Recommender Systems

Predicting movie ratings, collaborative filtering, and low rank matrix factorization.

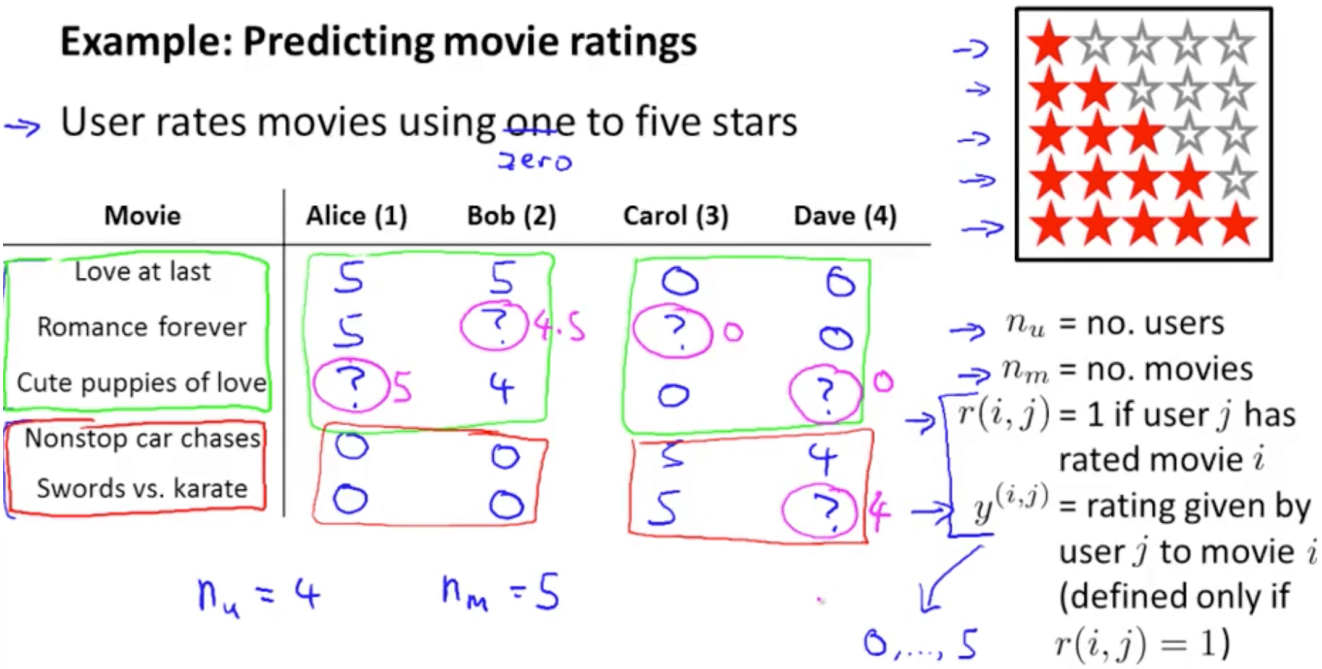
- 1. Predicting Movie Ratings
    - 1a. Problem Formulation
    - 1b. Content Based Recommendations
- 2. Collaborative Filtering
    - 2a. Introduction
    - 2b. Collaborative Filtering Algorithm
- 3. Low Rank Matrix Factorization

- 3a. Vectorization: Low Rank Matrix Factorization
- 3b. Implementation Detail: Mean Normalization

This is an important practical application of machine learning. Netflix, Spotify, Youtube, Amazon and other companies try to recommend things to you every time you use their services.

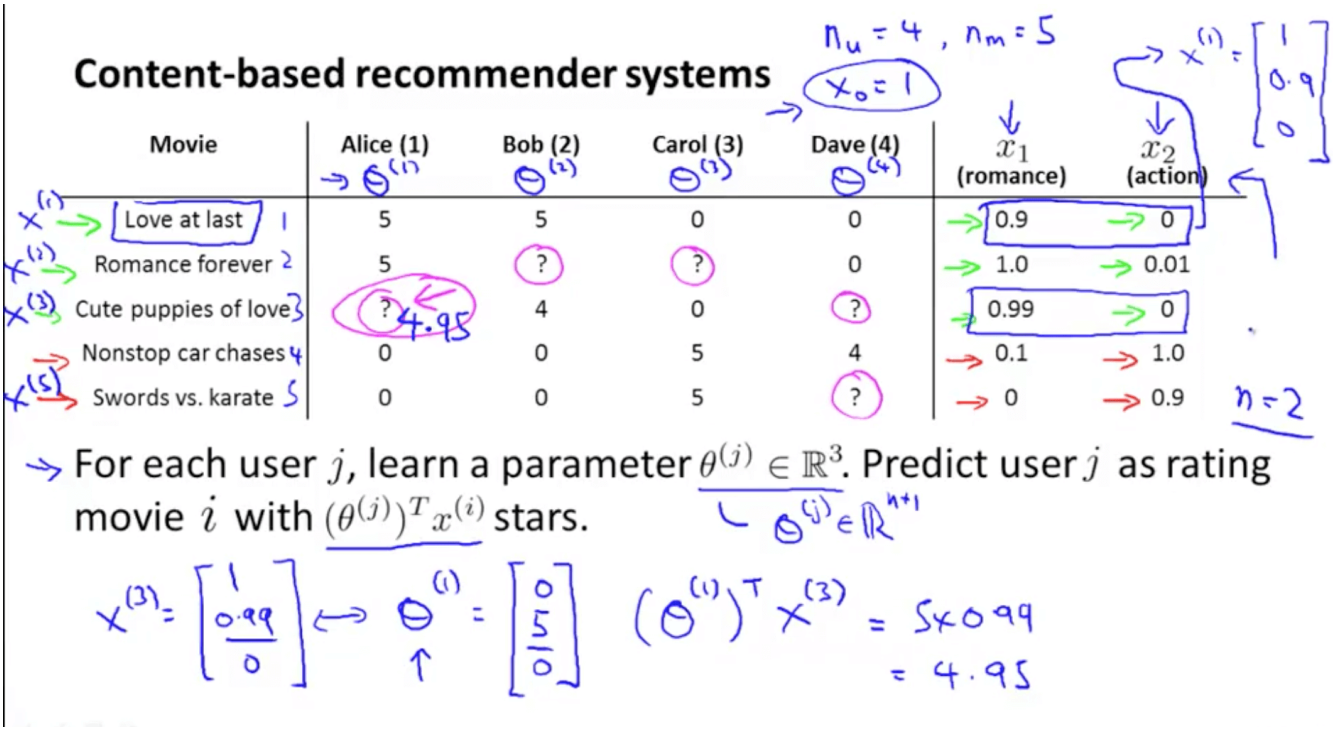
1. Predicting Movie Ratings

I would like to give full credits to the respective authors as these are my personal python notebooks taken from deep learning courses from Andrew Ng, Data School and Udemy :) This is a simple python notebook hosted generously through Github Pages that is on my main personal notes repository on https://github.com/ritchieng/ritchieng.github.io. They are meant for my personal review but I have open-source my repository of personal notes as a lot of people found it useful.



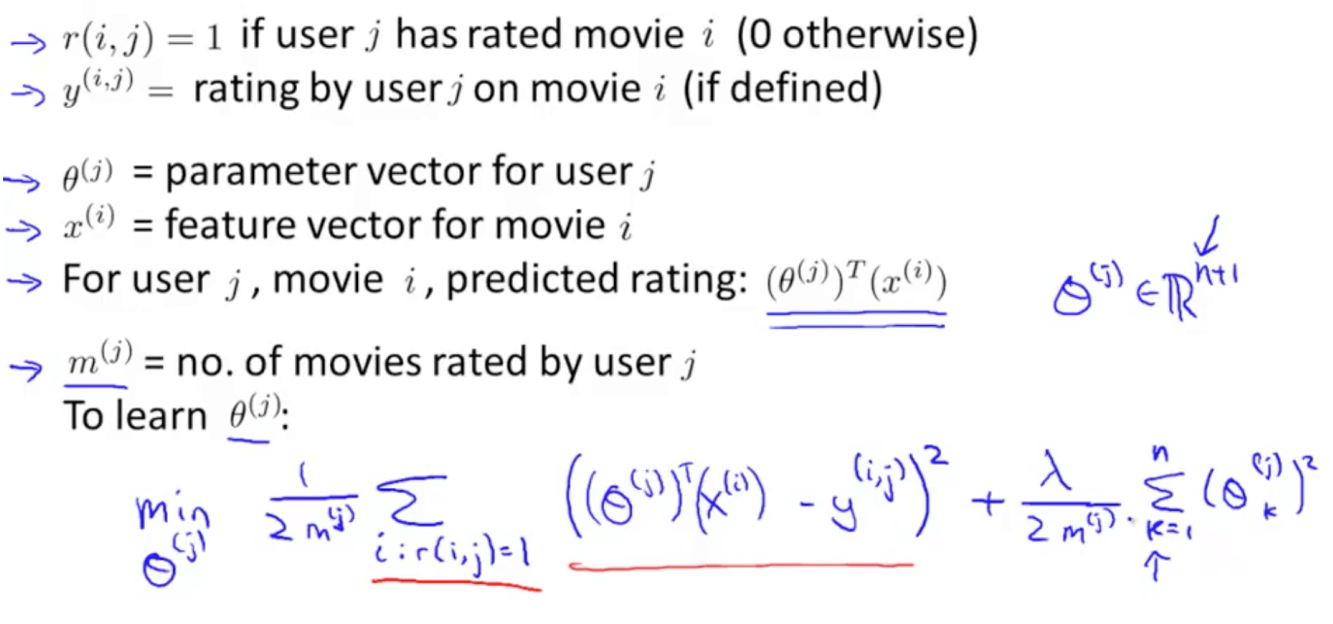
1b. Content Based Recommendations

- How do we predict the missing values?



Problem formulation

- If we minimize the following function, we get the parameters to predict



We can use other minimization algorithms (other than gradient descent)

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

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

Gradient descent update:

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

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

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

2. Collaborative Filtering

2a. Introduction

- Here we will be learning about “Feature Learning”
  - Feature Learning: learning what features to use
- Problem motivation
  - It is inefficient and difficult to ask someone to watch each movie and inform us how romantic or action-packed the movie is

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	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	?	?	?

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

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

$\theta^{(j)}$

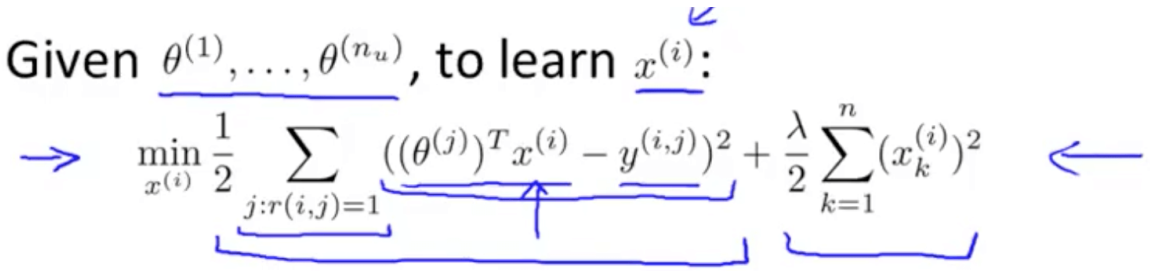
$(\theta^{(1)})^T x^{(1)} \approx 5$

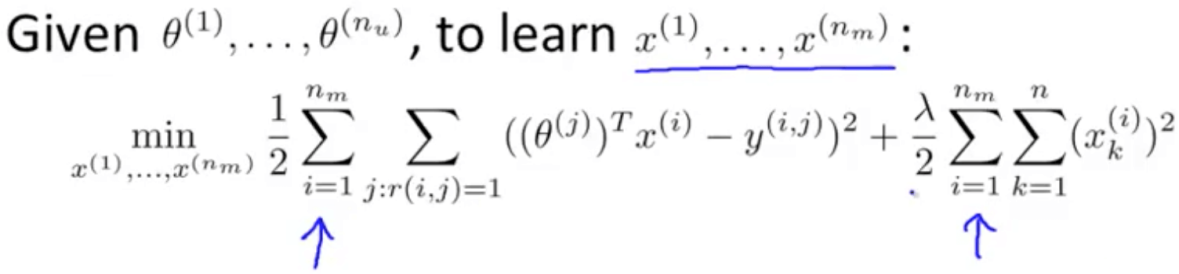
$(\theta^{(2)})^T x^{(1)} \approx 5$

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

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

- - Now we've no idea how each movie is romantic (x1) or action-packed (x2)
    - Let's say that
        - Alice (θ1): likes romance
        - Bob (θ2): likes romance
        - Carol (θ3): likes action
        - Dave (θ4): likes action
    - We can discover x_1 by making sure the following happens
        - θ1_transpose * x_1 = 5
        - θ2_transpose * x_1 = 5
        - θ3_transpose * x_1 = 0
        - θ4_transpose * x_1 = 0
- Optimization algorithm
    - This tries to choose features X_i so that for all the users J that have rated that movie, the algorithm also predicts a value for how that user would have rated that movie that is not too far, in the squared error sense, from the actual value that the user had rated that movie
    - Regularization term to prevent features from becoming too big





We can guess θ, solve for x, then solve for θ and continue

- There is a more efficient method to do this and we will be discussing this shortly

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

Guess $\theta \rightarrow x \rightarrow \theta \rightarrow x \rightarrow \theta \rightarrow x \rightarrow \cdots$

2b. Collaborative Filtering Algorithm

- Collaborative filtering optimization objective
  - We will take both of these optimization objectives and put them together
  - Now we get to minimize with respect to x and θ simultaneously

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

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

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

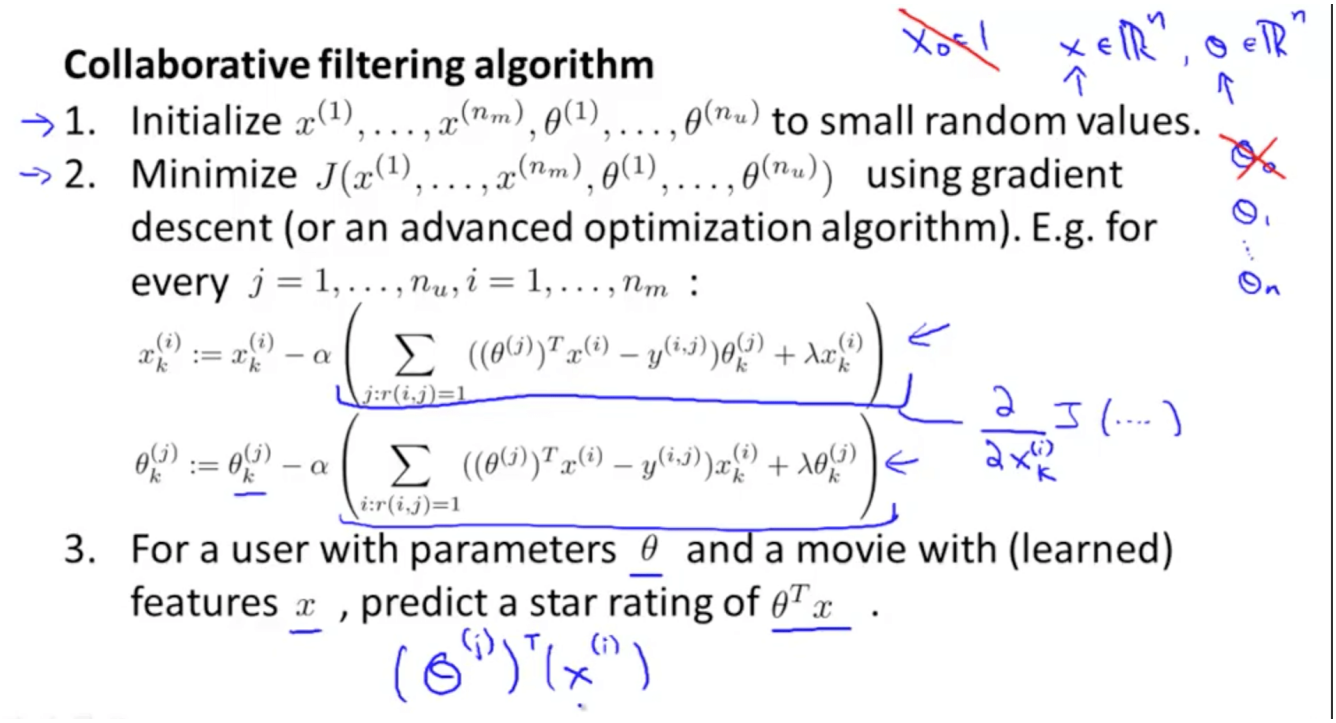
$$\min_{x^{(1)},\ldots,x^{(n_m)}} \frac{1}{2}\sum_{i=1}^{n_m}\sum_{j:r(i,j)=1}((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}(x_k^{(i)})^2$$

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

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

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

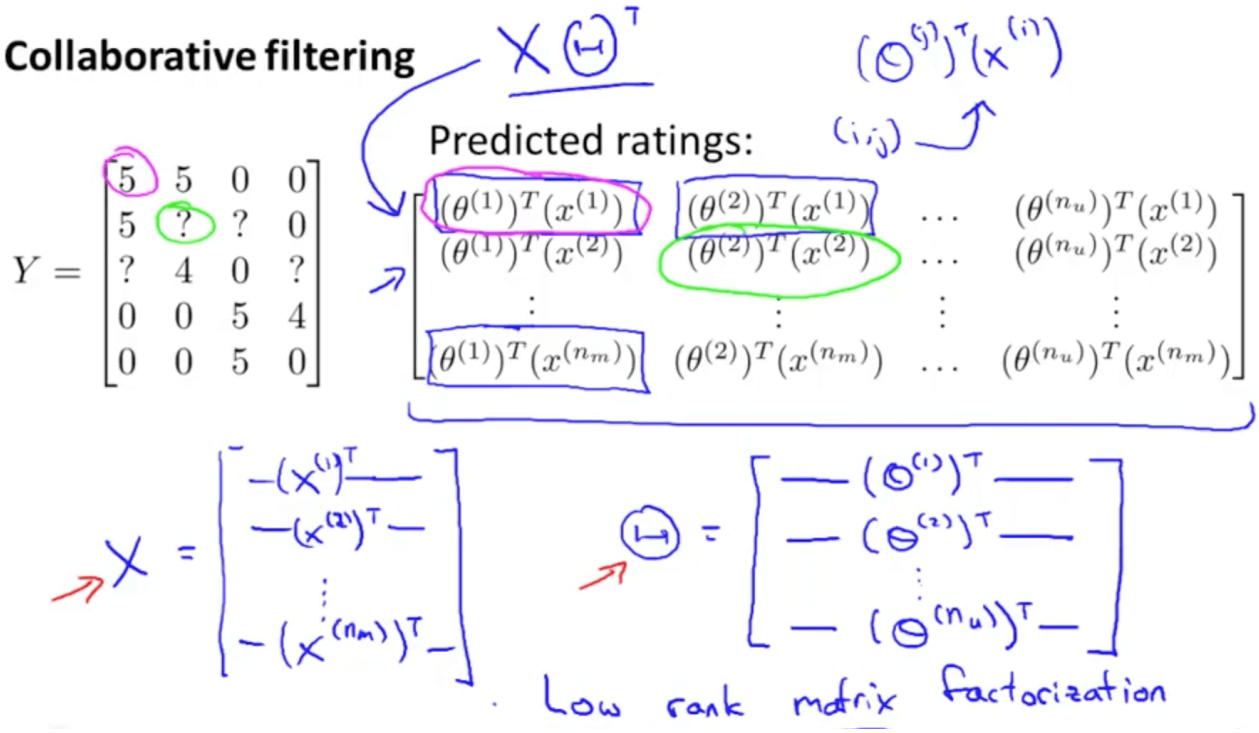
Collaborative filtering algorithm



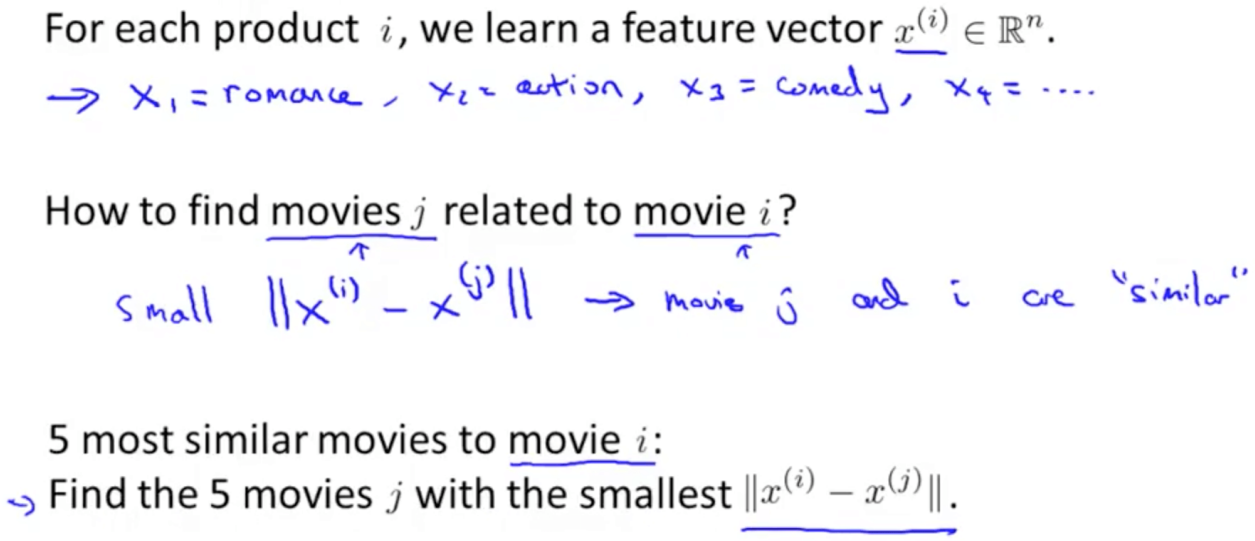
3. Low Rank Matrix Factorization

3a. Vectorization: Low Rank Matrix Factorization

- Low rank matrix factorization
  - Y matrix: all the predicted ratings
  - We conduct matrix factorization by decomposing the matrix into a product of matrices
    - X matrix: features of each movie stacked in rows
    - H matrix: parameters of each user stacked in rows
    - Product of X and H_transpose equals Y matrix



Finding related movies



3b. Implementation Detail: Mean Normalization

- Issue without mean normalization
  - And for even the Swords vs. Karate, someone rated it 5 stars
  - So some people do like some movies
  - It seems not useful to just predict that Eve is going to rate everything 0 stars
  - If we're predicting that eve is going to rate everything 0 stars, we also don't have any good way of recommending any movies to her, because you know all of these movies are getting exactly the same predicted rating for Eve so there's no one movie with a higher predicted rating that we could recommend to her

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	?	?

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

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

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

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

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

We can conduct mean normalization to solve this issue

- Our prediction would be the average of each movie

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix} \leftarrow \begin{matrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{matrix} = \mu$$

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

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

For user j, on movie i predict:

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

User 5 (Eve):

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

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