Term Project Recommendation System Report

Group48 / 106062314 蔡政諺

● Item-Item Collaborative Filtering

(1) RDD Variables

○ **ratings**

每個 key-value pair 為一個 movieId，對應所有對其評分的 userId 與 rating。

Format: (movieId, [[$userId_1$, $rating_1$], …, [$userId_i$, $rating_i$]])

○ **ratingsNorm**

每個 key-value pair 為一個 movieId，對應所有對其評分的 userId 與標準化後的 rating。

Format: (movieId, [[$userId_1$, $rating'_1$], …, [$userId_i$, $rating'_i$]])

○ **similarities**

每個 key-value pair 為兩個 movieId，對應兩者之間的 cosine similarity。

Format: (($movieId_i$, $movieId_j$), similarity)

(2) Mappers and Reducers

詳見 **Term_Project_Group48.ipynb**。

(3) Overall Process

首先讀取 **ratingsFile** (ratings.csv)，刪掉第一行 (第一行沒有資訊)，用 **mapper1** 將每一行的 userId, movieId, rating 取出，根據 movieId, userId 排序，接著用 **reducer1** 將同一部電影的 rating 合在一起，用 **mapper2** 將只被一個使用者評分的電影移除 (原因是 subtract mean 後評分為 0，會無法計算 similarity)，再根據 movieId 排序，得到 **ratings**。接著用 **mapper3** 對評分做 subtract mean，並將 subtract mean 後評分都是 0 的電影移除 (同上，無法計算 similarity)，得到 **ratingsNorm**。最後用 **mapper4** 計算兩兩電影之間的相似度，得到 **similarities**。這裡因為如果一次將全部的 similarity 算出來並 **collect**，處理太大的 dataset 時會導致 IO error，因此我改為執行一個 for loop，每次只算出部分的結果，批次將結果寫進 **similaritiesFile** (similarities.txt)。

```
ratings = sc.textFile(ratingsFile)
ratings = ratings.filter(lambda line: not line.startswith('userId'))
ratings = ratings.map(mapper1).sortBy(lambda x: (x[0], x[1][0]))
ratings = ratings.reduceByKey(reducer1).flatMap(mapper2).sortBy(lambda x: x[0])

ratingsNorm = ratings.flatMap(mapper3).sortBy(lambda x: x[0])
ratingsNormList = ratingsNorm.collect()

f = open(similaritiesFile, 'w')
batch_size = int(len(ratingsNormList)/iteration)
for i in range(iteration):
    start_time = time.time()
    similarities = ratingsNorm.flatMap(lambda x: mapper4(x, ratingsNormList[i*batch_size:(i+1)*batch_size])).sortBy(lambda x: (x
    similaritiesList = similarities.collect()
    for pair in similaritiesList:
        f.write("(%d, %d), %.6f\n" % (pair[0][0], pair[0][1], pair[1]))
    print("iter %d/%d: %ds" % (i+1, iteration, time.time()-start_time))
f.close()
```

- **Rating Predictions**

(1) RDD Variables

- **ratings**

每個 key-value pair 為一個 movieId，對應所有對其評分的 userId 與 rating。

Format: (movieId, [[$userId_1$, $rating_1$], …, [$userId_i$, $rating_i$]])

- **similarities**

每個 key-value pair 為兩個 movieId，對應兩者之間的 cosine similarity。

Format: (($movieId_i$, $movieId_j$), similarity)

- **neighbors**

每個 key-value pair 為一個 movieId，對應所有與之相似的電影。

Format: (movieId, [[$movieId_1$, $similarity_1$], …, [$movieId_i$, $similarity_i$]])

- **predictions**

每個 key-value pair 為 userId 與 movieId，對應我們預測該使用者對這部電影的評分。如果該評分本來就存在，則不會輸出。

Format: ((userId, movieId), rating)

(2) Mappers and Reducers

詳見 **Term_Project_Group48.ipynb**。

(3) Overall Process

首先使用與 **Item-Item Collaborative Filtering** 相同的步驟，得到 **ratings**。並讀取 **similaritiesFile** (similarities.txt)，用 **mapper_load** 將每一行的 (($movieId_i$, $movieId_j$), similarity) 取出，得到 **similarities**。接著用 **mapper5** 將兩兩電影之間的相似度的所有配對條列出來，根據 $movieId_i$, similarity(降序) 排序，用 **reducer1** 將同一部電影的所有相似電影降序排列，並用 **mapper6** 過濾相似度小於等於 0 的電影，得到 **neighbors**。最後用 **mapper7** 計算所有原本空缺的評分，再用 **mapper8** 將同一部電影的所有評分展開成多個 ((userId, movieId), rating) 的 key-value pairs，根據 userId, movieId 排序，得到 **predictions**，並寫進 **predictionsFile** (predictions.txt)。

```
ratings = sc.textFile(ratingsFile)
ratings = ratings.filter(lambda line: not line.startswith('userId'))
ratings = ratings.map(mapper1).sortBy(lambda x: (x[0], x[1][0]))
ratings = ratings.reduceByKey(reducer1).flatMap(mapper2).sortBy(lambda x: x[0])
ratingsList = ratings.collect()

similarities = sc.textFile(similaritiesFile)
similarities = similarities.map(mapper_load)

neighbors = similarities.flatMap(mapper5).sortBy(lambda x: (x[0], -x[1][1]))
neighbors = neighbors.reduceByKey(reducer1).map(mapper6)

predictions = neighbors.map(lambda x: mapper7(x, ratingsList))
predictions = predictions.flatMap(mapper8).sortBy(lambda x: (x[0][0], x[0][1]))
predictionsList = predictions.collect()

f = open(predictionsFile, 'w')
for pair in predictionsList:
    f.write("(%d, %d), %.6f\n" % (pair[0][0], pair[0][1], pair[1]))
f.close()
```

- **Outputfile**

similarities.txt

predictions.txt