



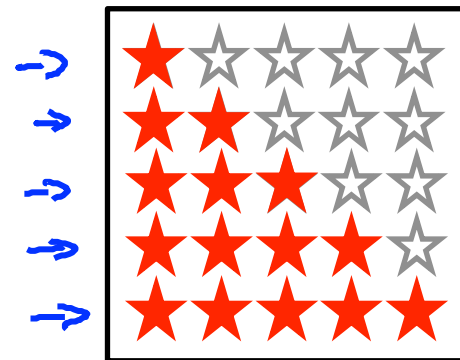
Machine Learning

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	?

$$n_u = 4$$

$$n_m = 5$$

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 $r(i, j) = 1$)



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	0	?
Nonstop car chases 4	0	0	5	4
Swords vs. karate 5	0	0	5	?

$x^{(i)} \rightarrow$

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

$n=2$

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

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

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

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 $\theta^{(j)}$:

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

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 $\theta^{(1)}$, $\theta^{(2)}$, ..., $\theta^{(n_u)}$:

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

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

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

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

Problem motivation

Movie	Alice (1) $\theta^{(1)}$	Bob (2) $\theta^{(2)}$	Carol (3) $\theta^{(3)}$	Dave (4) $\theta^{(4)}$	x_1 (romance) \downarrow	x_2 (action) \downarrow
$x^{(1)}$ Love at last	$\rightarrow 5$	$\rightarrow 5$	$\rightarrow 0$	$\rightarrow 0$	$\rightarrow 1.0$	$\rightarrow 0.0$
Romance forever	5	?	?	0	[?]	[?]
Cute puppies of love	?	4	0	?	[?]	[?]
Nonstop car chases	0	0	5	4	[?]	[?]
Swords vs. karate	0	0	5	?	[?]	[?]

$x^{(1)} = \begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix}$
 $\theta^{(j)}$
 $\theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}$, $\theta^{(2)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}$, $\theta^{(3)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}$, $\theta^{(4)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}$
 $(\theta^{(1)})^T x^{(1)} \approx 5$
 $(\theta^{(2)})^T x^{(1)} \approx 5$
 $(\theta^{(3)})^T x^{(1)} \approx 0$
 $(\theta^{(4)})^T x^{(1)} \approx 0$

Optimization algorithm

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

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

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

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

Collaborative filtering

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

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

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

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



Machine Learning

Recommender Systems

Collaborative
filtering algorithm

Collaborative filtering optimization objective

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

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

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

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

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

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

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

~~$x_0 = 1$~~

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

~~θ_0~~
 θ_1
 \dots
 θ_n

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




Machine Learning

Recommender Systems

Vectorization:
Low rank matrix
factorization

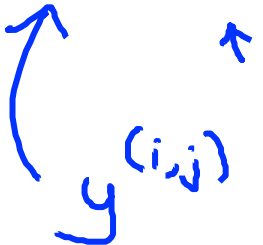
Collaborative filtering

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	?



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

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



$y^{(i,j)}$

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:

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

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

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

→ Low rank matrix factorization

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

Users who have not rated any movies

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>

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

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

$$n=2$$

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

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

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

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

Mean Normalization:

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

Handwritten annotations: Blue circles around the first row's first four elements (5, 5, 0, 0) and the last element (?). Blue arrows point from the first four elements to the right, with values 2.5, 2.5, 2, and 2.5 written next to them. A blue box highlights the last column's first four elements (0, 0, 5, 4), and a blue arrow points from this box to the last element (?).

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

Handwritten annotations: Blue circles around the first two elements (2.5, 2.5) and the last element (1.25). Blue arrows point from the first two elements to the right, and a blue box highlights the last element (1.25).

$\rightarrow \underline{Y} =$

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

Handwritten annotations: Blue circles around the first two elements (2.5, 2.5) and the last element (?). Blue arrows point from the first two elements to the right, and a blue box highlights the last column's first four elements (-2.5, -2.5, 2.75, 1.75).

For user j , on movie i predict:

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

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

User 5 (Eve):

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

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