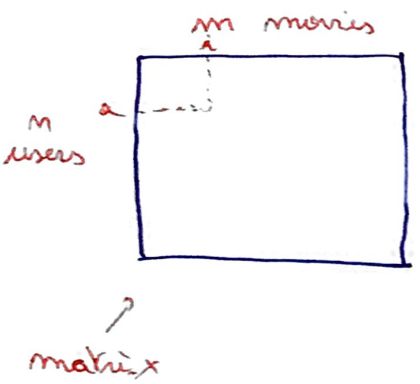
Unit 2 - Lecture 7: Recommender Systems

Objectives:

- Understand the problem definition and assumptions of Recommender Systems
- Understand the impact of similarity measures in the K-Nearest Neighbor method
- Understand the need to impose the low rank assumption in collaborative filtering
- iteratively find values of U and V (given $X = UV^T$) in collaborative filtering

Introduction: Netflix



Y_{ai}

Netflix challenge : 0.5 million users = n

18 000 movies = m

The interesting part about it is that most of this matrix is actually empty. (only ranked by 5000 users)

We want to predict what will be the opinion of this user for the movie he or she hasn't seen.

We don't want to use the regression here for many reasons:

1. not clear what features are we going to record, and it will be really huge feature vectors.
2. In order for us to do linear regression for a specific user, we actually need to have enough ranking of this user. So if we just have a few rankings, we would be unable to recommend this user any new products.

K-Nearest Neighbor Method:

K means, how big should be your advisory pool or how many neighbors you want to look at

predicted → $$\hat{Y}_{ai} = \frac{\sum_{b \in KNN(a,i)} Y_{bi}}{K}$$

I'm going to select all the nearest neighbors who did watches this movie and then divide by K

We will identify users similar to me, take their values and take the average.

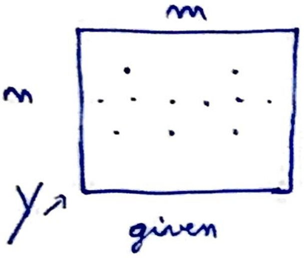
we assume we have similarity

$$\hat{Y}_{ai} = \frac{\sum_{b \in KNN(a,i)} sim(a,b)\, Y_{bi}}{\sum_{b \in KNN(a,i)} |sim(a,b)|}$$

similarity

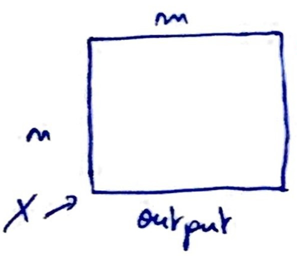
We take the score of KNN and compare the similarity between a and b and use it as a weighting factor.

Collaborative Filtering: The Naive approach



← Sparse matrix: very few entries

Our goal is to build another matrix X, same size.



← I need every single entry full.

The Algorithm is given Y and will output X, which will contain prediction for every single user and movie.

$$\underbrace{J(X)}_{\text{objective}} = \underbrace{\sum_{(a,i)\in D} (Y_{ai} - X_{ai})^2/2}_{\text{Loss}} + \underbrace{\frac{\lambda}{2} \sum_{(a,i)} X_{ai}^2}_{\text{Regularization}}$$

→ $\|X\|$

For each predicted matrix we need to see how good is this matrix

$$D = \{(a,i) \mid Y_{ai} \text{ is given}\}$$

1- $(a,i) \in D$

$$\frac{\partial J(X_{ai})}{\partial X_{ai}} = \frac{\partial\left((Y_{ai}-X_{ai})^2/2 + \frac{\lambda}{2} X_{ai}^2 \right)}{\partial X_{ai}} = 0 \Longrightarrow X_{ai} = \frac{Y_{ai}}{1+\lambda}$$

2. $(a,i) \notin D$

$$\frac{\partial J(X_{ai})}{\partial X_{ai}} = \frac{\partial \frac{\lambda}{2} X_{ai}^2}{\partial X_{ai}} = 0 \Longrightarrow X_{ai} = 0$$

When we did our estimation, we said for all the entries that you didn't know what the value should be, you just put zeros everywhere. And for those that you knew what the value should be you actually corrupt the value.

There is something wrong in this particular way of thinking.

We are taking every single user, compute the derivative and that will cost a lot of time. and the problem with this approach is that there is no connection between assigned values for all different entries of X.

We don't discover any groupings between either users or movies.

Collaborative Filtering with Matrix Factorization:

What we will do now is to somehow tie all these parameters, not to do independent estimation. and we want to decrease the number of parameters.

Assumption: X is low Rank (Rank captures how much dependency do you see between the entries of the matrix)

Rank 1: $$\begin{array}{c} \\ n \end{array}\underbrace{\overset{m}{\begin{bmatrix}1 & 2 & 3\\ 5 & 10 & 15\end{bmatrix}}}_{O(n\times m)} = \underbrace{\begin{bmatrix}1\\5\end{bmatrix}\begin{bmatrix}1 & 2 & 3\end{bmatrix}}_{O(n+m)} = \begin{bmatrix}10\\50\end{bmatrix}\begin{bmatrix}0.1 & 0.2 & 0.3\end{bmatrix}$$

← parameters.

Why is it interesting for us to factorize? To decrease the number of parameters

$$\underbrace{\begin{bmatrix}1\\5\end{bmatrix}}_{u}\underbrace{\begin{bmatrix}1 & 2 & 3\end{bmatrix}}_{v} = u \cdot v^T \qquad X_{ai} = u_a \cdot v_i$$

The Higher you make the Rank ⇒ The more multifaceted representation of the user and the movie you can have.

Alternating Minimization:

$$J(X) = \sum_{(a,i)\in D} (Y_{ai} - X_{ai})^2/2 + \frac{\lambda}{2}\sum_{(a,i)} X_{ai}^2 \quad \text{and} \quad X_{ai} = u_a v_i \text{ ; rank 1}$$

↓

$$J(u,v) = \sum_{(a,i)\in D} (Y_{ai} - u_a v_i)^2 + \frac{\lambda}{2}\sum_{a=1}^{n} u_a^2 + \frac{\lambda}{2}\sum_{i=1}^{m} v_i^2$$

Assume:

$$Y = \Big(\overbrace{\begin{bmatrix}5 & ? & 7\\ 1 & 2 & ?\end{bmatrix}}^{3 \text{ movies}} \qquad u = \begin{bmatrix}u_1\\u_2\end{bmatrix} \qquad v = \begin{bmatrix}v_1\\v_2\\v_3\end{bmatrix}$$

2 users

① Initialization: $v = \begin{bmatrix}2\\7\\8\end{bmatrix}$

$$u v^T = \begin{bmatrix}u_1\\u_2\end{bmatrix}\begin{bmatrix}2 & 7 & 8\end{bmatrix} = \begin{bmatrix}2u_1 & 7u_1 & 8u_1\\ 2u_2 & 7u_2 & 8u_2\end{bmatrix}$$

For user 1: u_1 $\quad J(u,v) = ?$

$$\frac{\partial}{\partial u_1}\left[\frac{(5-2u_1)^2}{2} + \frac{(7-8u_1)^2}{2} + \frac{\lambda}{2}u_1^2\right] = -66 + (68+\lambda)u_1 = 0 \Rightarrow u_1 = \frac{66}{\lambda+68}$$

$$u_2 = \frac{16}{\lambda+53}$$

if $\lambda = 1 \Rightarrow u_1 = \frac{66}{69}$ and $u_2 = \frac{16}{54}$

② We will take theses us and recompute the vs

$$u = \begin{bmatrix} \frac{66}{69} \\ \frac{16}{54} \end{bmatrix} \text{ and } V = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

$$u \cdot v^T = \begin{bmatrix} \frac{66}{69} \\ \frac{16}{54} \end{bmatrix} \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} = \begin{bmatrix} \frac{66}{69} v_1 & \frac{66}{69} v_2 & \frac{66}{69} v_3 \\ \frac{16}{54} v_1 & \frac{16}{54} v_2 & \frac{16}{54} v_3 \end{bmatrix}$$

Fixing V and Finding U:

2 users, 3 movies

$$Y = \begin{bmatrix} 1 & 8 & ? \\ 2 & ? & 5 \end{bmatrix}$$ The goal is to find U and V such that $X = UV^T$

We initialize $V = [4 \ 2 \ 1]^T$

missing: We ignore the missing values in the expression.

$$X = uv^T = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \begin{bmatrix} 4 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 4u_1 & 2u_1 & u_1 \\ 4u_2 & 2u_2 & u_2 \end{bmatrix}$$

missing

$$\frac{\partial}{\partial u_1} \left[\frac{(1-4u_1)^2}{2} + \frac{(8-2u_1)^2}{2} + \frac{\lambda}{2} u_1^2 \right] = -4(1-4u_1) - 2(8-2u_1) + \lambda u_1$$

$$\Rightarrow (\lambda + 20) u_1 - 20 = 0 \Rightarrow u_1 = \frac{20}{\lambda + 20}$$

$$\frac{\partial}{\partial u_2} \left[\frac{(2-4u_2)^2}{2} + \frac{(5-u_2)^2}{2} + \frac{\lambda}{2} u_2^2 \right] = -4(2-4u_2) - (5-u_2) + \lambda u_2$$

$$\Rightarrow (\lambda + 17) u_2 - 13 = 0 \Rightarrow u_2 = \frac{13}{\lambda + 17}$$

Kernels:

$k(x,q) = (x^T q + 1)^2$ Polynomial Kernel

$k(x,q) = \phi(x)^T \phi(q)$

$\phi(x) = ?$ $\quad \phi(x) = [f_1(x_1, x_2), \dots, f_N(x_1, x_2)]$

$$k(x,q) = (x^T q + 1)^2 = \left(1 + \sum_{i=1}^{2} x_i q_i\right)^2 = (x_1 q_1 + x_2 q_2 + 1)^2$$

$$= x_1^2 q_1^2 + x_2^2 q_2^2 + 2x_1 x_2 q_1 q_2 + 2x_1 q_1 + 2x_2 q_2 + 1$$

$$\Rightarrow \phi(x) = [x_1^2, x_2^2, \sqrt{2} x_1 x_2, \sqrt{2} x_1, \sqrt{2} x_2, 1]$$