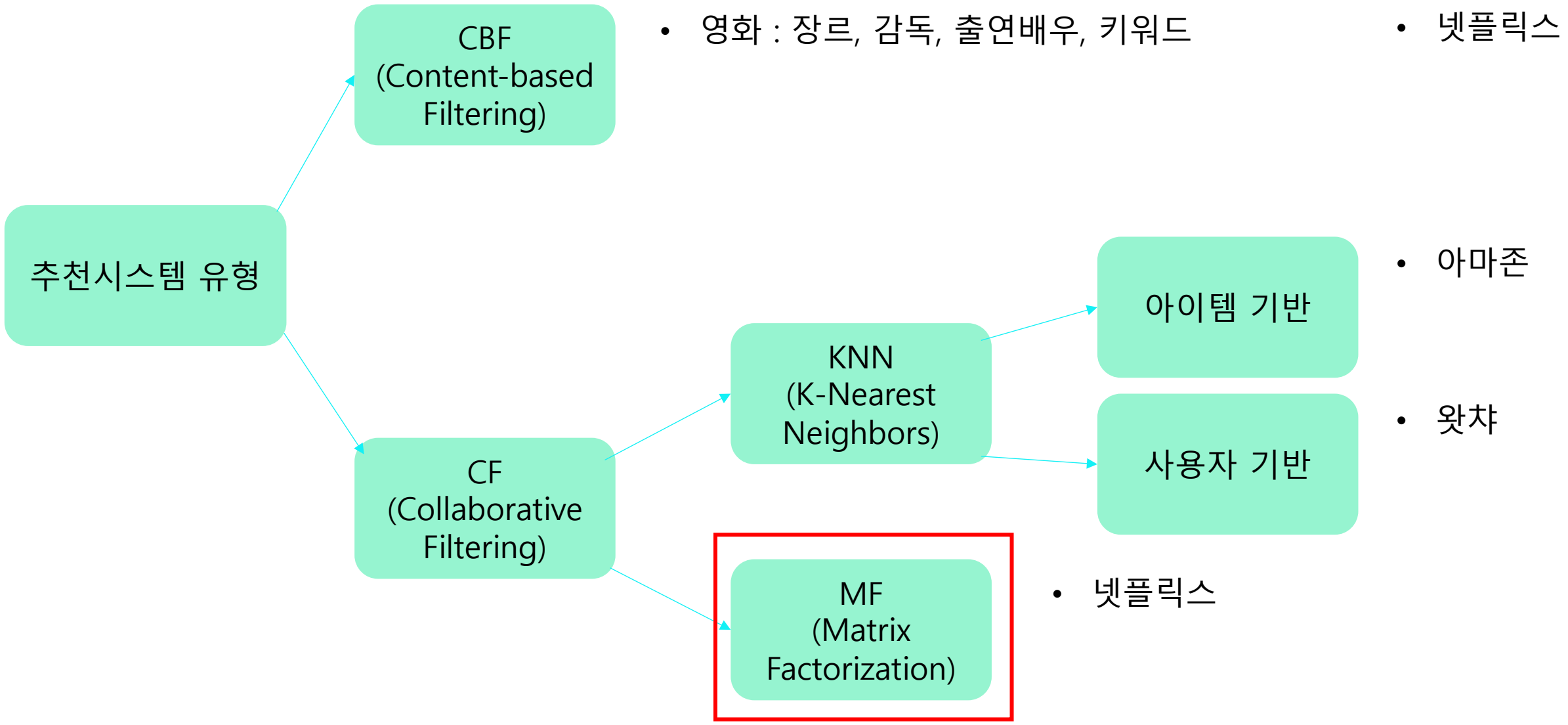
협업필터링(CF)
- Matrix Factorization

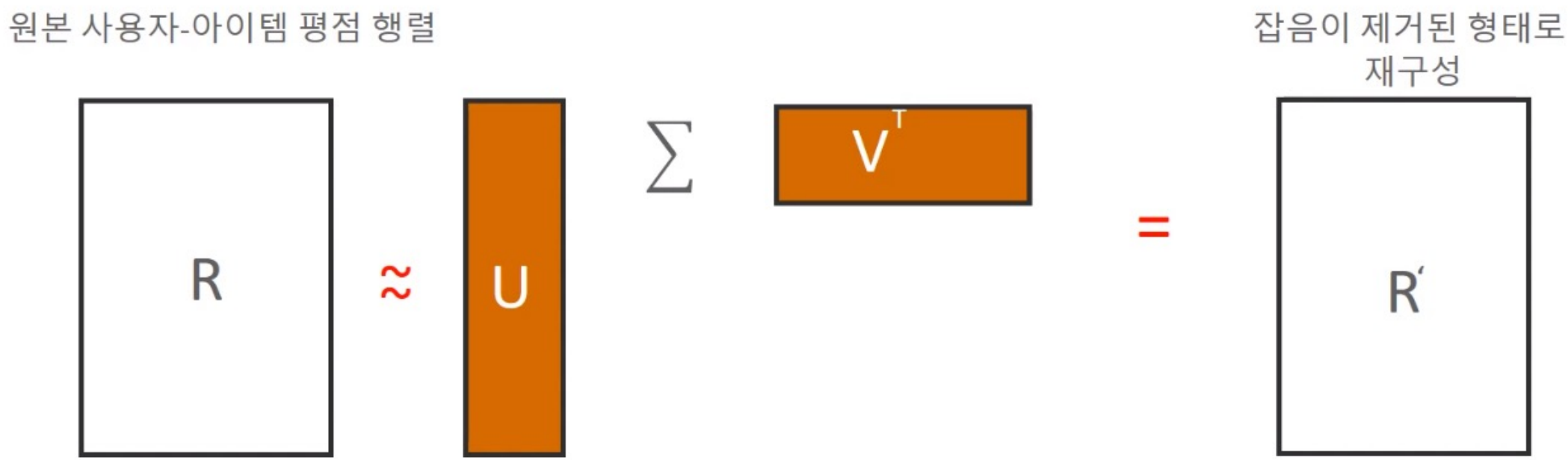
추천시스템 종류


추천시스템 유형
CBF
(Content-based
Filtering)
CF
(Collaborative
Filtering)
KNN
(K-Nearest
Neighbors)
MF
(Matrix
Factorization)
아이템 기반
사용자 기반
• 영화 : 장르, 감독, 출연배우, 키워드
• 넷플릭스
• 아마존
• 왓챠
• 넷플릭스

행렬 분해(MF) 기반의 잠재 요인 협업필터링

잠재 요인 협업 필터링의 개요

잠재 요인 협업 필터링은 사용자-아이템 평점 행렬 속에 숨어 있는 잠재 요인을 추출해 추천 예측을 할 수 있게 하는 기법입니다. 대규모 다차원 행렬을 SVD와 같은 행렬 분해(Matrix Factorization) 기법으로 분해하는 과정에서 잠재 요인을 추출하는데, 이 잠재 요인을 기반으로 사용자-아이템 평점 행렬을 재 구성하면서 추천을 구현합니다.



잠재 요인 협업 필터링의 이해

잠재 요인 협업 필터링은 사용자-아이템 평점 행렬 속에 숨어 있는 잠재 요인을 추출해 추천 예측을 할 수 있게 하는 기법입니다

사용자 레벨의 잠재요인

- Sci-fiction : 0.7
- Horror : 0.6
- Action : 0.5
- Drama : 0.8
- romance : 0.0

영화의 잠재 요인 추출

아이템 레벨의 잠재요인

- Sci-fiction : 0.8
- Horror : 0.6
- Action : 0.6
- Drama : 0.8
- romance : 0.1

	Item 1	Item 2	Item 3	Item M
User 1	3		3	
User 2	4	2		3
User 3		1	2	2

잠재 요인 협업 필터링 – 넷플릭스 추천 엔진 경연

백만 달러의 상금이 걸린 넷플릭스 추천 엔진 경연 대회 우승팀 사진

행렬 분해를 통한 잠재 요인 협업 필터링

잠재 요인 협업 필터링의 행렬 분해

$$R \simeq P * Q^T$$

- R: 사용자-아이템 평점 행렬
- $\simeq$: Decomposed by
- P: 사용자-잠재 요인 행렬
- Q^T: 잠재 요인-아이템 행렬

- 잠재 요인 협업 필터링의 행렬 분해 목표는 희소 행렬 형태의 사용자-아이템 평점 행렬을 밀집(Dense) 행렬 형태의 사용자-잠재 요인 행렬과 잠재 요인-아이템 행렬로 분해 한 뒤 이를 재 결합하여 밀집 행렬 형태의 사용자-아이템 평점 행렬을 생성하여 사용자에게 새로운 아이템을 추천하는 것입니다.

행렬 분해를 통한 잠재 요인 협업 필터링

원본 행렬

사용자 – 아이템 평점 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	

≈

행렬 분해

사용자 – 잠재 요인 행렬

	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

*

잠재 요인 - 아이템 행렬

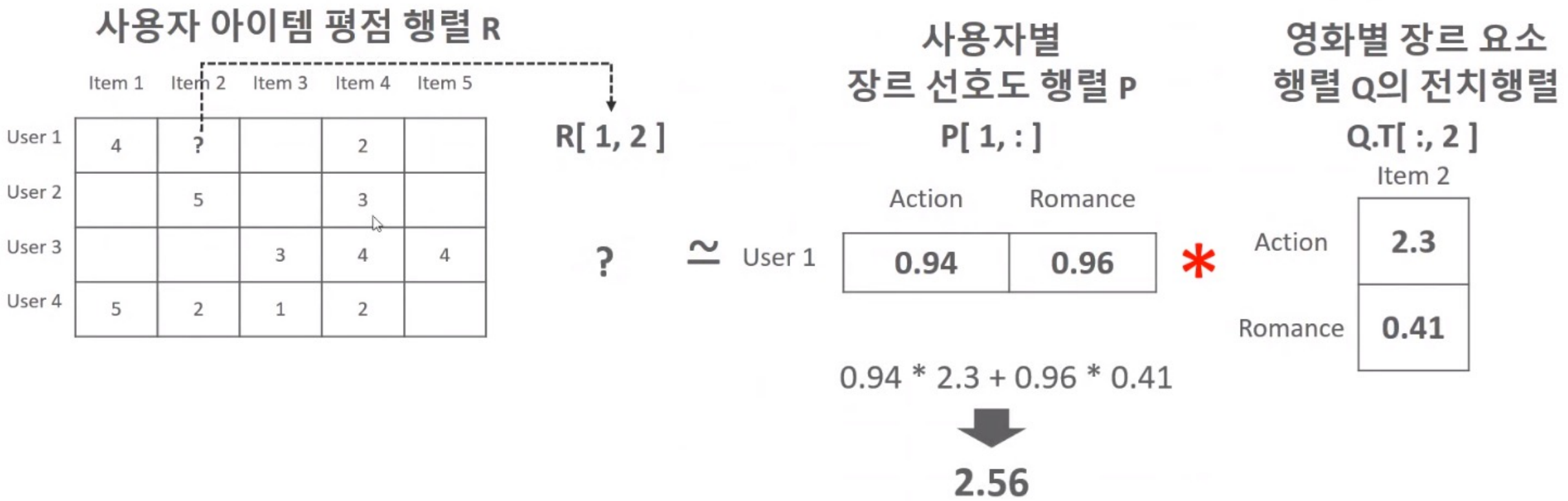
	Item 1	Item 2	Item 3	Item 4	Item 5
factor 1	1.7	2.3	1.41	1.36	0.41
factor 2	2.49	0.41	0.14	0.75	1.77

내적 곱 결과

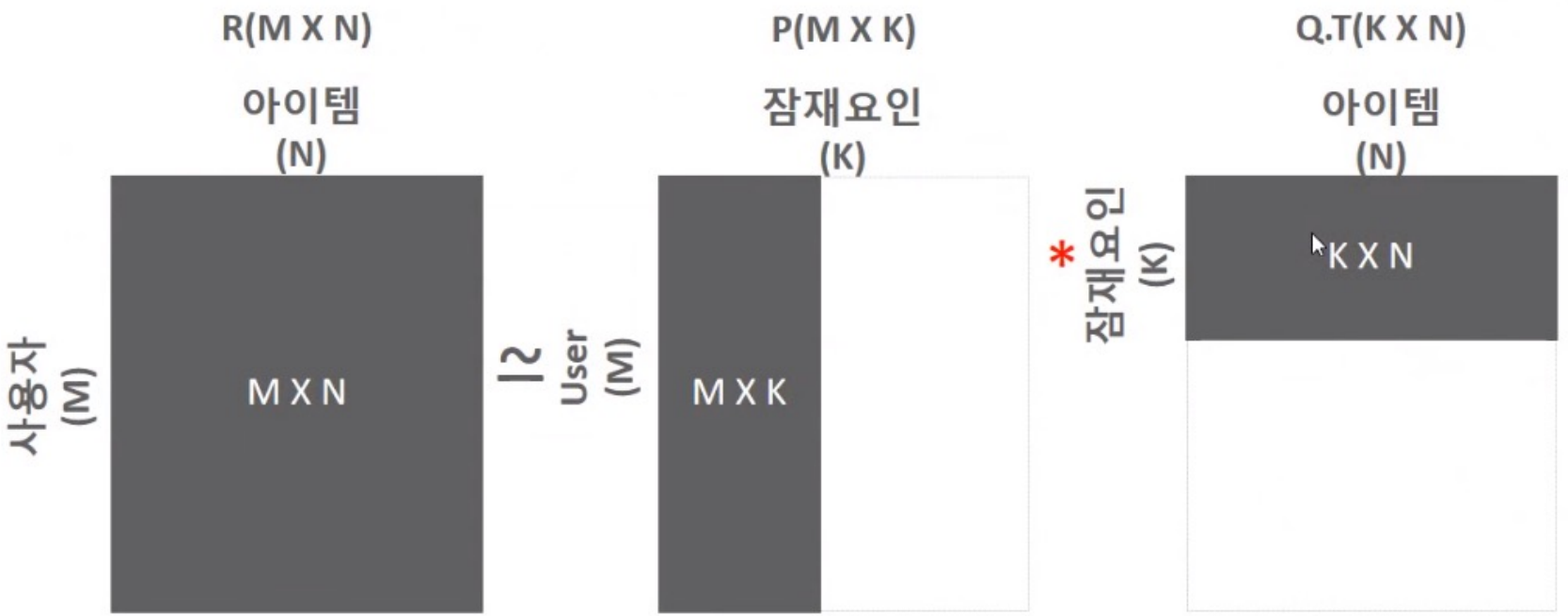
예측 평점

	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	4.96	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

행렬 분해를 통한 잠재 요인 협업 필터링

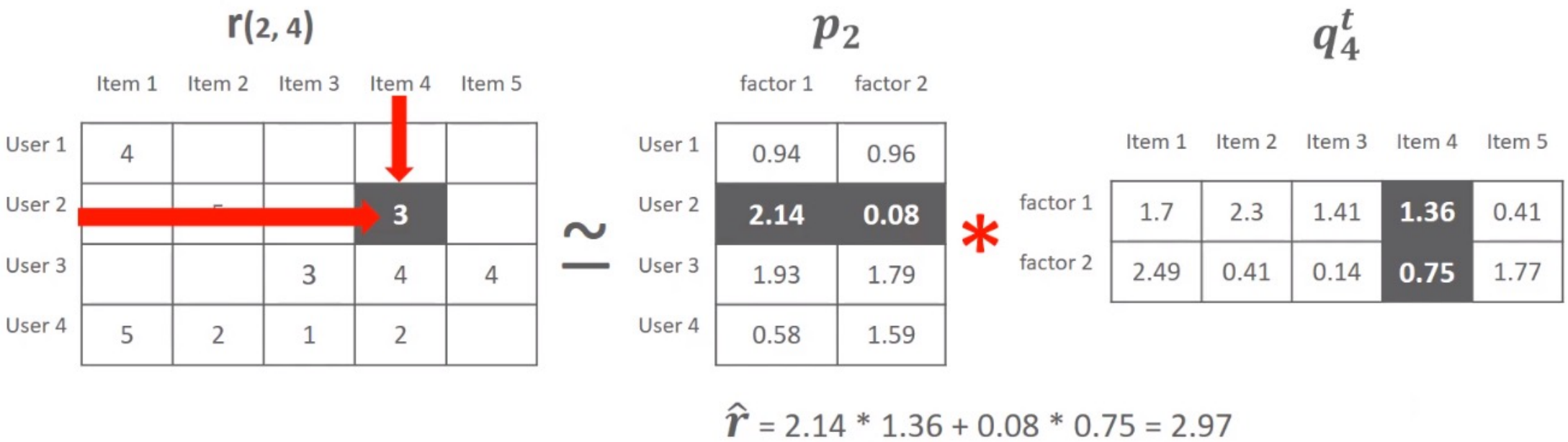


잠재 요인 기반의 행렬 분해 이해



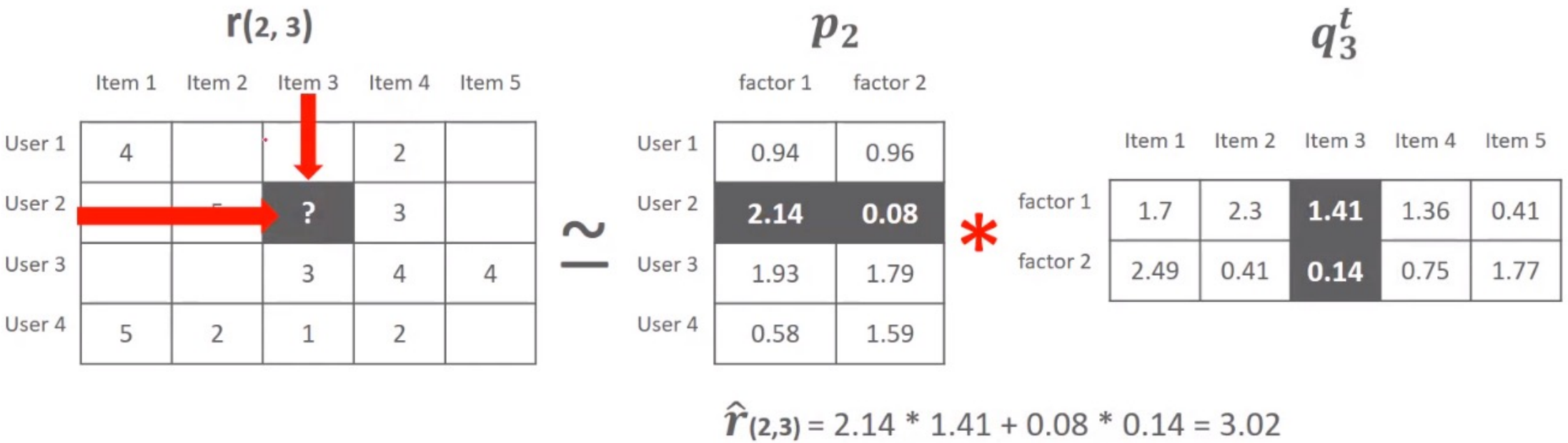
- M은 총 사용자 수
- N은 총 아이템 수
- K는 잠재 요인의 차원 수
- R은 M X N 차원의 사용자-아이템 평점 행렬
- P는 사용자와 잠재 요인과의 관계 값을 가지는 M X K 차원의 사용자-잠재 요인 행렬
- Q는 아이템과 잠재 요인과의 관계 값을 가지는 N X K 차원의 아이템-잠재 요인 행렬
- Q.T는 Q 매트릭스의 행과 열 값을 교환한 전치 행렬

행렬 분해를 통한 평점 예측



$\hat{r}$ = 2.14 * 1.36 + 0.08 * 0.75 = 2.97

행렬 분해를 통한 평점 예측



$\hat{r}_{(2,3)} = 2.14 * 1.41 + 0.08 * 0.14 = 3.02$

행렬 분해를 통한 평점 예측

P

	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

*

Q.T

	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

예측 평점

예측 사용자-아이템 평점 행렬

$\hat{R}$

	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	4.96	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-아이템 평점 행렬 분해 이슈

사용자-아이템 평점 행렬

	Item 1	Item 2	Item 3	.....	Item M
User 1	3	?	3	?	?
User 2	4	2			3
User 3		1	2		2
User 4	1				
.....		3	1		
User N	4	2			5

그러나 SVD 는 Missing Value 가 없는 행렬에 적용 가능 합니다. 따라서 P 와 Q 행렬을 일반적인 SVD 방식으로는 분해 할 수 없습니다.

P 와 Q 를 모르는데 어떻게 R 을 예측할 수가 있는가 ?

경사 하강법(Gradient Descent)를 이용하여 P 와 Q에 기반한 예측 R 값이 실제 R 값과 가장 최소의 오류를 가질 수 있도록 비용함수 최적화를 통해 P 와 Q를 최적화 유추.

경사 하강법 기반의 행렬 분해

경사 하강법을 이용한 행렬 분해 방법은 P와 Q 행렬로 계산된 예측 R 행렬 값이 실제 R 행렬 값과 가장 최소의 오류를 가질 수 있도록 반복적인 비용 함수 최적화를 통해 P와 Q를 유추해내는 것입니다

경사 하강법 기반의 행렬 분해 순서

1. P와 Q를 임의의 값을 가진 행렬로 설정합니다.
2. P와 Q.T 값을 곱해 예측 R 행렬을 계산하고 예측 R 행렬과 실제 R 행렬에 해당하는 오류 값을 계산합니다.
3. 이 오류 값을 최소화할 수 있도록 P와 Q 행렬을 적절한 값으로 각각 업데이트합니다.
4. 만족할 만한 오류 값을 가질 때까지 2, 3번 작업을 반복하면서 P와 Q 값을 업데이트해 근사화합니다

경사 하강법 기반의 행렬 분해 비용 함수

실제 값과 예측값의 오류 최소화와 L2 규제(Regularization)를 고려한 비용 함수식

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

실제값과 예측값의 오류 최소화

과적합 개선을 위한 L2 규제

경사 하강법 기반의 행렬 분해 업데이트 식

실제 R 행렬 값과 예측 R 행렬 값의 차이를 최소화하는 방향성을 가지고 P행렬과 Q행렬에 업데이트 값을 반복적으로 수행하면서 최적화된 예측 R 행렬을 구하는 방식이 경사 하강법 기반의 행렬 분해입니다

비용 함수를 최소화하기 위해서 새롭게 업데이트되는 $\acute{p_u}$와 $\acute{q_i}$

$$\acute{p}_u = p_u + \eta\left(e_{(u,i)} * q_i - \lambda * p_u\right)$$

$$\acute{q}_i = q_i + \eta\left(e_{(u,i)} * p_u - \lambda * q_i\right)$$

- p_u: P 행렬의 사용자 u행 벡터
- q_i^t: Q 행렬의 아이템 i행의 전치 벡터(transpose vector)
- $r_{(u,i)}$: R 행렬의 u행, i열에 위치한 값.
- $\hat{r}_{(u,i)}$: $p_u * q_i^t$로 계산하며, u행, i열에 위치한 행렬의 예측값
- $e_{(u,i)}$: $r_{(u,i)} - \hat{r}_{(u,i)}$의 값으로, u행, i열에 위치한 실제 행렬 값과 예측 행렬 값의 차이 오류
- α: SGD 학습률
- λ: L2 Regularization 계수

경사하강법을 이용한 행렬 분해 이해

In [1]:
```
import numpy as np

# 원본 행렬 R 생성,
# 분해 행렬 P와 Q 초기화, 잠재요인 차원 K는 3 설정.
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 ]])
R # 4X5 행렬
```

Out[1]:
```
array([[ 4., nan, nan,  2., nan],
       [nan,  5., nan,  3.,  1.],
       [nan, nan,  3.,  4.,  4.],
       [ 5.,  2.,  1.,  2., nan]])
```

행렬 R을 행렬 P, Q로 분해할 예정

In [2]:
```
num_users, num_items = R.shape
K=3  # 잠재 요인은 3개

# P와 Q 매트릭스의 크기를 지정하고 정규분포를 가진 random한 값으로 입력합니다.
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))
```

In [10]:

```
import numpy as np
from sklearn.metrics import mean_squared_error
from tqdm import tqdm_notebook

def get_rmse(R, P, Q, non_zeros):
    error = 0
    # 두개의 분해된 행렬 P와 Q.T의 내적 곱으로 예측 R 행렬 생성
    full_pred_matrix = np.dot(P, Q.T)

    # 실제 R 행렬에서 널이 아닌 값의 위치 인덱스 추출하여 실제 R 행렬과 예측 행렬의 RMSE 추출
    x_non_zero_ind = [non_zero[0] for non_zero in non_zeros]
    y_non_zero_ind = [non_zero[1] for non_zero in non_zeros]
    R_non_zeros = R[x_non_zero_ind, y_non_zero_ind]

    full_pred_matrix_non_zeros = full_pred_matrix[x_non_zero_ind, y_non_zero_ind]

    mse = mean_squared_error(R_non_zeros, full_pred_matrix_non_zeros)
    rmse = np.sqrt(mse)

    return rmse
```

In [11]:

```
def matrix_factorization(R, K, steps=200, learning_rate=0.01, r_lambda = 0.01):
    num_users, num_items = R.shape
    # P와 Q 매트릭스의 크기를 지정하고 정규분포를 가진 랜덤한 값으로 입력합니다.
    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

    # R > 0 인 행 위치, 열 위치, 값을 non_zeros 리스트 객체에 저장.
    non_zeros = [ (i, j, R[i,j]) for i in range(num_users) for j in range(num_items) if R[i,j] > 0 ]
```