



Machine Learning

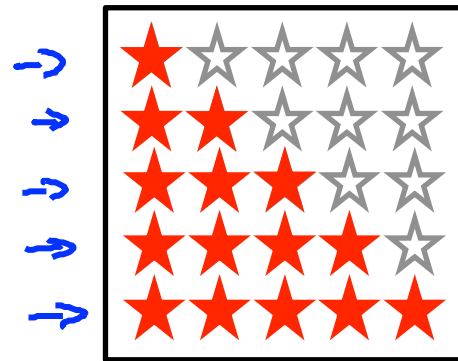
# Recommender Systems

---

## Problem formulation

## Example: Predicting movie ratings

→ User rates movies using ~~one~~ to five stars  
zero



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

$$n_u = 4$$

$$n_m = 5$$

→  $n_u$  = 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, ..., 5

于是 我们开发一个推荐系统 主要工作就是 想出一种学习算法 能够帮我们自动地填上这些缺失的数值 这样我们就能 比方说 看一下 用户还没看过哪些电影

然后向用户 推荐新电影 你试图预测 还有什么电影可能会让一位用户感兴趣 这些就是推荐系统问题的正式表述



Machine Learning

# Recommender Systems

---

Content-based  
recommendations

# Content-based recommender systems

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

Movie	Alice (1) $\theta^{(1)}$	Bob (2) $\theta^{(2)}$	Carol (3) $\theta^{(3)}$	Dave (4) $\theta^{(4)}$
Love at last 1	5	5	0	0
Romance forever 2	5	?	?	0
Cute puppies of love 3	4.95	4	0	?
Nonstop car chases 4	0	0	5	4
Swords vs. karate 5	0	0	5	?

$x^{(i)} \rightarrow$  [Love at last] 1  
 $x^{(2)} \rightarrow$  Romance forever 2  
 $x^{(3)} \rightarrow$  Cute puppies of love 3  
 $x^{(4)} \rightarrow$  Nonstop car chases 4  
 $x^{(5)} \rightarrow$  Swords vs. karate 5

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

$n=2$

→ For each user  $j$ , learn a parameter  $\theta^{(j)} \in \mathbb{R}^3$ . Predict user  $j$  as rating movie  $i$  (with  $x^{(i)}$ ) stars.  $\theta^{(j)} \in \mathbb{R}^{n+1}$

截距项, 都是1

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

愛情成份  
動作成份

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

$\theta$ 就相當於logistic reg 中的 $\theta$ , 是學習出來的, 我們根據 $\theta x$ 的值來預測電影的評分

仍然用  $n$  表示特征变量数 不包括截距项 这样  $n$ 就等于2

## Problem formulation

記憶: movie中有字母i

- $r(i, j) = 1$  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  $\theta^{(j)}$ :

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

我要去掉这个  $m(j)$  这只是一个常数 我可以去掉这一项  
不改变  $\theta(j)$  的最优化结果

## Optimization objective:

後面我會拿它跟  $x^{(i)}$  比較

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

注意是對  $i$  求和:

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

$r(i, j)$  的定義見上頁

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

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

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

## 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:

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

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

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



Machine Learning

# Recommender Systems



---

## Collaborative filtering



# Problem motivation

注意這裡的1,2是下標, 表示 $x^{(i)}$ 的分量

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

在这段视频中 我们要讲 一种构建推荐系统的方法 叫做collaborative filtering  
有一个值得一提的 特点 那就是它能实现 对features的学习  
我的意思是 这种算法能够 自行学习所要使用的features

要让每个人 看完每一部电影 告诉你你每一部电影有多浪漫 多动作 这是一件不容易的事情

假如我们 有某一个数据集 我们并不知道特征的值是多少 所以比如我们得到一些 关于电影的数据 不同用户对电影的评分 我们并不知道每部电影 到底有多少浪漫的成分 也不知道到底每部电影里面动作成分是多少 于是我把所有的问题都打上问号

# Problem motivation

如果我们能够从用户那里 得到这些  $\theta$  参考值 那么我们理论上就能推测出每部电影的  $x_1$  以及  $x_2$  的值

Movie	Alice (1) $\theta^{(1)}$	Bob (2) $\theta^{(2)}$	Carol (3) $\theta^{(3)}$	Dave (4) $\theta^{(4)}$	$x_1$ (romance)	$x_2$ (action)
<del>Love at last</del>	5	5	0	0	1.0	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	?	?	?

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

$\theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}$ , 
  $\theta^{(2)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}$ , 
  $\theta^{(3)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}$ , 
  $\theta^{(4)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}$

已知  $\theta$ , 求  $x$ :

$$\begin{aligned}
 (\theta^{(1)})^T x^{(1)} &\approx 5 \\
 (\theta^{(2)})^T x^{(1)} &\approx 5 \\
 (\theta^{(3)})^T x^{(1)} &\approx 0 \\
 (\theta^{(4)})^T x^{(1)} &\approx 0
 \end{aligned}$$

假如 Alice 告诉我们 她十分喜欢 爱情电影 于是 Alice 的特征  $x_1$  对应的值就是5  
 假设 Alice 告诉我们 她非常不喜欢动作电影 于是这一个特征就是0

# Optimization algorithm

Given  $\theta^{(1)}, \dots, \theta^{(n_u)}$ , to learn  $x^{(i)}$ :

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

注意是對j求和  
(p6中求 $\theta$ 時,  
是對i求和)

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

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

我们之前 这个视频中讲的是 如果用户愿意 为你提供参数 那么你就 可以为不同的电影估计特征

## Collaborative filtering

如果我们能知道  $\theta$  就能学习到  $x$   
如果我们知道  $x$  也会学出  $\theta$  来

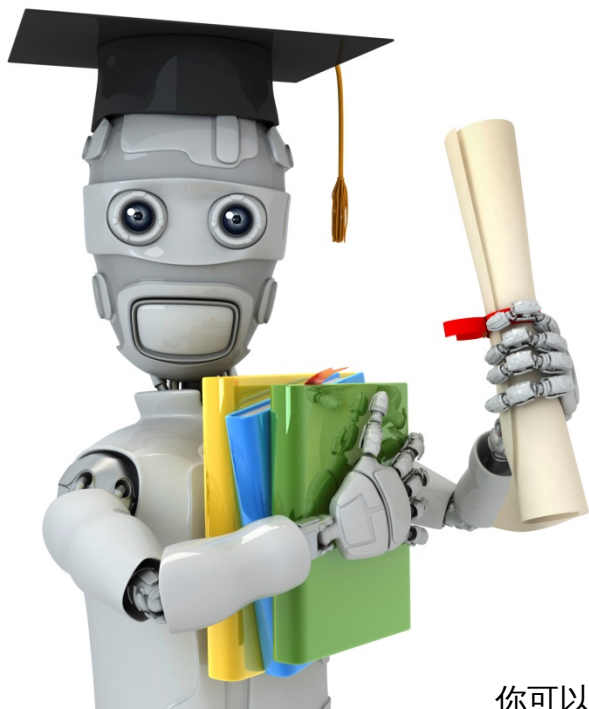
$$\sigma^{(i,j)} \\ y^{(i,j)}$$

Given  $x^{(1)}, \dots, x^{(n_m)}$  (and movie ratings),  
can estimate  $\theta^{(1)}, \dots, \theta^{(n_u)}$  ↗

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

Guess  $\Theta \rightarrow x \rightarrow \Theta \rightarrow x \rightarrow \Theta \rightarrow x \rightarrow \dots$

而这样一来 你能做的 就是 如果这真的可行的话 实际上你能做的就是 随机猜 $\theta$  的值  
基于你一开始随机 猜测出的  $\theta$  的值 继你可以继续下去 运用我们刚刚讲到的 步骤 我们可以学习出 不同电影的特征  
给出已有的一些电影的 原始特征 你可以运用 我们在上一个视频中讨论过的 第一种方法 可以得到 对参数  $\theta$  的更好估计  
这样就会为用户提供更好的参数  $\theta$  集 我们就可以用这些 得到更好的 特征集或者其他数据  
然后我们可以继续 迭代 不停重复 优化 $\theta \times \theta \times \theta$  这非常有效 如果你 这样做的话 你的算法将会收敛到 一组合理的电影的特征  
以及一组对合理的 对不同用户参数的估计  
这就是基本的协同过滤算法. 这实际并不是最后. 下一个视频中,我们将改进这个算法 让其在计算时更为高效



# Recommender Systems

---

## Collaborative filtering algorithm

Machine Learning

你可以做的事 是不停地重复这些计算 或许是随机地初始化这些参数 然后解出  $\theta$  解出  $x$  解出  $\theta$  解出  $x$  但实际上呢 存在一个更有效率的算法 让我们不再需要再这样不停地 计算  $x$  和  $\theta$  而是能够将  $x$  和  $\theta$  同时计算出来

# Collaborative filtering optimization objective

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

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

$(i,j) : r(i,j)=1$   
 $x \in \mathbb{R}^n$   
 $\theta \in \mathbb{R}^n$   
 $x_i = 1$

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

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

将这两个优化目标函数 给合为一个

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

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

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

$\theta \rightarrow x \rightarrow \theta \rightarrow x \rightarrow \dots$

# 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  $\theta^T x$ .

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

~~$x_0 = 1$~~

$x \in \mathbb{R}^n, \theta \in \mathbb{R}^n$

~~$\theta_0$~~   
 $\theta_1$   
 $\dots$   
 $\theta_n$

$\frac{\partial}{\partial x_k^{(i)}} J(\dots)$



Machine Learning

# Recommender Systems


---

Vectorization:  
Low rank matrix  
factorization



# Collaborative filtering

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	?



這裡面的問號就是 $r(i, j)$ 不為1的。

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 \\ 5 & ? & ? & 0 \\ ? & 4 & 0 & ? \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 5 & 0 \end{bmatrix}$$

$$n_m = 5$$
$$n_u = 4$$

  $y^{(i,j)}$

$x^{(i)}$  是第  $i$  個 movie 的,  $x^{(i)}$  是個向量

$\theta^{(j)}$  是第  $j$  個人的,  $\theta^{(j)}$  也是個向量

## Collaborative filtering

$$X \Theta^T \leftarrow$$

$$(\Theta^{(j)})^T (x^{(i)})$$

$$(i, j) \rightarrow$$

Predicted ratings:

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 \\ 5 & ? & ? & 0 \\ ? & 4 & 0 & ? \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 5 & 0 \end{bmatrix}$$

$$\begin{bmatrix} (\theta^{(1)})^T (x^{(1)}) & (\theta^{(2)})^T (x^{(1)}) & \dots & (\theta^{(n_u)})^T (x^{(1)}) \\ (\theta^{(1)})^T (x^{(2)}) & (\theta^{(2)})^T (x^{(2)}) & \dots & (\theta^{(n_u)})^T (x^{(2)}) \\ \vdots & \vdots & \vdots & \vdots \\ (\theta^{(1)})^T (x^{(n_m)}) & (\theta^{(2)})^T (x^{(n_m)}) & \dots & (\theta^{(n_u)})^T (x^{(n_m)}) \end{bmatrix}$$

$$X = \begin{bmatrix} -(x^{(1)})^T \\ -(x^{(2)})^T \\ \vdots \\ -(x^{(n_m)})^T \end{bmatrix}$$

$$\Theta =$$

$$\begin{bmatrix} -(\theta^{(1)})^T \\ -(\theta^{(2)})^T \\ \vdots \\ -(\theta^{(n_u)})^T \end{bmatrix}$$

Low rank matrix factorization

当用户在看 某部电影  $i$  的时候 如果你想找5部 与电影 非常相似的电影 为了能给用户推荐 5部新电影 你需要做的是 找出电影  $j$  在这些不同的电影中 与我们要找的电影  $i$  的距离最小 这样你就能给你的用户推荐几部不同的电影了

## Finding related movies

For each product  $i$ , we learn a feature vector  $\underline{x}^{(i)} \in \mathbb{R}^n$ .

→  $x_1 = \text{romance}$ ,  $x_2 = \text{action}$ ,  $x_3 = \text{comedy}$ ,  $x_4 = \dots$

How to find movies  $j$  related to movie  $i$ ?

small  $\|\underline{x}^{(i)} - \underline{x}^{(j)}\| \rightarrow$  movie  $j$  and  $i$  are "similar"

5 most similar movies to movie  $i$ :

Find the 5 movies  $j$  with the smallest  $\|\underline{x}^{(i)} - \underline{x}^{(j)}\|$ .



Machine Learning

# Recommender Systems

---

## Implementational detail: Mean normalization

到目前为止 你已经了解到了 推荐系统算法或者 协同过滤算法的所有要点  
在这节视频中 我想分享最后一点实现过程中的细节  
这一点就是均值归一化 有时它可以让算法 运行得更好

# Users who have not rated any movies

我现在加上了第五个用户 Eve 她没有给任何电影评分

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

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix}$$

所以影响  $\theta^{(5)}$  值的唯一一项 是这一项 这就是说 我们想选一个向量  $\theta^{(5)}$  使得最后的正则化项 尽可能地小

对用户 Eve 来说 没有电影 满足  $r(i,j)=1$  这个条件 所以这第一项 完全不影响  $\theta^{(5)}$  的值

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

$$n=2$$

$$\theta^{(5)} \in \mathbb{R}^2$$

$$\theta^{(5)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

只有当  $\theta^{(5)}_i$  都为0 时, 这项才最小

$$\frac{\lambda}{2} [(\theta^{(5)}_1)^2 + (\theta^{(5)}_2)^2] \leftarrow$$

$$(\theta^{(5)})^T x^{(i)} = 0$$

如果我们预测 Eve 会给所有电影零星的话 我们还是没有任何好方法 来把电影推荐给她 均值归一化的想法可以让我们解决这个问题 下面介绍它是如何工作的

# Mean Normalization:

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix}$$

每个电影所得评分的均值  $\mu =$

$$\begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix}$$

把所有的电影评分减去平均评分

$\rightarrow \underline{Y} =$

$$\begin{bmatrix} 2.5 & 2.5 & -2.5 & -2.5 & ? \\ 2.5 & ? & ? & -2.5 & ? \\ ? & 2 & -2 & ? & ? \\ -2.25 & -2.25 & 2.75 & 1.75 & ? \\ -1.25 & -1.25 & 3.75 & -1.25 & ? \end{bmatrix}$$

我做的就是 把每个电影都归一化为 平均评分为零

我要假设 这就是我从用户那儿得到的数据

当然这些问号没变

For user  $j$ , on movie  $i$  predict:

当我想要做电影评分预测

$$\rightarrow (\theta^{(j)})^T (x^{(i)}) + \mu_i$$

时, 就这样. 因为我已经对数据集 减去了均值 所以我要把这个均值加回来

User 5 (Eve):

用户5的参数 仍然还是 会等于 0

$$\theta^{(5)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

然后再加上  $\mu_i$  所以如果  $\theta^{(5)}$  等于0的话, 所以用户5对电影  $i$  的评分 我们最终会预测为  $\mu_i$

$$(\theta^{(5)})^T (x^{(i)}) + \mu_i$$

如果 Eve 没给任何电影评分 我们就对这个新用户 Eve 一无所知 我们要做的就是预测她对每个电影的评分 就是这些电影所得的平均评分

$$\text{learn } \underline{\theta^{(j)}}, \underline{x^{(i)}}$$

如果有些电影是没有评分的 你可以对不同的列 进行归一化 使得它们的均值为0  
如果你 真的有个电影没有评分 可能不管怎么说 你就不该把这个电影 推荐给任何人