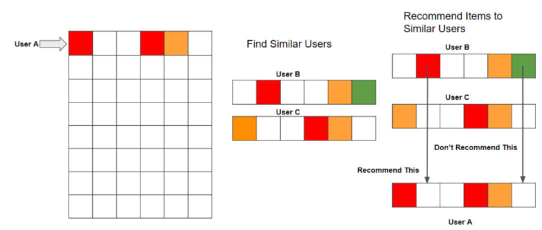
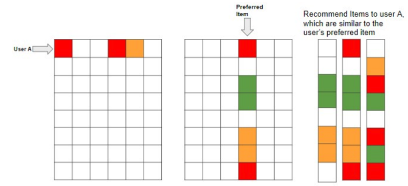
Collaborative Filtering

Description	What	Limitations/Assumptions	Example/Formula	Reference/Comments
Clustering	averaging method ussed to predict the unknown value of the user-item interaction matrix using the average of all the item ratings assuming all the users are the same. However, this method seems to be very naive, as it carries an assumption of all users being the same. One of the solutions for this problem is **clustering-based recommendation systems**	**While the clustering-based approach is better than the averaging approach, it is still weak** because what we do is identify user groups and recommend each user in this group the ssame items but the cluster might not be a good representative of the users **that are closer to the boundary of clusters.**		
How to build clusters	In clustering based recommendation systems, we can build clusters based on user-item interactions, and users within each cluster would receive recommendations by applying the averaging method within the individual clusters			
Collaborative Filtering	also known as personalized clustering is one of the most popular approaches used in recommendation systems. It helps in the providing personalized recommendations to users. In collaborative filtering, the recommendation is done based on the similarity between users or similarity between items	The set of features acquired by a user is transformed into a user vector and that of an item is transformed into an item vector. Hence for every user, there is a user vector and for every item, there is an item vector. The similarity between the user vectors or item vectors can be calculated by using various distance measurement methods like cosine-similarity, pearson coefficient, etc		
Types of collaborative Filtering	1. User-User collaborative Filtering 2. Item-iterm collaborative Filtering			
User-User Collaborative Filtering	is a technique userd to predict the items that a user might like based on the ratings given to items by other users who have similar tastes to the target user			
Item-Item Collaborative Filtering	is a technique used to predict the items that a user likes based on finding the similarities between items that the user had rated and the target item			

Singular Value Decomposition (SVD)

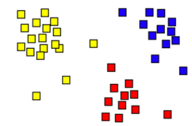
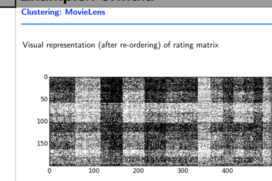
<table>
<tr><th>Description</th><th>What</th><th>Limitations/Assumptions</th><th>Example/Formula</th><th>Reference/Comments</th></tr>
<tr><td>Limitations of Collaborative Filtering</td><td>With a highly sparse user-item interaction matrix, where the majority of entries are missing or 0, the collaborative filtering strategy might produce unsatisfactory results</td><td></td><td></td><td></td></tr>
<tr><td>Single Value Decomposition</td><td>There is another method to perform collaborative filtering using matrix factorization where we can identify the relationship between items and users for sparse matrics as well. With the input of user's ratings on the items, we can predict how the users would rate the items. This way the user can get the recommendation based on the ratings.
The idea behind matrix factorization is to represent users and items in a lower-dimensional space and extract hidden features from the data which are constructed by some hidden relations. These hidden features called Latent features.
The latent features cannot be observed but can be extracted using matrix factorization algorithms. One of the most common and useful
matrix factorization algorithms is Singular Value Decomposition.</td><td>Usually , the user-item interaction matrix has many missing values, so the purpose of SVD is to estimate the matrix values by filling null values with 0.</td><td></td><td></td></tr>
<tr><td>SVD Matrix Decomposition</td><td>In SVD, we decompose a matrix into 3 small matrics that are relevant to the original matrix.
These matrics can be used to reconstruct the original matrix.
SVD decomposes the original matrix X into the following form X=US*V (exp of T)</td><td>X--> The original user-item interaction matrix with size m x n
U--> Matrix of Latent Features for Users with size m x r
V (exp of T)--> Matrix of Latent Features for Items with size r x n
S --> It is a diagonal matrix of single values with size r x r

The shape of U and V must be m x r. and r x n, respectively, because for matrix multiplication the number of columns of the first matrix must be equal to the number of rows of the second matrix

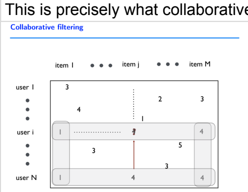
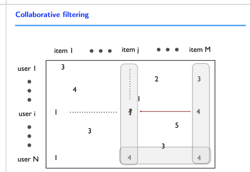
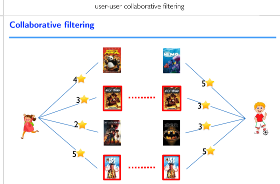
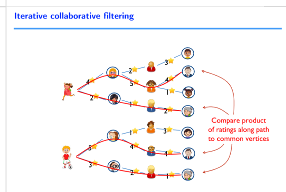
In reality, we have a large number of users and items, which makes the latent features non-interpretable</td><td></td><td></td></tr>
<tr><td>SVD Representation</td><td>Now, let's consider an example of movies rated by users. Suppose the user-item interaction matrix looks like the below figure:
<table>
<tr><th>USERS</th><th>Star Trek</th><th>Alien</th><th>Serenity</th><th>Titanic</th><th>Amelie</th></tr>
<tr><td>USER 1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>USER 2</td><td>3</td><td>3</td><td>3</td><td>0</td><td>0</td></tr>
<tr><td>USER 3</td><td>4</td><td>4</td><td>4</td><td>0</td><td>0</td></tr>
<tr><td>USER 4</td><td>5</td><td>5</td><td>5</td><td>0</td><td>0</td></tr>
<tr><td>USER 5</td><td>0</td><td>2</td><td>0</td><td>4</td><td>4</td></tr>
<tr><td>USER 6</td><td>0</td><td>0</td><td>0</td><td>5</td><td>5</td></tr>
<tr><td>USER 7</td><td>0</td><td>1</td><td>0</td><td>2</td><td>2</td></tr>
</table></td><td>Suppose, we got the following matrix U of latent features for the users:
<table>
<tr><th>USERS</th><th>F1</th><th>F2</th><th>F3</th></tr>
<tr><td>USER 1</td><td>0.13</td><td>-0.02</td><td>-0.01</td></tr>
<tr><td>USER 2</td><td>0.41</td><td>-0.07</td><td>-0.03</td></tr>
<tr><td>USER 3</td><td>0.55</td><td>-0.09</td><td>-0.04</td></tr>
<tr><td>USER 4</td><td>0.68</td><td>-0.11</td><td>-0.05</td></tr>
<tr><td>USER 5</td><td>0.15</td><td>0.59</td><td>0.65</td></tr>
<tr><td>USER 6</td><td>0.07</td><td>0.73</td><td>-0.67</td></tr>
<tr><td>USER 7</td><td>0.07</td><td>0.29</td><td>0.32</td></tr>
</table></td><td>And similarly, suppose, we got the following matrix V^T for latent features of movies:
<table>
<tr><th></th><th>Star Trek</th><th>Alien</th><th>Serenity</th><th>Titanic</th><th>Amelie</th></tr>
<tr><td>F1</td><td>0.56</td><td>0.59</td><td>0.56</td><td>0.09</td><td>0.09</td></tr>
<tr><td>F2</td><td>-0.12</td><td>0.02</td><td>-0.12</td><td>0.69</td><td>0.69</td></tr>
<tr><td>F3</td><td>0.40</td><td>-0.80</td><td>0.40</td><td>0.09</td><td>0.09</td></tr>
</table></td><td>And the sigma matrix S is given as,
<table>
<tr><td>12.4</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>9.5</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1.3</td></tr>
</table></td></tr>
<tr><td>Truncated SVD</td><td>produces a factorization where the number of columns can be specified for truncations.
For example, given an n x n matrix, truncated SVD generates the matrics with the specified number of columns, where SVD outputs n
columns of matrices. The truncated SVD better works on the sparse matrices for feature output.</td><td>$$\widehat{L}_{ij} = \frac{1}{\hat{p}} \sum_{k=1}^{r} s_k u_{ik} v_{jk}, \quad \text{for all } i, j$$
where $\hat{p}$ is fraction of observed entries

Matrix $\hat{L}$ is the estimation of the original matrix L.</td><td></td><td></td></tr>
</table>

Clustering

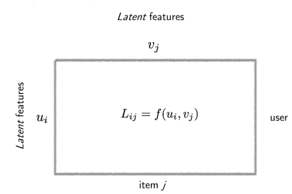
Description	What	Limitations/Assumptions	Example/Formula	Reference/Comments
What is a cluster	is a **collection** of **observations** that are **more similar** to **each other** than the **rest**	 An example of Three clusters	 top 200 users x top 500 items	
Why interested in clusters of users and items	To estimate L_{ij}, we compute average over **all rows**, **columns**. This assumes **all users** (or **items**) being **homogeneous** Most likely that may not be true However, it may be that users (and items) form multiple homogeneous enough **groups** or **clusters** Then **find** these **clusters** and **restrict averaging** method to the **cluster** in which **user/item** of **interest** belongs			
How to cluster users (or items) ?	1. Compute similarity between each pair of N users (how?) This gives N x N similarity matrix (that is symmetric) 2. Obtain representation of each users in low-dim space (How?) This assigns co-ordinates to each user in d dim space 3. Perform K-means clusterings (How>) Iteratively find K clusters till they make sense	Excercise: 1. Apply clustering algorithm for Yelp Data 2. Compare the estimation error with respect to global averaging 3. Did it work better?		

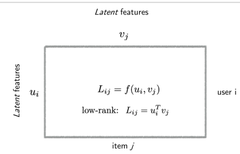
Collaborative filtering

Clustering Types	Aggregrate Clustering Personalized Clustering			
Aggregrate Clustering	leads to many user (or item) being closer to 'boundary' of clusters that is the cluster utilized for its estimation may not be representative enough			
Personalized Clustering	for each user (or item), find other users that are very similar to it declare them as it's personal cluster use the average over such a cluster to produce the estimate This is precisely what collaborative filtering attempts to do			
	 user-user collaborative filtering	 item-item collaborative filtering		
	 Computing similarity requires common observations (Birthday Paradox!) What if observations are too sparse?	 Thy Friend is My Friend!		
Iterative Collaborative filtering - Summary	What if there are very few neighbors? Or they have not yet experienced the item of interest? To overcome such sparse data challenge Find users similar to user of interest Next find users similar to these users And continue iterating this procedure to find more simiar users till enough similar users are found Use the experiences of such users to obtain the estimate			

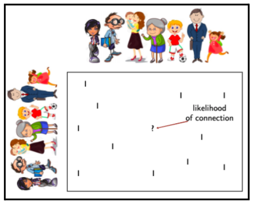
Collaborative Filtering	**Extensively used in practice** Scalable implementation using "**approximate nearest neighbors**" Closely related to non-parameteric nearest neighbor method Incremental and hence robust Interpretable: You are being **recommended GoodFellas because you liked Godfather** And, those who **liked Godfather also liked GoodFellas**	Excercise: Apply collaborative filtering algorithm for MovieLens Data Compare the estimation error of User-user collaborative filtering Item-item collaborative filtering Iterative collaborative filtering	

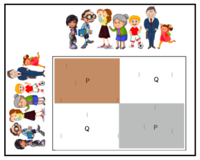
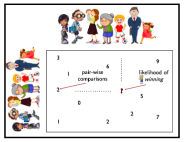
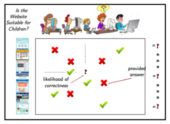
Single Value Thresholding

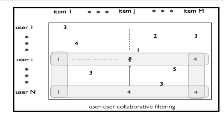
Description	What	Limitations/Assumptions	Example/Formula	Reference/Comments
Matrix Estimation: Generic Model	Prediction Problem: complete the matrix Collaborative Filtering is solving such a problem Using effectively "nearest Neighbor's" approach			
L is a matrix, Do Singular Value Decomposition	To estimate Lij for any user **i** and item **j** We need to **fill missing values** in a **matrix** Now, **any matrix obeys singular value decomposition**: rank (X) = r	$\underbrace{\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & \\ \vdots & \vdots & \ddots & \\ x_{m1} & & & x_{mn} \end{pmatrix}}_{m \times n}^{X} = \underbrace{\begin{pmatrix} u_{11} & \cdots & u_{1r} \\ \vdots & \ddots & \\ u_{m1} & & u_{mr} \end{pmatrix}}_{m \times r}^{U} \underbrace{\begin{pmatrix} s_{11} & 0 & \cdots \\ 0 & \ddots & \\ \vdots & & s_{rr} \end{pmatrix}}_{r \times r}^{S} \underbrace{\begin{pmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & \ddots & \\ v_{r1} & & v_{rn} \end{pmatrix}}_{r \times n}^{V^{\mathsf{T}}}$ An example: $\begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} = \begin{bmatrix} -1 & -1 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} -2 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ -2 & -1 \end{bmatrix}$		
How to use SVD to estimate L_{ij}	Let $L = U\Sigma V^T$ (assuming we know it) That is, $L_{ij} = \sum_{k \leq r} s_k u_{ik} v_{jk}$	Therefore, if we know U, S and V Then we can estimate all missing values And de-noise observed values Question: how do we find SVD of L		
Singular Value Thresholding	A natural algorithm: singular value thresholding 1. Fill missing values in Y by 0 [better way to fill missing values?] 2. Compute SVD 3. Truncate SVD by keeping on top r components (+ normalize)	Step 2. Compute SVD $Y_{ij} = \sum_{k=1}^{\min(M,N)} s_k u_{ik} v_{jk}, \quad \text{for all } i, j$ Step 3. Truncate SVD by keeping on top r components (+ normalize) $\widehat{L}_{ij} = \frac{1}{\hat{p}} \sum_{k=1}^{r} s_k u_{ik} v_{jk}, \quad \text{for all } i, j$ where $\hat{p}$ is fraction of observed entries [how to choose r?]		
Explore: MovieLens Data	Excercise: 1. Apply singular value thresholding algorithm for MovieLens Data 2. Compare the estimation error Across different thresholds 3. How close is it to collaborative filtering?			
Singular value decomposition: optimization perspective	Let U, V be solution of minimize $\sum_{i,j} (X_{ij} - \sum_{k=1}^{r} u_{ik} v_{jk})^2$ over $u_{ik} \in \mathbb{R},\ 1 \leq i \leq N,\ 1 \leq k \leq r$ over $v_{jk} \in \mathbb{R}, 1 \leq j \leq M,\ 1 \leq k \leq r.$	Then, U and V are (effectively) left/right singular vectors And UV(tranpsose of T) provides the best rank **r** approximate of **X**		
Singular value decomposition: optimization perspective	Find solution U, V of optimization problem	minimize $\sum_{(i,j):\text{obs}} (Y_{ij} - \sum_{k=1}^{r} u_{ik} v_{jk})^2$ over $u_{ik} \in \mathbb{R},\ 1 \leq i \leq N,\ 1 \leq k \leq r$ over $v_{jk} \in \mathbb{R}, 1 \leq j \leq M,\ 1 \leq k \leq r.$ For any i, j, produce estimate $\widehat{L}_{ij} = \sum_{k=1}^{r} u_{ik} v_{jk}$		

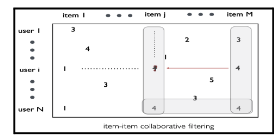
Singular value decomposition: optimization perspective	Algorithm uses only observed entries and does not require filling missing values as in for SVD The optimization problem can be solved via iteratively solving for U and V also known as **Alternating Least Squares** Food for thought: Will it converge?			
Singular value decomposition: optimization perspective	Excercise: **Singular Value Thresholding meets Alternative Least Squares** Initialize by filling missing values with 0 Singular Value Thresholding to obtain an estimate Use outcome to fill missing values and then perform Alt. Least. Sqs. Iterate What are the advantages? Use MovieLens and/or Yelp data to answer			
Matrix Estimation: Generic Model Prediction problem: complete the matrix		**Problem reduces** to **learning "factorization"** of the **matrix** either through **similarities** or **algebraic** approaches		
Matrix Estimation with Neural Networks	Singular Value Thresholding bi-linear function of latent features Generalized Singular Value Thresholding generic "activation" function of latent features multiple layers this provides neural network implementation Excercise: Compare performance with other methods			

Recommendation Systems

<table>
<tr><th>Description</th><th>What</th><th>Limitations/Assumptions</th><th>Example/Formula</th><th>Reference/Comments</th></tr>
<tr><td>Examples of Matrix Estimation - Social Networking</td><td>
</td><td>Value 1 implies connection between the corresponding people in the row and column
Value "?" (unknown) , we need to find the likelihood of a connection between them

We can think of this as an application for designing a recommendation system for recommending friend requests on Facebook and Linkedin</td><td></td><td></td></tr>
<tr><td>Examples of Matrix Estimation - Community Detection</td><td>
</td><td>community detection splits the network down into several small scale groups where more traditional recommendation approaches can be implemented
The matrix represents that the Users belonging to the same group have more similiar characteristics and typically strong ties than might normally be encountered in the rest of the network.
We need to find the density of connections in P and Q where, P and Q are the likelihood of edges between the communities and across the communities</td><td></td><td></td></tr>
<tr><td>Examples of MAtrix Estimation - Ranking players and team</td><td>$P_{ij} = P[i > j]$ (probability of winning of player i over player j)

</td><td>In the matrix, the values number of games played between the people of the corresponding row and column. Our aim in this task is to try to find the likelihood of one person winning over the other person to fill the matrix below.</td><td></td><td></td></tr>
<tr><td>Estimates of Matrix Estimation - Crowdsourcing</td><td>
</td><td>In the matrix, the symbols rated by different users represent whether a particular website is suitable for children or not. Our aim in this task is to find the likelihood of a website being suitable for children or not. For this, we try to find the likelihood of correctness (i.e the probability that the user correctly rates of an item) for each individual based on all their</td><td></td><td></td></tr>
<tr><td>Solutions of Matrix Estimation:</td><td>Clustering
Collaborative filtering (personalized clustering)
Singular Value thresholding, optimization</td><td></td><td></td><td></td></tr>
<tr><td>Clustering</td><td>Find the user-item cluster
Average within the cluster</td><td>We make clusters based on user-item interactions. We assume that within the clusters, the users and items are similiar
First we find these clusters and restrict the averaging method within the cluster in which the user/item of interest belongs</td><td></td><td></td></tr>
<tr><td>Collaborative filtering (personalized filtering)</td><td>Finding users and items similar to a given user, item
acveraging amongst user-item specific item specific similiar users, items</td><td></td><td></td><td></td></tr>
<tr><td>Singular Value thresholding, optimization</td><td>Find singular value decomposition of the matrix</td><td></td><td></td><td></td></tr>
<tr><td>How to form clusters</td><td>1. Compute similarity between each pair of N users/items
1. Alternative approach - A more evolved version is to take the average of each user's ratings on all the movies and
substract them from every rating given by that user and then compute the similarity score.
2. Obtain a representation of each user in low-dim space
3. Perform k-means clusterings</td><td>The drawback of Aggregrate clustering is that there may be many users (or items) being closer to the boundary of clusters. For those users (or items), the cluster utilized for its estimation may not be representative enough. We will overcome this problem with Collaborative Filtering</td><td></td><td></td></tr>
<tr><td>1. Compute similarity between each pair of N users/items</td><td>We calculate the normalized Euclidean distance or cosine similarity distance between users and
form a matrix of dimension N x N where each value represents the simiarity of the users/items in the row to the users/items in the column.
We can also find the similiarity score by calculating normalized Euclidean distance instead of cosine-similarity</td><td>Example:
<table>
<tr><td>Users/items</td><td>Movie 1</td><td>Movie 2</td><td>Movie 3</td><td>Movie 4</td></tr>
<tr><td>User 1</td><td>3</td><td>4</td><td>*</td><td>2</td></tr>
<tr><td>User 2</td><td>*</td><td>4</td><td>5</td><td>1</td></tr>
</table>
Here, we only consider movie 2 and movie 4 to find the similarity as there were unknowns in movie 1 and movie 3. So, the new matrix is:
<table>
<tr><td>Users/items</td><td>Movie 2</td><td>Movie 4</td></tr>
<tr><td>User 1</td><td>4</td><td>2</td></tr>
<tr><td>User 2</td><td>4</td><td>1</td></tr>
</table>
Cosine Similarity = $\frac{(4\times4)+(2\times1)}{\sqrt{(4^2+2^2)\times(4^2+1^2)}}$ = 0.97
Similarity score between User 1 and User 2 is 0.97. This implies that User 1 and User 2 are very similar.</td><td></td><td></td></tr>
</table>

<table>
<tr><td>1. Alternate Approach</td><td>In the above table, we take the average rating of movies by every user
User 1 = (4 + 2) / 2 = 3
User 2 = (4 + 1) / 2 = 2.5
and now substract these averages with all the ratings of the movies given by their corresponding values
Note: In case there are no common movie ratings between two users, then we consider the similarity score between that particular pair of users as 0</td><td><table><tr><td>Users/items</td><td>Movie 2</td><td>Movie 4</td></tr><tr><td>User 1</td><td>(4 - 3) = 1</td><td>(2 - 3) = -1</td></tr><tr><td>User 2</td><td>(4 - 2.5) = 1.5</td><td>(1 - 2.5) = -1.5</td></tr></table>Cosine Similarity = $\frac{(1\times1.5)+(-1\times-1.5)}{\sqrt{(1^2+-1^2)}\times\sqrt{(1.5^2+-1.5^2)}} = 1$</td><td></td><td></td></tr>
<tr><td>2. Obtain a representation of each user in low-dim space</td><td>After getting N X N similarity matrix from Step 1, we can do SVD on the matrix nd keep the top d components to get the representation of each user in a lower dimension or we can also use PCA to reduce the matrix into a lower dimension</td><td></td><td></td><td></td></tr>
<tr><td>3. Perform K-means clusterings</td><td>Iteratively find k clusters till they make sens and after finding these clusters and restric averaging method within the cluster in which the user/item of interest belongs</td><td>The drawback of Aggregrate clustering is that there may be many users (or items) being closer to the boundary of clusters. For those users (or items), the cluster utilized for its estimation may not be representative enough. We will overcome this problem with Collaborative Filtering</td><td></td><td></td></tr>
<tr><td>Collaborative Filtering or Personalized Clustering</td><td>In personalized clustering, for every user or item L_{ij}, we find other users that are very similar to it and declare them as its cluster and now we use the average over the declared cluster to produce the matrix estimate
1. User-based (aslo known as User-User Collaborative Filtering)
2. Item-based (also knnown as Item-iTem Collaborative Filtering)</td><td></td><td></td><td></td></tr>
<tr><td>User-based (User-User Collaborative Filtering)</td><td>is a technique used to predict the items that a user might like on the basis of ratings given to items by other users who have the similar tastes with that of the target user</td><td></td><td></td><td></td></tr>
<tr><td>Item-based (Item-Item Collaborative Filtering)</td><td>is a technique used to predict the items that a user likes on the basis of finding similarities between items that the user had rated with that of the target items</td><td></td><td></td><td></td></tr>
<tr><td>User-User collaborative Filtering</td><td>To find the likelihood of user i on item j
We observe all the users who had rated item j from the matrix
We find the similarity score between user i and all other users who had rated item j
Consider the top k nearest neighbors based on the above-calculated similarity score
Take the average of these top k user ratings on item j to estimate the value</td><td></td><td>
The cosine similarity score of user i and user N is $\frac{(1\times1)+(4\times4)}{\sqrt{(1^2+4^2)\times(1^2+4^2)}} = 1$</td><td></td></tr>
<tr><td>Item-Item collaborative filtering</td><td>To find the likelihood of user i on item j:
We observe all the items which user i had rated from the matrix
We find the similarity score between item j and all other items which had been rated by user i
Consider the top k nearest neighbors based on the above-calculated similarity score
Take the average of these top k item ratings rated by user i to estimate the value</td><td>This technique is similiar to User-User collaborative filtering with the difference that instead of users here we find the similarity between items that the users had rated and find the matrix estimates

Instead of going either only User-User or Item-Item interactions, we can do both User-User and Item-Item collaborative filtering and take their average to find the estimate</td><td></td><td>The formula for considering both User-User and Item-Item interactions is:
$\widehat{L_{ij}} = \frac{\left(\sum_{i' \in I_{i'}, j' \in I_j} L_{i'j'}\right)}{|I_i||I_j|}$</td></tr>
</table>