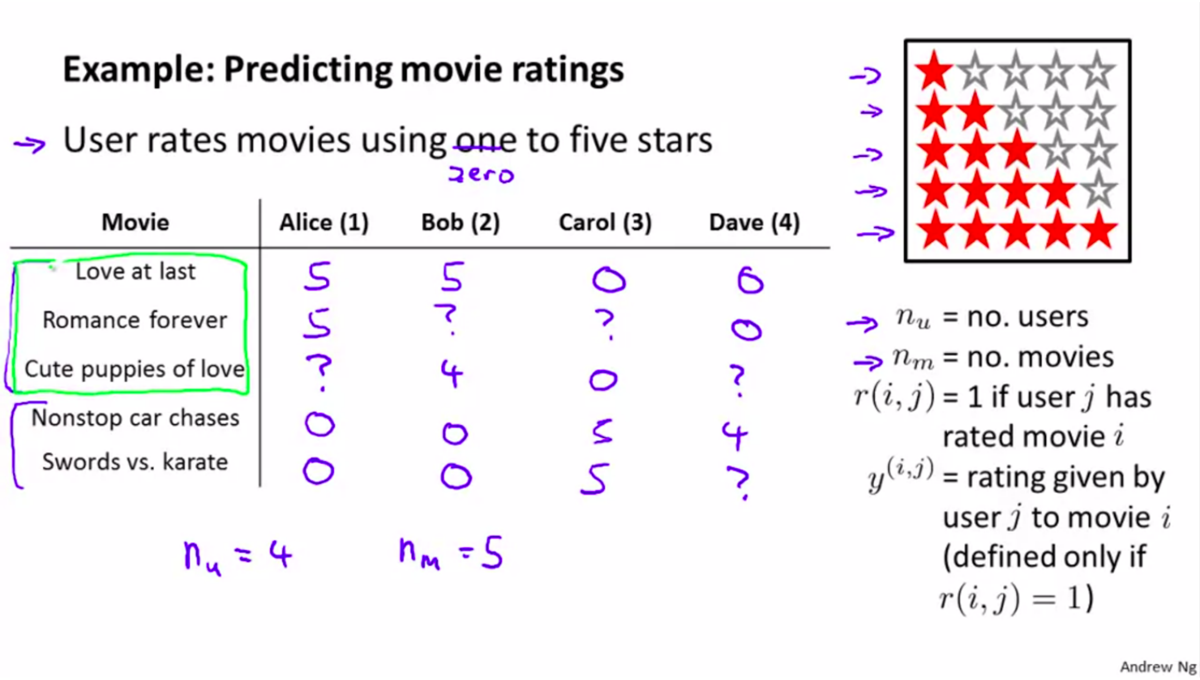
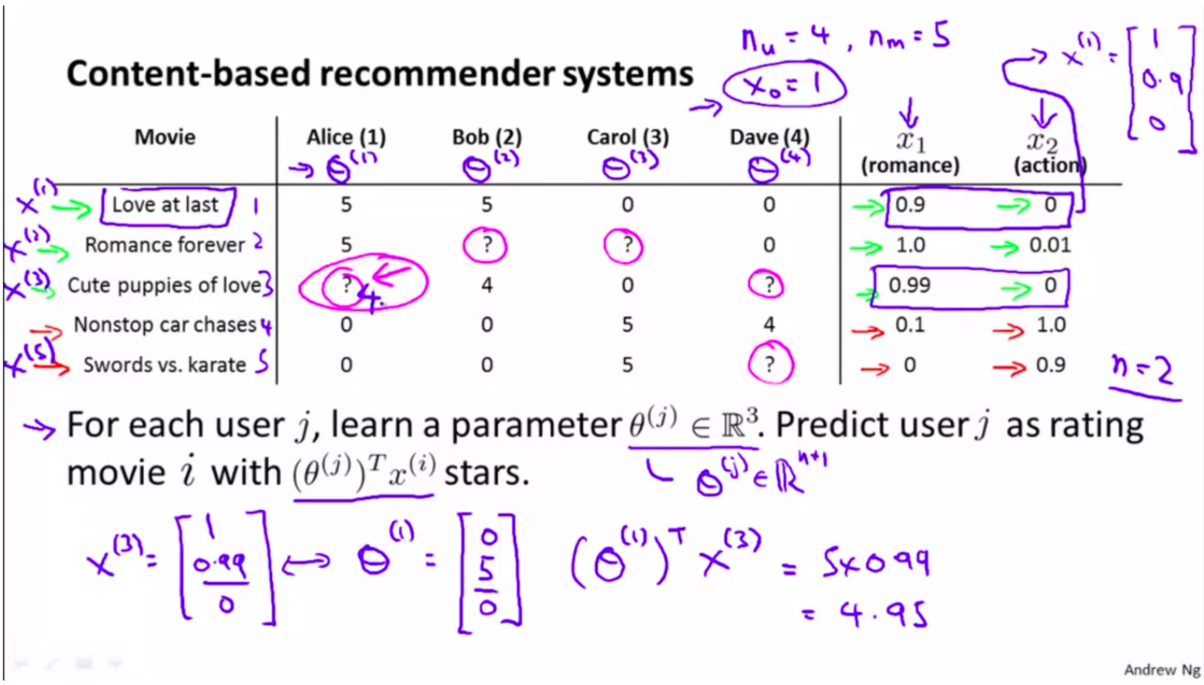
Week 9 - Lecture 1

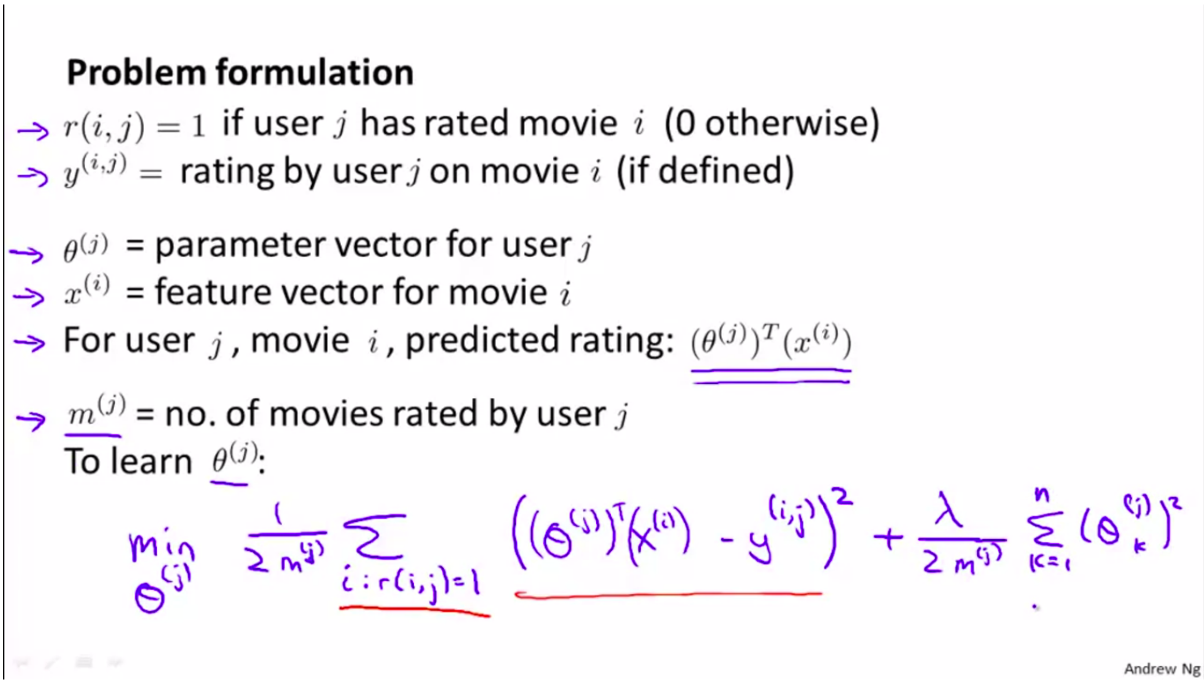
Predicting Movie Ratings (Recommender Systems)

Problem Formulation



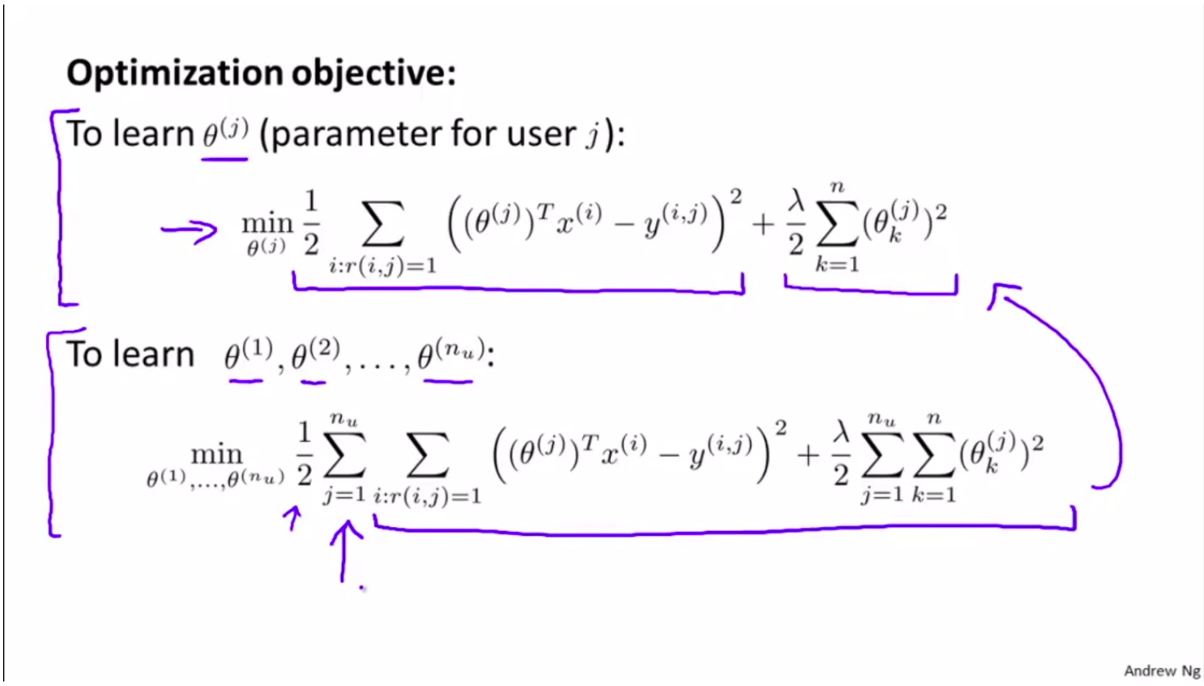
Content Based Recommendations





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

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



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

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:

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

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

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

$x_0 = 1$

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

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

Andrew Ng

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.

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	?	?	?

$x^{(1)}$

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

$\theta^{(j)}$

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

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

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

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

Andrew Ng

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

Optimization algorithm

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

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

Andrew Ng

Collaborative filtering

$r^{(i,j)}$

$y^{(i,j)}$

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

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

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

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

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

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

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

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

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

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

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

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

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

$\cancel{\theta_0}$, θ_1, …, θ_n

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

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

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

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

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

$X \Theta^T$

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

(i,j)

Predicted ratings:

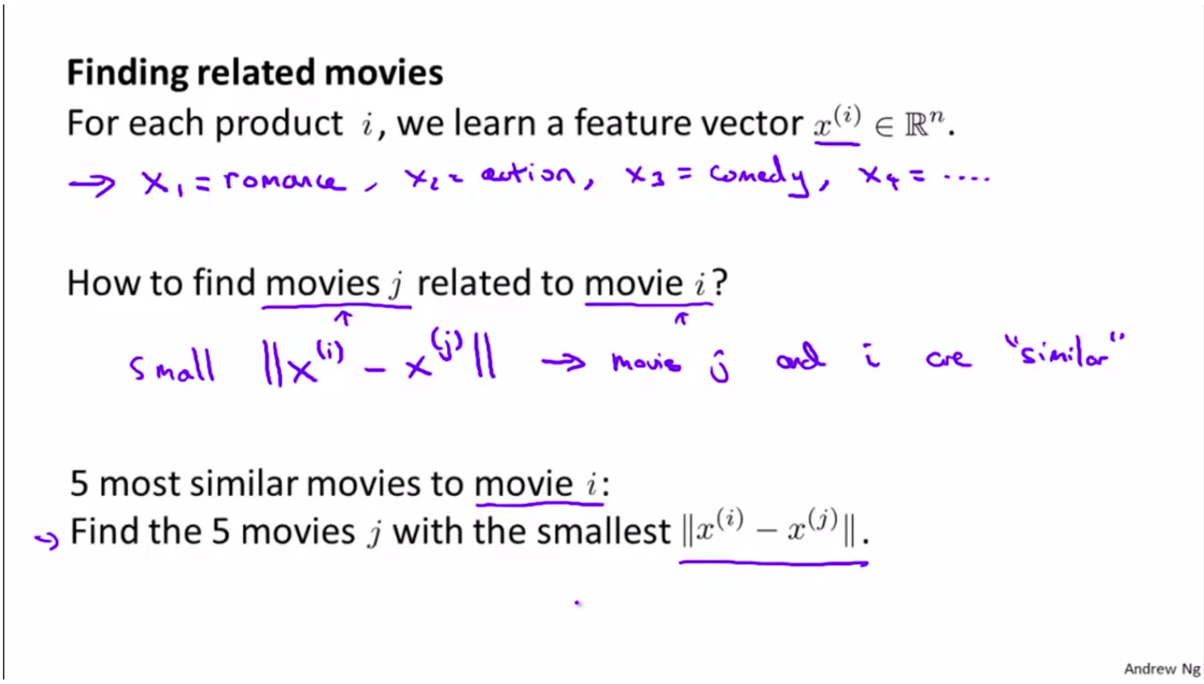
$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 \\ 5 & ? & ? & 0 \\ ? & 4 & 0 & ? \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 5 & 0 \end{bmatrix} \quad \underbrace{\begin{bmatrix} \boxed{(\theta^{(1)})^T(x^{(1)})} & \boxed{(\theta^{(2)})^T(x^{(1)})} & \dots & (\theta^{(n_u)})^T(x^{(1)}) \\ (\theta^{(1)})^T(x^{(2)}) & (\theta^{(2)})^T(x^{(2)}) & \dots & (\theta^{(n_u)})^T(x^{(2)}) \\ \vdots & \vdots & \vdots & \vdots \\ \boxed{(\theta^{(1)})^T(x^{(n_m)})} & (\theta^{(2)})^T(x^{(n_m)}) & \dots & (\theta^{(n_u)})^T(x^{(n_m)}) \end{bmatrix}}$$

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

Low rank matrix factorization

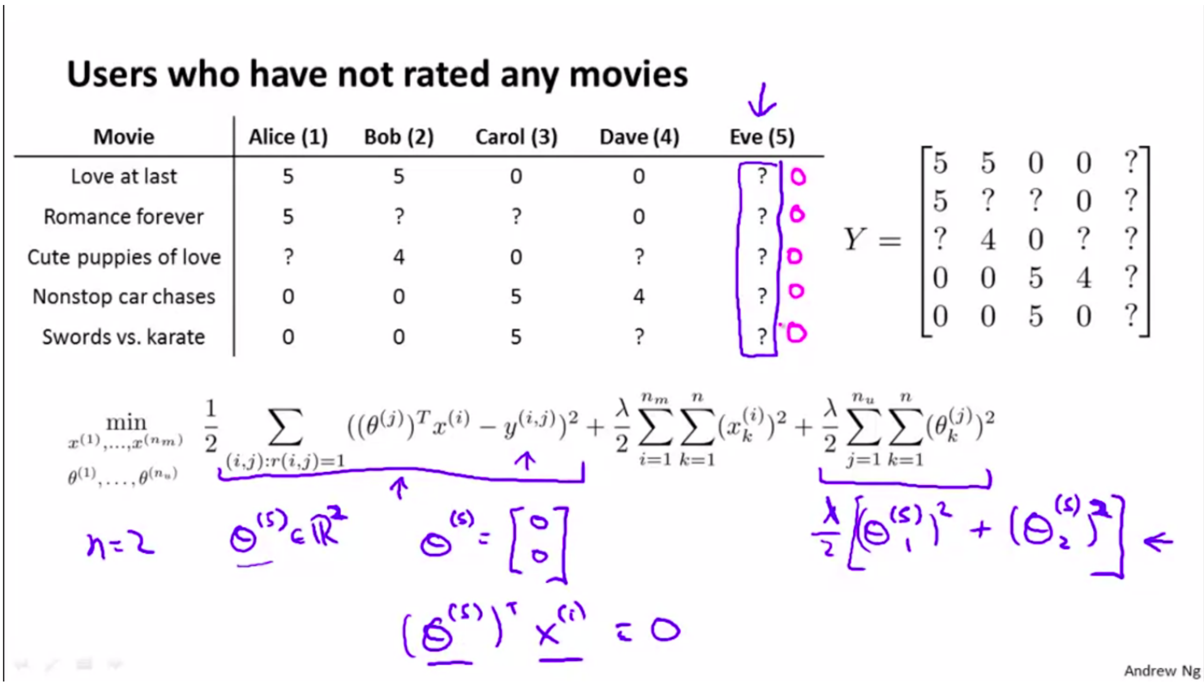
Andrew Ng

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

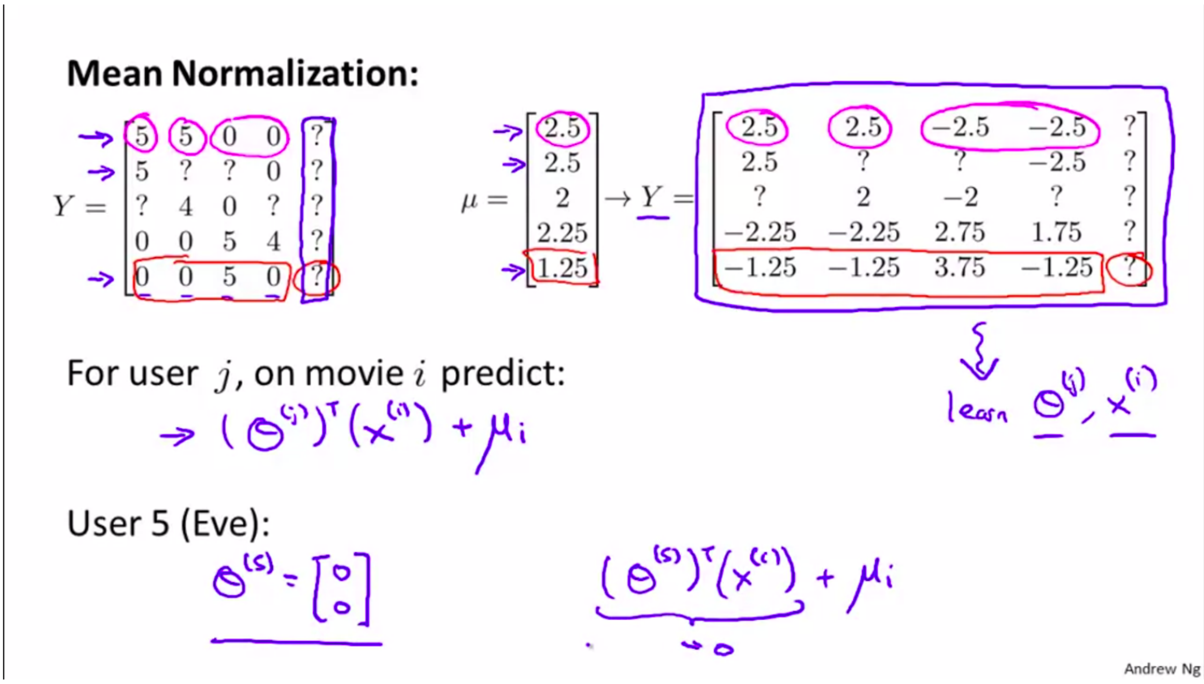


Implementational Detail: Mean Normalization

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).



How to deal with this?



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)