문제 1.User based CF와 Item based CF에 대해 간단히 서술해주세요. 둘 중 어느 추천이 더 좋을까요?

User based CF는 '서로 비슷한 유저' 를 찾아 추천해주는 방식이며 Item based CF는 '서로 비슷한 아이템' 을 찾아 추천해주는 방식입니다. 일반적으로 한 유저가 보거나 구입한 아이템이 매우 적기 때문에 (Sparsity) 정확하게 유저를 표현하는 벡터를 찾기 어려울 수 있습니다.
User based CF는 유저들의 유사도를 계산할 때에 유저들이 공통적으로 평가한 아이템들만 사용해 유사도를 계산하므로 많은 경우 이는 결과에 나쁜 영향을 줄 수 있습니다. (이를 해결 하기 위해 adjust cosine similarity가 도입되었습니다. 자세한 건 1일차 학습자료3 GroupLens 블로그 글을 참고해주세요)

- 상대적으로 덜 유명한 아이템들을 추천에 고려할 수 없다 유명한 아이템들에 bias가 걸리는 현상이 발생(Popularity bias)

또한 많은 경우 아이템이 늘어나는 것보다 **유저가 늘어나는 것이 더 빠르기 때문에 user based CF보다 item based CF가 많은 경우 추천 결과가 더 좋습니다**.(한 유저가 보거나 구입한 아이템이 유저에 비해 상대적으로 매우 적기에 정확히 유저를 표현하는 벡터를 찾기 어렵다.)

- Collaborative filtering은 특정한 방법론 하나를 의미하는 것이 아니라 방법론들을 총칭하는 말 ex) methodology - neighborhood method, matrix factorization

- Methogology를 푸는 방법을 알고리즘이라 부른다. ex)SGD, ALS

User based CF

$$\bar{r}_u + \frac{\sum_{v \in \text{Users}} \text{sim}(u, v)(r_{vi} - \bar{r}_v)}{\sum_{v \in \text{Users}} \text{sim}(u, v)}$$

user u와 v가 같이본 item i

Item based CF

$$\frac{\sum_{j \in \text{rated items by u}} \text{sim}(i, j) r_{ui}}{\sum_{j \in \text{rated items by u}} \text{sim}(i, j)}$$

item i와 j를 같이 본 user u

문제 2. Neighborhood method를 사용한다고 하였을 때에 user based CF와 item based CF 각각의 방법에서 user u가 item i를 얼마나 좋아할지 예측하는 방법을 서술해주세요.

User based CF

$$\bar{r}_u + \frac{\sum_{v \in \text{Users}} \text{sim}(u, v)(r_{vi} - \bar{r}_v)}{\sum_{v \in \text{Users}} \text{sim}(u, v)}$$

user u와 v가 같이본 item i

맨 왼쪽 항은 유저 u의 평균 rating 이며, 오른 쪽 항의 분모는 item i를 관측한 모든 유저들과의 similarity의 합을 의미하며, 분자는 item i를 관측한 모든 유저들의 rating과 (정확히는 rating - rating의 평균) similarity를 곱한 후 합한 값을 의미합니다.

Item based CF

$$\frac{\sum_{j \in \text{rated items by u}} \text{sim}(i, j) r_{ui}}{\sum_{j \in \text{rated items by u}} \text{sim}(i, j)}$$

item i와 j를 같이 본 user u

u가 평가한 모든 아이템들과 item i와의 similarity와 실제 u가 i를 평가한 값을 곱한 후, similarity 들의 합으로 나누어줍니다.

실제 추천을 할 때에는 이 방법을 사용하여 가장 예상 점수가 높은 순으로 아이템을 추천합니다. 여기에 보통 추가적인 로직들이 들어가게 됩니다 (이미 본 아이템을 뺀다거나, 특정한 아이템 분류는 제외하고 추천한다거나 등)

문제 3.3일차 학습자료 1과 보충자료 1에 의하면 MF 문제는 ₩sum_ui ₩| r_ui - p_i q_u ₩|^2 + λ ₩| p_i ₩|^2 + λ ₩| q_u ₩|^2 를 가장 최소화하는 p와 q를 찾는 문제이며, 이 문제를 푸는 알고리즘은 gradient descent와 ALS가 있습니다. (자세한 수식은 보충자료의 eq 2를 참고해주세요) 각각이 p와 q를 어떻게 찾는지 간단하게 설명해주시고 둘의 장단점을 비교해서 설명해주세요. 어떤 알고리즘이 더 좋을까요?

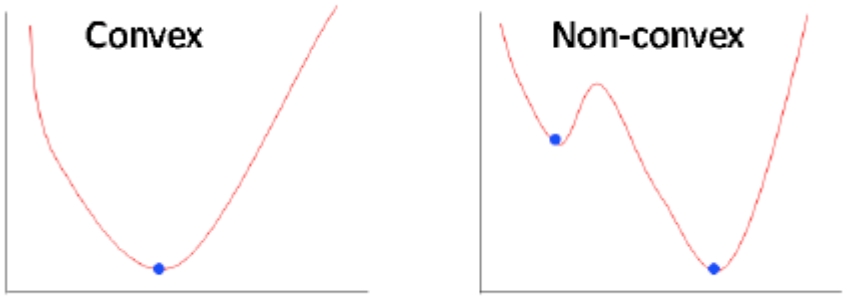
gradient descent는 p와 q의 gradient를 계산하여 gradient 방향으로 step size만큼 움직이는 방법론입니다.

- SGD 분산처리가 불가능 – 한 item, user pair가 업데이트 되고 나면 그 item, user와 연관되었던 모든 pair들의 결과에 영향을 준다. Optimization 결과가 달라질 수 있다.

ALS는 alternating least squares의 약자로 p와 q를 각각 고정하였을 때에 문제 꼴이 간단한 least square라는 선형문제로 나타내어지고, 이를 번갈아가면서 계속 풀어서 p와 q를 구하는 방식입니다.

- ALS는 분산처리가 가능하다 - ALS는 p나 q를 고정시키고나면 optimize해야하는 parameter들 (q나 p는) 서로서로 independent해지기 때문에 그냥 따로따로 계산하고 합쳐도 동일한 답을 얻을 수 있다

Gradient descent보다 ALS을 좀더 많이 사용한다. ALS에서 p나 q 한 쪽을 고정하면 그냥 least square 라는 엄청 간단한 closed form으로 정답이 나오는 (근의 공식처럼 그냥 수식 대입하면 답 나오는걸 closed form이라고 합니다) 문제로 바뀐다.

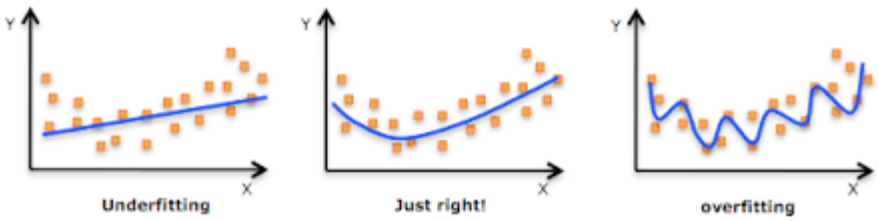


p나 q를 고정하면 아까 least square 문제가 된다. least square가 convex 문제라서 그렇게 표현합니다. 그 문제는 정확히 답을 효율적으로 구할 수 있다는 의미

문제 4.MF에서 사용하는 파라미터로는 latent feature들의 dimension (즉, p와 q의 dimension), regularization term λ가 있습니다. 이 중 latent feature의 dimension은 어떤 의미를 가지고 있을까요? 또한 각각의 값이 클 때와 작을 때 어떤 현상이 발생할까요?

기본적으로 MF는 rating matrix가 low rank라고 가정하며 그 rank가 정확하게 latent feature dimension, 혹은 factor size가 됩니다. 따라서 latent feature dimension은 모델의 complexity를 표현한다고 할 수 있는데요 이 값이 작을수록 rating matrix가 low rank라고 가정하게 되며 이 값이 클 수록 rating matrix가 high rank라고 가정하게 됩니다.

따라서 이 값이 너무 크면 overfitting이 발생할 수 있으므로 너무 큰 값은 피해야 합니다. 반면 이 값이 너무 작아지면 matrix의 rank가 너무 작아져 데이터를 제대로 표현할 수 없게 됩니다.(under fitting) 보통은 100보다 작은 값이면 충분하며, 20이나 40정도의 값으로도 충분합니다. (실제로는 적당한 validation을 통해 찾아야한다.)



문제 5. Day 4 학습자료1, 학습자료2의 part 4 및 보충자료에서 나오는 "ALS for Implicit Feedback" 알고리즘에서 preference p와 confidence c의 역할은 무엇인가요? 자료에 따르면 confidence를 1+ αr 로 정의하는데, 이때 α 는 무슨 의미를 가지게 될까요? (Hint, 원 논문에서 r_ui는 user u가 item i를 관측하거나 구매한 'implicit feedback'의 횟수를 의미합니다. 이전처럼 explicit rating score가 아닌 implict feedback을 가정하고 답안을 작성해주세요)

Implicit feedback은 주로 pageview 등의 '간접적인' 지표를 의미합니다. 이 지표는 굉장히 노이즈가 많아서 한 두 번 정도 클릭하였다고 하여 유저가 그 아이템을 꼭 좋아한다고 말하기는 어려운데요, 그러나 한 두 번 본 것 보다는 수십 번 본 것이 더 관심이 많은 아이템이라고는 말할 수 있습니다.
이게 implicit ALS의 기본 가정인데요, pageview를 직접 맞추는 것이 매우 어렵기 때문에 (너무 노이지하기도 하구요) 해당 논문에서는 한 번이라도 본 경우에는 preference p를 1로 만들어줍니다. 이제 objective function에서는 latent vector p와 q가 (논문 용어로는 x, y) 0 또는 1을 맞추는 값이 됩니다.
preference는 이처럼 noisy한 implicit feedback 데이터를 좀 더 맞추기 용이하게 만들어줍니다. 또한 confidence c는 모든 preference에 같은 정도의 중요도를 주는 대신에 feedback이 큰 user-item 페어를 더 강하게 맞추도록 강제해주는 항입니다. α는 그 중요도를 조정하는 항으로, α가 클수록 많이 본 아이템에 힘을 더 실어주고 (작은 아이템을 무시하고), α가 작을수록 그 반대의 역할을 하도록 합니다.