.StructField("iid", T.IntegerType(), False),  # 项目id
    T.StructField("merged_pr_cnt", T.IntegerType(), True),  # 被成功接收的PR数
    T.StructField("total_pr_cnt", T.IntegerType(), True),  # 总共提交的PR数
    T.StructField("opened_pr_cnt", T.IntegerType(), True),  # 打开状态的PR数
    T.StructField("closed_pr_cnt", T.IntegerType(), True),  # 关闭状态的PR数
    T.StructField("reoponed_pr_cnt", T.IntegerType(), True),  # 重新打开状态的PR数
    T.StructField("syncrhonize_pr_cnt", T.IntegerType(), True),  # 同步新Commit的PR数
    T.StructField("intra_branch_pr_cnt", T.IntegerType(), True),  # 在同项目不同分支下提交PR数
    T.StructField("inter_branch_pr_cnt", T.IntegerType(), True),  # Fork原项目后提交的PR数
])

samples = spark.read.csv(path=samplesFilePath, schema=samplesSchema, sep=",", nullValue="").fillna(0).cache()
samples.createOrReplaceTempView("samples")

samples.count()
```

① First import related libraries such as numpy, panda and s park related packages

②Set configuration information initialization environment.

③Create Spark Session

④ Set the original data file path

⑤ Obtain the required data content

the second part:

[3]:
```
# 推荐系统，固定uid-iid，选取features
select = spark.sql("""SELECT uid,
                             iid,
                             merged_pr_cnt as features
                      FROM samples""").cache()
```

①According to the database content obtained in the first part, further process and generate the data table that needs to be applied to the recommendation system.

②After analysis, I found that merged_pr_cnt is the main feature that affects u id and i