

Week 9 - Lecture 2

Predicting Movie Ratings (Recommender Systems)

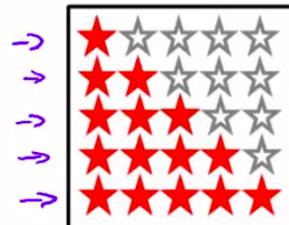
Problem Formulation

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	6
Romance forever	5	?	?	0
Cute puppies of love	?	4	0	?
Nonstop car chases	0	0	5	4
Swords vs. karate	0	0	5	?

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



$\rightarrow n_u = \text{no. users}$
 $\rightarrow n_m = \text{no. movies}$
 $r(i, j) = 1$ 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

Content Based Recommendations

Content-based recommender systems

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	$x_0 = 1$	$x^{(1)} = \begin{bmatrix} 1 \\ 0.9 \\ 0 \end{bmatrix}$
Love at last	1	5	0	0	0.9	0
Romance forever	2	5	?	0	1.0	0.01
Cute puppies of love	3	4	0	?	0.99	0
Nonstop car chases	4	0	5	4	0.1	1.0
Swords vs. karate	5	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

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: $\underline{\underline{(\theta^{(j)})^T(x^{(i)})}}$
- $m^{(j)}$ = no. of movies rated by user j

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

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

Andrew Ng

To simplify this problem a little bit, we can do the following modifications:

- As $m^{(j)}$ is just a constant, it doesn't interfere in the minimization problem solution. So, we can remove it from our optimization objective formula.

Optimization objective:

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

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

Andrew Ng

So, to optimize our $J(\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(n_u)})$ function, we can apply the Gradient Descent algorithm as follows:

Optimization algorithm:

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

\downarrow

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

Gradient descent update:

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

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

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

Andrew Ng

Note: This particular algorithm is called **content based recommendations**, or a **content based approach**, because we assume that we have available to us features for the different movies. And so where features that capture what is the content of these movies, of how romantic is this movie, how much action is in this movie.

But for many movies, we don't actually have such features. Or maybe it is very difficult to get such features for all of our movies, for all of whatever items we're trying to sell.

Collaborative Filtering

Suppose now that we don't have exact features (x_1 and x_2) that tells us how much of romance, action, etc. there is in each movie. However, each of our users told us somehow how much they like each kind of movie (like exemplified in the next figure)

Problem motivation

Movie	Alice (1) $\theta^{(1)}$	Bob (2) $\theta^{(2)}$	Carol (3) $\theta^{(3)}$	Dave (4) $\theta^{(4)}$	x_1 (romance)	x_2 (action)	$x_0 = 1$
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	?	?	?	

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

$\Theta^{(j)}$

By combining these information each user provided, we are able to construct features x_1 and x_2 accordingly to how they should be, given each user preference.

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	\downarrow	\downarrow	$x_0=1$
	$\Theta^{(1)}$	$\Theta^{(2)}$	$\Theta^{(3)}$	$\Theta^{(4)}$	x_1 (romance)	x_2 (action)	
Love at last	5	5	0	0	1.0	0.0	\times
Romance forever	5	?	?	0	?	?	
Cute puppies of love	?	4	0	?	?	?	
Nonstop car chases	0	0	5	4	?	?	
Swords vs. karate	0	0	5	?	?	?	$\times^{(1)}$

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

$\Theta^{(j)}$

$(\Theta^{(1)})^T x^{(1)} \approx 5$
 $(\Theta^{(2)})^T x^{(1)} \approx 5$
 $(\Theta^{(3)})^T x^{(1)} \approx 0$
 $(\Theta^{(4)})^T x^{(1)} \approx 0$

Andrew Ng

Now, our problem is to find values for x in order to minimize our cost function:

Optimization algorithm

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

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

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

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

Andrew Ng

Collaborative filtering

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

$$\begin{matrix} r^{(i,j)} \\ y^{(i,j)} \end{matrix}$$

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

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

Andrew Ng

Collaborative Filtering Algorithm

In the previous section, we developed an algorithm that could do back and forth in the following steps to learn theta and \$x\$'s and enhance itself iteratively.

Collaborative filtering optimization objective

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

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

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

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

Andrew Ng

What we are going to do in this section is to come up with an algorithm that can learn both things at the same time.

Collaborative filtering optimization objective

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

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

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

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

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

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

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

Andrew Ng

Note: we basically put the two objective functions together in a smart way and transformed the two problems into one.

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

Andrew Ng

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	?

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

Andrew Ng

Collaborative filtering

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

$$\text{Predicted ratings: } \hat{y}_{ij} = (\Theta^{(i)})^T (\Phi^{(j)})$$

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

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

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

Low rank matrix factorization

Andrew Ng

How do we use this information to recommend movies to the users of our system?

Finding related movies

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

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

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 :

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

Andrew Ng

Implementational Detail: Mean Normalization

A problem that might arise in our algorithm is that, when a new user starts to use our system, we don't know anything about his/hers preferences yet and, as a result, our algorithm would predict that this new user would rate every movie \$0\$ (because the only parameter that affects this new user rating in our model is the regularization term, and we want it to be as small as possible).

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	?
Romance forever	5	?	?	0	?
Cute puppies of love	?	4	0	?	?
Nonstop car chases	0	0	5	4	?
Swords vs. karate	0	0	5	?	?

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

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

$n=2$

$$\underline{\Theta}^{(s)} \in \mathbb{R}^2 \quad \underline{\Theta}^{(s)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\frac{\lambda}{2} [\underline{\Theta}_1^{(s)2} + \underline{\Theta}_2^{(s)2}] \leftarrow$$

$$(\underline{\Theta}^{(s)})^T \underline{x}^{(i)} = 0$$

Andrew Ng

How to deal with this?

Mean Normalization:

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

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

For user j , on movie i predict:

$$\rightarrow (\underline{\Theta^{(s)}})^T (\underline{x^{(i)}}) + \mu_i$$

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

User 5 (Eve):

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

$$\underbrace{(\underline{\Theta^{(s)}})^T (\underline{x^{(i)}})}_{\sim \Theta} + \mu_i$$

Andrew Ng

This means that, if we don't know anything from a user, we're going to predict that he/she will rate a particular movie as the mean of the ratings given by all other users.

References

[1] Machine Learning - Stanford University (<https://www.coursera.org/learn/machine-learning>).