Recommendation Systems

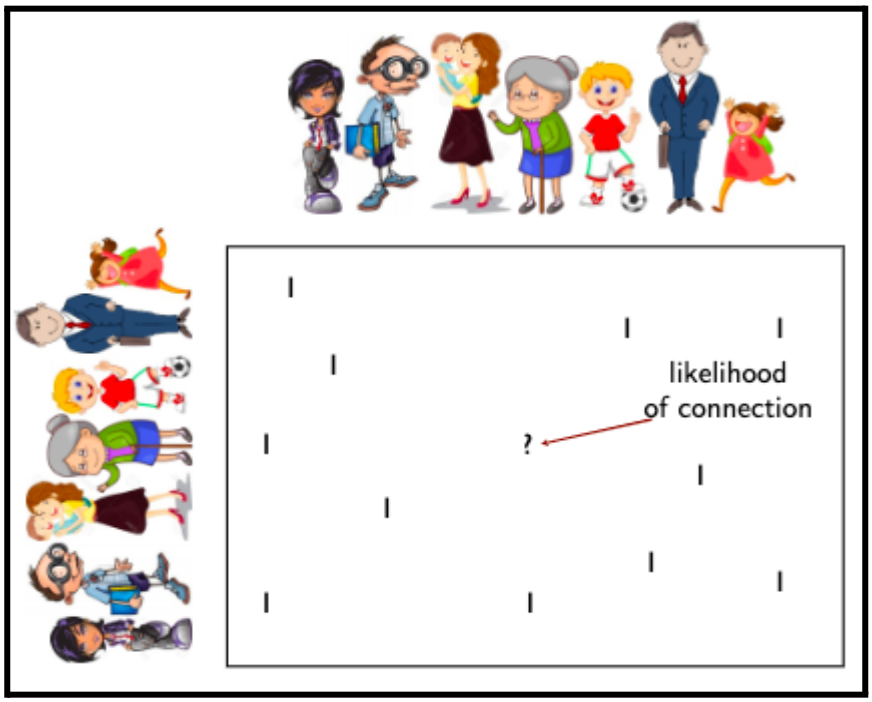
LVC 2: Recommendation Systems Part 2

In the last module, we learned about a simple method where we estimated the matrix values by taking the average of all the items assuming all the users are the same. But this method seems to be very naïve, as it carries an assumption of all the users being the same. In this module, we will learn more advanced approaches for matrix estimation.

Some of the examples of Matrix Estimation:

Social Networking:

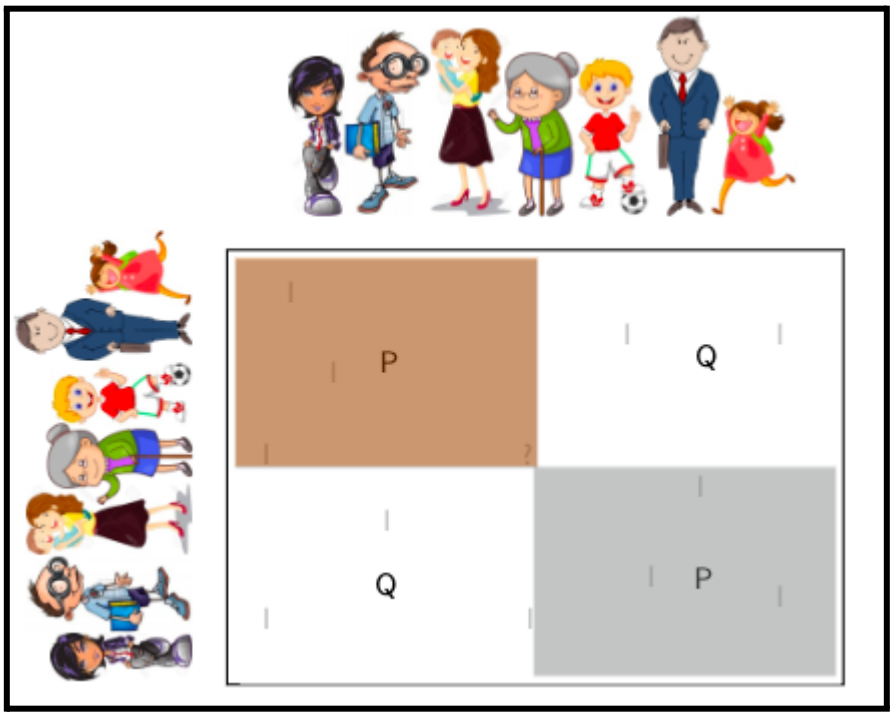
Here, in the below matrix, value 1 implies that there is a connection between the corresponding people in the row and column. For those entries which are unknown (?), we need to find the likelihood of a connection between them. We can think of this as an application for designing a recommendation system for recommending friend requests on Facebook and LinkedIn.



Community Detection:

Community detection splits the network down into several small-scale groups where more traditional recommendation approaches can be implemented. The below matrix represents that the Users belonging to the same group have more similar social 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

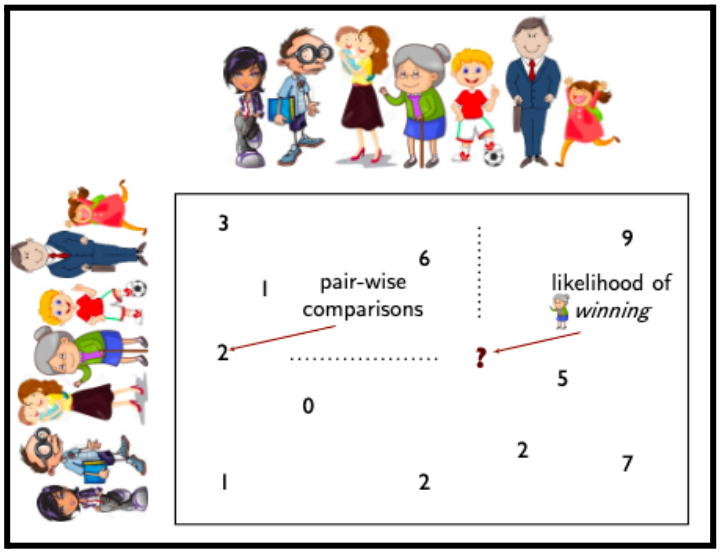
communities.



Ranking players and team:

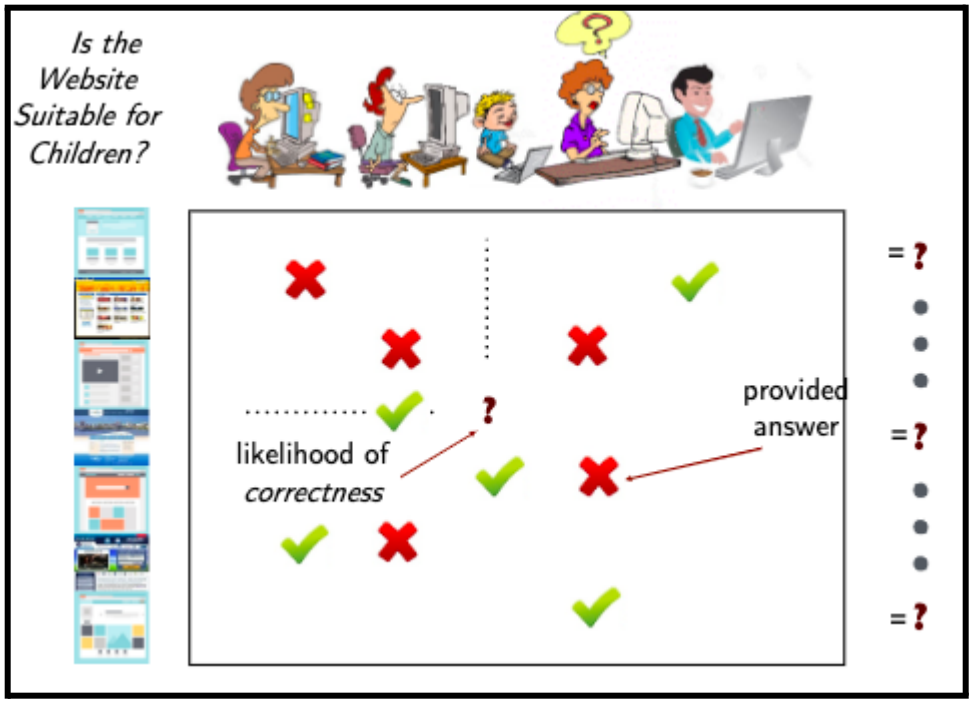
In the below matrix, the values represent the number of games played between the people of the corresponding row and column. Our aim in this task is to try to find the likelihood of one person winning over the other person to fill the matrix below

$$P_{ij} = P[i > j] \quad \text{(probability of winning of player } i \text{ over player } j\text{)}$$



Crowdsourcing:

In the below 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 on an item) for each individual based on all their ratings and aggregate them to decide the likelihood of a website being suitable for children or not.



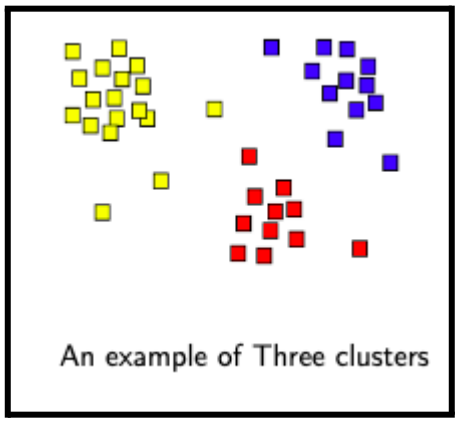
Solutions of Matrix Estimation:

- Clustering:
  - Find the user-item cluster
  - Average within the cluster
- Collaborative filtering (personalized clustering)
  - Finding users and items similar to a given user, item
  - Averaging amongst user-item specific similar users, items
- Singular value thresholding, optimization
  - Find Singular Value Decomposition of the matrix

Let's go through each of these techniques one-by-one.

Clustering:

We try to relax the assumption that either all users are similar or all items are similar (i.e., homogeneous users or items) which is most likely not true. However, we may make the assumption that users (or items) form multiple homogeneous-enough groups or clusters.

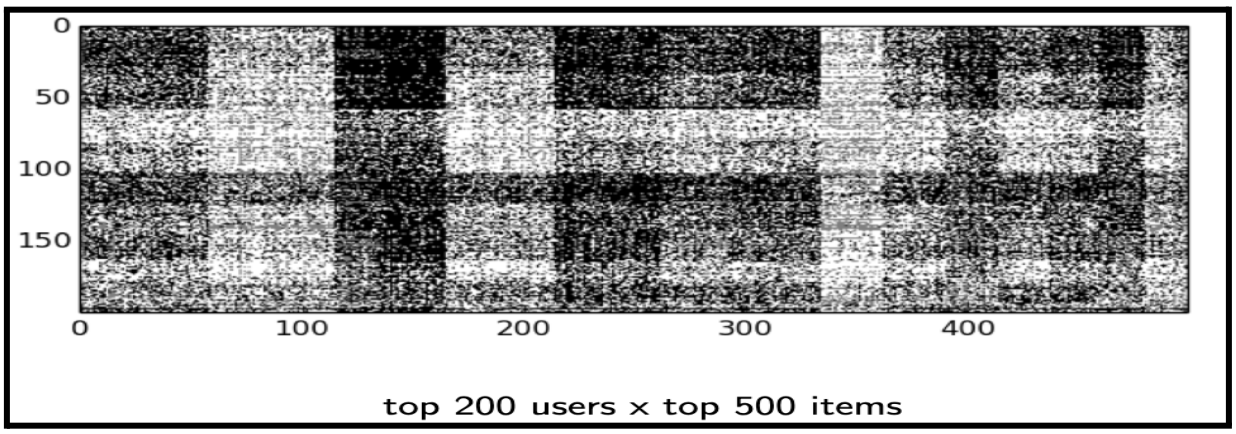


Above, we can see an example of three clusters. We have learned about clustering in the earlier course-weeks and how it can be used to find groups of similar types of data.

In this method, we make clusters based on user-item interactions. We assume that within the clusters, the users and items are similar. So, first we find these clusters and restrict the averaging method within the cluster in which the user/item of interest belongs.

Example:

Clustering: MovieLens Visual



The above image is the representation (after reordering) of the rating matrix.

Here, we take the top 200 users who have rated most movies and the top 500 movies which have been rated by users. We reorganize the users and items in such a way that we can see the checkered box pattern in the grayscale.

We can see that the values of user-items were similar within the clusters but different from across clusters. Now, we can use the averaging method within the clusters.

How to form Clusters:

Step 1: Compute similarity between each pair of N users/items.

We calculate the normalized Euclidean distance or cosine similarity distance between users and form a matrix of dimension $N \times N$ where each value represents the similarity of the users/items in the row to the users/items in the column.

Example:

Users/items	Movie 1	Movie 2	Movie 3	Movie 4
User 1	3	4	*	2
User 2	*	4	5	1

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:

Users/items	Movie 2	Movie 4
User 1	4	2
User 2	4	1

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.

We can also find the similarity score by calculating normalized Euclidean distance instead of Cosine-similarity.

Step 1 Alternative approach:

A more evolved version is to take the average of each user's ratings on all the movies and subtract them from every rating given by that user and then compute the similarity score.

Example:

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 subtract these averages with all the ratings of the movies given by their corresponding users.

Users/items	Movie 2	Movie 4
User 1	(4 - 3) = 1	(2 - 3) = -1
User 2	(4 - 2.5) = 1.5	(1 - 2.5) = -1.5

Cosine Similarity = $\frac{(1\times1.5)+(-1\times-1.5)}{\sqrt{(1^2+-1^2)}\times\sqrt{(1.5^2+-1.5^2)}} = 1$

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.

Step 2: Obtain a representation of each user in low-dim space

After getting $N\ X\ N$ similarity matrix from Step 1, we can do SVD on the matrix and 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.

Step 3: Perform k-means clusterings

Iteratively find k clusters till they make sense and after finding these clusters and restrict averaging method within the cluster in which the user/item of interest belongs.

The **drawback of Aggregate 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.

Collaborative Filtering or Personalized Clustering:

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.

There are two types of Collaborative Filtering:

- **User-based (also known as User-User Collaborative Filtering):** It 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 similar tastes with that of the target user.
- **Item-based (also known as Item-Item Collaborative Filtering):** It 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.

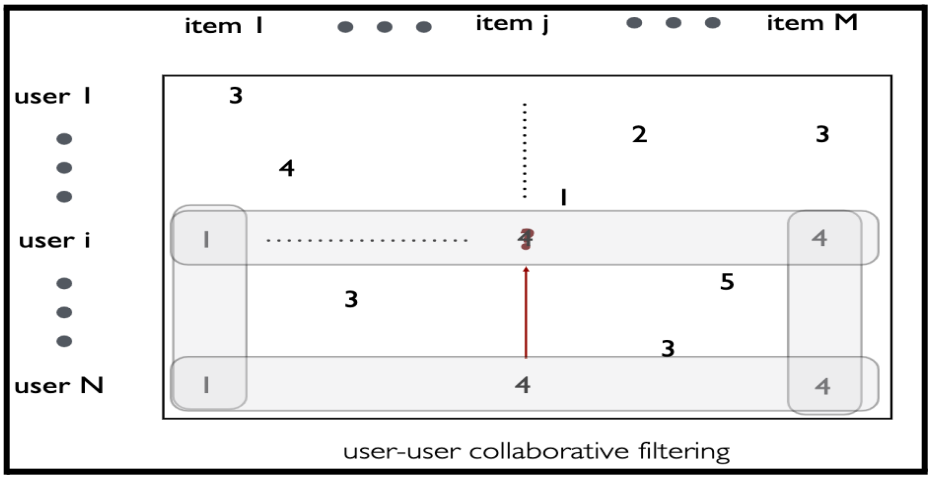
User-User Collaborative Filtering:

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

Example for User-User Collaborative Filtering:

In the below matrix, we need to estimate the matrix values L_{ij} i.e., likelihood of user i on item j. We observe that item j is rated by user N only. So, now we consider user i and user N, both of them have rated the item i and item M. Now, we calculate the similarity score between user i and user N by considering item 1 and item M.



The cosine similarity score of user i and user N is $\frac{(1\times1)+(4\times4)}{\sqrt{(1^2+4^2)\times(1^2+4^2)}} = 1$

As the similarity score is 1, i.e., both the users are completely similar. The value of user i x item j is the same as the value of user N x item j, which is equal to 4.

Remark: In case if the similarity is not 1, then we can take the weighted average or we can do exponential weightage or gaussian weightage of the top k user's ratings of that particular item to find the likelihood.

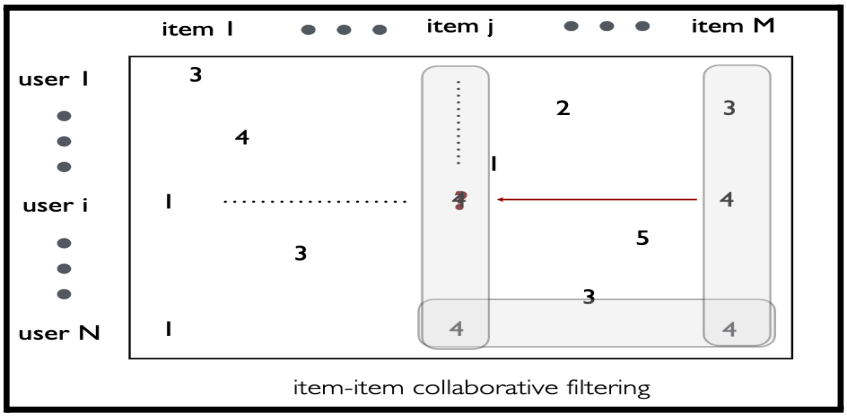
Item-Item Collaborative Filtering:

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 knearest 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

Example for Item-Item Collaborative Filtering:

This technique is similar 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.

The formula for considering both User-User and Item-Item interactions is:

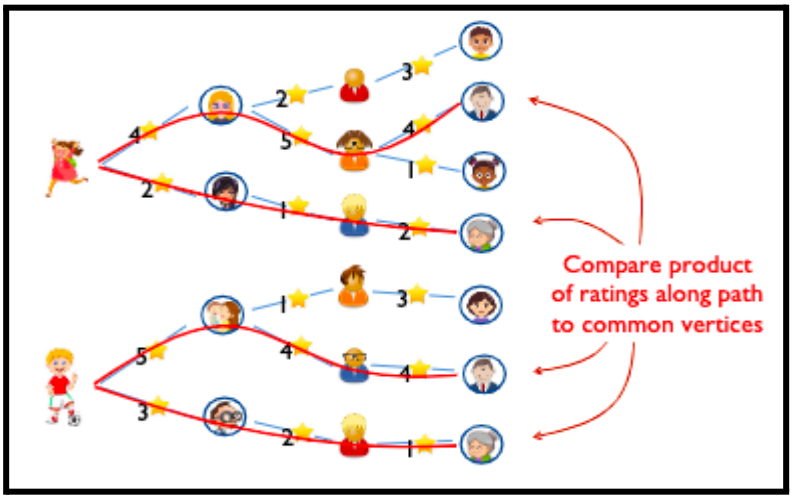
$$\widehat{L_{ij}} = \frac{\left(\sum_{i' \in I_i, j' \in I_j} L_{i'j'}\right)}{\left|I_i\right|\left|I_j\right|}$$

Where, I_i are all the users similar to user i and I_j are all the items similar to item j.

Iterative Collaborative Filtering:

In case our matrix is too sparse and there were no common predictions between 2 users, then we use an iterative collaborative filtering approach for the matrix estimation.

First, we find users similar to users of interest, next, we find users similar to these users, and continue iterating this procedure to find more similar users still enough similar users are found. Use the experiences of such users to obtain the estimate. This is illustrated in the below image.



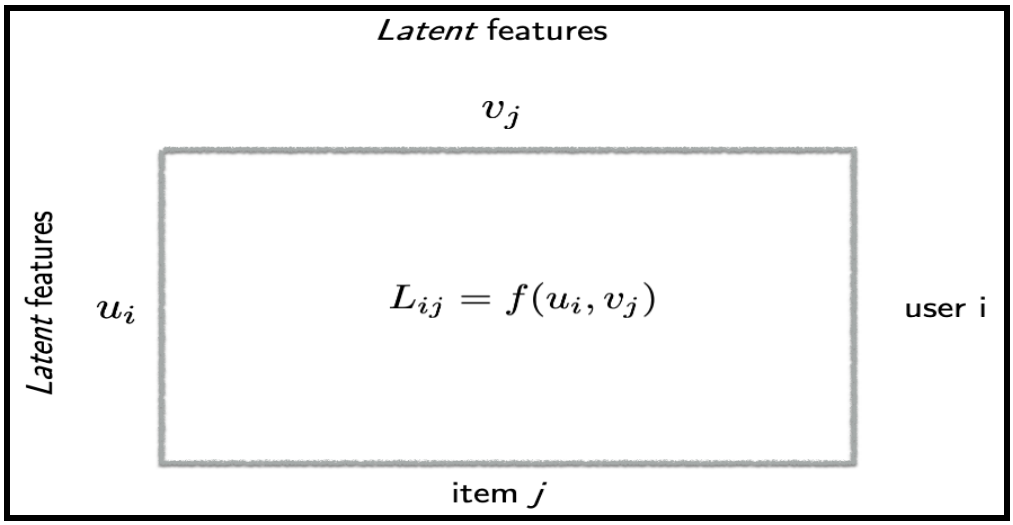
Advantage of Collaborative filtering:

- Extensively used in practice
- Scalable implementation using “approximate nearest neighbors”
- Closely related to the non-parametric nearest neighbor method
- Incremental and hence robust

Now, let’s see another technique of matrix estimation.

Singular Value Thresholding (SVT):

Prediction problem: Complete the matrix



To estimate L_{ij} for any user i and item j using SVD, we extract the latent features that explain the behavior of both users and items, i.e., we try to find u_i and v_j (where u_i and v_j represent latent features of the rows and columns, respectively).

Let's suppose there are some latent features U and V for the users and items. The features have many missing values, so the purpose of SVD is to estimate the matrix by filling null values with 0 and applying the following formula to estimate the feature matrix.

$$A = USV^T$$

Where, U and V are Left Singular matrix and Right Singular matrix, respectively and S is a diagonal matrix.

$$\underset{m \times n}{X} = \underset{m \times r}{U} \; \underset{r \times r}{S} \; \underset{r \times n}{V^T}$$

An example:

$$\begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} = \begin{bmatrix} -1 & -1 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} -2 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ -2 & -1 \end{bmatrix}$$

Now, any matrix obeys singular value decomposition if rank(X) = r

To find SVD of L :

Step 1: Fill missing values in L by 0. (In case of causal dependency a better way is to fill the missing values by taking the average of its the row and column values)

Step 2: Compute SVD

$$L_{ij} = \sum_{k=1}^{min(M,N)} s_k u_{ik} v_{jk}$$

Step 3: Apply truncated SVD by keeping the top r components (+ normalize)

$$\widehat{L_{ij}} = \frac{1}{\hat{p}} \sum_{k=1}^{r} s_k u_{ik} v_{jk} \text{ for all i,j}$$

Where $\hat{p}$ is the fraction of observed entries.

Question: How do we choose r from step 3?

Ans: We first sort the singular values which we obtained in step 1 in descending order and plot the values in a graph. We consider r as the value where the knee point appears in the plot. In case the graph is smooth, then it means that this algorithm doesn't work properly for the dataset

Example of SVD:

Question: We were given that r=1, s_1=1 and we know the values of the first row and first column. Find the values of the matrix.

The SVD formula is

$$\widehat{L_{ij}} = \frac{1}{\hat{p}} \sum_{k=1}^{r} s_k u_{ik} v_{jk}$$

Given r=1 and s_1=1 then the equation becomes

$$L_{ij} = u_i v_j$$

Now, we adjust the formula as (multiplying and dividing by $u_1 v_1$)

$$L_{ij} = \frac{(u_i v_1)^* (u_1 v_j)}{(u_1 v_1)}$$

We know that $(u_i v_1)$ is L_{i1} and $(u_1 v_j)$ is L_{1j}

$$L_{ij} = \frac{(L_{i1})^* (L_{1j})}{(L_{11})}$$

As given in the question, we know the values of the first row and column, i.e., we know the values of L_{i1} and the values of the first column L_{1j}. Now, we can recover all the values of the matrix by changing values for i and j.

- From the above matrix with $m + n - 1$ observations, we can recover $m \times n$ values of the matrix.
- We can generalize the above statement as if we have rank r matrix we roughly need to know $(m + n) \times r$ entries to recover the entire $m \times n$ matrix.

Optimization of SVD:

Singular value decomposition: An optimization perspective

In case we only observe a few observations of the matrix which are noisy we can still find the U and V matrices such that the sum of the squared error on the observed entries is minimized.

Mathematically, the optimization problem of finding U and V can be written as:

$$\text{minimize} \sum_{(i,j) \in obs} \left(L_{ij} - \sum_{k=1}^{r} u_{ik} v_{jk} \right)^2$$

over

$$u_{ik} \epsilon R,\ 1 \le i \le N, 1 \le k \le r$$

$$v_{jk} \epsilon R,\ 1 \le j \le M, 1 \le k \le r$$

We can solve the above optimization problem by making it into 2 subproblems:

Step 1: Fix V (items) and solve for U (items) by minimization of the cost function, we can think of it as a linear regression problem on i keeping j as fixed.

$$\text{i.e., minimize} \sum_{i \epsilon obs} \left(L_{ij} - \sum_{k=1}^{r} (u_{ik}) v_{jk} \right)^2$$

as we know L_{ij} and u_{ik} let's think of v_{jk} as $\beta's$ and the equation becomes of the form:

$$\text{minimize} \sum_{i \epsilon obs} \left(Y_i - x_i \times \beta \right)^2$$

Step 2: Similarly, now Fix U (users) and solve for V (items) by minimization of the cost function, we can think of it as a linear regression problem on j keeping i as constant.

$$\text{i.e., minimize} \sum_{j \epsilon obs} \left(L_{ij} - \sum_{k=1}^{r} u_{ik} (v_{jk}) \right)^2$$

$$\text{minimize} \sum_{j \epsilon obs} \left(Y_j - x_j \times \beta \right)^2$$

- After finding the U and V

  For any i, j, produce estimate as follows:

$$\widehat{L_{ij}} = \sum_{k=1}^{r} u_{ik} v_{jk}.$$

Note: The above algorithm uses only observed entries and does not require filling missing values as in for SVD.