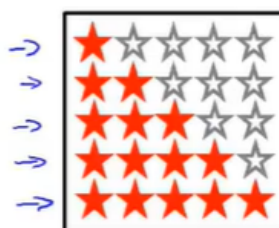


Example: Predicting movie ratings

→ User rates movies using one to five stars

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)
Love at last	5	5	0	0
Romance forever	5	?	?	0
Cute puppies of love	?	4	0	?
Nonstop car chases	0	0	5	4
Swords vs. karate	0	0	5	?

$$n_u = 4 \quad n_m = 5$$



→ n_u = no. users
 → n_m = no. movies
 $r(i, j) = 1$ if user j has rated movie i
 $y^{(i, j)}$ = rating given by user j to movie i (defined only if $r(i, j) = 1$)
 $0, \dots, 5$

Andrew Ng

在预测电影评分这个例子中，我们应该预测用户会给他没有看过的电影打多少分，再推荐他会打高分的电影给他。

Content-based recommender systems

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

→ 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.

$$x^{(3)} = \begin{bmatrix} 1 \\ 0.99 \\ 0 \end{bmatrix} \leftrightarrow \theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix} \quad (\theta^{(1)})^T x^{(3)} = 5 \times 0.99 = 4.95$$

Andrew Ng

我们预测问号上的喜爱程度，是将电影中的成分当作特征，再添加一个截距项 1，得到 x 向量，再用某种方式得到参数向量，之后我们利用参数向量的转置乘以 x 向量，得到了喜爱程度。

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: $\underline{(\theta^{(j)})^T x^{(i)}}$

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

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

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

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

Andrew Ng

为了得到sita参数向量，最小化代价方程。将预测喜爱值看作线性回归问题，得到代价方程，加入正则化项，截距项不计入正则化项，所以k从1开始。代价方程中去掉m，因为是常数项。

Optimization objective:

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

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

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

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

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

Andrew Ng

第二个代价方程针对所有用户。

利用梯度下降来求解参数向量

k=0时没有正则化项，当k! =0，正则化项会对导数产生影响，所以更新梯度下降方程针对两种情况。

Optimization algorithm:

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

Gradient descent update:

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

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

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

