Recommender Systems

Consider the example of movie ratings, where users can rate from 1 to 5. For simplicity sake, we can allow rating from 0 to 5. We will have the following notations:

- n_u = Number of users
- n_m = Number of 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 $r(i,j) = 1$

Let's take the following example:

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	Romance (x_1)	Action (x_2)
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
Non-stop Car Chases	0	0	5	4	0.1	1.0
Swords vs Karate	0	0	5	?	0	0.9

Here, $n_u = 4$ and $n_m = 5$

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.

So we would have feature vectors like:

$$x^{(1)} = \begin{bmatrix} 1 \\ 0.9 \\ 0 \end{bmatrix} \qquad x^{(2)} = \begin{bmatrix} 1 \\ 1 \\ 0.01 \end{bmatrix} \qquad \ldots \text{and so on}$$

For reach user j, learn a parameter $\theta^{(j)} \in \mathbb{R}^3$ as $\theta^{(j)} \in \mathbb{R}^{n+1}$. Predict user j as rating movie i with $(\theta^{(j)})^T x^{(i)}$ stars.

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: $(\theta^{(j)})^T (x^{(i)})$
- $m^{(j)} =$ number of movies rated by user j

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

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

Optimization Objective

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

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

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

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

Gradient descent update:

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

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

Collaborative Filtering

Suppose we have a list of movies but we don't know the metrics

Movie	Alice ($\theta^{(1)}$)	Bob ($\theta^{(2)}$)	Carol ($\theta^{(3)}$)	Dave ($\theta^{(4)}$)	Romance (x_1)	Action (x_2)
Love at Last	5	5	0	0	?	?
Romance Forever	5	?	?	0	?	?
Cute Puppies of Love	?	4	0	?	?	?
Non-stop Car Chases	0	0	5	4	?	?
Swords vs Karate	0	0	5	?	?	?

Now suppose the audience has provided us with their ratings on how much they love romantic movies and action movies:

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

From these values, it's possible to infer the values of x_1 and x_2.

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 $x^{(i)}, \ldots, x^{(n_m)}$:

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

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

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

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

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

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

Summary:

So we can do either of these:

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

Collaborative filtering algorithm:

1. Initialize $x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_m)}$ to small random values
2. Minimize $J(x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_m)})$ using gradient descent or advanced optimization algorithm
3. For an user with parameters θ and a movie with (learned) features x, predict a star rating of $\theta^T x$

Vectorization: Low Rank Matrix Factorization

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	?
Non-stop Car Chases	0	0	5	4
Swords vs Karate	0	0	5	?

Getting the values into a matrix:

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

Predicted Ratings:

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

Finding related movies:

- For each product i, we learn a feature vector $x^{(i)} \in \mathbb{R}^n$ (x_1 = romance, x_2 = action, etc)
- 5 most similar movies to movie i: find 5 movies j with the smallest $||x^{(i)} - x^{(j)}||$

Mean Normalization

Let's suppose we have a new user Eve(5) for whom the values are unknown. We'd have the matrix as:

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

Computing the average rating:

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

Use the above calculated matrix to learn $\theta^{(j)}, x^{(i)}$

For user j, on movie i predict: $(\theta^{(j)})^T(x^{(i)}) + \mu_i$

For user Eve(5):

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