



# 17주차 발표

최하경 문수인

# 목차

01 추천시스템

02 콘텐츠 기반 필터링

03 최근접 이웃 협업 필터링

04 잠재 요인 협업 필터링

05 콘텐츠 기반 필터링 실습

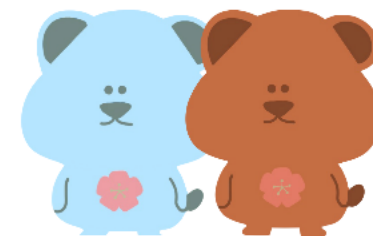
06 아이템 기반 최근접 이웃 협업 필터링 실습

07 잠재 요인 협업 필터링 실습

08 Surprise 패키지



## 01 추천시스템



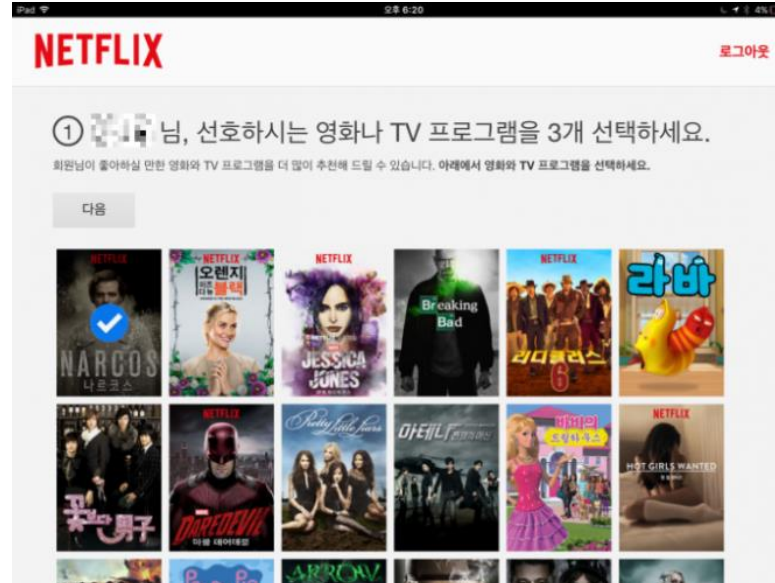
# #1.1 추천시스템의 개요



## 추천시스템

사용자의 선호도 및 과거 행동, 아이템의 특성, 그 외 정보 등을 토대로

특정 사용자가 관심을 가질 만한 아이템 (영화, 음악, 책, 뉴스, 이미지, 웹 페이지 등)을 추천하는 것



# #1.2 추천시스템의 유형

## 1. 콘텐츠 기반 필터링

사용자가 선호하는 아이템과 비슷한 아이템을 추천하는 방식

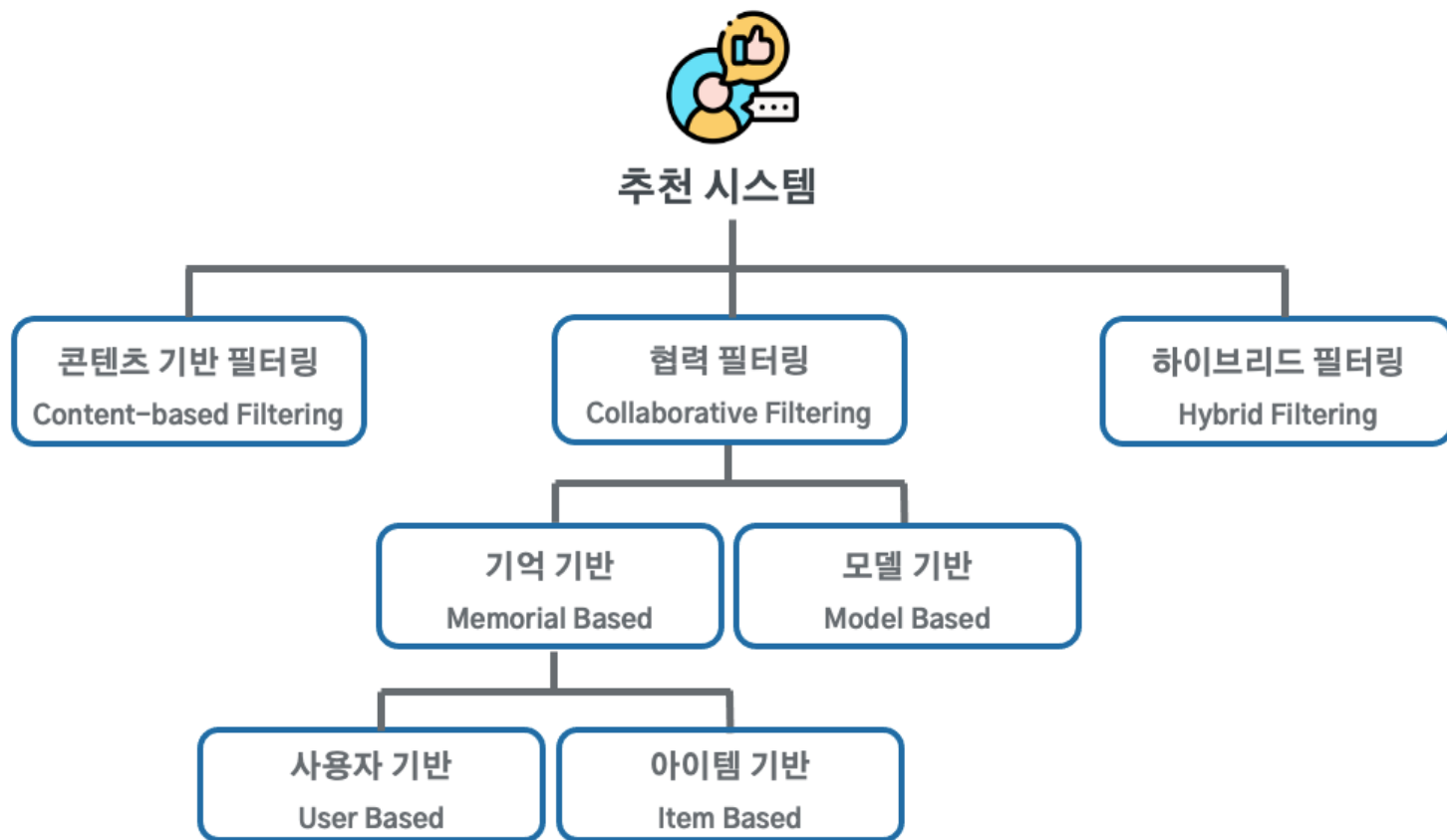
## 2. 협업 필터링

User-Item 매트릭스를 사용, User가 내린 평가와 Item이 받은 평가를 활용한  
평가 예측 및 추천 방식

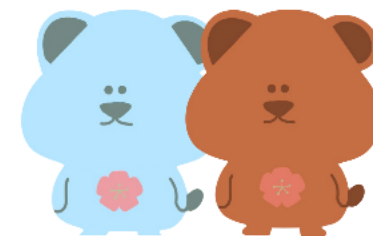
2.1 최근접 이웃 협업 필터링

2.2 잠재 요인 협업 필터링

# #1.2 추천시스템의 유형



## 02 콘텐츠 기반 필터링

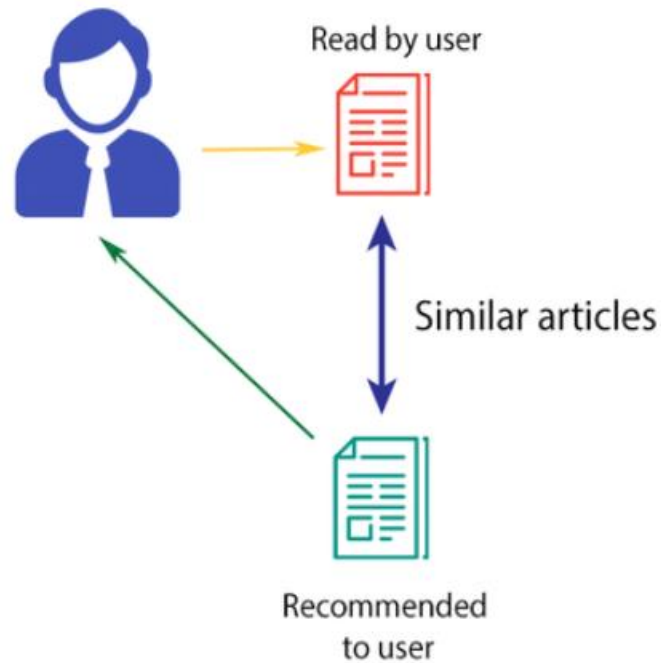


## #2 콘텐츠 기반 필터링



### Content based filtering

사용자가 선호하는 아이템과 비슷한 콘텐츠를 가진 다른 아이템을 추천하는 방식





# #2 콘텐츠 기반 필터링

예)

item : 영화

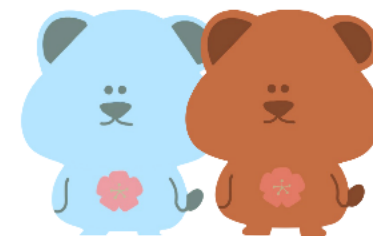
item의 콘텐츠 : 장르, 출연 배우, 감독, 영화 키워드 등 -> 곧, feature

	id	imdb_id	original_title	director	production	genre	cast	budget	revenue	runtime	release_year	vote_count
0	135397	tt0369610	Jurassic World	Colin Trevorrow	Universal Studios	Action	Chris Pratt	150000000	1513528810	124	2015	5562
1	76341	tt1392190	Mad Max Fury Road	George Miller	Village Roadshow Pictures	Action	Tom Hardy	150000000	378436354	120	2015	6185
2	262500	tt2908446	Insurgent	Robert Schwentke	Summit Entertainment	Adventure	Shailene Woodley	110000000	295238201	119	2015	2480
3	140607	tt2488496	Star Wars The Force Awakens	JJ Abrams	Lucasfilm	Action	Harrison Ford	200000000	2068178225	136	2015	5292
4	168259	tt2820852	Furious	James Wan	Universal Pictures	Action	Vin Diesel	190000000	1506249360	137	2015	2947
5	281957	tt1663202	The Revenant	Alejandro Gonzlez Iritu	Regency Enterprises	Western	Leonardo DiCaprio	135000000	532950503	156	2015	3929
6	87101	tt1340138	Terminator Genisys	Alan Taylor	Paramount Pictures	Science Fiction	Arnold Schwarzenegger	155000000	440603537	125	2015	2598
7	286217	tt3659388	The Martian	Ridley Scott	Twentieth Century Fox Film Corporation	Drama	Matt Damon	108000000	595380321	141	2015	4572
8	211672	tt2293640	Minions	Kyle Balda Pierre Coffin	Universal Pictures	Family	Sandra Bullock	74000000	1156730962	91	2015	2893
9	150540	tt2096673	Inside Out	Pete Docter	Walt Disney Pictures	Comedy	Amy Poehler	175000000	853708609	94	2015	3935

비슷한 item?  
by item의 유사도

item 콘텐츠 (feature)  
벡터 형태의 표현  
-> 벡터 유사도

## 03 최근접 이웃 협업 필터링

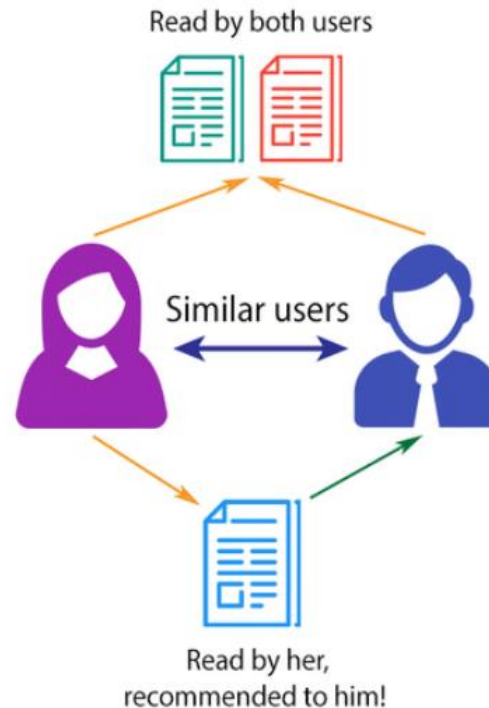


# #3.1 협업 필터링



## Collaborative filtering























User-Item 행렬 기반으로, User가 Item에 대해 내린 평가와 Item이 받은 평가들을 활용한 추천 및 평가 예측 방법



# #3.1 협업 필터링

## User-Item 행렬

일반적으로 사용자가 일부의 아이템에 대해서만 사용 및 평가하기 때문에 희소 행렬이다.

					
	Book 1	Book 2	Book 3	Book 4	Book 5
 User A					
 User B					
 User C					
 User D					

## #3.2 최근접 이웃 협업 필터링



### Nearest neighbor collaborative filtering

Memory 협업 필터링이라고도 하며, 과거 데이터 (memory)를 바탕으로 한 유사도 기반 추천 방식

#### 1. 사용자 기반 (User-User) 유사도

특정 사용자와 유사한 사용자들이 선호하는 아이템 추천  
“당신과 비슷한 고객들이 다음 상품도 구매했습니다 ”

#### 2. 아이템 기반 (Item-Item) 유사도

특정 아이템과 평점 분포가 유사한 다른 아이템 추천  
“이 상품을 선택한 다른 고객들은 다음 상품도 구매했습니다 ”

## #3.2 최근접 이웃 협업 필터링



### 1. 사용자 기반 (User-User)

사용자 A는 영화 평점 측면에서 사용자 C 보다 사용자 B와 유사도가 높음  
=> A에게 B가 선호하는 프로메테우스 추천

		다크 나이트	인터스텔라	엣지오브 투모로우	프로메테우스	스타워즈 라스트제다이
상호간 유사도 높음	사용자 A	5	4	4		
	사용자 B	3	3	4	5	3
	사용자 C	4	3	3	2	5

사용자 유사도 : 나이, 성별 등 특성의 유사도가 아니라,  
아이템에 대한 평가의 유사도를 말함

## #3.2 최근접 이웃 협업 필터링



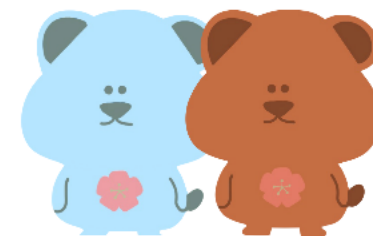
### 2. 아이템 기반 (Item-Item)

영화 (아이템) 프로메테우스는 평점 분포 측면에서 스타워즈보다 다크 나이트와 유사도가 높음  
=> 다크 나이트를 선호하는 사용자 D에게 스타워즈보다는 프로메테우스 추천

		사용자 A	사용자 B	사용자 C	사용자 D	사용자 E
상호간 유사도 높음	다크 나이트	5	4	5	5	5
	프로메테우스	5	4	4		5
	스타워즈 라스트제다이	4	3	3		4

아이템 유사도 : 아이템 속성의 유사도가 아니라,  
사용자들이 아이템에 대해 매긴 평가 분포의 유사도를 말함

## 04 잠재 요인 협업 필터링





# #4.1 잠재 요인 협업 필터링의 이해



## Latent factor collaborative filtering

사용자-아이템 평점 행렬의 잠재 요인을 추출

사용자-잠재요인 행렬과 잠재요인-아이템 행렬로 분해하고,  
분해된 두 행렬을 내적을 통해 결합함으로써  
사용자가 평가하지 않은 아이템에 대한 평가를 예측하는 방법

## #4.2 행렬 분해



### 행렬 분해

다차원 행렬을 저차원 행렬로 분해하는 기법  
SVD, NMF 등

사용자-아이템 행렬  $R$  ( $4 \times 5$ ) 를 행렬 분해를 통해  
사용자-잠재요인 행렬  $P$  ( $4 \times 2$ ) 와 잠재요인-아이템 행렬  $Q.T$  ( $2 \times 5$ )로 분해

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	4			2	
User 2		5		3	
User 3			3	4	4
User 4	5	2	1	2	

사용자-아이템평점 행렬  $R$



	Factor 1	Factor 2
User 1	0.94	0.96
User 2	2.14	0.08
User 3	1.93	1.79
User 4	0.58	1.59

사용자-잠재요인 행렬  $P$



	Item 1	Item 2	Item 3	Item 4	Item 5
Factor 1	1.7	2.3	1.41	1.36	0.41
Factor 2	2.49	0.41	0.14	0.75	1.77

잠재요인-아이템 행렬  $Q.T$

## #4.2 행렬 분해

User 1의 Item 3에 대한 예측 평점

: User 1에 대한 잠재요인과 Item 3에 대한 잠재요인의 전치행렬의 곱으로 계산 가능

$$r_{(u,i)} = p_u * q_i^t$$

	Factor 1	Factor 2		Item 1	Item 2	Item 3	Item 4	Item 5		Item 3		
User 1	0.94	0.96	X	Factor 1	1.7	2.3	1.41	1.36	0.41	=	User 1	1.46
User 2	2.14	0.08		Factor 2	2.49	0.41	0.14	0.75	1.77			
User 3	1.93	1.79										
User 4	0.58	1.59										

$$\widehat{R(1,3)} \approx 0.94 * 1.41 + 0.96 * 0.14 = 1.46$$

## #4.2 행렬 분해

행렬 분해를 통해 분해된 사용자-잠재요인 행렬 P와 잠재요인-아이템 행렬 Q.T의 내적을 통해 모든 예측 평점 행렬  $\hat{R}$  계산 가능

	Factor 1	Factor 2
User 1	0.94	0.96
User 2	2.14	0.08
User 3	1.93	1.79
User 4	0.58	1.59

사용자-잠재요인 행렬 P

X

	Item 1	Item 2	Item 3	Item 4	Item 5
Factor 1	1.7	2.3	1.41	1.36	0.41
Factor 2	2.49	0.41	0.14	0.75	1.77

잠재요인-아이템 행렬 Q.T

=

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	3.98	2.56	1.46	2	2.08
User 2	3.82	5	3.02	2.97	1.02
User 3	5	5	2.96	3.97	4.95
User 4	4.95	1.99	1.04	1.99	3.05

사용자-아이템평점 예측 행렬  $\hat{R}$

## #4.3 확률적 경사 하강법을 이용한 행렬 분해

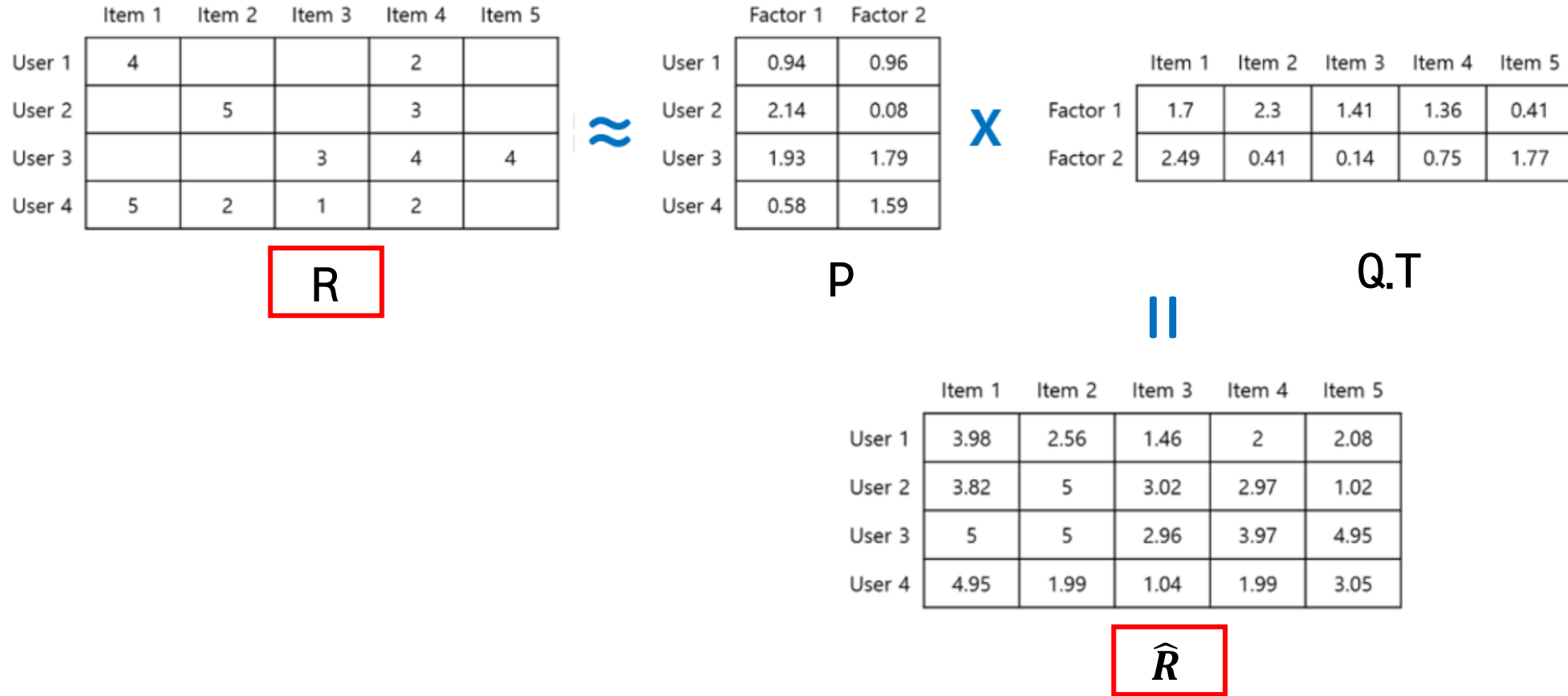
행렬 분해는 주로 SVD를 사용하지만,  
SVD는 null 값이 없는 행렬에만 사용 가능

user-item 평점 행렬 R은 null 값이 많은 희소 행렬

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	4	?	?	2	?
User 2	?	5	?	3	?
User 3	?	?	3	4	4
User 4	5	2	1	2	?

=> 확률적 경사 하강법을 이용한 SVD

## #4.3 확률적 경사 하강법을 이용한 행렬 분해



P와 Q로 계산된 예측 행렬  $\hat{R}$  이 실제 행렬 R과 같아지도록  
P와 Q를 조정하는 방법

## #4.3 확률적 경사 하강법을 이용한 행렬 분해

확률적 경사 하강법을 이용한 행렬 분해 절차

- [1] P와 Q를 임의의 값을 가진 행렬로 초기화
- [2]  $P * Q.T$  로 예측 행렬 R을 계산하고, 실제 행렬 R과의 오차 계산
- [3] 이 오차를 최소화하는 방향으로 P와 Q 행렬 값 업데이트
- [4] 오차가 특정 값 이하가 될 때까지 2, 3번 반복

## #4.3 확률적 경사 하강법을 이용한 행렬 분해

오차값 최소화와 과적합 방지를 위한 L2 규제를 위한 비용 함수

$$\min \sum (r_{(u,i)} - p_u q_i)^2 + \lambda (\|q_i\|^2 + \|p_u\|^2)$$

$r_{(u,i)}$  : 실제 R 행렬의 u행, i열에 위치한 값  
 $p_u$  : P 행렬의 사용자 u행 벡터  
 $q_i$  : Q 행렬의 아이템 i행 전치벡터  
 $\lambda$  : L2 정규화 계수

위 비용 함수를 최소화하기 위해 업데이트되는 p, q 값

$$p_{u_{new}} = p_u + \eta (e_{(u,i)} * q_i - \lambda * p_u)$$

$$q_{i_{new}} = q_i + \eta (e_{(u,i)} * p_u - \lambda * q_i)$$

$\hat{e}_{(u,i)}$  : u행, i열에 위치한 실제 행렬 값과 예측 행렬 값 차이  
 $\eta$  : 학습률



## #4.3 확률적 경사 하강법을 이용한 행렬 분해

### 1. 원본행렬 R 생성, P와 Q를 임의의 값을 가진 행렬로 초기화

```
import numpy as np

R = np.array([[4, np.NaN, np.NaN, 2, np.NaN ],
              [np.NaN, 5, np.NaN, 3, 1 ],
              [np.NaN, np.NaN, 3, 4, 4 ],
              [5, 2, 1, 2, np.NaN ]])

num_users, num_items = R.shape
K=3

np.random.seed(1)
P = np.random.normal(scale=1./K, size=(num_users, K))
Q = np.random.normal(scale=1./K, size=(num_items, K))
```

null 값을 포함한 행렬 R 생성

잠재 요인 차원 = 3으로 설정

(user, 잠재요인) 차원의 행렬 P  
(item, 잠재요인) 차원의 행렬 Q 를  
정규분포를 가진 랜덤 값으로 초기화

# #4.3 확률적 경사 하강법을 이용한 행렬 분해

## 2. steps=1000번 동안 p, q 업데이트

```
non_zeros = [ (i, j, R[i,j]) for i in range(num_users) for j in range(num_items) if R[i,j] > 0 ]

steps=1000
learning_rate=0.01
r_lambda=0.01

for step in range(steps):
    for i, j, r in non_zeros:
        eij = r - np.dot(P[i, :], Q[j, :].T)
        P[i, :] = P[i, :] + learning_rate*(eij * Q[j, :] - r_lambda*P[i, :])
        Q[j, :] = Q[j, :] + learning_rate*(eij * P[i, :] - r_lambda*Q[j, :])

    rmse = get_rmse(R, P, Q, non_zeros)
    if (step % 50) == 0 :
        print("### iteration step : ", step, " rmse : ", rmse)
```

-> R에서 널 값 제외한 데이터의 행렬 인덱스 추출

Steps: SGD를 반복해서 업데이트할

횟수

Learning rate: SGD의 학습률

R\_lambda: L2 regularization계수

$$p_{u_{new}} = p_u + \eta(e_{(u,i)} * q_i - \lambda * p_u)$$

$$q_{i_{new}} = q_i + \eta(e_{(u,i)} * p_u - \lambda * q_i)$$

## #4.3 확률적 경사 하강법을 이용한 행렬 분해

### 3. P, Q 행렬을 통해 예측 행렬 만들어 출력

```
1 pred_matrix = np.dot(P, Q.T)
2 print('예측 행렬:\n', np.round(pred_matrix, 3))
```

예측 행렬 :

```
[[3.991 0.897 1.306 2.002 1.663]
 [6.696 4.978 0.979 2.981 1.003]
 [6.677 0.391 2.987 3.977 3.986]
 [4.968 2.005 1.006 2.017 1.14 ]]
```

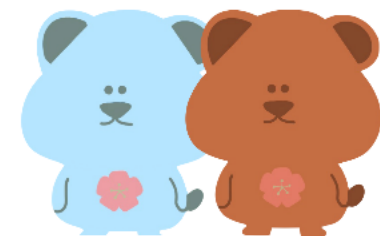
예측 행렬

```
[[[4, np.NaN, np.NaN, 2, np.NaN ]
 [np.NaN, 5, np.NaN, 3, 1 ],
 [np.NaN, np.NaN, 3, 4, 4 ],
 [5, 2, 1, 2, np.NaN ]])]
```

원본 행렬

null 값 => 새로운 예측 값으로 채워짐  
null이 아닌 값 => 큰 차이가 나지 않음

## 05 콘텐츠 기반 필터링 실습



# #5 콘텐츠 기반 필터링 실습

## #1 장르 콘텐츠 유사도 측정

- 문자열로 변환된 장르 칼럼을 countvectorizer를 이용해 피처 벡터화
- 피처 벡터화한 행렬 데이터 값을 코사인 유사도를 이용해 비교
  - 데이터셋의 레코드별로 타 레코드와 장르에서 코사인 유사도 값을 가지는 객체 생성
- 장르 유사도가 높은 영화 중 평점이 높은 순으로 영화 추천

```
def find_sim_movie(df, sorted_ind, title_name, top_n=10):  
  
    # 인자로 입력된 movies_df DataFrame에서 'title' 컬럼이 입력된 title_name 값인 DataFrame추출  
    title_movie = df[df['title'] == title_name]  
  
    # title_name을 가진 DataFrame의 index 객체를 ndarray로 반환하고  
    # sorted_ind 인자로 입력된 genre_sim_sorted_ind 객체에서 유사도 순으로 top_n 개의 index 추출  
    title_index = title_movie.index.values  
    similar_indexes = sorted_ind[title_index, :(top_n)]  
  
    # 추출된 top_n index들 출력. top_n index는 2차원 데이터 임.  
    # dataframe에서 index로 사용하기 위해서 1차원 array로 변경  
    print(similar_indexes)  
    similar_indexes = similar_indexes.reshape(-1)  
  
    return df.iloc[similar_indexes]
```

# #5 콘텐츠 기반 필터링 실습

## #2 장르 콘텐츠 필터링을 이용한 영화 추천

- 앞서 장르 유사도에 따라 영화를 추천하는 함수 생성
- 평점이 낮은 작품들이 추천되는 경우 발생 -> 개선 필요!
- 더 많은 후보군을 선정한 다음 영화의 평점에 따라 필터링해 최종 추천하는 방식으로 변경
- 왜곡된 평점 데이터를 회피할 수 있도록 평점에 평가횟수를 반영할 수 있는 새로운 평가 방식 필요

-> 가중 평점 방식

$$\text{가중 평점(Weighted Rating)} = \left( \frac{v}{(v+m)} \right) * R + \left( \frac{m}{(v+m)} \right) * C$$

- v: 개별 영화에 평점을 투표한 횟수 = `movies_df['vote_count']`
- m: 평점을 부여하기 위한 최소 투표 횟수
- R: 개별 영화에 대한 평균 평점 = `movies_df['vote_average']`
- C: 전체 영화에 대한 평균 평점 = `movies_df['vote_average'].mean()`

```
percentile = 0.6
m = movies_df['vote_count'].quantile(percentile)
C = movies_df['vote_average'].mean()

def weighted_vote_average(record):
    v = record['vote_count']
    R = record['vote_average']

    return ( (v/(v+m)) * R ) + ( (m/(m+v)) * C )

movies_df['weighted_vote'] = movies_df.apply(weighted_vote_average, axis=1)
```

## 06 아이템 기반 최근접 이웃 협업 필터링 실습



# #6 아이템 기반 최근접 이웃 협업 필터링 실습

## #1 협업 필터링을 사용하기 위한 데이터 전처리

- 협업 필터링 : 사용자와 아이템 간 평점에 기반해 추천
- 사용자-평점 데이터 (row : 사용자, col : 영화) 생성

User ID	Item ID	Rating
User 1	Item 1	3
User 1	Item 3	3
User 2	Item 1	4
User 2	Item 2	1
User 3	Item 4	5



변환

	Item 1	Item 2	Item 3	Item 4
User 1	3		3	
User 2	4	1		
User 3				5

## #2 영화 간 유사도 산출

- Cosine similarity 이용해 유사도 계산
- 사용자가 기준인 행 레벨 데이터이기 때문에, 영화 간의 유사도가 아닌 사용자 간의 유사도가 만들어짐.
- Transpose()를 적용해 “영화” 간 코사인 유사도를 구해야 함.

```
from sklearn.metrics.pairwise import cosine_similarity

item_sim = cosine_similarity(ratings_matrix_T, ratings_matrix_T)

# cosine_similarity() 로 반환된 넘파이 행렬을 영화명을 매핑하여 DataFrame으로 변환
item_sim_df = pd.DataFrame(data=item_sim, index=ratings_matrix.columns,
                           columns=ratings_matrix.columns)
```



# #6 아이템 기반 최근접 이웃 협업 필터링 실습

## #3 아이템 기반 최근접 이웃 협업 필터링으로 개인화된 영화 추천

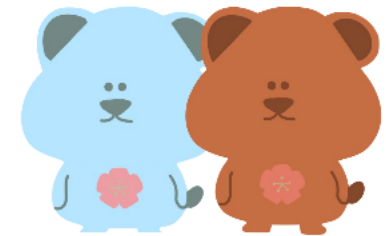
- 앞선 추천 방법은 개인적인 취향을 반영하지 않고, 오직 영화 간의 유사도만을 갖고 추천한 것
- 개인화된 영화 추천의 특징 : 아직 관람하지 않은 영화를 추천한다는 것
- 아직 관람하지 않은 영화에 대한 유사도와 기존에 관람한 영화의 평점 데이터를 기반으로 예측 평점 계산

$$\hat{R}_{u,i} = \frac{\sum (S_{i,N} * R_{u,N})}{\sum |S_{i,N}|}$$

- $\hat{R}_{u,i}$  : 사용자 u, 아이템 i의 개인화된 예측 평점 값
- $S_{i,N}$  : 아이템 i와 가장 유사도가 높은 Top-N개 아이템의 유사도 벡터
- $R_{u,N}$  : 사용자 u의 아이템 i와 가장 유사도가 높은 Top-N개 아이템에 대한 실제 평점 벡터

```
def predict_rating(ratings_arr, item_sim_arr):  
    ratings_pred = ratings_arr.dot(item_sim_arr) / np.array([np.abs(item_sim_arr).sum(axis=1)])  
    return ratings_pred
```

## 07 잠재 요인 협업 필터링 실습



# #7 행렬 분해를 이용한 잠재 요인 협업 필터링 실습

## #1 SGD 기반의 행렬 분해 이용

- 사용자-아이템 평점 행렬을 행렬 분해

```
def matrix_factorization(R, K, steps=200, learning_rate=0.01, r_lambda = 0.01):
    num_users, num_items = R.shape
    np.random.seed(1)
    P = np.random.normal(scale=1./K, size=(num_users, K))
    Q = np.random.normal(scale=1./K, size=(num_items, K))

    break_count = 0

    non_zeros = [ (i, j, R[i,j]) for i in range(num_users) for j in range(num_items) if R[i,j] > 0 ]

    for step in range(steps):
        for i, j, r in non_zeros:
            eij = r - np.dot(P[i, :], Q[j, :].T)
            P[i, :] = P[i, :] + learning_rate*(eij * Q[j, :] - r_lambda*P[i,:])
            Q[j, :] = Q[j, :] + learning_rate*(eij * P[i, :] - r_lambda*Q[j,:])

        rmse = get_rmse(R, P, Q, non_zeros)
        if (step % 10) == 0 :
            print("### iteration step : ", step, " rmse : ", rmse)

    return P, Q
```

```
P, Q = matrix_factorization(ratings_matrix.values, K=50, steps=200, learning_rate=0.01, r_lambda = 0.01)
pred_matrix = np.dot(P, Q.T)
```

- 잠재 요인 협업 필터링으로 추천

```
unseen_list = get_unseen_movies(ratings_matrix, 9)

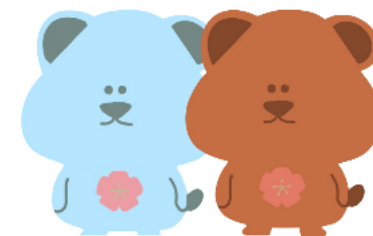
recomm_movies = recomm_movie_by_userid(ratings_pred_matrix, 9, unseen_list, top_n=10)

recomm_movies = pd.DataFrame(data=recomm_movies.values, index=recomm_movies.index, columns=[ 'pred_score' ])
recomm_movies
```

	pred_score
title	
Rear Window (1954)	5.704612
South Park: Bigger, Longer and Uncut (1999)	5.451100
Rounders (1998)	5.298393
Blade Runner (1982)	5.244951
Roger & Me (1989)	5.191962
Gattaca (1997)	5.183179
Ben-Hur (1959)	5.130463
Rosencrantz and Guildenstern Are Dead (1990)	5.087375
Big Lebowski, The (1998)	5.038690
Star Wars: Episode V - The Empire Strikes Back (1980)	4.989601

반환된 행렬을 영화 타이틀을 칼럼으로 갖는 데이터프레임으로 변경

## 08 Surprise 패키지



# #8 Surprise 패키지

## #1 Surprise 패키지

- 파이썬 기반의 추천 시스템 구축을 위한 전용 패키지

## #2 장점

- 다양한 추천 알고리즘 기반의 잠재 요인 협업 필터링을 쉽게 적용해 추천 시스템 구축할 수 있음
- 핵심 API는 사이킷런의 핵심 API와 유사한 API 명으로 작성됨

# #8 Surprise 패키지

## #3 Surprise 추천 알고리즘 클래스

클래스명	설명
SVD	행렬 분해를 통한 잠재 요인 협업 필터링을 위한 SVD 알고리즘
KNNBasic	최근접 이웃 협업 필터링을 위한 KNN 알고리즘
BaselineOnly	사용자 Bias와 아이템 Bias를 고려한 SGD 베이스라인알고리즘

## #4 교차검증과 하이퍼 파라미터 튜닝

- 교차검증과 하이퍼 파라미터 튜닝을 위해 사이킷런과 유사한 cross\_validate()와 gridsearchCV 클래스 제공

함수	설명
cross_validate()	<ul style="list-style-type: none"><li>- 폴드된 데이터 세트의 개수와 성능 측정 방법을 명시해 교차 검증 수행</li><li>- 인자로 알고리즘 객체, 데이터, 성능 평가 방법, 폴드 데이터 세트 개수(cv) 입력</li><li>- 폴드별 성능 평가 수치와 전체 폴드의 평균 성능 평가 수치 출력 결과로 나옴</li></ul>
GridSearchCV	<ul style="list-style-type: none"><li>- 사이킷런과 유사하게 교차 검증을 통한 하이퍼파라미터 최적화 수행</li><li>- SVD 의 경우, 주로 SGD의 반복 횟수를 지정하는 n_epochs와 SVD의 잠재 요인 K의 크기를 지정하는 n_factors 튜닝</li></ul>

# #0 아이디어션 공지

#1. 다음주 화요일 오후 6시 30분 전까지 아이디어션 주제를 정해 저(최하경)에게 개인톡으로 알려주세요.

#2. 아이디어션 발표는 1인당 총 10분씩 (발표 8분 + 질문 2분) 진행

#3. 유런 템플릿을 이용해 발표 ppt를 준비해주세요.

- 들어가야 할 내용 : 주제, 선정 배경, 사용할 데이터 출처(프로젝트 시작 후 바로 사용 가능해야 함), 방향성

#4. 진행 순서

- 발표 순서는 세션 시작 10분 전에 단독방 사다리 타기를 통해 결정합니다.
- 모든 멤버들의 발표가 끝나면 멤버들이 발표한 주제들 중 하고 싶은 주제를 총 3개씩 투표합니다. (독방 투표 이용)
- 상위 2~3개의 주제를 선정하고, 다시 투표를 통해 선정된 주제들 중 하고 싶은 팀을 선택합니다.  
(이때 선정된 주제를 발표한 멤버는 해당 팀의 팀장이 되기 때문에 투표에 참여하지 않습니다.)
- 팀원 수는 운영진들의 참여 여부에 따라 확정될 예정입니다.
- 팀이 확정되면, 12월 21일부터 프로젝트는 시작됩니다.

#5. 정규세션은 종료되며, 화요일 오후 6시 30분까지 회의록 제출하는 것이 출석체크에 해당됩니다. (하지만 2주에 한번씩 화요일 오후 6시 30분에 진행상황 보고시간이 있을 예정입니다.)

# THANK YOU

