

머신러닝 - 추천 시스템

이론적 설명

- 콘텐츠 기반 : 특정 사용자 중심. 사용자가 특정 아이템을 선호하는 경우, 그와 유사한 콘텐츠를 가진 타 아이템 추천
- 협업 필터링
 - 1 최근접 이웃 기반 : 다른 사람들에 대한 정보를 통해 아이템 추천.
 - 사용자 기반 : 유사한 다른 사람들이 구매한 다음 상품의 추천
 - 아이템 기반 : 사용자가 선택한 상품을 동일하게 구매한 사람들이 다음번 구매한 항목을 추천
 - 2. 잠재 요인 기반(Latent, SVD) : 행렬 분해 기반

DATASET_ ML-100K

| movies.csv - Windows 메모장 | |
|--------------------------|--------------------------------------------------------------|
| 파일(F) | 편집(E) 서식(O) 보기(V) 도움말(H) |
| movieId,title,genres | |
| 1 | Toy Story (1995),Adventure Animation Children Comedy Fantasy |
| 2 | Jumanji (1995),Adventure Children Fantasy |
| 3 | Grumpier Old Men (1995),Comedy Romance |
| 4 | Waiting to Exhale (1995),Comedy Drama Romance |
| 5 | Father of the Bride Part II (1995),Comedy |
| 6 | Heat (1995),Action Crime Thriller |
| 7 | Sabrina (1995),Comedy Romance |
| 8 | Tom and Huck (1995),Adventure Children Fantasy |
| 9 | Sudden Death (1995),Action Thriller |
| 10 | GoldenEye (1995),Action Adventure Thriller |
| 11 | "American President, The (1995),Comedy Drama Romance |
| 12 | Dracula: Dead and Loving It (1995),Comedy Horror Romance |
| 13 | Balto (1995),Adventure Animation Children Comedy |
| 14 | Nixon (1995),Drama |
| 15 | Cutthroat Island (1995),Action Adventure Thriller |

| ratings.csv - Windows 메모장 | |
|---------------------------------|--------------------------|
| 파일(F) | 편집(E) 서식(O) 보기(V) 도움말(H) |
| userId,movieId,rating,timestamp | |
| 1 | 1,4.0,964982703 |
| 1 | 3,4.0,964981247 |
| 1 | 6,4.0,964982224 |
| 1 | 47,5.0,964983815 |
| 1 | 50,5.0,964982931 |
| 1 | 70,3.0,964982400 |
| 1 | 101,5.0,964980868 |
| 1 | 110,4.0,964982176 |
| 1 | 151,5.0,964984041 |
| 1 | 157,5.0,964984100 |
| 1 | 163,5.0,964983650 |
| 1 | 216,5.0,964981208 |
| 1 | 223,3.0,964980985 |
| 1 | 231,5.0,964981179 |
| 1 | 235,4.0,964980908 |

| links.csv - Windows 메모장 | |
|-------------------------|--------------------------|
| 파일(F) | 편집(E) 서식(O) 보기(V) 도움말(H) |
| movieId,imdbId,tmdbId | |
| 1 | 0114709,862 |
| 2 | 0113497,8844 |
| 3 | 0113228,15602 |
| 4 | 0114885,31357 |
| 5 | 0113041,11862 |
| 6 | 0113277,949 |
| 7 | 0114319,11860 |
| 8 | 0112302,45325 |
| 9 | 0114576,9091 |
| 10 | 0113189,710 |
| 11 | 0112346,9087 |
| 12 | 0112896,12110 |
| 13 | 0112453,21032 |
| 14 | 0113987,10858 |
| 15 | 0112760,1408 |

Surprise를 이용한 추천 시스템 구축

```
In [2]: import surprise
```

```
In [3]: surprise.__version__
```

```
Out [3]: '1.1.1'
```

```
In [4]: from surprise import SVD
```

```
In [5]: from surprise import Dataset
```

```
In [6]: from surprise import accuracy
```

```
In [7]: from surprise.model_selection import train_test_split
```

```
In [8]: data = Dataset.load_builtin('ml-100k')
```

```
In [9]: trainset, testset = train_test_split(data, test_size=.25, random_state=0)
```

```
In [10]: trainset
```

```
Out [10]: <surprise.trainset.Trainset at 0x17e6f665c50>
```

```
In [11]: testset
```

```
Out [11]: [('120', '282', 4.0),  
           ('882', '291', 4.0),  
           ('535', '507', 5.0),  
           ('697', '244', 5.0),  
           ('751', '385', 4.0),  
           ('219', '82', 1.0),  
           ('279', '571', 4.0),  
           ('429', '568', 3.0),  
           ('456', '100', 3.0),  
           ('249', '23', 4.0),  
           ('493', '183', 5.0),  
           ('325', '469', 4.0),  
           ('631', '682', 2.0),  
           ('276', '121', 4.0),  
           ('269', '405', 1.0),  
           ('159', '1095', 5.0),  
           ...]
```

```
In [12]: algo = SVD()
```

```
In [13]: algo.fit(trainset)
```

```
Out [13]: <surprise.prediction_algorithms.matrix_factorization.SVD at 0x17e70eab9e8>
```

```
In [14]: predictions = algo.test(testset)
```

```
In [15]: predictions[:5]
```

```
Out [15]: [Prediction(uid='120', iid='282', r_ui=4.0, est=3.6320611937072074, details={'was_impossible': False}),  
           Prediction(uid='882', iid='291', r_ui=4.0, est=3.743511328401798, details={'was_impossible': False}),  
           Prediction(uid='535', iid='507', r_ui=5.0, est=4.241697418758268, details={'was_impossible': False}),  
           Prediction(uid='697', iid='244', r_ui=5.0, est=3.6660659901542494, details={'was_impossible': False}),  
           Prediction(uid='751', iid='385', r_ui=4.0, est=3.296022519971667, details={'was_impossible': False})]
```

Surprice 추천 시스템의 fit과 확률

```
In [16]: accuracy.rmse(predictions)
```

```
RMSE: 0.9487
```

```
Out [16]: 0.948677632302057
```

```
In [17]: uid=str(196)
         iid=str(302)
         pred=algo.predict(uid, iid)
         pred
```

```
Out [17]: Prediction(uid='196', iid='302', r_ui=None, est=4.467098364098535, details={'was_impossible': False})
```

```
In [18]: import pandas as pd
```

```
In [19]: ratings = pd.read_csv('C:/Users/tjoen709-12/Desktop/ml-latest-small/ratings.csv')
```

```
In [20]: ratings.to_csv('C:/Users/tjoen709-12/Desktop/ml-latest-small/ratings_noHeader.csv', index=False)
```

```
In [21]: from surprise import Reader
```

```
In [22]: reader=Reader(line_format="user item rating timestamp", sep=',', rating_scale=(0.5, 5))
```

```
In [23]: data=Dataset.load_from_file('C:/Users/tjoen709-12/Desktop/ml-latest-small/ratings_noHeader.csv', reader=reader)
```

```
In [24]: data
```

```
Out [24]: <surprise.dataset.DatasetAutoFolds at 0x17e70eab668>
```

```
In [25]: trainset, testset=train_test_split(data, test_size=.25, random_state=0)
```

```
In [26]: algo = SVD(random_state=0)
```

```
In [27]: algo
```

```
Out [27]: <surprise.prediction_algorithms.matrix_factorization.SVD at 0x17e6f3b7940>
```

```
In [28]: algo.fit(trainset)
```

```
Out [28]: <surprise.prediction_algorithms.matrix_factorization.SVD at 0x17e6f3b7940>
```

```
In [29]: predictions = algo.test(testset)
```

```
In [30]: accuracy.rmse(predictions)
```

```
RMSE: 0.8692
```

```
Out [30]: 0.8691518972016722
```

```
In [ ]:
```

```
In [29]: algo = SVD()
         algo.fit(trainset)
```

```
Out [29]: <surprise.prediction_algorithms.matrix_factorization.SVD at 0x7f82ad5fc0d0>
```

```
In [30]: predictions = algo.test( testset )
         print('prediction type :', type(predictions), ' size:', len(predictions))
         print('prediction 결과의 최초 5개 추출')
         predictions[:5]
```

```
prediction type : <class 'list'> size: 25000
prediction 결과의 최초 5개 추출
```

```
Out [30]: [Prediction(uid='120', iid='282', r_ui=4.0, est=3.7836175713351237, details={'was_
           ssible': False}),
           Prediction(uid='882', iid='291', r_ui=4.0, est=3.6589028213478634, details={'was_
           ssible': False}),
           Prediction(uid='535', iid='507', r_ui=5.0, est=4.138458490697659, details={'was_ir
           sible': False}),
           Prediction(uid='697', iid='244', r_ui=5.0, est=3.294684812013692, details={'was_ir
           sible': False}),
           Prediction(uid='751', iid='385', r_ui=4.0, est=3.304936965664121, details={'was_ir
           sible': False})]
```

```
In [31]: [ (pred.uid, pred.iid, pred.est) for pred in predictions[:3] ]
```

```
Out [31]: [('120', '282', 3.7836175713351237),
           ('882', '291', 3.6589028213478634),
           ('535', '507', 4.138458490697659)]
```

```
In [32]: # 사용자 아이디, 아이템 아이디는 문자열로 입력해야 함.
         uid = str(196)
         iid = str(302)
         pred = algo.predict(uid, iid)
         print(pred)
```

```
user: 196      item: 302      r_ui = None  est = 4.26  {'was_impossible': Fa
```

```
In [33]: accuracy.rmse(predictions)
```

```
RMSE: 0.9465
```

```
Out [33]: 0.946530325860163
```

```
RMSE: 0.9465
```

```
Out [33]: 0.946530325860163
```

Surprise 주요 모듈 소개

```
In [10]: import pandas as pd
```

```
ratings = pd.read_csv('ratings.csv')
# ratings_noh.csv 파일로 unload 시 index 와 header를 모두 제거한 새로운 파일 생성.
ratings.to_csv('ratings_noh.csv', index=False, header=False)
```

```
In [11]: from surprise import Reader
```

```
reader = Reader(line_format='user item rating timestamp', sep=',', rating_scale=(0.5,
5))
data=Dataset.load_from_file('ratings_noh.csv', reader=reader)
```

```
In [12]: trainset, testset = train_test_split(data, test_size=.25, random_state=0)
```

```
# 수행시마다 동일한 결과 도출을 위해 random_state 설정
algo = SVD(n_factors=50, random_state=0)
```

```
# 학습 데이터 세트로 학습 후 테스트 데이터 세트로 평점 예측 후 RMSE 평가
algo.fit(trainset)
predictions = algo.test( testset )
accuracy.rmse(predictions)
```

```
RMSE: 0.8682
```

```
Out [12]: 0.8681952927143516
```

```
In [13]: import pandas as pd
         from surprise import Reader, Dataset
```

```
ratings = pd.read_csv('ratings.csv')
reader = Reader(rating_scale=(0.5, 5.0))
```

```
# ratings DataFrame 에서 컬럼은 사용자 아이디, 아이템 아이디, 평점 순서를 지켜야 한다.
data = Dataset.load_from_df(ratings[['userId', 'movieId', 'rating']], reader)
trainset, testset = train_test_split(data, test_size=.25, random_state=0)
```

```
algo = SVD(n_factors=50, random_state=0)
algo.fit(trainset)
predictions = algo.test( testset )
accuracy.rmse(predictions)
```

```
RMSE: 0.8682
```

```
Out [13]: 0.8681952927143516
```

교차 검증(Cross Validation)과 하이퍼 파라미터 튜닝

```
In [14]: from surprise.model_selection import cross_validate

# Pandas DataFrame에서 Surprise Dataset으로 데이터 로딩
ratings = pd.read_csv('ratings.csv') # reading data in pandas df
reader = Reader(rating_scale=(0.5, 5.0))
data = Dataset.load_from_df(ratings[['userId', 'movieId', 'rating']], reader)

algo = SVD(random_state=0)
cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

| | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 | Mean | Std |
|----------------|--------|--------|--------|--------|--------|--------|--------|
| RMSE (testset) | 0.8808 | 0.8675 | 0.8711 | 0.8712 | 0.8706 | 0.8722 | 0.0045 |
| MAE (testset) | 0.6763 | 0.6674 | 0.6683 | 0.6690 | 0.6682 | 0.6698 | 0.0033 |
| Fit time | 4.82 | 4.76 | 4.79 | 4.77 | 4.81 | 4.79 | 0.02 |
| Test time | 0.14 | 0.24 | 0.16 | 0.18 | 0.26 | 0.19 | 0.05 |

```
Out[14]: {'fit_time': (4.8206703662872314,
4.764417409896851,
4.79180908203125,
4.772138833999634,
4.809448957443237),
'test_mae': array([0.67631873, 0.66738312, 0.66832042, 0.66897396, 0.66819498]),
'test_rmse': array([0.88079396, 0.86746231, 0.87111614, 0.87123139, 0.87056727]),
'test_time': (0.1420269012451172,
0.23880982398986816,
0.15640544891357422,
0.17791128158569336,
0.25979161262512207)}
```

```
In [15]: from surprise.model_selection import GridSearchCV

# 최적화할 파라미터들을 딕셔너리 형태로 지정.
param_grid = {'n_epochs': [20, 40, 60], 'n_factors': [50, 100, 200] }

# CV를 3개 폴드 세트로 지정, 성능 평가는 rmse, mae 로 수행 하도록 GridSearchCV 구성
gs = GridSearchCV(SVD, param_grid, measures=['rmse', 'mae'], cv=3)
gs.fit(data)

# 최고 RMSE Evaluation 점수와 그때의 하이퍼 파라미터
print(gs.best_score['rmse'])
print(gs.best_params['rmse'])

0.8757833991640306
{'n_epochs': 20, 'n_factors': 50}
```

Surprise 를 이용한 개인화 영화 추천 시스템 구축

Surprise 를 이용한 개인화

```
In [16]: # 아래 코드는 train_test_split()
# 를 발생시킵니다.
data = Dataset.load_from_df(rat
algo = SYD(n_factors=50, random
algo.fit(data)
```

```
AttributeError
<ipython-input-16-33c08dace4bd>
2 data = Dataset.load_from_d
3 algo = SYD(n_factors=50)
----> 4 algo.fit(data)
```

```
/usr/local/lib/python3.7/dist-packa
tion.pyx in surprise.prediction
/usr/local/lib/python3.7/dist-packa
tion.pyx in surprise.prediction
```

```
AttributeError: 'DatasetAutoFolds' object has no attribute 'fit'
```

```
In [17]: from surprise.dataset import
```

```
reader = Reader(line_format='us
5))
# DatasetAutoFolds 클래스를 rat
data_folds = DatasetAutoFolds(r
#전체 데이터를 학습데이터로 생성
trainset = data_folds.build_full_trainset()
```

```
In [ ]: algo = SYD(n_epochs=20, n_factors=50, random_state=0)
algo.fit(trainset)
```

```
In [21]: # 영화에 대한 상세 속성 정보 DataFrame로딩
movies = pd.read_csv('movies.csv')

# userID=9 의 movied 데이터 추출하여 movied=42 데이터가 있는지 확인.
movieds = ratings[ratings['userID']==9]['movied']
if movieds[movieds==42].count() == 0:
    print('사용자 아이디 9는 영화 아이디 42의 평점 없음')

print(movies[movies['movied']==42])
```

```
사용자 아이디 9는 영화 아이디 42의 평점 없음
   movied  title  genres
38      42  Dead Presidents (1995)  Action|Crime|Drama
```

```
In [34]: uid = str(9)
iid = str(42)
```

```
In [34]: uid = str(9)
iid = str(42)
```

```
pred = algo.predict
```

```
user: 9
```

```
In [35]: def get_unseen_sur
```

```
#입력값으로 넘
```

```
seen_movies =
```

```
# 모든 영화들
```

```
total_movies
```

```
# 모든 영화들
```

```
unseen_movies
```

```
print('평점 0
```

```
s), #
```

```
'전체 0
```

```
return unseen
```

```
unseen_movies = g
```

```
평점 매긴 영화수:
```

```
unseen_movies = get_unseen_surprise(ratings, movies, 9)
```

평점 매긴 영화수: 46 추천대상 영화수: 9696 전체 영화수: 9742

```
In [36]: def recomm_movie_by_surprise(algo, userID, unseen_movies, top_n=10):
# 알고리즘 객체의 predict() 메서드를 평점이 없는 영화에 반복 수행한 후 결과를 lis
t 객체로 저장
predictions = [algo.predict(str(userID), str(movied))] for movied in unseen_movi
es]
```

```
# predictions list 객체는 surprise의 Predictions 객체를 원소로 가지고 있을.
# [Prediction(uid='9', iid='1', est=3.69), Prediction(uid='9', iid='2', est=2.9
8),...]
# 이를 est 값으로 정렬하기 위해서 아래의 sortkey_est 함수를 정의함.
# sortkey_est 함수는 list 객체의 sort() 함수의 키 값으로 사용되어 정렬 수행.
def sortkey_est(pred):
    return pred.est
```

```
# sortkey_est( ) 반환값의 내림 차순으로 정렬 수행하고 top_n개의 최상위 값 추출.
predictions.sort(key=sortkey_est, reverse=True)
top_predictions= predictions[:top_n]
```

```
# top_n으로 추출된 영화의 정보 추출. 영화 아이디, 추천 예상 평점, 제목 추출
top_movie_ids = [int(pred.iid) for pred in top_predictions]
top_movie_rating = [pred.est for pred in top_predictions]
top_movie_titles = movies[movies.movied.isin(top_movie_ids)][['title']]
top_movie_preds = [(id, title, rating) for id, title, rating in zip(top_movie_id
s, top_movie_titles, top_movie_rating)]
```

```
return top_movie_preds
```

```
unseen_movies = get_unseen_surprise(ratings, movies, 9)
top_movie_preds = recomm_movie_by_surprise(algo, 9, unseen_movies, top_n=10)
print('##### Top-10 추천 영화 리스트 #####')
```

```
for top_movie in top_movie_preds:
    print(top_movie[1], ":", top_movie[2])
```

```
평점 매긴 영화수: 46 추천대상 영화수: 9696 전체 영화수: 9742
##### Top-10 추천 영화 리스트 #####
Usual Suspects, The (1995) : 4.840787856443575
Free Willy 2: The Adventure Home (1995) : 4.723660700223147
Love & Human Remains (1993) : 4.682973291927803
Madness of King George, The (1994) : 4.620637976239669
Shawshank Redemption, The (1994) : 4.594653203319725
Four Weddings and a Funeral (1994) : 4.585005406992522
8 Seconds (1994) : 4.5771747273727605
Radioland Murders (1994) : 4.5279555303167545
Land and Freedom (Tierra y libertad) (1995) : 4.52789793004861
Inspector General, The (1949) : 4.527445503118298
```