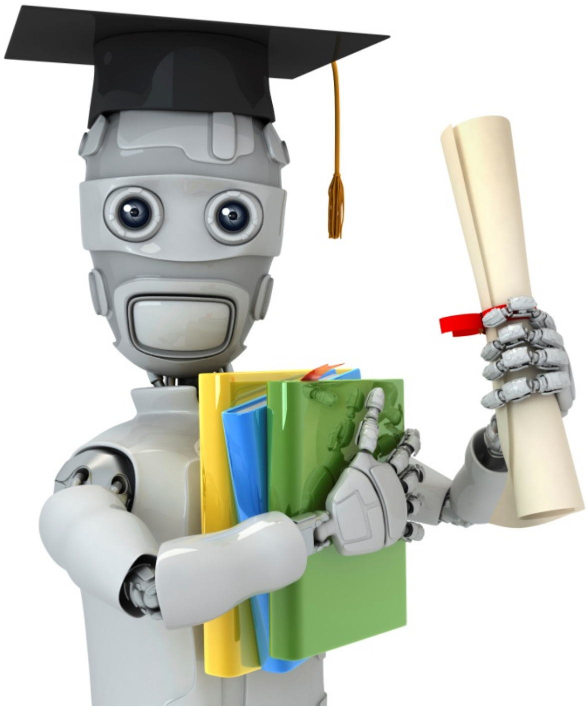


Machine Learning

Recommender Systems

Problem formulation

try to predict the missing value "?"

Example: Predicting movie ratings

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

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

alice and bob like movie 1-3

carol and dave like movie 4-5

$n_u = 4$ $n_m = 5$

→ 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 $r(i,j) = 1$)

$0, \ldots, 5$

when r(i,j) = 1, y(i,j) has value 0-5

Andrew Ng

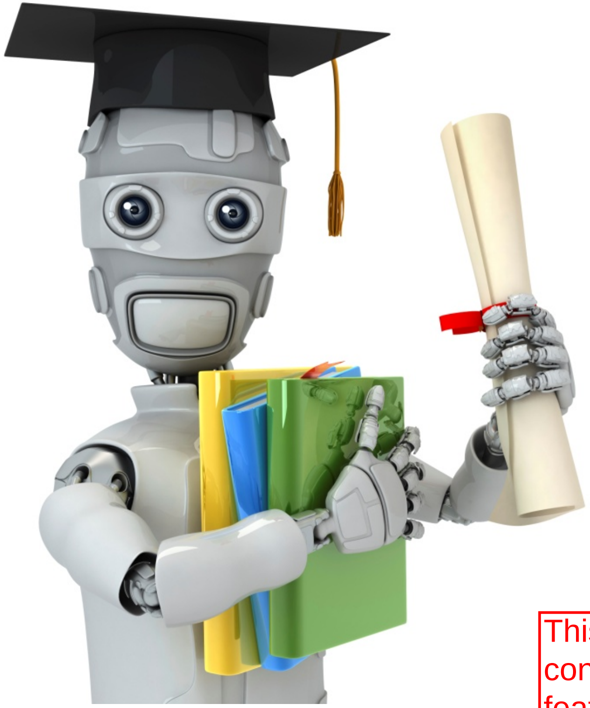
In our notation, $r(i,j) = 1$ if user j has rated movie i, and $y^{(i,j)}$ is his rating on that movie. Consider the following example (no. of movies $n_m = 2$, no. of users $n_u = 3$):

.	User 1	User 2	User 3
Movie 1	0	1	?
Movie 2	?	5	5

What is $r(2,1)$? How about $y^{(2,1)}$?

- $r(2,1) = 0,\ y^{(2,1)} = 1$
- $r(2,1) = 1,\ y^{(2,1)} = 1$
- $r(2,1) = 0,\ y^{(2,1)} = \text{undefined}$

Correct



Machine Learning

Recommender Systems

Content-based recommendations

This particular algorithm is called a 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. And we're really using features of a content of the movies to make our predictions.

Content-based recommender systems

$n_u = 4, \; n_m = 5$

$x_0 = 1$

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

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.

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

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

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

Content-based recommender systems

$n_u = 4, \; n_m = 5$

$x_0 = 1$

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

Movie	Alice (1) $\theta^{(1)}$	Bob (2) $\theta^{(2)}$	Carol (3) $\theta^{(3)}$	Dave (4) $\theta^{(4)}$	x_1 (romance)	x_2 (action)
$x^{(1)}$ $\boxed{\text{Love at last}}$ 1	5	5	0	0	0.9	0
$x^{(2)}$ Romance forever 2	5	?	?	0	1.0	0.01
$x^{(3)}$ Cute puppies of love 3	? 4.95	4	0	?	0.99	0
Nonstop car chases 4	0	0	5	4	0.1	1.0
$x^{(5)}$ Swords vs. karate 5	0	0	5	?	0	0.9

$n = 2$

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.

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

$x^{(3)} = \begin{bmatrix} 1 \\ 0.99 \\ 0 \end{bmatrix} \longleftrightarrow \theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix} \qquad (\theta^{(1)})^T x^{(3)} = 5 \times 0.99 = 4.95$

Consider the following set of movie ratings:

Movie	Alice (1)	Bob (2)	Carol (3)	David (4)	(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

Which of the following is a reasonable value for $\theta^{(3)}$? Recall that $x_0 = 1$.

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

Correct

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 $\theta^{(j)}$:

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

Optimization objective:

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

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

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

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

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

Optimization algorithm:

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

Gradient descent update:

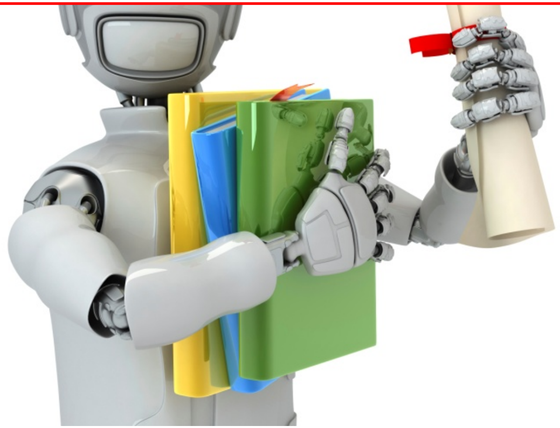
$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \sum_{i:r(i,j)=1}((\theta^{(j)})^T x^{(i)} - y^{(i,j)})x_k^{(i)} \quad \text{(for } k = 0\text{)}$$

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

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

Andrew Ng

for many movies, we don't actually have such features. Or maybe very difficult to get such features for all of our movies, And so, we'll talk about an approach to recommender systems that isn't content based and does not assume that we have someone else giving us all of these features for all of the movies in our data set.The algorithm that we're talking about has a very interesting property that it does what is called feature learning and by that I mean that this will be an algorithm that can start to learn for itself what features to use.



Recommender Systems

Collaborative filtering

in this video we've seen an initial collaborative filtering algorithm. The term collaborative filtering refers to the observation that when you run this algorithm with a large set of users, what all of these users are effectively doing are sort of collaboratively--or collaborating to get better movie ratings for everyone because with every user rating some subset with the movies, every user is helping the algorithm a little bit to learn better features, and then by rating a few movies myself, I will be helping the system learn better features and then these features can be used by the system to make better movie predictions for everyone else. And so there is a sense of collaboration where every user is helping the system learn better features for the common good.

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

But as you can imagine it can be very difficult and time consuming and expensive to actually try to get someone to watch each movie and tell you how romantic each movie and how action packed is each movie, and often you'll want even more features than just these two. So where do you get these features from?

Let's say we've gone to each of our users, and told us how much they like the romantic movies and how much they like action packed movies.

Problem motivation

$x_0 = 1$

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	?	?	?

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

$x^{(i)}$

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

$\theta^{(j)}$

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

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

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

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

Consider the following movie ratings:

.	User 1	User 2	User 3	(romance)
Movie 1	0	1.5	2.5	?

Note that there is only one feature x_1. Suppose that:

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

What would be a reasonable value for $x_1^{(1)}$ (the value denoted "?" in the table above)?

- ◉ 0.5

Correct

- ○ 1
- ○ 2
- ○ Any of these values would be equally reasonable.

Optimization algorithm

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

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

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

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

Collaborative filtering

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

$r^{(i,j)}$

$y^{(i,j)}$

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

Guess $\Theta \rightarrow x \rightarrow \Theta \rightarrow x \rightarrow \Theta \rightarrow x \rightarrow \ldots$

And what you can do is, randomly guess some value of the thetas, Now based on your initial random guess for the thetas, you can then go ahead and use the procedure to learn features for your different movies. Now given some initial set of features for your movies you can then use this first method to try to get an even better estimate for your parameters theta. Now that you have a better setting of the parameters theta for your users, we can use that to maybe even get a better set of features and so on. We can sort of keep iterating, going back and forth and optimizing theta, x theta, x theta, and this actually works and if you do this, this will actually cause your album to converge to a reasonable set of features for you movies and a reasonable set of parameters for your different users.

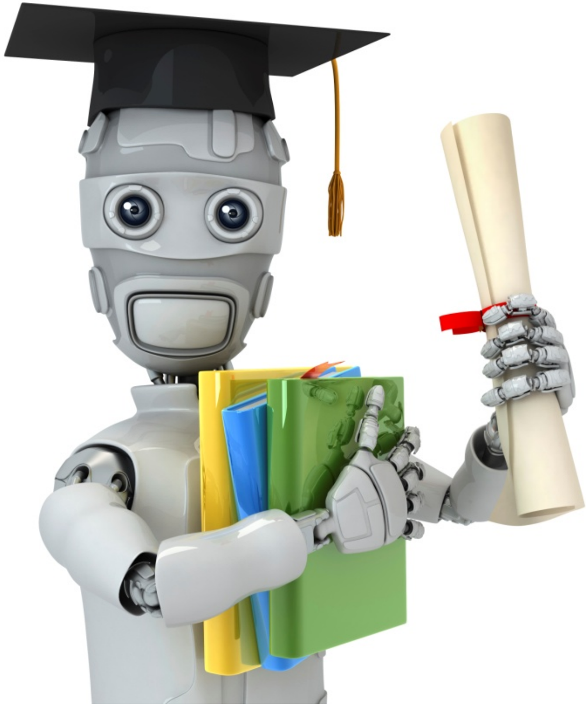
Suppose you use gradient descent to minimize:

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

Which of the following is a correct gradient descent update rule for $i \neq 0$?

- ○ $x_k^{(i)} := x_k^{(i)} + \alpha\left(\sum_{j:r(i,j)=1}\left((\theta^{(j)})^T(x^{(i)}) - y^{(i,j)}\right)\theta_k^{(j)}\right)$
- ○ $x_k^{(i)} := x_k^{(i)} - \alpha\left(\sum_{j:r(i,j)=1}\left((\theta^{(j)})^T(x^{(i)}) - y^{(i,j)}\right)\theta_k^{(j)}\right)$
- ○ $x_k^{(i)} := x_k^{(i)} + \alpha\left(\sum_{j:r(i,j)=1}\left((\theta^{(j)})^T(x^{(i)}) - y^{(i,j)}\right)\theta_k^{(j)} + \lambda x_k^{(i)}\right)$
- ◉ $x_k^{(i)} := x_k^{(i)} - \alpha\left(\sum_{j:r(i,j)=1}\left((\theta^{(j)})^T(x^{(i)}) - y^{(i,j)}\right)\theta_k^{(j)} + \lambda x_k^{(i)}\right)$

Correct



Machine Learning

Recommender Systems

Collaborative filtering algorithm

there is a more efficient algorithm that doesn't need to go back and forth between the x's and the thetas, but that can solve for theta and x simultaneously.

Collaborative filtering optimization objective

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

using this algorithm we dont need to use x0=1, because the algorithm has the flexibility to learn it by itself, no need to hard core it

$\cancel{x_0 = 1}$

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

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

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

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

for every user, the sum of all the movies rated by that user

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

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

for every movie I, sum over all users J that have rated that movie

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

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

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

$\cancel{\theta \rightarrow x \rightarrow \theta \rightarrow x \rightarrow \ldots}$

Collaborative filtering algorithm

we no longer have x0=1 and theta_0 term

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

1. Initialize $x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)}$ to small random values.

   ~~θ_0~~ θ_1 ... θ_n

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 $j = 1, \ldots, n_u, i = 1, \ldots, n_m$:

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

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

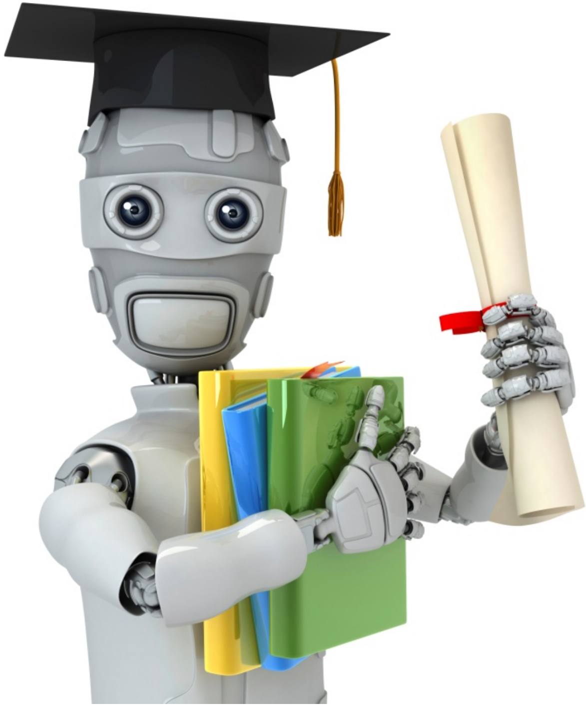
3. For a user with parameters θ and a movie with (learned) features x, predict a star rating of $\theta^T x$.

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

In the algorithm we described, we initialized $x^{(1)}, \ldots, x^{(n_m)}$ and $\theta^{(1)}, \ldots, \theta^{(n_u)}$ to small random values. Why is this?

- This step is optional. Initializing to all 0's would work just as well.
- Random initialization is always necessary when using gradient descent on any problem.
- This ensures that $x^{(i)} \neq \theta^{(j)}$ for any i, j.
- This serves as symmetry breaking (similar to the random initialization of a neural network's parameters) and ensures the algorithm learns features $x^{(1)}, \ldots, x^{(n_m)}$ that are different from each other.

Correct



Machine Learning

Recommender Systems

Vectorization: Low rank matrix factorization

vectorization implementation of collaborative filtering algorithm, and also talk about other things you can do with this algorithm. For example, given one product can you find other products that are related to this so that for example, a user has recently been looking at one product. Are there other related products that you could recommend to this user?

Collaborative filtering

$n_m = 5$
$n_u = 4$

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	?

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

$y^{(i,j)}$

its low rank matrix factorization, this term comes from the property that this matrix x times theta transpose has a mathematical property in linear algebra called that this is a low rank matrix

$X\Theta^T$

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

(i,j)

Predicted ratings:

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

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

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

m x # of feature

u x # of feature

Low rank matrix factorization

Andrew Ng

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 = ...

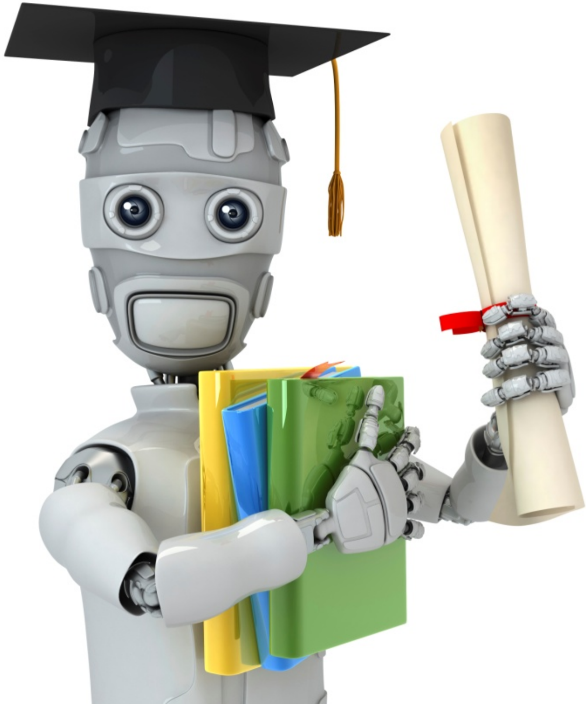
How to find movies j related to movie i?

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

Recommender Systems

Implementational detail: Mean normalization



Machine Learning

Users who have not rated any movies

we want to know theta_5

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

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

no movies for which Rij=1 for Eve so this term plays no role at all in determining theta 5

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

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

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

then

then

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

give all 0 to rating not really useful !!

the only term that effects theta 5, if your goal is to minimize this, then what you're going to end up with theta 5=0 0, because a regularization term encourages us to set parameters close to 0 and there is no data (first term) to try to pull the parameters away from 0 compute the average rating that each movie obtained.

subtract off the mean rating

normalizing each movie to have an average rating of zero.

Mean Normalization:

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

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

For user j, on movie i predict:

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

User 5 (Eve):

$$\Theta^{(5)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \underbrace{(\Theta^{(5)})^T (x^{(i)})}_{\to 0} + \boxed{\mu_i}$$

Andrew Ng

in this video we talked about mean normalization, where we normalized each row of the matrix y, to have mean 0. In case you have some movies with no ratings, so it is analogous to a user who hasn't rated anything, but in case you have some movies with no ratings, you can also play with versions of the algorithm, where you normalize the different columns to have means zero, instead of normalizing the rows to have mean zero, although that's maybe less important, because if you really have a movie with no rating, maybe you just shouldn't recommend that movie to anyone, anyway. And so, taking care of the case of a user who hasn't rated anything might be more important than taking care of the case of a movie that hasn't gotten a single rating.

So to summarize, that's how you can do mean normalization as a sort of pre-processing step for collaborative filtering. Depending on your data set, this might some times make your implementation work just a little bit better.

We talked about mean normalization. However, unlike some other applications of feature scaling, we did not scale the movie ratings by dividing by the range (max – min value). This is because:

- This sort of scaling is not useful when the value being predicted is real-valued.
- All the movie ratings are already comparable (e.g., 0 to 5 stars), so they are already on similar scales.

Correct

- Subtracting the mean is mathematically equivalent to dividing by the range.
- This makes the overall algorithm significantly more computationally efficient.