추천시스템 실습하기

"무비렌즈 데이터를 이용한 실습"

카이스트

김현우

Contents

1. 협업필터링이란?

- SGD
- ALS

2. 분석 실습

무비렌즈 데이터를 이용한 실습
- Stochastic Gradient Descent
- Alternating Least Squares
- 자료 : https://github.com/choco9966/Daejeon-Learning-Day

01 협업 필터링이란?

1 협업필터링 개요

정의

협업필터링은 사용자의 구매 패턴이나 평점을 가지고 다른 사람들의 구매 패턴, 평점을 통해서 추천을 하는 방법입니다. 추가적인 사용자의 개인정보나 아이템의 정보가 없이도 추천할 수 있는게 큰 장점이며 2006부터 2009년동안 열린 Netflix Prize Competition에서 우승한 알고리즘으로 유명세를 떨쳤습니다.

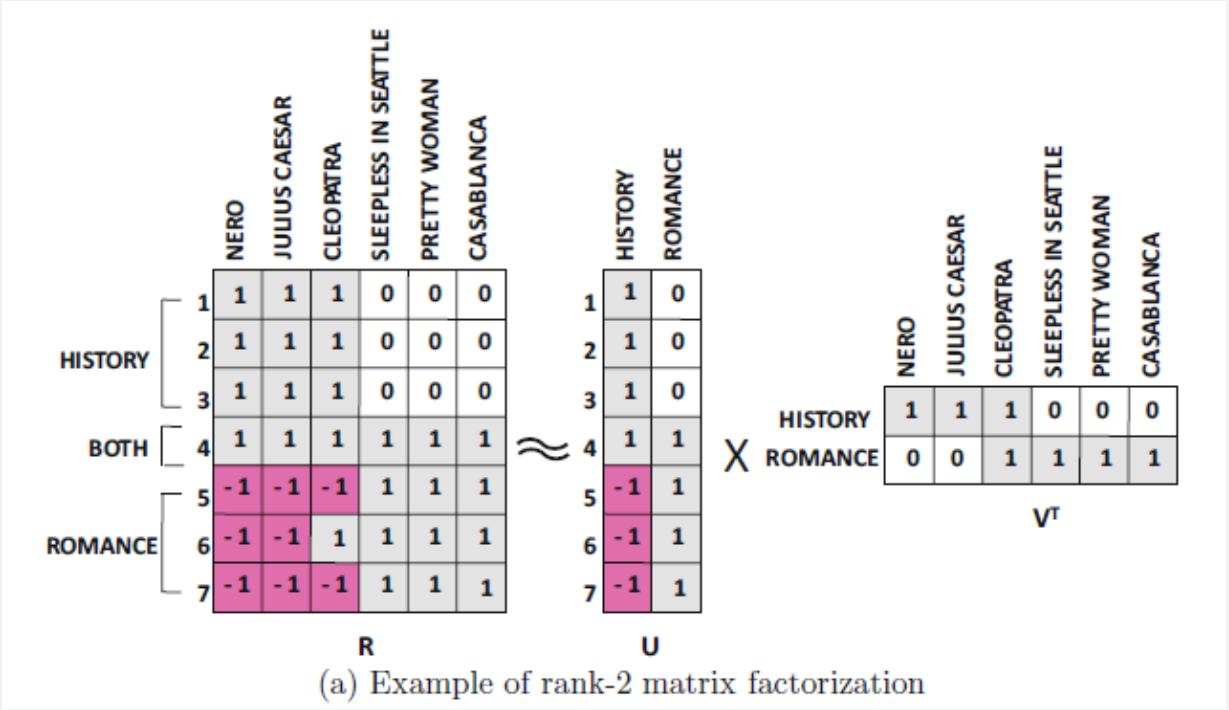
종류

1. 최근접 이웃기반
2. 잠재 요인기반

1 Latent Factor Collaborative Filtering

정의

잠재 요인 협업 필터링은 Rating Matrix에서 빈 공간을 채우기 위해서 사용자와 상품을 잘 표현하는 차원(Latent Factor)을 찾는 방법입니다. 행렬 분해 알고리즘은 사용자-아이템 상호 작용 행렬을 두 개의 저 차원 직사각형 행렬의 곱으로 분해하여 작동합니다. 이 방법은 Netflix 챌린지에서 널리 알려지게되었습니다



(a) Example of rank-2 matrix factorization

1 SGD

정의

고유값 분해(eigen value Decomposition)와 같은 행렬을 대각화 하는 방법

$$\text{Minimize } J = \frac{1}{2}||R - UV^T||^2$$
$$\text{subject to:}$$
$$\text{No constraints on } U \text{ and } V$$
$$S = \{(i,j) : r_{ij} \text{ is observed}\}$$

$$\text{Minimize } J = \frac{1}{2}\sum_{(i,j)\in S} e_{ij}^2 = \frac{1}{2}\sum_{(i,j)\in S}\left(r_{ij} - \sum_{s=1}^{k} u_{is}\cdot v_{js}\right)^2$$
$$\text{subject to:}$$
$$\text{No constraints on } U \text{ and } V$$

Gradient Descent에 의해 편미분 된 값

$$\frac{\partial J}{\partial u_{iq}} = \sum_{j:(i,j)\in S}\left(r_{ij} - \sum_{s=1}^{k} u_{is}\cdot v_{js}\right)(-v_{jq}) \quad \forall i \in \{1\dots m\}, q \in \{1\dots k\}$$
$$= \sum_{j:(i,j)\in S}(e_{ij})(-v_{jq}) \quad \forall i \in \{1\dots m\}, q \in \{1\dots k\}$$
$$\frac{\partial J}{\partial v_{jq}} = \sum_{i:(i,j)\in S}\left(r_{ij} - \sum_{s=1}^{k} u_{is}\cdot v_{js}\right)(-u_{iq}) \quad \forall j \in \{1\dots n\}, q \in \{1\dots k\}$$
$$= \sum_{i:(i,j)\in S}(e_{ij})(-u_{iq}) \quad \forall j \in \{1\dots n\}, q \in \{1\dots k\}$$

1 SGD

Regularization

고유값 분해(eigen value Decomposition)와 같은 행렬을 대각화 하는 방법

$$\text{Minimize } J = \frac{1}{2}\sum_{(i,j)\in S} e_{ij}^2 + \frac{\lambda}{2}\sum_{i=1}^{m}\sum_{s=1}^{k} u_{is}^2 + \frac{\lambda}{2}\sum_{j=1}^{n}\sum_{s=1}^{k} v_{js}^2$$

$$= \frac{1}{2}\sum_{(i,j)\in S}\left(r_{ij} - \sum_{s=1}^{k} u_{is}\cdot v_{js}\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{m}\sum_{s=1}^{k} u_{is}^2 + \frac{\lambda}{2}\sum_{j=1}^{n}\sum_{s=1}^{k} v_{js}^2$$

Gradient Descent에 의해 편미분 된 값

$$\frac{\partial J}{\partial u_{iq}} = \sum_{j:(i,j)\in S}\left(r_{ij} - \sum_{s=1}^{k} u_{is}\cdot v_{js}\right)(-v_{jq}) + \lambda u_{iq} \quad \forall i \in \{1\ldots m\}, q \in \{1\ldots k\}$$

$$= \sum_{j:(i,j)\in S}(e_{ij})(-v_{jq}) + \lambda u_{iq} \quad \forall i \in \{1\ldots m\}, q \in \{1\ldots k\}$$

$$\frac{\partial J}{\partial v_{jq}} = \sum_{i:(i,j)\in S}\left(r_{ij} - \sum_{s=1}^{k} u_{is}\cdot v_{js}\right)(-u_{iq}) + \lambda v_{jq} \quad \forall j \in \{1\ldots n\}, q \in \{1\ldots k\}$$

$$= \sum_{i:(i,j)\in S}(e_{ij})(-u_{iq}) + \lambda v_{jq} \quad \forall j \in \{1\ldots n\}, q \in \{1\ldots k\}$$

$$u_{iq} \Leftarrow u_{iq} + \alpha\left(\sum_{j:(i,j)\in S} e_{ij}\cdot v_{jq} - \lambda\cdot u_{iq}\right) \quad \forall q \in \{1\ldots k\}$$

$$v_{jq} \Leftarrow v_{jq} + \alpha\left(\sum_{i:(i,j)\in S} e_{ij}\cdot u_{iq} - \lambda\cdot v_{jq}\right) \quad \forall q \in \{1\ldots k\}$$

1 SGD

예시

Explict Feedback 된 형태의 4명의 유저에 대한 3개의 아이템에 대한 평점 Matrix

Rating Matrix

?	3	2
5	1	2
4	2	1
2	?	4

=

User Latent (U)

np.dot

Item Latent (V)의 Transpose

1 SGD

예시

1. User Latent 와 Item Latent의 차원(아래는 2)을 설정하고 임의로 초기화

User Latent (U)

0.5756	1.4534
-0.199	-1.218
2.7297	0.48
-0.039	-2.506

np.dot

Item Latent (V)의 Transpose

0.3668	-1.1078	1.4593
-0.3392	0.8972	0.4528

=

-0.2819	0.6663	1.4981
0.3403	-0.8728	-0.8421
0.8384	-2.5933	4.2008
0.8354	-2.2043	-1.1928

1 SGD

예시

2. Gradient Descent 진행

?	3	2
5	1	2
4	2	1
2	?	4

-0.2819	0.6663	1.4981
0.3403	-0.8728	-0.8421
0.8384	-2.5933	4.2008
0.8354	-2.2043	-1.1928

0.5756	1.4534
-0.199	-1.218
2.7297	0.48
-0.039	-2.506

0.3668	-1.1078	1.4593
-0.3392	0.8972	0.4528

Error : 3 – 0.6663 = 2.3337
dUser = -2.3337*[-1.1078 , 0.8972] + 0.01 *[0.5756, 1.4534]
dItem = -2.3337*[0.5756, 1.4534] + 0.01 *[-1.1078 , 0.8972]

$$\frac{\partial J}{\partial u_{iq}} = \sum_{j:(i,j)\in S} \left(r_{ij} - \sum_{s=1}^{k} u_{is} \cdot v_{js} \right) (-v_{jq}) + \lambda u_{iq} \quad \forall i \in \{1 \dots m\}, q \in \{1 \dots k\}$$

$$= \sum_{j:(i,j)\in S} (e_{ij})(-v_{jq}) + \lambda u_{iq} \quad \forall i \in \{1 \dots m\}, q \in \{1 \dots k\}$$

$$\frac{\partial J}{\partial v_{jq}} = \sum_{i:(i,j)\in S} \left(r_{ij} - \sum_{s=1}^{k} u_{is} \cdot v_{js} \right) (-u_{iq}) + \lambda v_{jq} \quad \forall j \in \{1 \dots n\}, q \in \{1 \dots k\}$$

$$= \sum_{i:(i,j)\in S} (e_{ij})(-u_{iq}) + \lambda v_{jq} \quad \forall j \in \{1 \dots n\}, q \in \{1 \dots k\}$$

1 SGD

예시

0.5756	1.4534
-0.199	-1.218
2.7297	0.48
-0.039	-2.506

0.3668	-1.1078	1.4593
-0.3392	0.8972	0.4528

2. Gradient Descent 진행

Error : 3 – 0.6663 = 2.3337
dUser = -2.3337*[-1.1078 , 0.8972] + 0.01 *[0.5756, 1.4534]
dItem = -2.3337*[0.5756, 1.4534] + 0.01 *[-1.1078 , 0.8972]

Updated User Latent = [0.5756, 1.4534] - Learning rate * dUser
[0.446, 1.5574] = [0.5756, 1.4534] – 0.05 * [2.591, -2.0793]

Updated Item Latent = [-1.1078, 0.8972] - Learning rate * dItem
[-1.0401, 1.0663] = [-1.1078 , 0.8972] – 0.05 * [-1.3544, -3.3828]

$$u_{iq} \Leftarrow u_{iq} + \alpha \left(\sum_{j:(i,j)\in S} e_{ij} \cdot v_{jq} - \lambda \cdot u_{iq} \right) \quad \forall q \in \{1 \dots k\}$$

$$v_{jq} \Leftarrow v_{jq} + \alpha \left(\sum_{i:(i,j)\in S} e_{ij} \cdot u_{iq} - \lambda \cdot v_{jq} \right) \quad \forall q \in \{1 \dots k\}$$

0.446	1.5574
-0.199	-1.218
2.7297	0.48
-0.039	-2.506

0.3668	-1.0401	1.4593
-0.3392	1.0663	0.4528

-0.3647	1.1967	1.3560
0.3403	-0.8728	-0.8421
0.8384	-2.5933	4.2008
0.8354	-2.2043	-1.1928

1 SGD

예시

3. 모든 평점에 대해서 반복 (epoch : 1)

- ?를 제외한 모든 평점에 대해서 진행

?	3	2
5	1	2
4	2	1
2	?	4

-0.3522	1.1628	1.5157
0.3403	-1.0924	-0.9057
0.8384	-2.3273	4.2620
0.8354	-2.6308	-1.3184

0.4928	1.5711
-0.199	-1.218
2.7297	0.48
-0.039	-2.506

0.3668	-1.0401	1.4729
-0.3392	1.0663	0.5027

1 SGD

예시

3. 모든 평점에 대해서 반복 (epoch : 1)

- ?를 제외한 모든 평점에 대해서 진행

?	3	2
5	1	2
4	2	1
2	?	4

-0.8209	1.1628	1.5157
0.7715	-1.2650	-0.8190
0.5752	-2.3273	4.2619
1.5482	-2.6308	-1.3184

0.4928	1.5711
-0.113	-1.296
2.7297	0.48
-0.039	-2.506

0.3203	-1.0401	1.4729
-0.6230	1.0663	0.5027

1 SGD

예시

3. 모든 평점에 대해서 반복 (epoch : 1)

- ?를 제외한 모든 평점에 대해서 진행

?	3	2
5	1	2
4	2	1
2	?	4

-0.5947	1.3736	-0.0028
0.6763	-1.0994	0.3561
1.4991	-0.4721	2.3222
1.7260	-2.5722	1.0869

0.4927	1.5711
-0.015	-1.101
2.3345	0.5363
0.2363	-2.464

0.7858	-0.4137	1.0724
-0.6250	1.0040	-0.3381

1 SGD

예시

4. 2~3의 과정을 10번 반복 (epoch : 10)

?	3	2
5	1	2
4	2	1
2	?	4

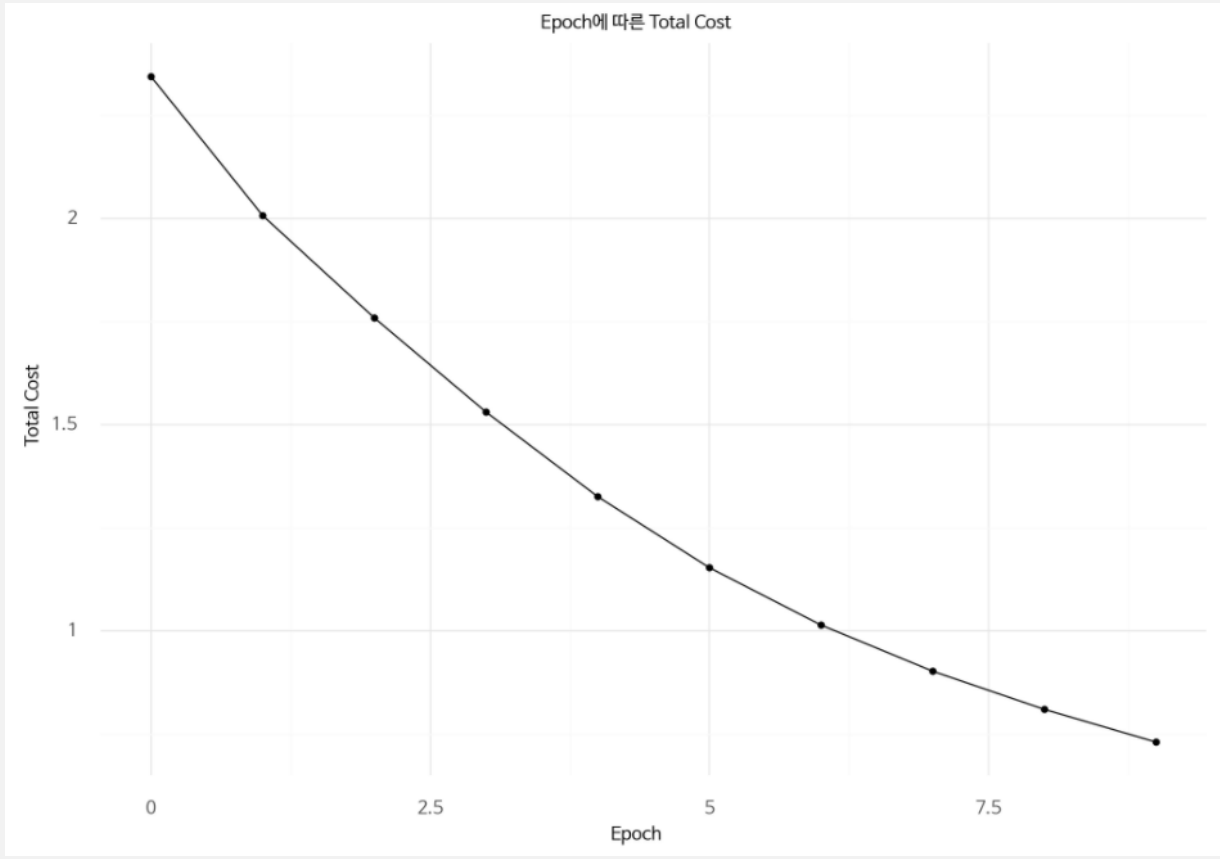
평점이 높은 1번 상품은 1번 유저에게 추천 o

2.7458	2.4147	0.3873
4.2476	0.5806	2.8256
3.9181	2.3825	1.3030
2.4323	-1.9994	3.2637

평점이 낮은 2번 상품은 4번 유저에게 추천 x

1.6462	1.0993
1.6740	-1.072
2.0550	0.6342
0.3135	-2.663

2.1118	0.8951	0.9768
-0.6646	0.8560	-1.1103



2 ALS

정의

기존의 SGD가 두개의 행렬(User Latent, Item Latent)을 동시에 최적화하는 방법이라면, ALS는 두 행렬 중 하나를 고정시키고 다른 하나의 행렬을 순차적으로 반복하면서 최적화 하는 방법입니다. 이렇게 하면, 기존의 최적화 문제가 convex 형태로 바뀌기에 수렴된 행렬을 찾을 수 있는 장점이 있습니다.

알고리즘

1. 초기 아이템, 사용자 행렬을 초기화
2. 아이템 행렬을 고정하고 사용자 행렬을 최적화
3. 사용자 행렬을 고정하고 아이템 행렬을 최적화
4. 위의 2, 3 과정을 반복

2 ALS

수렴

기존의 SGD가 두개의 행렬(User Latent, Item Latent)을 동시에 최적화하는 방법이라면, ALS는 두 행렬 중 하나를 고정시키고 다른 하나의 행렬을 순차적으로 반복하면서 최적화 하는 방법입니다. 이렇게 하면, 기존의 최적화 문제가 convex 형태로 바뀌기에 수렴된 행렬을 찾을 수 있는 장점이 있습니다.

$$||y - X\beta||_2$$

$$\beta = (X^T X)^{-1} X^T y$$

$$\forall u_i : J(u_i) = ||R_{i.} - u_i \times V^T||_2 + \lambda \cdot ||u_i||_2$$

$$\forall v_j : J(v_j) = ||R_{i.} - U \times v_j{}^T||_2 + \lambda \cdot ||v_j||_2$$

$$u_i = (V^T \times V + \lambda I)^{-1} \times V^T \times R_{i.}$$

$$v_j = (U^T \times U + \lambda I)^{-1} \times U^T \times R_{.j}$$

0	3	2
5	1	2
4	2	1
2	0	4

?의 경우는 모두 0으로 바꿔줍니다.

출처 : Codelog (통계쟁이 엔지니어), Matrix Factorization에 대해 이해, Alternating Least Square (ALS) 이해

2 ALS

알고리즘

1. 초기 아이템, 사용자 행렬을 초기화

User Latent

0.5756	1.4534
-0.199	-1.218
2.7297	0.48
-0.039	-2.506

Item Latent 의 Transpose

0.3668	-1.1078	1.4593
-0.3392	0.8972	0.4528

2 ALS

알고리즘

2. 아이템 행렬을 고정하고 사용자 행렬을 최적화 $u_i = (V^T \times V + \lambda I)^{-1} \times V^T \times R_i.$

0.3151	3.2962
1.1118	0.5423
0.3225	0.9144
2.1187	1.8521

모든 Row에 대해서 진행

$u_1 = (V^T \times V + \lambda I)^{-1} \times V^T \times R_1. = [0.3151, 3.2962]$

$u_2 = (V^T \times V + \lambda I)^{-1} \times V^T \times R_2.$

$u_3 = (V^T \times V + \lambda I)^{-1} \times V^T \times R_3.$

$u_4 = (V^T \times V + \lambda I)^{-1} \times V^T \times R_4.$

2 ALS

알고리즘

3. 사용자 행렬을 고정하고 아이템 행렬을 최적화 $v_j = (U^T \times U + \lambda I)^{-1} \times U^T \times R_{.j}$

1.9557	-0.090
-0.525	0.9939
1.5017	0.4663

모든 Row에 대해서 진행

$v_1 = (U^T \times U + \lambda I)^{-1} \times U^T \times R_{.1} = [1.9557, -0.0900]$

$v_2 = (U^T \times U + \lambda I)^{-1} \times U^T \times R_{.2}$

$v_3 = (U^T \times U + \lambda I)^{-1} \times U^T \times R_{.3}$

2 ALS

알고리즘

4. 2와 3의 과정을 설정한 Iterations 만큼 반복

02 실습

2 실습

데이터

무비렌즈 (https://grouplens.org/datasets/movielens/)

MovieLens

GroupLens Research has collected and made available rating data sets from the MovieLens web site (http://movielens.org). The data sets were collected over various periods of time, depending on the size of the set. Before using these data sets, please review their README files for the usage licenses and other details.

usedId	moiveId	rating	timestamp
1	31	2.5	1260759144
1	1029	3.0	1260759179
1	1061	3.0	1260759182
1	1129	2.0	1260759185

감사합니다