

## Coursera 16-4 Collaborative-filtering Algorithm

<https://www.coursera.org/learn/machine-learning/lecture/f26nH/collaborative-filtering-algorithm>

### Collaborative filtering optimization objective

→ Given  $x^{(1)}, \dots, x^{(n_m)}$ , estimate  $\theta^{(1)}, \dots, \theta^{(n_u)}$ :

$$\min_{\theta^{(1)}, \dots, \theta^{(n_u)}} \left[ \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2 \right]$$

→ Given  $\theta^{(1)}, \dots, \theta^{(n_u)}$ , estimate  $x^{(1)}, \dots, x^{(n_m)}$ :

$$\min_{x^{(1)}, \dots, x^{(n_m)}} \left[ \frac{1}{2} \sum_{i=1}^{n_m} \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^n (x_k^{(i)})^2 \right]$$

Minimizing  $x^{(1)}, \dots, x^{(n_m)}$  and  $\theta^{(1)}, \dots, \theta^{(n_u)}$  simultaneously:

$$J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}) = \frac{1}{2} \sum_{(i,j):r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^n (x_k^{(i)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2$$

$$\min_{\substack{x^{(1)}, \dots, x^{(n_m)} \\ \theta^{(1)}, \dots, \theta^{(n_u)}}} J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)})$$

- 협업 필터링은 앞에서 배운 2가지의 알고리즘을 동시에 실행하는 것이다.

- 협업 필터링의 첫 번째 항은 평점을 매긴 모든 사용자에게 대한  $X(i)$ , 세타에 대한 것이다. 나머지 두 개의 항은 세타와  $X(i)$ 의 정규화 항이다.

- 동시에 학습을 진행하고 있기 때문에  $x_0$ (절편값)이 필요하지 않다.  $x \in \mathbb{R}^n$ ,  $\theta \in \mathbb{R}^n$ .

### Collaborative filtering algorithm

→ 1. Initialize  $x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}$  to small random values.

→ 2. Minimize  $J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)})$  using gradient descent (or an advanced optimization algorithm). E.g. for every  $j = 1, \dots, n_u, i = 1, \dots, n_m$ :

$$x_k^{(i)} := x_k^{(i)} - \alpha \left( \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) \theta_k^{(j)} + \lambda x_k^{(i)} \right)$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \left( \sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) x_k^{(i)} + \lambda \theta_k^{(j)} \right)$$

3. For a user with parameters  $\theta$  and a movie with (learned) features  $x$ , predict a star rating of  $\theta^T x$ .

$$(\theta^{(i)})^T (x^{(i)})$$

- small random values 초기값으로 학습을 진행하는 것이 neural network와

-  $x_0$ 가 없기 때문에  $k=0$ 인 특수케이스의 Gradient descent 모델이 존재하지 않는다.

In the algorithm we described, we initialized  $x^{(1)}, \dots, x^{(n_m)}$  and  $\theta^{(1)}, \dots, \theta^{(n_u)}$  to small random values. Why is this?

- ☐ This step is optional. Initializing to all 0's would work just as well.
- ☐ Random initialization is always necessary when using gradient descent on any problem.
- ☐ This ensures that  $x^{(i)} \neq \theta^{(j)}$  for any  $i, j$ .
- ☒ This serves as symmetry breaking (similar to the random initialization of a neural network's parameters) and ensures the algorithm learns features  $x^{(1)}, \dots, x^{(n_m)}$  that are different from each other.



\*단어공부\*

Sequentially: 순차적으로

Convention: 관습, 관례, 대회