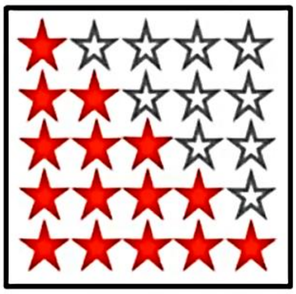
16. Recommender Systems

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

$n_u = 4$ $n_m = 5$

→ 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 $r(i,j) = 1$)

$0, \ldots, 5$

The movie prediction system simply try to predict the values of these missing values (with "?") .. like what rating would the user have given to those films

---

CONTENT BASED RECOMMENDER SYSTEM:

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 1	5	5	0	0	0.9	0
$x^{(2)}$ → Romance forever 2	5	?	?	0	1.0	0.01
→ Cute puppies of love 3	?	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

x_1 = degree to which a movie is a romantic movie

x_2 = degree to which a movie is an action movie

n =2 = no of features, not counting the extra feature(x_0→always =1)

$X^{(1)}$ = feature values vector for movie 1

$\Theta^{(1)}$ = parameters vector learned for a particular user

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.

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

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

Procedure:

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: $(\theta^{(j)})^T(x^{(i)})$ $\quad \theta^{(j)} \in \mathbb{R}^{n+1}$

- $m^{(j)}$ = no. of movies rated by user j

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

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

Minimizing the sum over all **i such that r(i, j) ==1**

Optimization objective:

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

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

But we need to do this for all users:

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

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

We get a separate parameter vector for each user, $\Theta^{(j)}$

Optimization algorithm:

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

Gradient descent update:

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

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

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

Not all movies can be rated based on their content, or it's just very difficult → so, **collaborative learning** comes to our rescue

---

COLLABORATIVE FILTERING: feature learning → algo automatically learns what features to use

If we have no idea how to rate a movie..

Problem motivation

Movie	Alice (1) $\theta^{(1)}$	Bob (2) $\theta^{(1)}$	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	?	?	?

Let’s say, **our users have told us the values of their own parameters Θ:**

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

We try to rate the movies automatically:

We try to infer what should the **vector $X^{(1)}$** be so that:

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

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

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

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

Andrew Ng

Optimization algorithm

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

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

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

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

For each movie i:

For each user j:

If j has rated that movie:

We minimize the diff b/w predicted rating (0 to 5) and its actual rating

For each movie i:

For each feature k:

We also try to minimize the values of parameters, so as to achieve regularization.

Diff b/w content based recommender vs Collaborative filtering:

In content-based recommender:

Given $\underline{x^{(1)}, \ldots, x^{(n_m)}}$ (and movie ratings),

can estimate $\underline{\theta^{(1)}, \ldots, \theta^{(n_u)}}$

In collaborative filtering:

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

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

So, what we can do to train the model is that:

We can guess an initial Θ and obtain X from that

Use that to obtain better fitting parameters (Θ)

Use that to obtain better set of features (X)

And so on …

Guess $\Theta \rightarrow X \rightarrow \Theta \rightarrow X \rightarrow \Theta \rightarrow x$

This actually works, as eventually this will converge to a reasonable set of features and a reasonable set of parameters for each user

- This algo is called collaborative filtering as each user is collaborating to advance the algo a little bit, i.e., they are helping the algo little bit

---

Combining Collaborative filtering and content-based recommender:

In content-based recommender:

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

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

In collab. Featuring:

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

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

Combing both:

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

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

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

Andrew Ng

Here, if we keep **X→ constant** ➔ it's minimized over Θ (content-based recommender)

If we keep Θ → constant ➔ it's minimized over X (collaborative filtering)

Here: we **don't need x0 =1** and also **don't need Θ0 →** because since our algo is learning its own features, we don't need to hardcode any features, **it will learn x1==1 by itself** (since we don't give x0)

$\cancel{x_0 = 1}$ $\quad x \in \mathbb{R}^n \quad \cancel{x \in \mathbb{R}^{n+1}}$

$\cancel{\theta_0}$ $\quad \theta \in \mathbb{R}^n$

Procedure:

Collaborative filtering algorithm

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

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

$\cancel{\theta_0}$, θ_1, …, θ_n

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

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

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

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

VECTORIZATION: LOW RANK FACTORIZATION ALGORITHM:

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	?

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

$y^{(i,j)}$

Collaborative filtering

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

(i,j)

Predicted ratings:

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

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

We can create an X vector which contains features of all examples row-wise

And we can also create a Θ vector which contains parameters of all examples row-wise

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

So, we can obtain the predicted rating matrix as:

$X\Theta^T$ **This is a low rank matrix** (→ a property of linear algebra)

Application:

Finding related movies

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

→ 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 $\|x^{(i)} - x^{(j)}\|$.

MEAN NORMALIZATION:

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	?	?

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

$$\min_{\substack{x^{(1)},\ldots,x^{(n_m)} \\ \theta^{(1)},\ldots,\theta^{(n_u)}}} \frac{1}{2} \sum_{(i,j):r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} (x_k^{(i)})^2 + \underbrace{\frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2}_{\frac{\lambda}{2}\left[(\Theta_1^{(5)})^2 + (\Theta_2^{(5)})^2\right]}$$

$n = 2 \quad \Theta^{(5)} \in \mathbb{R}^2$

In such case, the last regularization term will give a 0 vector for **Θ**$^{(5)}$ (parameter vector for Eve). This will cause all predicted ratings of Eve to be 0.

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

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

But that's not useful, as all the movies will be equally likely for Eve

Mean Normalization: it can be used as a data pre-processing step:

First, we find the mean rating of each movie by different users

Mean Normalization:

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

Then we Subtract the mean from each user's ratings

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

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

Now, we use this matrix to learn our parameters for each user and feature-values of each movie.

Also, we add the calculated mean to our predicted ratings at the end of prediction

For user j, on movie i predict:

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

This way we have:

User 5 (Eve):

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

So, the user who has not rated any movie will get an average rating for each movie.

- Mean normalization can also be used in case when a movie has no rating from any user, by modifying the algo a little bit.