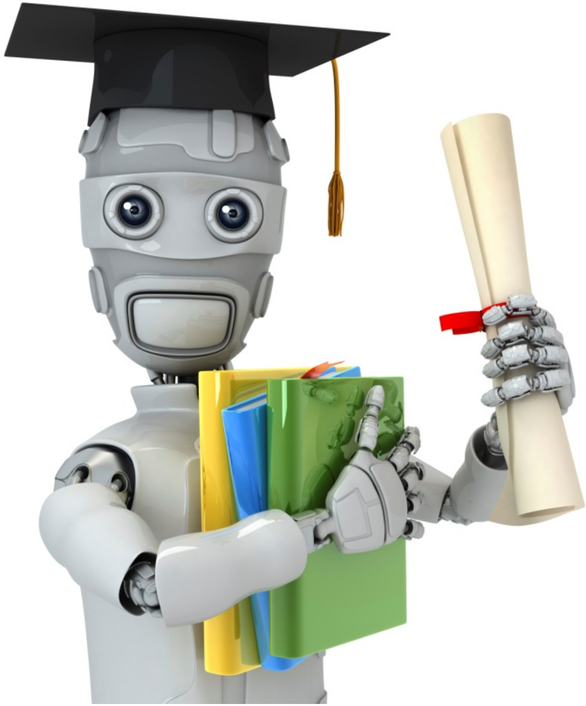


Machine Learning

Amazon,ebay,netflix,...etc
Little attention in Academia but used very often in industry!

Recommender Systems

Problem formulation

Example: Predicting movie ratings

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

romantic movies

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)
Love at last	5	5	0	0
Romance forever	5	? 4.5	? 0	0
Cute puppies of love	? 5	4	0	? 0
Nonstop car chases	0	0	5	4
Swords vs. karate	0	0	5	? 4

action movies

$n_u = 4$ $n_m = 5$

n_u = no. users

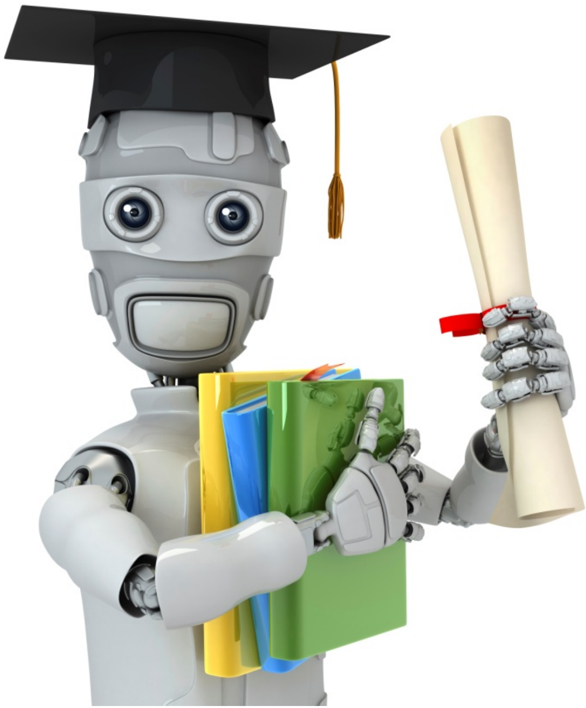
n_m = no. movies

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

$y^{(i,j)}$ = rating given by user j to movie i (defined only if $r(i,j) = 1$)

$0, \ldots, 5$

Andrew Ng



Machine Learning

Recommender Systems

Content-based recommendations

1st approach to building a recommender system

we have some ratings from users (content) available; content-based.

Content-based recommender systems

$n_u = 4$, $n_m = 5$

$x_0 = 1$

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

learned parameter

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

$n = 2$

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.

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

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

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

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

linear regression problem

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

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

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

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

prediction

actual value

Optimization objective:

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

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

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

learn for all users

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

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

Andrew Ng

Optimization algorithm:

objective

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

Gradient descent update:

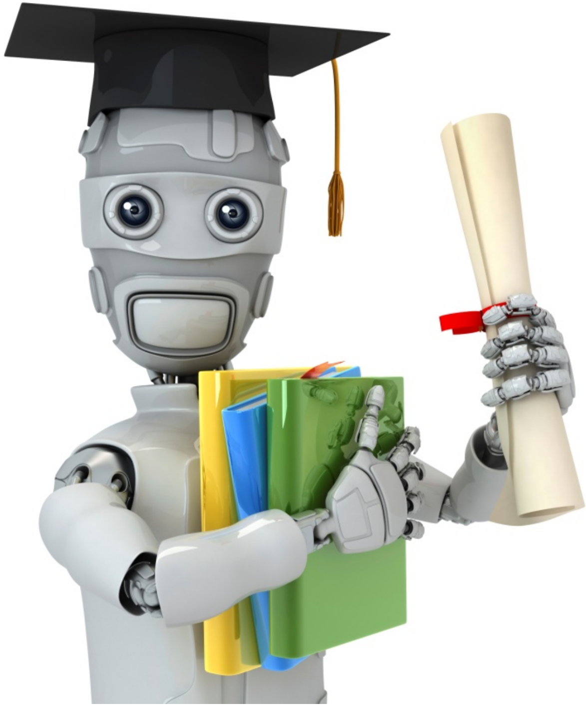
Gradient descent algorithm

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

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

$\cancel{\frac{1}{m^{(j)}}}$

Andrew Ng



Machine Learning

Recommender Systems

Collaborative filtering

Another recommender algorithm

Problem motivation

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	x_1 (romance)	x_2 (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

Problem motivation

Now, we do not have values for these features

$x_0 = 1$

Movie	Alice (1) $\theta^{(1)}$	Bob (2) $\theta^{(2)}$	Carol (3) $\theta^{(3)}$	Dave (4) $\theta^{(4)}$	x_1 (romance)	x_2 (action)
$x^{(1)}$ Love at last	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	?	?	?

$$x^{(1)} = \begin{bmatrix} 1 \\ 1.0 \\ 0.0 \end{bmatrix}$$

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

$\theta^{(j)}$

$x^{(1)}$ s.t.

$(\theta^{(1)})^T x^{(1)} \approx 5$

$(\theta^{(2)})^T x^{(1)} \approx 5$

$(\theta^{(3)})^T x^{(1)} \approx 0$

$(\theta^{(4)})^T x^{(1)} \approx 0$

Formalize this algorithm

Optimization algorithm

feature for one specific movie

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

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

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

features for all movies

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

simultaneously learn the parameters and features!!!

Collaborative filtering

$r^{(i,j)}$

$y^{(i,j)}$

Given $x^{(1)}, \ldots, x^{(n_m)}$ (and movie ratings),

can estimate $\theta^{(1)}, \ldots, \theta^{(n_u)}$

Given $\theta^{(1)}, \ldots, \theta^{(n_u)}$,

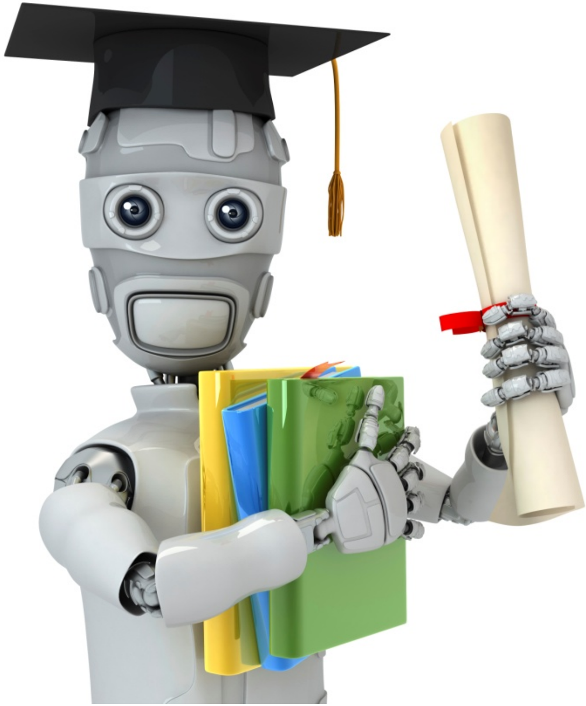
can estimate $x^{(1)}, \ldots, x^{(n_m)}$

initial guess of parameters

can actually converge to a parameter set theta

Guess $\theta \rightarrow x \rightarrow \theta \rightarrow x \rightarrow \theta \rightarrow x \rightarrow \ldots$

kind like the chicken and egg problem!



Machine Learning

Recommender Systems

Collaborative filtering algorithm

Collaborative filtering optimization objective

use this formalism, we do not need to hard code of a feature x0=1.

$(i,j): r(i,j)=1$

$\cancel{x_0 = 1}$ $\boxed{x \in \mathbb{R}^n \quad \theta \in \mathbb{R}^n}$ $\cancel{x \in \mathbb{R}^{n+1}}$ $x_1 = 1$

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

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

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

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

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

combine two objective together so that we do not need to go back and forth!!!

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

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

$\theta \to x \to \theta \to x \to \ldots$

To formalize the algorithm!

Collaborative filtering algorithm

~~$x_0 = 1$~~ $\boxed{x \in \mathbb{R}^n, \theta \in \mathbb{R}^n}$

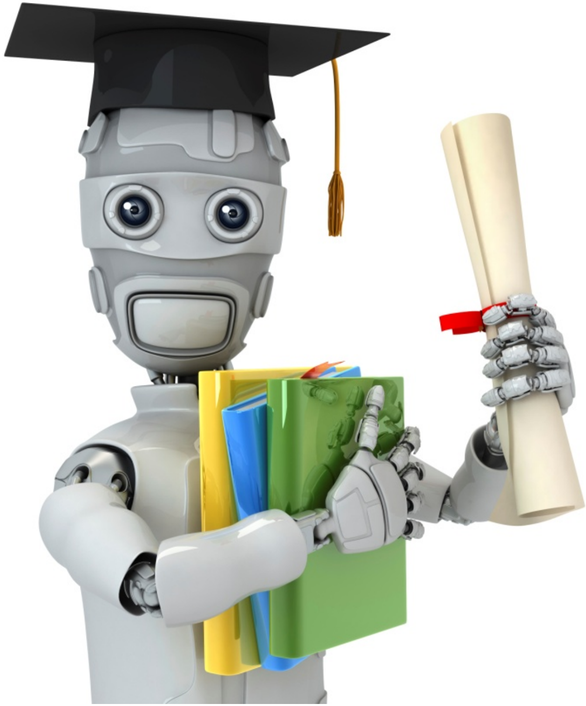
1. Initialize $x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}$ to small random values.
2. Minimize $J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)})$ using gradient descent (or an advanced optimization algorithm). E.g. for every $j = 1, \dots, n_u, i = 1, \dots, n_m$:

$$\boxed{\begin{aligned} x_k^{(i)} &:= x_k^{(i)} - \alpha \underbrace{\left(\sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})\theta_k^{(j)} + \lambda x_k^{(i)} \right)}_{\frac{\partial}{\partial x_k^{(i)}} J(\dots)} \\ \theta_k^{(j)} &:= \theta_k^{(j)} - \alpha \underbrace{\left(\sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})x_k^{(i)} + \lambda \theta_k^{(j)} \right)} \end{aligned}}$$

$\boxed{\cancel{\theta_0}, \theta_1, \dots, \theta_n}$

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

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



Machine Learning

Recommender Systems

Vectorization: Low rank matrix factorization

Vectorization implementation of this algorithm!

Collaborative filtering

$n_m = 5$ movies

$n_u = 4$ users

Movie	Alice (1) theta1	Bob (2) theta2, ...	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	?

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

$y^{(i,j)}$ movie i by user j

Andrew Ng

Collaborative filtering

$$\boxed{X \Theta^T}$$

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

(i,j)

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

Predicted ratings:

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

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

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

low rank!

$\rightarrow$ $\boxed{\text{Low rank matrix factorization}}$

Finding related movies

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

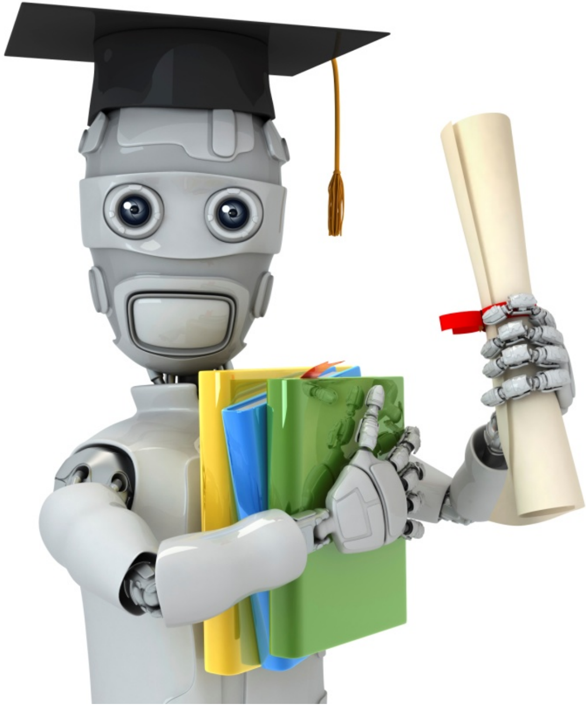
n features

→ x_1 = romance, x_2 = action, x_3 = comedy, x_4 = ...

How to find movies j related to movie i ?

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

5 most similar movies to movie i:
Find the 5 movies j with the smallest $\|x^{(i)} - x^{(j)}\|$.



Machine Learning

Recommender Systems

Implementational detail: Mean normalization

one last implementation detail

Users who have not rated any movies

mean normalization fixes this problem!

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

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

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

no term of theta5

$n=2$ $\quad \theta^{(5)} \in \mathbb{R}^2$ $\quad \theta^{(5)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$$\frac{\lambda}{2}\left[(\theta_1^{(5)})^2 + (\theta_2^{(5)})^2\right]$$

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

Mean Normalization:

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix} \begin{array}{l} \leftarrow 2.5 \\ \leftarrow 2.5 \\ \leftarrow 2 \\ \\ \\ \end{array}$$

avg rating of all movies

= 5 - 2.5

$$\mu = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix} \rightarrow \underline{Y} = \begin{bmatrix} 2.5 & 2.5 & -2.5 & -2.5 & ? \\ 2.5 & ? & ? & -2.5 & ? \\ ? & 2 & -2 & ? & ? \\ -2.25 & -2.25 & 2.75 & 1.75 & ? \\ -1.25 & -1.25 & 3.75 & -1.25 & ? \end{bmatrix}$$

Mean normalization!

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

For user j, on movie i predict:

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

User 5 (Eve):

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

output the avg movie rating

$$\boxed{\underbrace{(\Theta^{(5)})^T (x^{(i)})}_{\rightarrow 0} + \boxed{\mu_i}}$$