CH9

추천시스템

- 하나의 콘텐츠를 선택했을 때 연관된 추천 콘텐츠가 얼마나 사용자의 관심을 끌고, 개인에게 맞춘 콘텐츠를 추천했는지 중요한 요소 → 사용자 해당 사이트 신뢰 + 접근 증가 → 더 많은 데이터 축적되어 더 정확한 추천 시스템이 구현
- 전자 상거래나 온라인 콘텐츠 업계 → 온라인 기반에서 가치 높음
- 지나친 정보 + 한정된 시간의 제약을 해결
    - 어떤 상품 구매?
    - 어떤 상품 위주로 둘러봄? 클릭?
    - 작성한 평점이나 리뷰?
    - 사용자가 작성한 취향?

콘텐츠 기반 필터링

- 사용자가 특정 아이템을 선호하는 경우, 그와 비슷한 콘텐츠를 가진 다른 아이템을 추천하는 방식
- 장르나 감독, 출연배우, 키워드 등이 비슷한 콘텐츠를 추천

협업 필터링

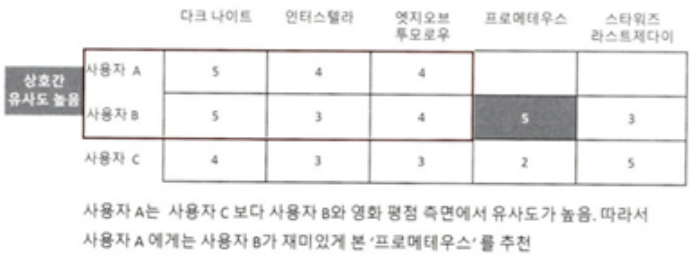
- 친구에게 물어보는 것과 유사하게, 사용자가 아이템을 매긴 평점 정보나 상품 구매 이력과 같은 **사용자 행동 양식**만을 기반으로 추천
- 사용자-아이템 평점 매트릭스와 같은 축적된 사용자 행동 데이터를 기반으로 아직 평가하지 않은 아이템을 예측 평가
- 행은 개별 사용자, 열은 개별 아이템, 각 행렬에 해당하는 값이 평점을 나타내야 함 → 희소 행렬

1) 최근접 이웃 협업 필터링

= 메모리 협업 필터링

- 아마존은 아직도 아이템 기반 최근접 이웃 협업 필터링 방식 이용

1. 사용자 기반 : 당신과 비슷한 고객들이 다음 상품도 구매
    - 특정 사용자와 유사한 사용자를 TOP-N으로 선정해, 좋아하는 아이템을 추천
    - 사용자 간 유사도 측정 후 가장 유사도가 높은 TOP-N 사용자 추출



2. 아이템 기반 : 이 상품을 선택한 고객들이 다음 상품도 구매
    - 아이템이 가지는 속성과는 상관없이! 사용자들이 아이템을 좋아하는지 평가 척도가 유사한 아이템을 추천하는 기준이 되는 알고리즘
    - 1과 행.열이 반대
    - **일반적으로 아이템 기반 협업 필터링이 정확도가 더 높음**
    - 비슷한 것을 좋아한다고 취향이 비슷하다고 판단하긴 어렵기 때문

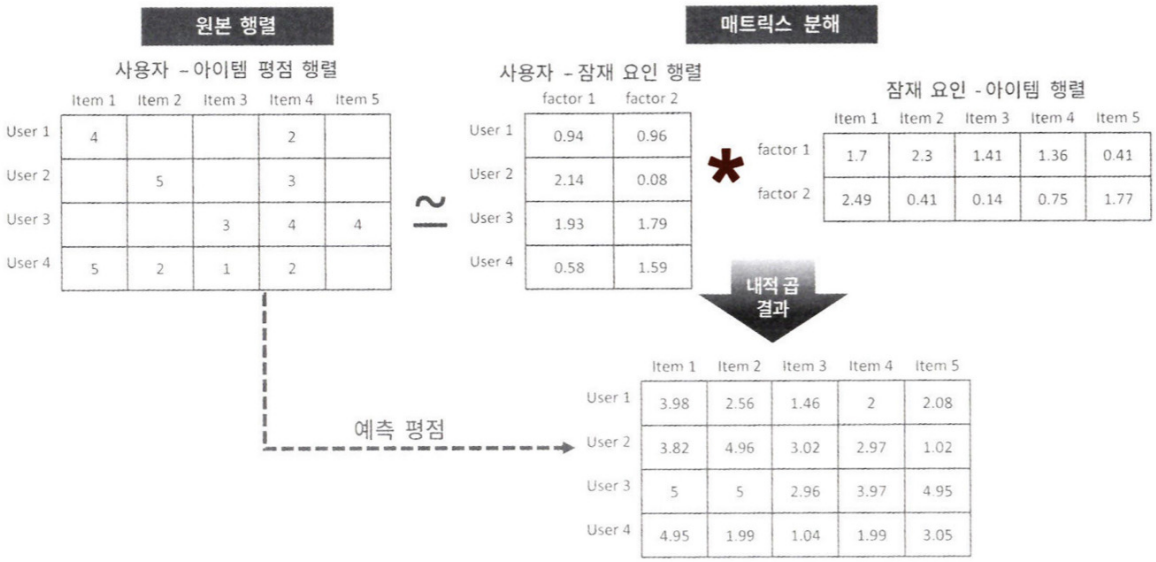
** 유사도 측정에는 주로 코사인 유사도 사용

2) 잠재 요인 협업 필터링

- 넷플릭스 추천 시스템 경연 대회에서 행렬 분해 기법을 이용한 잠재 요인 협업 필터링 방식이 우승하면서 대부분 이를 이용하고 있음
- 사용자-아이템 평점 매트릭스 속 숨어 있는 잠재 요인을 추출해 추천 예측

  ⇒ 행렬 분해 : 대규모 다차원 행렬을 SVD 등 차원 감소 기법을 통해 분해하는 과정에서

  잠재 요인 추출
- 다차원 희소 행렬인 사용자-아이템 행렬 데이터를 저차원 밀집 행렬의 사용자-잠재 요인 행렬 + 아이템-잠재 요인 행렬의 전치 행렬로 분해 → 내적하여 새로운 예측 사용자-아이템 평점 행렬 데이터를 만들어서 예측 평점 생성
- 잠재 요인은 정확히 알 수는 없지만 예를 들어 영화 평점 기반 행렬이라면 장르별 특성 선호도로 가정할 수 있음 → 두 행렬 : 사용자 장르 선호도와 영화 장르별 특성 값



〈 행렬 분해를 통한 잠재 요인 협업 필터링 〉

행렬 분해

- 다차원 매트릭스를 저차원으로 분해
- R → P*Q = (M,N) → (M,K) * (K,N)
- SVD : 주로 사용, NAN 없을 때에만 가능 → R 행렬에는 부적합
- SGD : 경사 하강법, 최소의 LOSS를 가지도록 반복적인 비용함수 최적화를 통해 유추

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.

실제 값과 예측값의 오류 최소화와 L2 규제(Regularization)를 고려한 비용 함수식은 다음과 같습니다.

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

```
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 ]])
num_users, num_items = R.shape
K=3
```

```
# 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))
from sklearn.metrics import mean_squared_error

def get_rmse(R, P, Q, non_zeros):
    error = 0
    # 두개의 분해된 행렬 P와 Q.T의 내적으로 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
# R > 0 인 행 위치, 열 위치, 값을 non_zeros 리스트에 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 ]

steps=1000
learning_rate=0.01
r_lambda=0.01

# SGD 기법으로 P와 Q 매트릭스를 계속 업데이트.
for step in range(steps):
    for i, j, r in non_zeros:
        # 실제 값과 예측 값의 차이인 오류 값 구함
        eij = r - np.dot(P[i, :], Q[j, :].T)
        # Regularization을 반영한 SGD 업데이트 공식 적용
        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)

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

콘텐츠 기반 필터링 실습 - TMDB 5000 영화 데이터셋

아이템 기반 최근접 이웃 협업 필터링 실습 - MovieLens 데이터셋

행렬 분해 이용한 잠재 요인 협업 필터링 실습

파이썬 추천 시스템 패키지 - Surprise

- 사용자 아이디, 아이템 아이디, 평점 칼럼 순서 지켜야 함
- 베이스라인
  : 개인의 성향을 반영해 아이템 평가에 **편향성 요소**를 반영하여 평점을 부과하는 방식

  = 전체 평균 평점 + 사용자 편향 점수 + 아이템 편향 점수