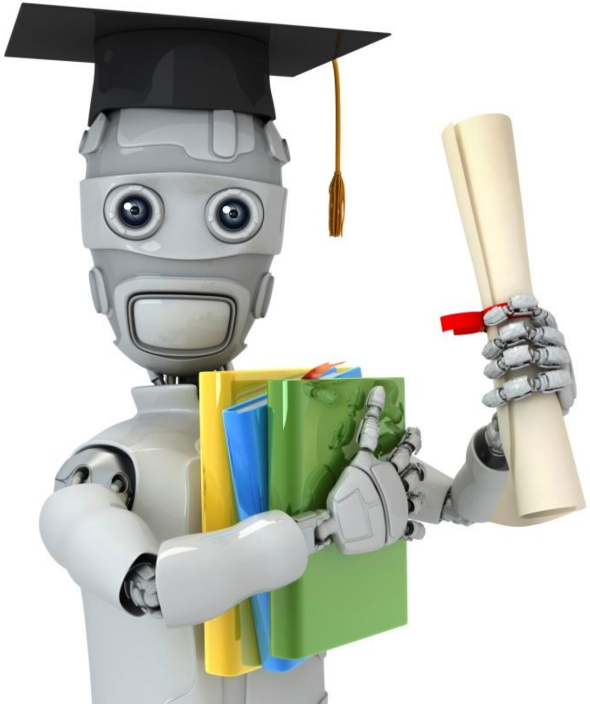


Machine Learning

Recommender Systems

Problem formulation

Example: Predicting movie ratings

User rates movies using zero 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	3	4
Swords vs. karate	0	0	5	?

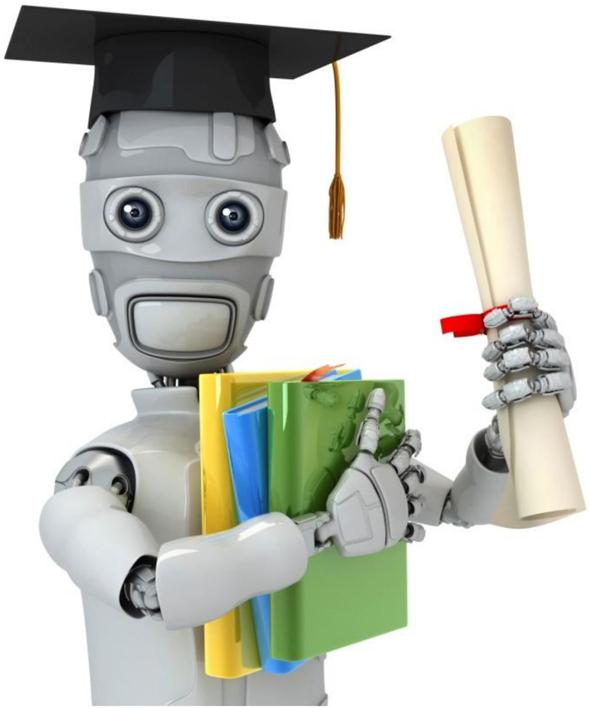
n_u = no. users

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 $r(i,j) = 1$)

Andrew Ng



Machine Learning

Recommender Systems

Content-based recommendations

Content-based recommender systems n_u=4 n_m=5

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

Let $x_0 = 1$,the intercept term, then the feature vector of movie 1,(Love at last) is

$x^1 = [1,0.9,0]^T$,and $x^2 = [1,1.0,0.01]^T$, $x^5 = [1,0,0.9]^T$

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

Example: θ^1=$[0,5,0]^T$, $x^3 = [1,0.99,0]^T$,then,$(\theta^1)^T x^3 = 4.95$.

Andrew Ng

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

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

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

Optimization objective:

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

n: number of feature

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

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

n_u: number of user

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

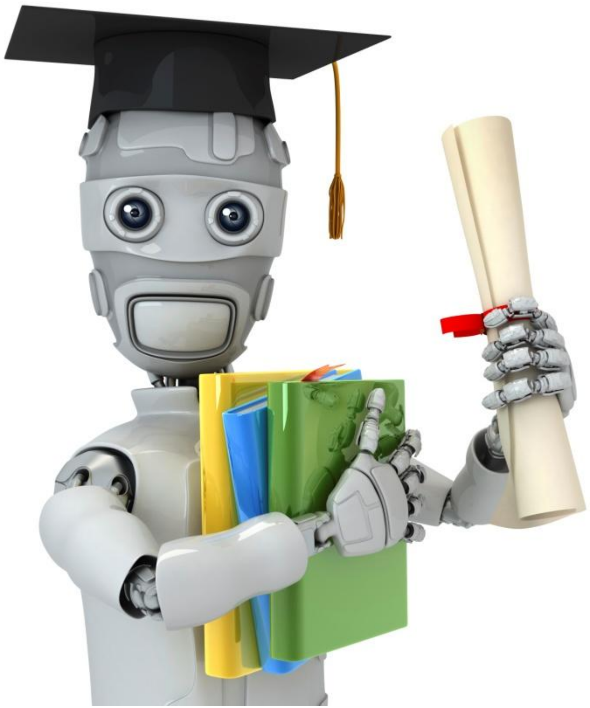
Optimization algorithm:

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

Gradient descent update:

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

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



Machine Learning

Recommender Systems

Collaborative filtering

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

	Movie	Alice (1) θ^1	Bob (2) θ^2	Carol (3) θ^3	Dave (4) θ^4	x_1 (romance)	x_2 (action)
x^1	Love at last	5	5	0	0	?	?
x^2	Romance forever	5	?	?	0	?	?
$\vdots$	Cute puppies of love	?	4	0	?	?	?
$\vdots$	Nonstop car chases	0	0	5	4	?	?
x^5	Swords vs. karate	0	0	5	?	?	?

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

Given $\theta^{(j)}$,what feature vector should $x^{(i)}$ be?

Optimization algorithm

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

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

Collaborative filtering

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

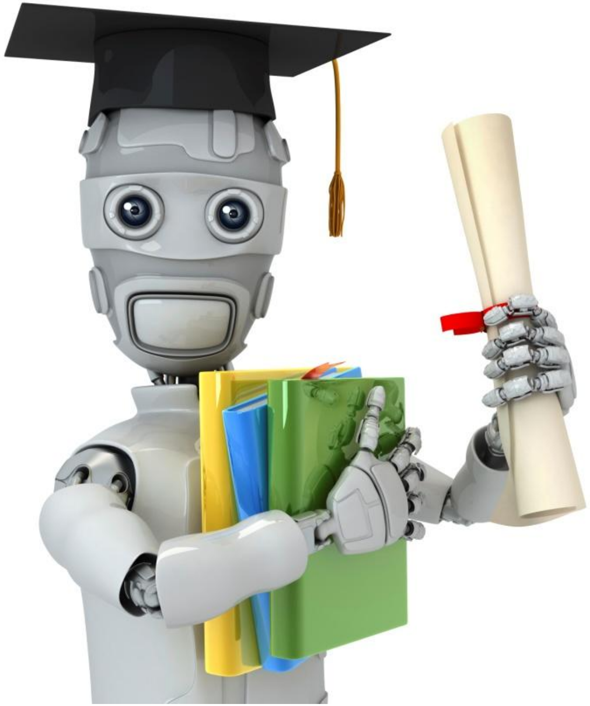
can estimate $\theta^{(1)}, \ldots, \theta^{(n_u)}$

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

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

Randomly Guess θ,use θ learn x(θ->x), the use x learn θ(x->θ).

θ->x->θ->x->θ->x->θ->x……………



Machine Learning

Recommender Systems

Collaborative filtering algorithm

Collaborative filtering optimization objective

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

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

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

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

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

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

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

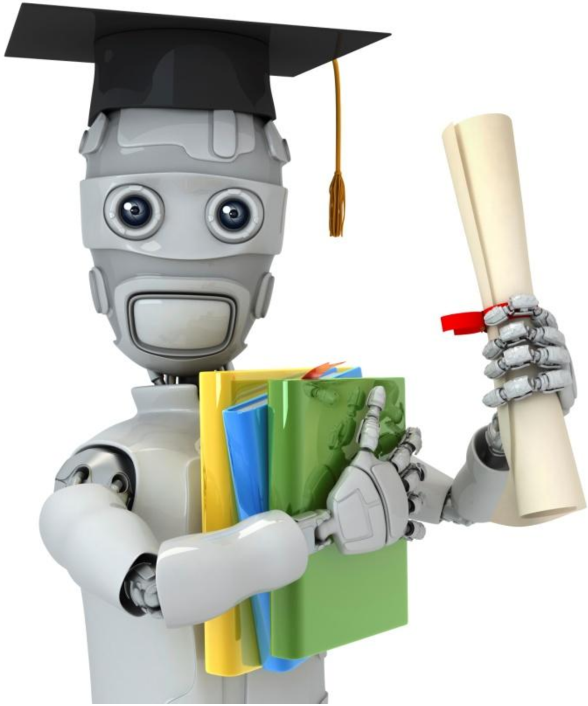
Collaborative filtering algorithm

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

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

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

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



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	?

n_u=4 n_m=5

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

Collaborative filtering

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

Predicted ratings:

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

Where

$$X = \begin{bmatrix} (x^{(1)})^T \\ (x^{(2)})^T \\ \vdots \\ (x^{(n_m)})^T \end{bmatrix} \qquad \Theta = \begin{bmatrix} (\theta^{(1)})^T \\ (\theta^{(2)})^T \\ \vdots \\ (\theta^{(n_u)})^T \end{bmatrix}$$

Finding related movies

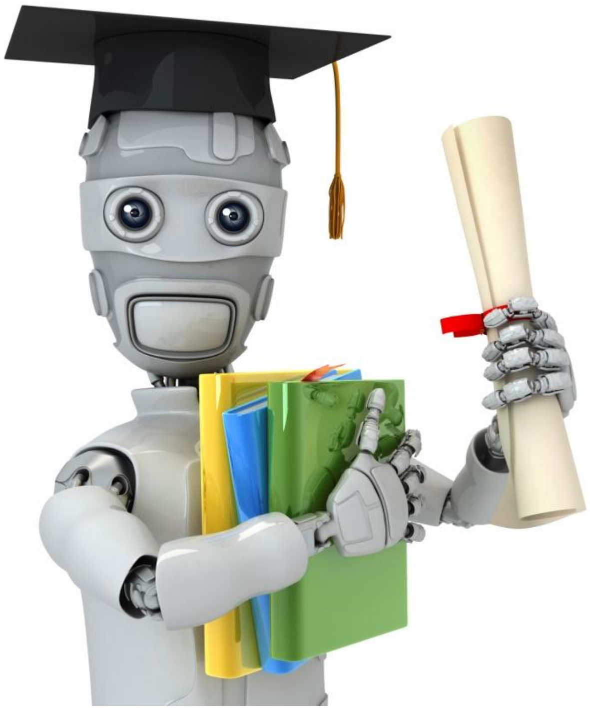
For each product i, we learn a feature vector $x^{(i)} \in \mathbb{R}^n$.

X_1=romace,x_2=action,….,

How to find movies j related to movie i?

$$small \quad ||x^{(i)} - x^{(j)}||$$

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

Users who have not rated any movies

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

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

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

Let $n = 2, \theta^{(5)} \in \mathbb{R}^2$

Because no movie that Eve is rated.so $\theta^{(5)}$=0,then for every movie, $\theta^{(5)} x^{(i)}$=0,Eve will rate 0.

Mean Normalization:

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix} \qquad \mu = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix} \rightarrow Y = \begin{bmatrix} 2.5 & 2.5 & -2.5 & -2.5 & ? \\ 2.5 & ? & ? & -2.5 & ? \\ ? & 2 & -2 & ? & ? \\ -2.25 & -2.25 & 2.75 & 1.75 & ? \\ -1.25 & -1.25 & 3.75 & -1.25 & ? \end{bmatrix}$$

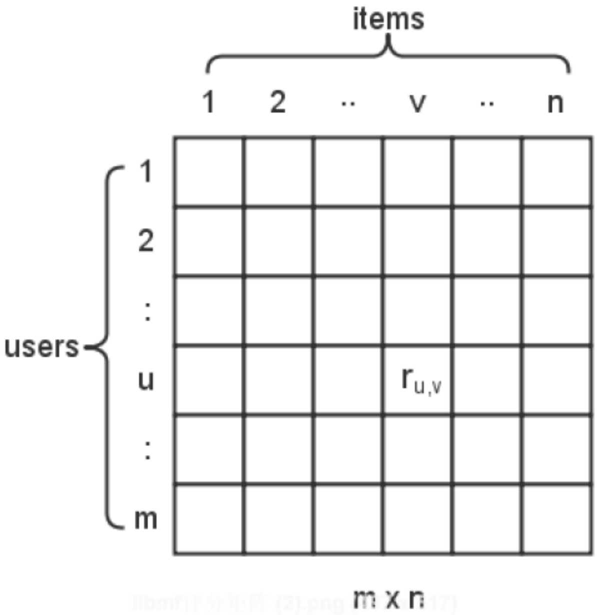
For user j, on movie i predict:

$$\left(\theta^{(j)}\right)^T \left(x^{(i)}\right) + u_i$$

User 5 (Eve): $\theta^{(5)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ $\qquad \left(\theta^{(j)}\right)^T \left(x^{(i)}\right) + u_i = u_i$

Matrix Factorization

- Matrix Factorization is an effective method for recommender systems.

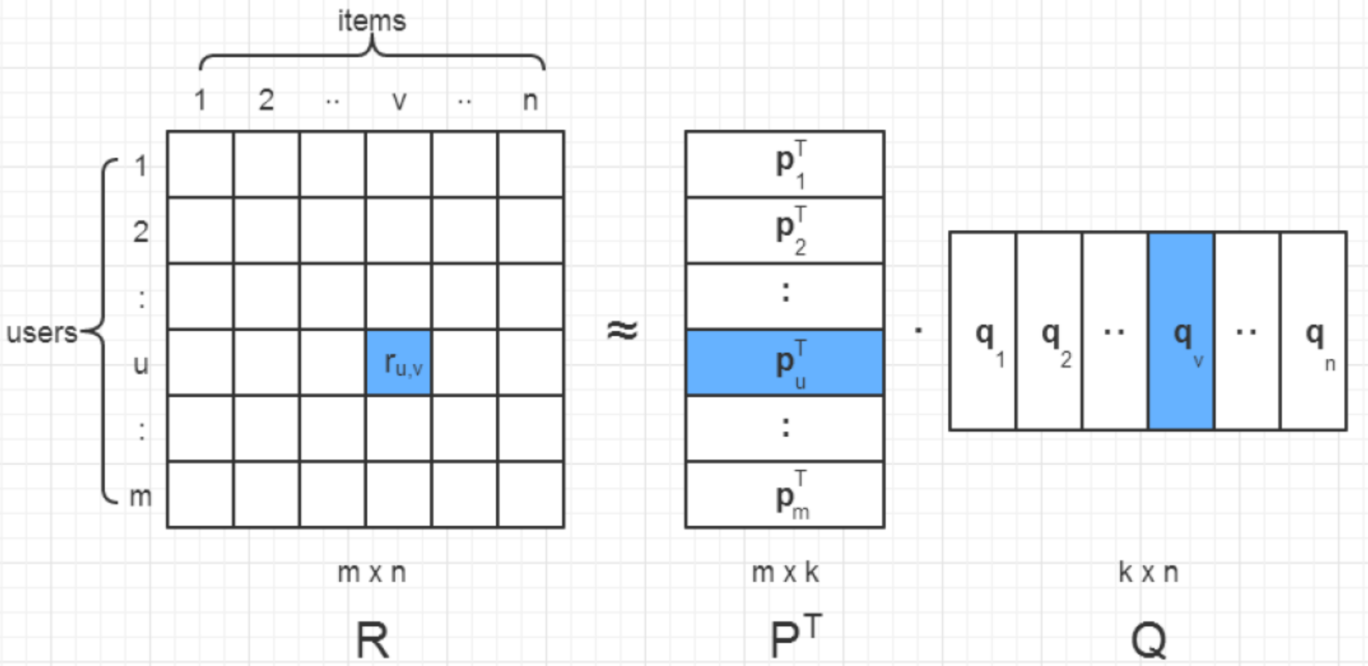


$m, n: \textit{numbers of users and items}$

$u, v: \textit{index for } u_{th} \textit{ user and } v_{th} \textit{ item}$

$r_{u,v}: u_{th} \textit{ user gives a rating } r_{u,v} \textit{ to } v_{th} \textit{ item}$

Matrix Factorization (Cont'd)



$k:$ *number of latent dimensions*

$$r_{u,v} = \mathbf{p}_u^T \mathbf{q}_v$$

$$\textit{Objective function}: \min_{P,Q} \sum_{(u,v)\in R} (r_{u,v} - \mathbf{p}_u^T \mathbf{q}_v)^2 + \lambda_P \|\mathbf{p}_u\|_F^2 + \lambda_Q \|\mathbf{q}_v\|_F^2 \quad (1)$$

Andrew Ng

Stochastic Gradient Descent (SGD)

- The basic idea of SGD is that, instead of expensively calculating the gradient of

$$\min_{P,Q} \sum_{(u,v)\in R} (r_{u,v} - \mathbf{p}_u^T \mathbf{q}_v)^2 + \lambda_P \|\mathbf{p}_u\|_F^2 + \lambda_Q \|\mathbf{q}_v\|_F^2$$

, it randomly selects a (u,v) entry from the summation and calculates the corresponding gradient.

- Once $r_{u,v}$ is chosen, the objective function is reduced to

$$(r_{u,v} - \mathbf{p}_u^T \mathbf{q}_v)^2 + \lambda_P \mathbf{p}_u^T \mathbf{p}_u + \lambda_Q \mathbf{q}_v^T \mathbf{q}_v$$

SGD (Cont'd)

$$J = (r_{u,v} - \mathbf{p}_u^T \mathbf{q}_v)^2 + \lambda_P \mathbf{p}_u^T \mathbf{p}_u + \lambda_Q \mathbf{q}_v^T \mathbf{q}_v$$

$$J = (e_{u,v})^2 + \lambda_P \mathbf{p}_u^T \mathbf{p}_u + \lambda_Q \mathbf{q}_v^T \mathbf{q}_v \quad where\ e_{u,v} = (r_{u,v} - \mathbf{p}_u^T \mathbf{q}_v)$$

$$\frac{\partial J}{\partial \mathbf{p}_u} = 2e_{u,v}(-\mathbf{q}_v) + 2\lambda_P \mathbf{p}_u = -2(e_{u,v}\mathbf{q}_v - \lambda_P \mathbf{p}_u)$$

$$\frac{\partial J}{\partial \mathbf{q}_v} = 2e_{u,v}(-\mathbf{p}_u) + 2\lambda_Q \mathbf{q}_v = -2(e_{u,v}\mathbf{p}_u - \lambda_Q \mathbf{q}_v)$$

$$\mathbf{p}_u = \mathbf{p}_u + \gamma(e_{u,v}\mathbf{q}_v - \lambda_P \mathbf{p}_u)$$

$$\mathbf{q}_v = \mathbf{q}_v + \gamma(e_{u,v}\mathbf{p}_u - \lambda_Q \mathbf{q}_v)$$

SGD (Cont'd)

$$J = (r_{u,v} - \mathbf{p}_u^T \mathbf{q}_v)^2 + \lambda_P \mathbf{p}_u^T \mathbf{p}_u + \lambda_Q \mathbf{q}_v^T \mathbf{q}_v$$

$$J = (e_{u,v})^2 + \lambda_P \mathbf{p}_u^T \mathbf{p}_u + \lambda_Q \mathbf{q}_v^T \mathbf{q}_v \quad where\ e_{u,v} = (r_{u,v} - \mathbf{p}_u^T \mathbf{q}_v)$$

consider the user bias and item bias :

$\mathbf{ub}$: *the user bias*, $\mathbf{ub}_u$: *the bias of* u_{th} *user*

$\mathbf{ib}$: *the item bias*, $\mathbf{ib}_i$: *the bias of* i_{th} *item*

avg : the average of all rates

so $e_{u,v}$ *now equels* $r_{u,v} - (\mathbf{p}_u^T \mathbf{q}_v + avg + \mathbf{ub}_u + \mathbf{ib}_v)$

SGD (Cont'd)

$$J = [r_{u,v} - (\mathbf{p}_u^T \mathbf{q}_v + avg + \mathbf{ub}_u + \mathbf{ib}_v)]^2 + \lambda_P \mathbf{p}_u^T \mathbf{p}_u + \lambda_Q \mathbf{q}_v^T \mathbf{q}_v + \lambda_{ub} \mathbf{ub}_u^2 + \lambda_{ib} \mathbf{ib}_v^2$$

$$J = (e_{u,v})^2 + \lambda_P \mathbf{p}_u^T \mathbf{p}_u + \lambda_Q \mathbf{q}_v^T \mathbf{q}_v + \lambda_P \mathbf{ub}_u^2 + \lambda_Q \mathbf{ib}_v$$

$$where\ e_{u,v} = r_{u,v} - (\mathbf{p}_u^T \mathbf{q}_v + avg + \mathbf{ub}_u + \mathbf{ib}_v)$$

$$\frac{\partial J}{\partial \mathbf{p}_u} = 2e_{u,v}(-\mathbf{q}_v) + 2\lambda_P \mathbf{p}_u = -2(e_{u,v}\mathbf{q}_v - \lambda_P \mathbf{p}_u), \quad \mathbf{p}_u = \mathbf{p}_u + \gamma(e_{u,v}\mathbf{q}_v - \lambda_P \mathbf{p}_u)$$

$$\frac{\partial J}{\partial \mathbf{q}_v} = 2e_{u,v}(-\mathbf{p}_u) + 2\lambda_Q \mathbf{q}_v = -2(e_{u,v}\mathbf{p}_u - \lambda_Q \mathbf{q}_v), \quad \mathbf{q}_v = \mathbf{q}_v + \gamma(e_{u,v}\mathbf{p}_u - \lambda_Q \mathbf{q}_v)$$

$$\frac{\partial J}{\partial \mathbf{ub}_u} = 2e_{u,v}(-1) + 2\lambda_{ub} \mathbf{ub}_u = -2(e_{u,v} - \lambda_{ub} \mathbf{ub}_u), \quad \mathbf{ub}_u = \mathbf{ub}_u + \gamma(e_{u,v} - \lambda_{ub} \mathbf{ub}_u$$

$$\frac{\partial J}{\partial \mathbf{ib}_v} = 2e_{u,v}(-1) + 2\lambda_{ib} \mathbf{ib}_v = -2(e_{u,v} - \lambda_{ib} \mathbf{ib}_v), \quad \mathbf{ib}_v = \mathbf{ib}_v + \gamma(e_{u,v} - \lambda_{ib} \mathbf{ib}_v)$$