

## Coursera 16-2 Content Based Recommendations

<https://www.coursera.org/learn/machine-learning/lecture/uG59z/content-based-recommendations>

**Content-based recommender systems**

$n_u = 4, n_m = 5$   
 $x_0 = 1$

| Movie                            | Alice (1) | Bob (2) | Carol (3) | Dave (4) | $x_1$ (romance) | $x_2$ (action) |
|----------------------------------|-----------|---------|-----------|----------|-----------------|----------------|
| $x^{(1)}$ Love at last 1         | 5         | 5       | 0         | 0        | 0.9             | 0              |
| $x^{(2)}$ Romance forever 2      | 5         | ?       | ?         | 0        | 1.0             | 0.01           |
| $x^{(3)}$ Cute puppies of love 3 | ?         | 4       | 0         | ?        | 0.99            | 0              |
| $x^{(4)}$ Nonstop car chases 4   | 0         | 0       | 5         | 4        | 0.1             | 1.0            |
| $x^{(5)}$ Swords vs. karate 5    | 0         | 0       | 5         | ?        | 0               | 0.9            |

$x^{(1)} = \begin{bmatrix} 1 \\ 0.9 \\ 0 \end{bmatrix}$

→ For each user  $j$ , learn a parameter  $\theta^{(j)} \in \mathbb{R}^3$ . Predict user  $j$  as rating movie  $i$  with  $(\theta^{(j)})^T x^{(i)}$  stars.

$\theta^{(j)} \in \mathbb{R}^{n+1}$

$x^{(3)} = \begin{bmatrix} 1 \\ 0.99 \\ 0 \end{bmatrix} \leftrightarrow \theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}$

$(\theta^{(1)})^T x^{(3)} = 5 \times 0.99 = 4.95$

$n=2$

- 영화의 특성에 따라  $x_1, x_2$ 로 측정, 예를들어 Love at last의 경우 romance적 요소가 많이 포함되어있다. 반면 Sword vs karate는 action요소가 포함되어있다.

-  $x_0 = 1$ 라는 절편 값을 이용하여  $x^{(1)}$ 의 벡터를 만들어준다. 나머지  $x^{(2)}, x^{(3)}...$ 도 같은 방법으로 만들어준다.

-  $\theta^{(j)} \in \mathbb{R}^{n+1}$ ,  $n=2$ , 세타값은 사용자에게 따라 각각 다르다. 여기서는 세타1이 주어져있다고 가정

$(\theta^{(j)})^T x^{(i)}$  계산을 하면 Alice가 평가한 Cute puppies of love는 4.95가 나오는 것을 확인 할 수 있다.

### Problem formulation

→  $r(i, j) = 1$  if user  $j$  has rated movie  $i$  (0 otherwise)

→  $y^{(i, j)}$  = rating by user  $j$  on movie  $i$  (if defined)

→  $\theta^{(j)}$  = parameter vector for user  $j$

→  $x^{(i)}$  = feature vector for movie  $i$

→ For user  $j$ , movie  $i$ , predicted rating:  $(\theta^{(j)})^T x^{(i)}$

$\theta^{(j)} \in \mathbb{R}^{n+1}$

→  $m^{(j)}$  = no. of movies rated by user  $j$

To learn  $\theta^{(j)}$ :

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

- 예측된 값이 Trainig set의 값과 원자료의 값이 비슷할 수 있도록 세타를 선택해준다.
- 선형회귀의 최소제곱법을 이용하여 식을 만들어준다. 그 후 Regularization항을 넣어준다.
- 추천시스템은 위에 식을 조금 변형한 형태이다.

### Optimization objective:

To learn  $\theta^{(j)}$  (parameter for user  $j$ ):

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

To learn  $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(n_u)}$ :

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

$\theta^{(1)}, \dots, \theta^{(n_u)}$

- 한 명의 유저의 최소화 세타 값을 구하고 싶으면 1번식, 여러 명의 유저의 최소화 세타 값을 구하고 싶으면 2번 (질문필수)

### Optimization algorithm:

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

$J(\theta^{(1)}, \dots, \theta^{(n_u)})$

Gradient descent update:

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \sum_{i:r(i,j)=1} \left( (\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right) x_k^{(i)} \quad (\text{for } k = 0)$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \left( \sum_{i:r(i,j)=1} \left( (\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right) x_k^{(i)} + \lambda \theta_k^{(j)} \right) \quad (\text{for } k \neq 0)$$

$\frac{\partial}{\partial \theta_k^{(j)}} J(\theta^{(1)}, \dots, \theta^{(n_u)})$

- 선형회귀와 유일한 차이점은  $1/m$ 을 사용하지 않는다는 점이다.
- 배운 알고리즘을 이용하여 좀더 발전된 최적화 알고리즘을 사용할 수 있다. Ex) conjugate gradient, LBFGS

Subtract : 추출하다, 빼다

for all of whatever : 어쨌든, 뭐든간에