기계학습개론 과제4 리포트

202011047 김승태

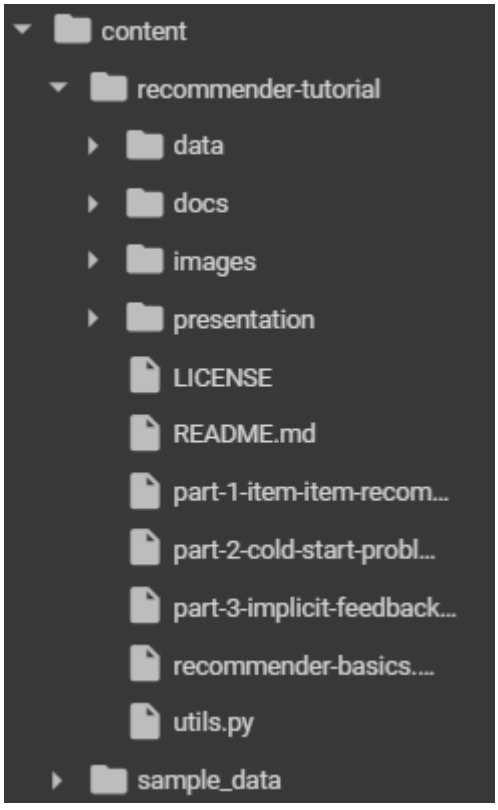
1. Run the code below and arrange the results.

https://github.com/topspinj/recommender-tutorial/blob/master/part-1-item-item-recommender.ipynb

환경은 google colab에서 진행하였으며, 먼저, 기존의 git에서 추후에 data에서 파일을 사용하므로 Git clone을 이용하여서, 레포지토리를 가져오도록 하였다.

```
1 !git clone https://github.com/topspinj/recommender-tutorial.git

Cloning into 'recommender-tutorial'...
remote: Enumerating objects: 463, done.
remote: Counting objects: 100% (35/35), done.
remote: Compressing objects: 100% (27/27), done.
remote: Total 463 (delta 18), reused 23 (delta 8), pack-reused 428
Receiving objects: 100% (463/463), 8.01 MiB | 12.98 MiB/s, done.
Resolving deltas: 100% (180/180), done.
```

그 결과, /content 안에 recommender-tutorial 레포지토리가 생기고, 참조할 수 있는 조건이 만족된다. Git clone 결과는 다음과 같다.



이는, 아래와 같은 코드에서 에러가 나기 때문에 실시하

였다.

```
[19]  1 from scipy.sparse import save_npz
      2
      3 save_npz('/content/recommender-tutorial/data/user_item_matrix.npz', X)
```

그리하여, content 주소를 포함한 위와 같은 코드로 진행하였을 때, 에러가 나지 않는다.

이제, results를 arrange 하겠다.

```
1 ratings = pd.read_csv("https://s3-us-west-2.amazonaws.com/recommender-tutorial/ratings.csv")
2 ratings.head()
```

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

```
1 movies = pd.read_csv("https://s3-us-west-2.amazonaws.com/recommender-tutorial/movies.csv")
2 movies.head()
```

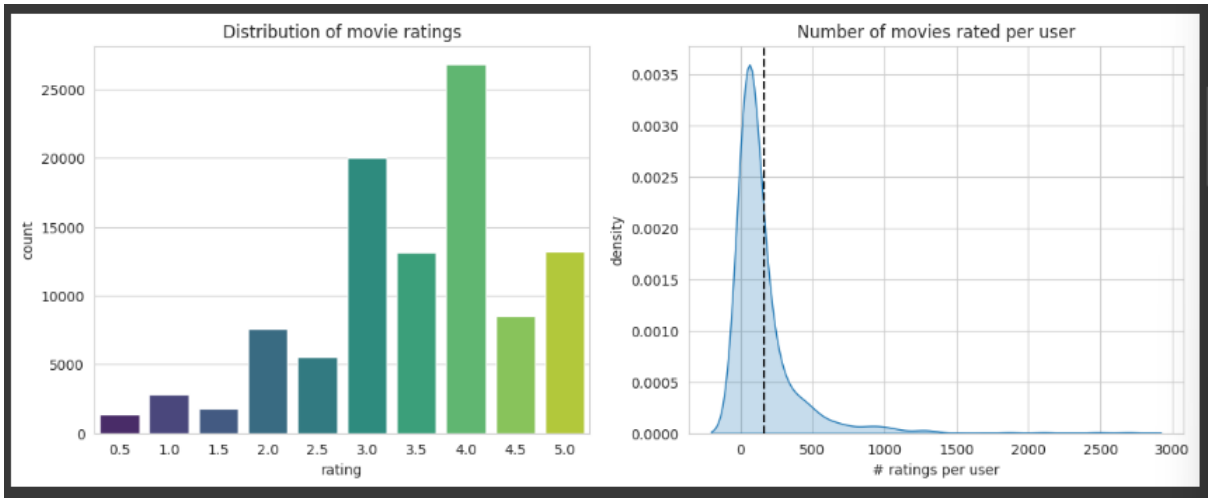
	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

이는 데이터의 rating과 , movies의 title, genres들을 보여주는 코드이다. 그냥 load하는 부분이니, 크게 중요하지는 않았다.

```
Number of ratings: 100836
Number of unique movieId's: 9724
Number of unique users: 610
Average number of ratings per user: 165.3
Average number of ratings per movie: 10.37
```

	userId	n_ratings
0	1	232
1	2	29
2	3	39
3	4	216
4	5	44

```
Mean number of ratings for a given user: 165.30.
```



이렇게 총 ratings, movie ID, user들의 수부터, 평균 등등을 보여주고, distribution을 보여준다.

그 후, 어떤 movie가 average rate가 가장 낮은지, 높은지도 보여주었다.

```
movie_stats = movie_stats.merge(movies[['movieId', 'title']])
movie_stats.sort_values('bayesian_avg', ascending=False).head()
```

	movieId	count	mean	bayesian_avg	title
277	318	317	4.429022	4.392070	Shawshank Redemption, The (1994)
659	858	192	4.289062	4.236457	Godfather, The (1972)
2224	2959	218	4.272936	4.227052	Fight Club (1999)
224	260	251	4.231076	4.192646	Star Wars: Episode IV - A New Hope (1977)
46	50	204	4.237745	4.190567	Usual Suspects, The (1995)

Using the Bayesian average, we see that `Shawshank Redemption`, `The Godfather`, and `Fight Club` are the most highly rated movies. This result makes much more sense since these movies are critically acclaimed films.

Now which movies are the worst rated, according to the Bayesian average?

```
movie_stats.sort_values('bayesian_avg', ascending=True).head()
```

	movieId	count	mean	bayesian_avg	title
1172	1556	19	1.605263	2.190377	Speed 2: Cruise Control (1997)
2679	3593	19	1.657895	2.224426	Battlefield Earth (2000)
1372	1882	33	1.954545	2.267268	Godzilla (1998)
1144	1499	27	1.925926	2.296800	Anaconda (1997)
1988	2643	16	1.687500	2.306841	Superman IV: The Quest for Peace (1987)

이는 Bayesian average를 적용한 결과이다.

그 후에는 data를 transforming하고, k-Nearest Neighbours를 이용하여서 similar movie들을 찾아내는 과정을 거쳤다.

```
Because you watched Toy Story (1995)
Toy Story 2 (1999)
Jurassic Park (1993)
Independence Day (a.k.a. ID4) (1996)
Star Wars: Episode IV - A New Hope (1977)
Forrest Gump (1994)
Lion King, The (1994)
Star Wars: Episode VI - Return of the Jedi (1983)
Mission: Impossible (1996)
Groundhog Day (1993)
Back to the Future (1985)
```

results above show the 10 most similar movies to Toy Story. Most movies in this list are family movies from the 1990s, which seems ty reasonable. Note that these recommendations are based solely on user-item ratings. Movie features such as genres are not taken consideration in this approach.

can also play around with the kNN distance metric and see what results you would get if you use "manhattan" or "euclidean" instead cosine".

```
movie_titles = dict(zip(movies['movieId'], movies['title']))

movie_id = 1
similar_ids = find_similar_movies(movie_id, X, k=10, metric="euclidean")

movie_title = movie_titles[movie_id]
print(f"Because you watched {movie_title}:")
for i in similar_ids:
    print(movie_titles[i])
```

```
Because you watched Toy Story (1995):
Toy Story 2 (1999)
Mission: Impossible (1996)
Independence Day (a.k.a. ID4) (1996)
Bug's Life, A (1998)
Nutty Professor, The (1996)
Willy Wonka & the Chocolate Factory (1971)
Babe (1995)
Groundhog Day (1993)
Mask, The (1994)
Honey, I Shrunk the Kids (1989)
```

이렇게 나오게 된다. 아래에서의 metric은 유클리드이다. 위와 아래의 결과에서는, 두 거리 척도는 일부 유사한 영화를 추천하지만, 추천 목록에서 차이점을 볼 수 있었다는 점이다.

2. Show how the results differ based on the distance metrics.

```
[24]  1 movie_titles = dict(zip(movies['movieId'], movies['title']))
      2
      3 movie_id = 1
      4 similar_ids = find_similar_movies(movie_id, X, k=10, metric="euclidean")
      5
      6 movie_title = movie_titles[movie_id]
      7 print(f"Because you watched {movie_title}:")
      8 for i in similar_ids:
      9     print(movie_titles[i])
```

```
Because you watched Toy Story (1995):
Toy Story 2 (1999)
Mission: Impossible (1996)
Independence Day (a.k.a. ID4) (1996)
Bug's Life, A (1998)
Nutty Professor, The (1996)
Willy Wonka & the Chocolate Factory (1971)
Babe (1995)
Groundhog Day (1993)
Mask, The (1994)
Honey, I Shrunk the Kids (1989)
```

```
[25]  1 movie_titles = dict(zip(movies['movieId'], movies['title']))
      2
      3 movie_id = 1
      4 similar_ids = find_similar_movies(movie_id, X, k=10, metric="manhattan")
      5
      6 movie_title = movie_titles[movie_id]
      7 print(f"Because you watched {movie_title}:")
      8 for i in similar_ids:
      9     print(movie_titles[i])
```

```
Because you watched Toy Story (1995):
Toy Story 2 (1999)
Bug's Life, A (1998)
Groundhog Day (1993)
Nutty Professor, The (1996)
Willy Wonka & the Chocolate Factory (1971)
Mission: Impossible (1996)
Babe (1995)
Monsters, Inc. (2001)
Toy Story 3 (2010)
Honey, I Shrunk the Kids (1989)
```

```
      1 movie_titles = dict(zip(movies['movieId'], movies['title']))
      2
      3 movie_id = 1
      4 similar_ids = find_similar_movies(movie_id, X, k=10, metric="cosine")
      5
      6 movie_title = movie_titles[movie_id]
      7 print(f"Because you watched {movie_title}:")
      8 for i in similar_ids:
      9     print(movie_titles[i])
```

```
Because you watched Toy Story (1995):
Toy Story 2 (1999)
Jurassic Park (1993)
Independence Day (a.k.a. ID4) (1996)
Star Wars: Episode IV - A New Hope (1977)
Forrest Gump (1994)
Lion King, The (1994)
Star Wars: Episode VI - Return of the Jedi (1983)
Mission: Impossible (1996)
Groundhog Day (1993)
Back to the Future (1985)
```

결과는 이렇다.

유클리드 거리는 영화의 실제 평점 간 차이를 기준으로 유사도를 측정한다. 이로 인해 다양한 장르의 영화들이 추천 리스트에 포함된다.

맨해튼 거리는 유클리드 거리와 유사하지만, 계산 방식이 달라 결과에 약간의 차이를 보인다. 주로 코미디와 가족 영화를 추천한다.

코사인 유사도는 각도를 기준으로 유사도를 측정해, 주로 장르와 테마가 유사한 영화를 추천한다. SF와 가족 영화들이 많이 포함된다.

결과적으로, 세 가지 거리 척도 모두 "Toy Story"와 유사한 영화를 추천하지만, 추천되는 영화의 종류와 순위에는 차이가 있다. 코사인 유사도는 장르 유사성을 더 잘 반영하는 반면, 유클리드와 맨해튼 거리는 실제 평점 차이를 반영해 다양한 영화를 추천한다.

3. Instead of K-NN, implement matrix factorization using Singular Value Decomposition (SVD) for collaborative filtering. Show how the results differ. If SVD does not work due to the large size of the matrix, check the results using only a subset of the matrix.

KNN 대신 SVD를 적용하여 진행하고자 하였다. 코드는 다음과 같다.

```
from sklearn.decomposition import TruncatedSVD

def svd_recommender(X, movie_id, k):
    svd = TruncatedSVD(n_components=k)
    latent_matrix = svd.fit_transform(X)
    movie_vec = latent_matrix[movie_mapper[movie_id]].reshape(1, -1)

    similarities = np.dot(latent_matrix, movie_vec.T).flatten()
    similar_indices = similarities.argsort()[::-1][1:k+1]

    similar_ids = [movie_inv_mapper[i] for i in similar_indices]
    return similar_ids

similar_movies_svd = svd_recommender(X, movie_id, k)
print(f"SVD Recommendations for {movie_title}:")
for i in similar_movies_svd:
    print(movie_titles[i])

```

```
SVD Recommendations for Toy Story (1995):
Shawshank Redemption, The (1994)
Pulp Fiction (1994)
Silence of the Lambs, The (1991)
Star Wars: Episode IV - A New Hope (1977)
Matrix, The (1999)
Jurassic Park (1993)
Star Wars: Episode V - The Empire Strikes Back (1980)
Terminator 2: Judgment Day (1991)
Braveheart (1995)
Star Wars: Episode VI - Return of the Jedi (1983)
```

다른 KNN의 유클리드, 맨해튼은 평점 간의 실제 차이를 기반으로 영화들을 추천하고, 코사인

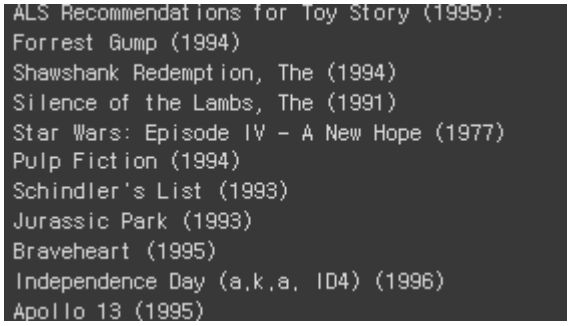
KNN은 장르와 테마가 유사한 영화들을 추천하는 경향이 있다. 그러나, 여기서 SVD는 잠재 요인을 기반으로 영화를 추천하고, 다양한 장르와 높은 평가를 받은 영화들을 포함하고 있다는 것을 알 수 있다. SVD가 특히나, 다른 방식에 비해 패턴을 잘 포착하여서 다양한 고평점 영화를 추천한다는 것을 알 수 있었다.

3. Implement matrix factorization using Alternating Least Square (ALS) for collaborative filtering. Show how the results differ.

KNN 대신 ALS를 적용하여 진행하고자 하였다. 코드는 다음과 같다.

```python
import implicit
from scipy import sparse

def als_recommender(X, movie_id, k, factors=20, regularization=0.1, iterations=20):
    sparse_item_user = sparse.csr_matrix(X.T)
    model = implicit.als.AlternatingLeastSquares(factors=factors, regularization=regularization, iterations=iterations)
    model.fit(sparse_item_user)

    movie_ind = movie_mapper[movie_id]
    movie_vec = model.item_factors[movie_ind]

    similarities = np.dot(model.item_factors, movie_vec)
    similar_indices = similarities.argsort()[::-1][1:k+1]

    similar_ids = [movie_inv_mapper[i] for i in similar_indices]
    return similar_ids

similar_movies_als = als_recommender(X, movie_id, k)
print(f"ALS Recommendations for {movie_title}:")
for i in similar_movies_als:
    print(movie_titles[i])
```

```
/usr/local/lib/python3.10/dist-packages/implicit/cpu/als.py:95: RuntimeWarning: OpenBLAS is configured to use 2 threads. It is highly recommended to disable its internal threadpool by set
  check_blas_config()
100%|██████████| 20/20 [00:17<00:00,  1.21s/it]
ALS Recommendations for Toy Story (1995):
Forrest Gump (1994)
Shawshank Redemption, The (1994)
Silence of the Lambs, The (1991)
Star Wars: Episode IV - A New Hope (1977)
Pulp Fiction (1994)
Schindler's List (1993)
Jurassic Park (1993)
Braveheart (1995)
Independence Day (a.k.a. ID4) (1996)
Apollo 13 (1995)
```



비교 결과는, KNN의 유클리드, 맨해튼은 평점 간의 실제 차이를 기반으로 영화들을 추천하고, 코사인 KNN은 장르와 테마가 유사한 영화들을 추천하는 경향이 있고, SVD는 잠재 요인을 기반으로 영화를 추천하고, 다양한 장르와 높은 평가를 받은 영화들을 포함하고 있다는 것을 이전에 알아내었고, 이 ALS는 SVD와 유사하게 잠재 요인을 기반으로 하면서, 더욱 복잡한 모델을 사용하기 때문에 더 다양한 장르와 평가가 높은 영화를 추천한다는 것을 알 수 있었다.