

데이터과학



[6주차] 영화추천

목표

- 추천 알고리즘 이해하기
- 가장 가까운(비슷한) 유저
- 영화 데이터 사용해 별점 예측하기

이번주


























- 추천시스템
- 아프리카 TV 추천시스템
- 영화(Open Data) 추천

RecSys Begins
(A.K.A **Re**commender **Sys**tems)

최규민

Collaborative Filtering

- 사용자의 지난 행위 정보 기반
- 별점예측 혹은 유사상품 추천

				
				
				
				?
				?
				

참조 : 위키 Collaborative Filtering

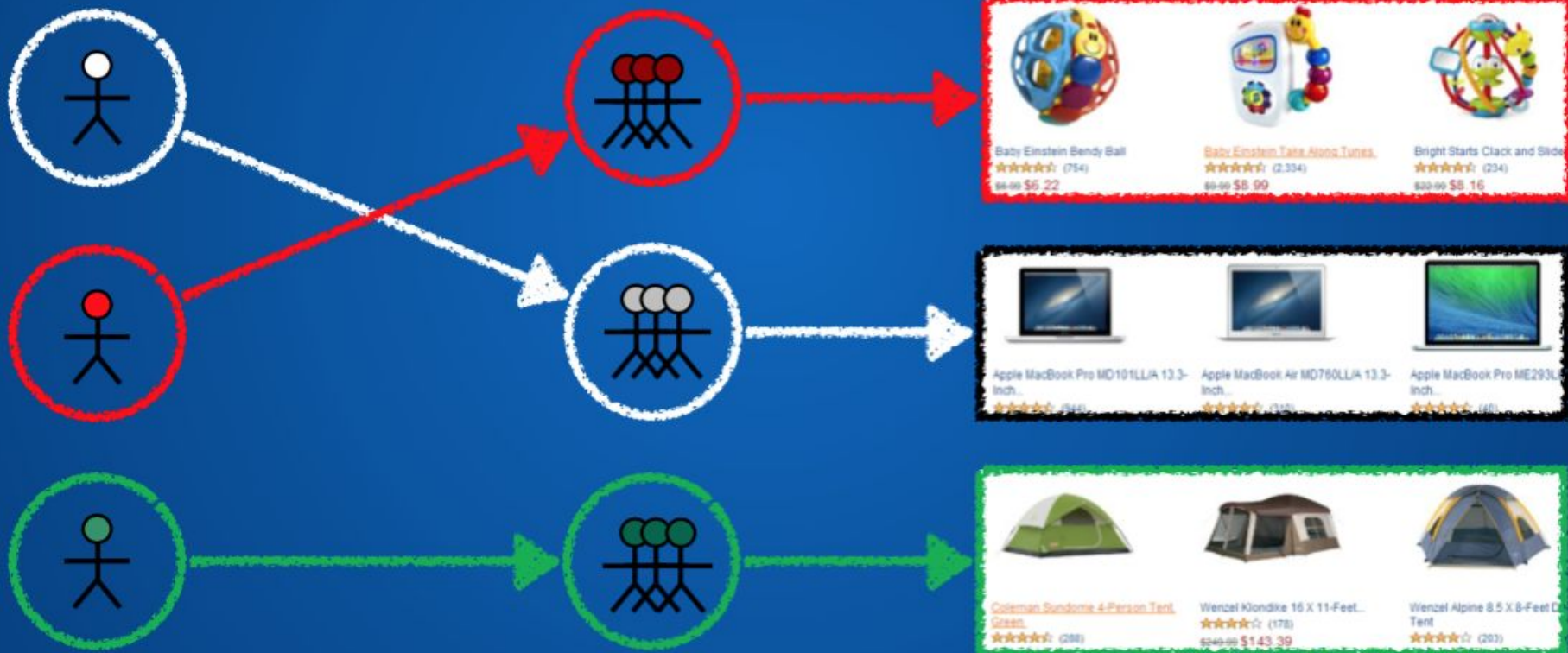
유저 기반 추천

- 나와 유사한 유저가 본 아이템을 추천

나

유사한 유저

그들이 본 아이템



영화 별점 데이터

- MovieLens
- <https://grouplens.org/datasets/movielens/>
- 20MB크기의 별점 데이터 사용

recommended for new research

MovieLens 20M Dataset

Stable benchmark dataset. 20 million ratings and 465,000 tag applications applied to 27,000 movies by 138,000 users. Includes tag genome data with 12 million relevance scores across 1,100 tags. Released 4/2015; updated 10/2016 to update links.csv and add tag genome data.

- [README.html](#)
- [ml-20m.zip](#) (size: 190 MB, [checksum](#))

Permalink: <http://grouplens.org/datasets/movielens/20m/>

영화 별점 데이터

- 총 2천만 Line의 별점 데이터
- userId, movieId, rating, timestamp 형태

20000256	138493,61160,4.0,1258390537
20000257	138493,65682,4.5,1255816373
20000258	138493,66762,4.5,1255805408
20000259	138493,68319,4.5,1260209720
20000260	138493,68954,4.5,1258126920
20000261	138493,69526,4.5,1259865108
20000262	138493,69644,3.0,1260209457
20000263	138493,70286,5.0,1258126944
20000264	138493,71619,2.5,1255811136

추천 과정

1. Data Processing

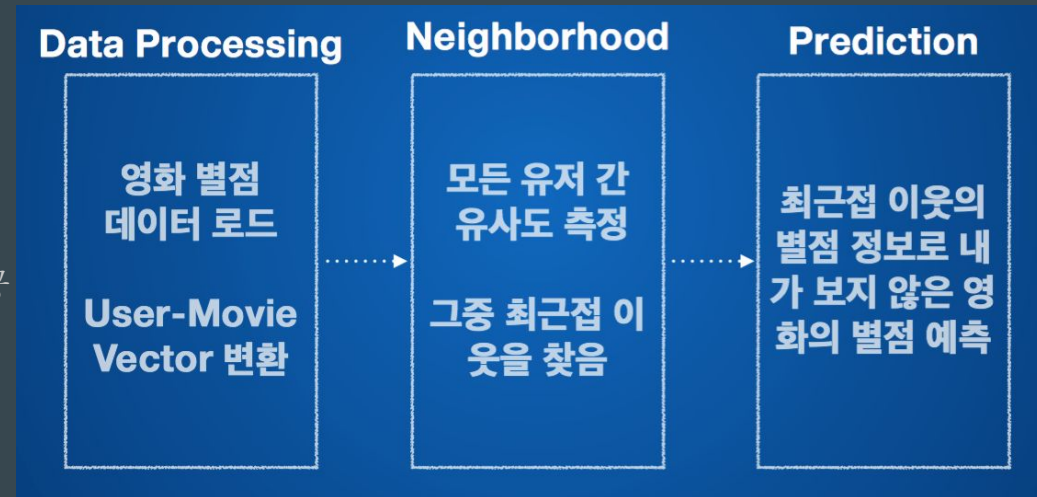
- 영화 별점 데이터 로딩
- 사용자 - 영화 형태의 데이터로 가공

2. Neighborhood

- 모든 유저간 유사도 측정
- 가장 가까운 이웃 찾기

3. Prediction

- 가장가까운 이웃들의 별점 정보로 내가 보지 않은 영화의 별점 예측



데이터 추출

- 마지막에 예측정확도를 비교하기 위해 ratings.csv데이터중 10개의 데이터 추출
- ratings.csv에서 1000, 2000, 3000, 4000...10000 번째 데이터를 추출
- 해당 결과값들을 예측할 것

```
1 userId,movieId,rating,timestamp
2 11,500,4.5,1230858949
3 18,4428,3.5,1262461295
4 24,4308,2.0,1015727461
5 35,60,4.0,1164498353
6 50,924,5.0,1182349293
7 54,3194,4.0,975440547
8 58,32587,4.5,1144061539
9 70,2890,3.0,102029424
10 83,296,5.0,1112724196
11 91,2863,4.5,1111558557
```

1. Data Processing

- 데이터 로드

```
def movieLensDataLoad(type):  
    ## user 영화 별점 data  
    ratings = pd.read_csv("/Users/goodvc/Documents/data-analytics/movie-recommendation/ratings.csv")  
  
    ## movie meta(타이틀, 장르) data  
    movies = pd.read_csv("/Users/goodvc/Documents/data-analytics/movie-recommendation/movies.csv")  
  
    ## user가 영화에 tag를 기입한 data  
    tags = pd.read_csv("/Users/goodvc/Documents/data-analytics/movie-recommendation/tags.csv")  
    # tags = pd.read_csv("/Users/goodvc/Documents/data-analytics/movie-recommendation/tags.csv")  
    return ( ratings, movies, tags )  
  
# ratings, movies, tags = movieLensDataLoad('ml-latest-small')  
ratings, movies, tags = movieLensDataLoad('ml-latest-small')
```

ratings

	userId	movieId	rating	timestamp
0	1	6	2	980730861
1	1	22	3	980731380
...
100021	706	1183	4	850465137
100022	706	1356	4	852915765

100023 rows x 4 columns

706 users.
8,552 movies
100,022 ratings

1. Data Processing

- 데이터 가공
 - User x Movie 형태로 가공

```
## 1. U X M vector Matrix를 만든다.
```

```
UM_matrix_ds = ratings.pivot(index='userId', columns='movieId', values='rating')
```

	userId	movieId	rating
0	1	6	2
1	1	22	3
...
100021	706	1183	4
100022	706	1356	4

100023 rows x 4 columns

movieId	1	2	3	4	5	...	124857	125916
userId								
1	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN
2	NaN	4	NaN	NaN	NaN	...	NaN	NaN
...
705	NaN	NaN	3	NaN	NaN	...	NaN	NaN
706	4	NaN	NaN	NaN	NaN	...	NaN	NaN

706 rows x 8552 columns

2. Neighborhood

- 유사도 측정
 - 교차로 평가한 영화의 별점 정보로 유사도 측정
 - 유사도값을 신뢰하기 위해 교차평가한 영화가 일정 수 이상일때만 유사도 측정
 - 유사도 측정의 대표적인 함수
 - Jaccard, Cosine, Euclidean, Correlation

2. Neighborhood

- 가장 가까운 이웃 구하기

```
def nearest_neighbor_user(user, top_n, similarity_func):
    base_user = UM_matrix_ds.loc[user].dropna()
    rated_index = base_user.index
    nearest_neighbor = {}

    for user_id, row_data in UM_matrix_ds.iterrows():
        intersection_user1 = []
        intersection_user2 = []

        if user == user_id:
            continue
        for index in rated_index:
            if math.isnan(row_data[index]):
                intersection_user1.append(base_user[index])
                intersection_user2.append(row_data[index])

        if len(intersection_user1) < 3:
            continue

        similarity = similarity_func(intersection_user1, intersection_user2)

        if math.isnan(similarity) == False:
            nearest_neighbor[user_id] = similarity
    return sorted(nearest_neighbor.items(), key=itemgetter(1))[:-(top_n+1):-1]
```

2. Neighborhood

- 가장 가까운 이웃 10명씩 찾기
- test.csv 에 존재하는 유저들의 가장 가까운 이웃 3명씩 구하기
 - 가까운 사용자 ID와 거리출력

11번, 18번, 24번 유저와 가장 가까운 이웃
3명씩

[사용자 ID, 거리] 형태

```
[(74142, 50.663102944845377), (116900, 45.417507637473896), (37253, 44.141816908686486)]  
[(50168, 21.412613105363857), (37253, 21.13646138784825), (93024, 20.530465167647808)]  
[(37253, 40.11545836706842), (74142, 39.859126934743564), (116900, 38.7556447501522)]
```

3. Prediction

- 가까운 이웃들의 별점을 베이스로 자신이 보지 않은 영화의 별점 예측
- test.csv 에 존재하는 사용자,영화의 별점예측
 - 즉, test.csv의 데이터는 ground-truth데이터
 - 예측한 결과와 test.csv데이터를 비교해 정확도 측정

```
def predict_rating(user_id, nearest_neighbor = 10, similarity_func = distance_euclidean):
    neighbor = nearest_neighbor_user(user_id, nearest_neighbor, similarity_func)
    neighbor_id = [id for id, sim in neighbor]

    neighbor_movie = UM_matrix_ds.loc[neighbor_id]\
        .dropna(1, how='all', thresh = 4)
    neighbor_dic = (dict(neighbor))
    ret = []

    for movieId, row in neighbor_movie.iteritems():
        jsum, wsum = 0, 0
        for v in row.dropna().iteritems():
            sim = neighbor_dic.get(v[0],0)
            jsum += sim
            wsum += (v[1]*sim)
        ret.append([movieId, wsum/jsum])
    return ret
```

3. Prediction

	userId	movieId	rating
0	11	500	4.5
1	18	4428	3.5
2	24	4308	2.0
3	35	60	4.0
4	50	924	5.0
5	54	3194	4.0
6	58	32587	4.5
7	70	2890	3.0
8	83	296	5.0
9	91	2863	4.5

실제 별점 데이터

	userId	movieId	rating
0	11	500	4.232143
1	18	4428	2.687921
2	24	4308	2.668541
3	35	60	4.541213
4	50	924	4.433242
5	54	3194	4.113210
6	58	32587	4.451231
7	70	2890	2.658785
8	83	296	4.875677
9	91	2863	4.323123

예측 데이터

4. Accuracy

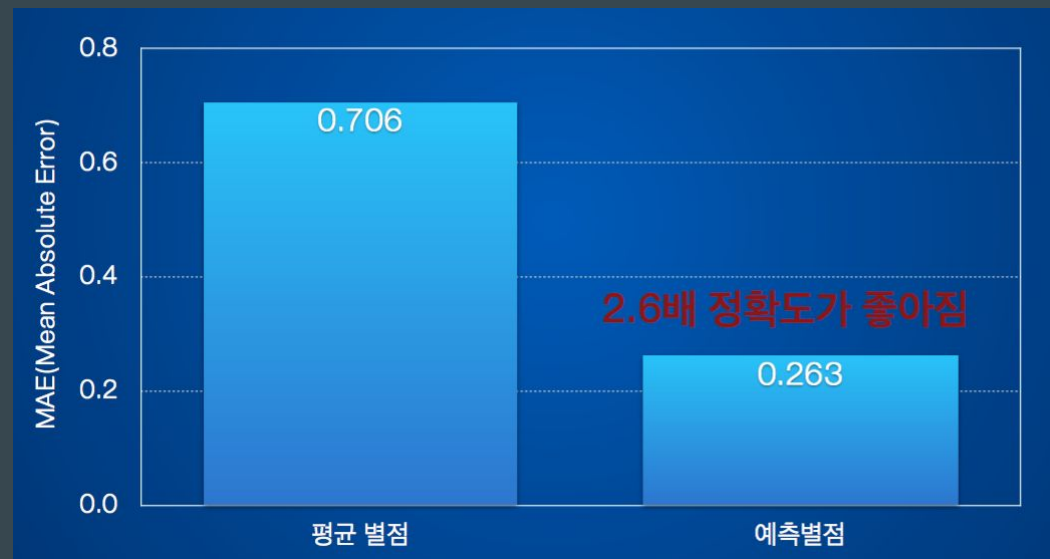
- 정확도 구하기

- Mean Square Error

- http://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html

- Mean Absoulte Error

- http://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_absolute_error.html



```
>>> print('Error rate : ' + str(((result_array - truth_array) **2).mean()))  
... )  
Error rate : 0.197065641262
```

과제 정리

1. test.csv 에 존재하는 사용자들의 가장가까운 이웃 3명씩 찾기
2. test.csv에 존재하는 사용자와 영화의 별점 예측하기
 - a. 유사도 비교 알고리즘은 선택
3. 예측 정확도 구하기
 - 데이터는 20M, 100K, 200K 선택적으로 사용
 - 데이터량따라 계산시간이 달라짐

과제 제출 방법

- 과제 제출 기한 : 2017년 4월 26일 **오후 6시까지!**
- Google Classroom에 제출!
 - **24시간 경과시마다 20% 감점**
- 파일 제목 : **DS_학번_이름_주차.pdf, DS_학번_이름_주차.zip**
 - 보고서(PDF형태) : **HWP, DOC**일경우 채점 안함
 - 파일 제목 및 형태 틀리면 -1점

질문 사항

- 방문
 - 606호 (데이터네트워크 연구실)
- 메일
 - dbgustlr92@cs-cnu.org
- Google Classroom
 - Good!