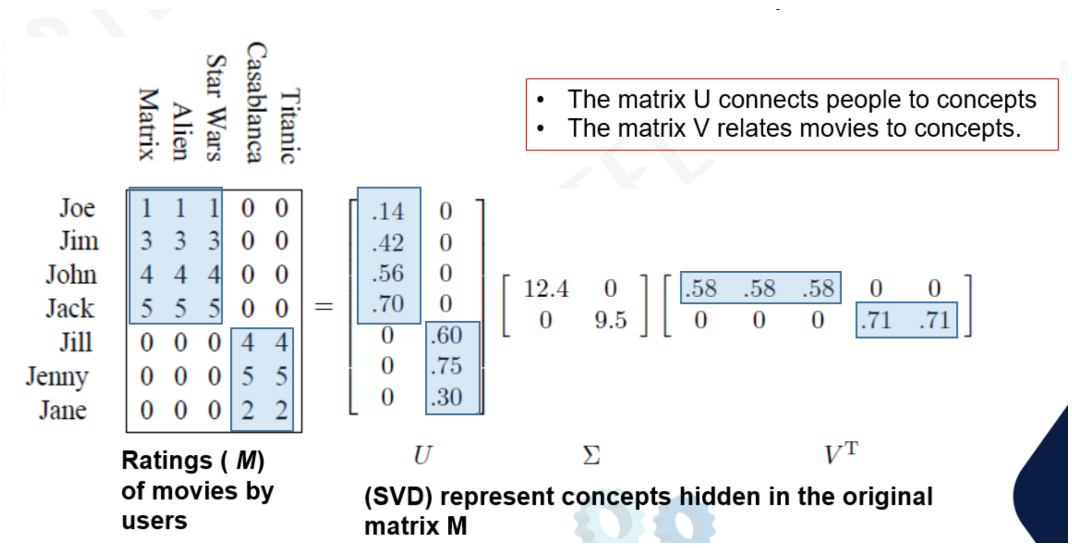
Course Name - Recommender Systems
Professor Name - Prof. Mamata Jenamani
Department Name - Industrial and Systems Engineering
Institute Name - Indian Institute of Technology Kharagpur
Week - 04
Lecture – 17

Lecture 17: UV Decomposition

Hello everyone. Today we are going to start a new topic. The topic is extension of collaborative filtering, but specifically we will start talking on model based collaborative filtering. In this context in today's lecture we are going to cover UPD composition which is the basis of this latent factor model. Now, what are this latent factor models? So far in model based collaborative filtering we have seen in the last class that almost all the machine learning algorithms can be used in case of collaborative filtering situation. Like in case of traditional machine learning algorithms you have a set of training data and you have a set of test data, row wise a set of training and testing data.

But in case of collaborative filtering it is a matrix completion problem. By matrix completion problem we mean that in a matrix some of the elements are available and some are known are not known. So, therefore, you are supposed to fill up those values ok. So, using extending this idea now latent factor models have come up.



- The matrix U connects people to concepts
- The matrix V relates movies to concepts.

$$\begin{array}{c|ccccc} & \text{Matrix} & \text{Alien} & \text{Star Wars} & \text{Casablanca} & \text{Titanic} \\ \hline \text{Joe} & 1 & 1 & 1 & 0 & 0 \\ \text{Jim} & 3 & 3 & 3 & 0 & 0 \\ \text{John} & 4 & 4 & 4 & 0 & 0 \\ \text{Jack} & 5 & 5 & 5 & 0 & 0 \\ \text{Jill} & 0 & 0 & 0 & 4 & 4 \\ \text{Jenny} & 0 & 0 & 0 & 5 & 5 \\ \text{Jane} & 0 & 0 & 0 & 2 & 2 \end{array} = \underbrace{\begin{bmatrix} .14 & 0 \\ .42 & 0 \\ .56 & 0 \\ .70 & 0 \\ 0 & .60 \\ 0 & .75 \\ 0 & .30 \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} 12.4 & 0 \\ 0 & 9.5 \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} .58 & .58 & .58 & 0 & 0 \\ 0 & 0 & 0 & .71 & .71 \end{bmatrix}}_{V^{T}}$$

Ratings (*M*) of movies by users

(SVD) represent concepts hidden in the original matrix M

Let me tell you. So, far we have covered neighborhood based collaborative filtering. In neighborhood based collaborative filtering the idea is you make the recommendation based on the closest neighbors choices or based on the similarity of the items that you are you have already purchased ok. So, those are fairly simple and even item based recommender systems are also widely adopted in commercial setting. However, in today's scenario this latent factor model have become very popular because they are much faster than the neighborhood based methods.

$$\begin{tikzcd} & M = U\Sigma V^{T} \arrow[dl, bend right] \arrow[dr, bend left] & \\ MM^{\mathrm{T}}U = U\Sigma^{2} & & M^{\mathrm{T}}MV = V\Sigma^{2} \end{tikzcd}$$

So, this latent factor based model they leverage the well known dimensionality reduction technique and fill in the missing entries. So, so far this dimensionality reduction technique we have got some overview in many places. While talking about the fundamentals of the of the various fundamental fundamental topics related to recommender system and we are also covered a bit at the time of neighborhood based system. So, today we are going to talk little bit elaborately on this particular topic. This diagram must be familiar to you in one of the earlier lecture while talking about the fundamental while discussed it.

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \\ u_{31} & u_{32} \\ u_{41} & u_{42} \\ u_{51} & u_{52} \end{bmatrix} \times \begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} & v_{15} \\ v_{21} & v_{22} & v_{23} & v_{24} & v_{25} \end{bmatrix}$$

So, once again I bring to your notice that when we try singular value decomposition what we do? We at one go only we try extracting the hidden concepts that that is there in the data set. In this particular example in this particular example we had 5 movies and as the name indicates both this belong to 2 different categories. And purposefully in this example this first 4 people like this first 3 movies and second 3 people like the second 2 the next 2 movies. So, when we make the SVD and try making factorizing this matrix what was SVD that we are going to see at a next slide M is U sigma V transpose. What is U? U is the user latent factor and V is the item latent factors.

So, these are basically the item features. Now, these item features are extracted not from the item details rather they are extracted from the rating behavior assuming that people like similar the movies with similar concepts this has been decided. So, in this particular example which is of course, very purposefully designed this first 4 people based on only the first latent factor can be understood that they have similar choices. Similarly, the second 3 from the second component you can know that they are the they choose the second genre of the movie. So, now individually this is the first persons detail this is the second person this is the third person this is the fourth person and so on.

So, from this person likes the first concept that is the genre here second person the first concept and so on. But when I move ahead we see that the next 3 people like the second concept. So, this matrix U connects the people to concept. So, similarly this is your this data is for this these are the latent factors for the movies. So, this is the first movie, second movie, third movie I am making this because this is the transpose this is the third movie this is the sorry this is the fourth movie and this is the fifth movie.

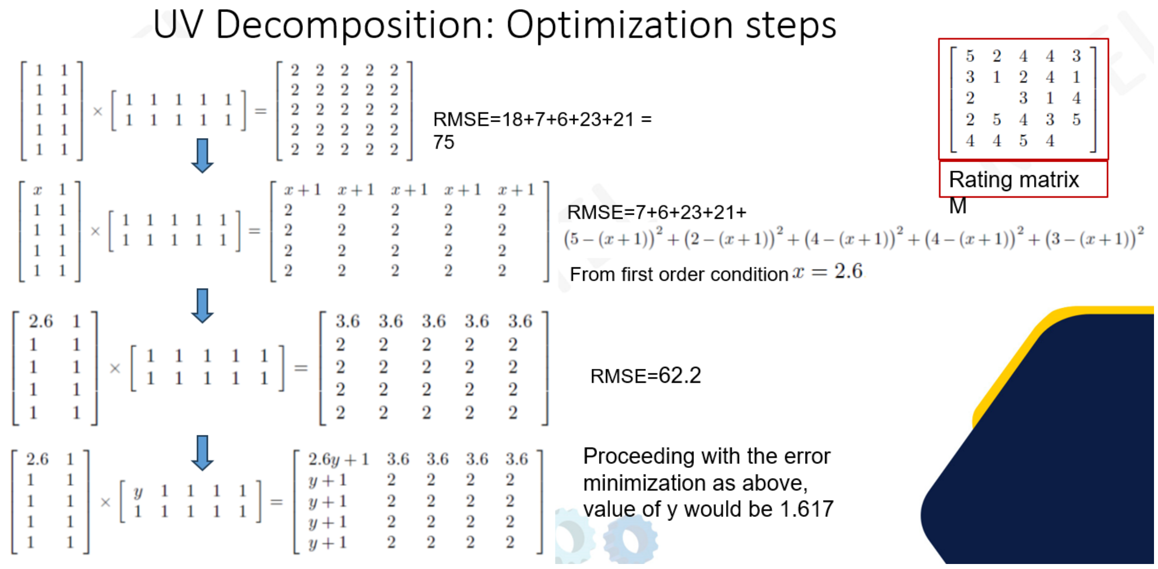
$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}$$

Now, here look at this here all these movies this first 3 this particular factor this is factor 1 this is factor 2 this first factor is becoming important for next 3 movies the second factor sorry the second factor is important in case of first category movie and this is for the second category of movie. So, this matrix view V also relates the movies to concepts. So, using this latent factor models at one go we can extract both item features and user features latent features. So, let us understand what was this singular value decomposition here we are decomposing matrix M into U sigma and V transpose where sigma is the set of singular values. And we also understood that U is a matrix which is column orthonormal and V is also column orthonormal and sigma is a diagonal matrix where the diagonal elements are the singular values of M.

$$\begin{bmatrix} x & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} x+1 & x+1 & x+1 & x+1 & x+1 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}$$

Now, we also understood that we can mathematically find out what is U V and sigma it turns out that U and sigma can be found out from M tem transpose and M transpose M. So, this is the eigen vector of M transpose and this is these are the eigen values eigen vector of M transpose M eigen values of M transpose M. And we though mathematically it was looking very nice while computing it may become little cumbersome because when the matrix becomes very large finding this eigen values and eigen vectors will be difficult to remind you how did we discover this eigen values and eigen vectors. For eigen values we constructed a constructed the characteristics equation, then from the characteristics equation we found out the eigen values from eigen values we found out the eigen vectors. And there are certain iterative method as well using power iteration.

So, however, these are only possible when this matrix M is full besides if we drop this orthonormality assumption then this U sigma V we can think of this as U some sigma to the power half then sigma to the power half and V transpose. And this we can call our U nu and V nu or we have some new U matrix and new V matrix which are decomposing the original matrix M, but without this orthonormality assumption to remind you what is orthonormality assumption if we take any two columns of the matrix then if we multiply then it becomes 0. So, they are perpendicular to each other. So, they are they are orthogonal as well as if we I mean they are unit vectors as well. Now, we have this problem the problem in this is so far in our discussion we know that matrix M was a full matrix.



UV Decomposition: Optimization steps

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}$$

RMSE=18+7+6+23+21 = 75

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix}$$

Rating matrix M

$$\begin{bmatrix} x & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} x+1 & x+1 & x+1 & x+1 & x+1 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}$$

RMSE=7+6+23+21+
$(5-(x+1))^2+(2-(x+1))^2+(4-(x+1))^2+(4-(x+1))^2+(3-(x+1))^2$

From first order condition $x = 2.6$

$$\begin{bmatrix} 2.6 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 3.6 & 3.6 & 3.6 & 3.6 & 3.6 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}$$

RMSE=62.2

$$\begin{bmatrix} 2.6 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} y & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2.6y+1 & 3.6 & 3.6 & 3.6 & 3.6 \\ y+1 & 2 & 2 & 2 & 2 \\ y+1 & 2 & 2 & 2 & 2 \\ y+1 & 2 & 2 & 2 & 2 \\ y+1 & 2 & 2 & 2 & 2 \end{bmatrix}$$

Proceeding with the error minimization as above, value of y would be 1.617

Therefore, there was no computational inconvenience observed while even determining this using such formula or through power iteration. So, now, we have a situation here. In this matrix M this value is missing and this value is missing. So, in these circumstances how do I get UV decomposition of course, the formula that we saw that we cannot use. So, what are we going to do? We are going to adopt some machine learning algorithm which will be some learning algorithm which will be simply looking at the available entries and try to minimize the error with respect to the available entries.

And because this process some the values which are missing will also keep getting updated. So, how do we do proceed about it? We have to first reprocess this matrix M ok. Then ah this of course, depends on the situation. So, if it is necessary you do it otherwise it is not and we have already discussed why preprocessing is necessary, why normalization has to happen. So, those things if necessary you do.

Then you have to initialize matrix U and V. When you initialize you can randomize the values. Then you have to start the optimization and you find out certain criteria to end the optimization process. Now, let us look at why and why you should try preprocessing the matrix M. So, in a typical rating matrix of course, the method that we are going to study it is it is not related to rating matrix only.

$$\begin{bmatrix} x & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} x+1 & x+1 & x+1 & x+1 & x+1 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}$$

$$\begin{bmatrix} 2.6 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} y & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2.6y+1 & 3.6 & 3.6 & 3.6 & 3.6 \\ y+1 & 2 & 2 & 2 & 2 \\ y+1 & 2 & 2 & 2 & 2 \\ y+1 & 2 & 2 & 2 & 2 \\ y+1 & 2 & 2 & 2 & 2 \end{bmatrix}$$

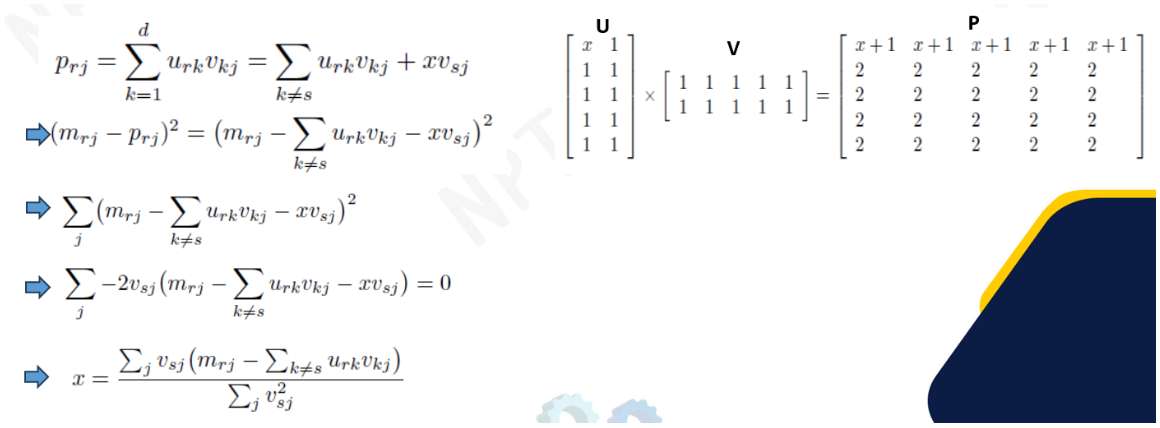
This matrix M can be any incomplete matrix and through UVD composition you can think of completing it. So, it is a matrix completion problem. So, ah in case in particular when this particular this matrix M is a rating matrix, then what kind of situation happen to this matrix that normalization becomes necessary. So, many times when in you get the values to rating matrix you restrict the users to give very high low values etcetera. Rather you provide them ah with certain discrete values most of the time this values will be discrete.

So, you give them a rating scale 4 star, 3 star, 1 star some star rating you allow, thumbs up, thumbs down such kind of things you allow. So, if the values are not distorted there is no chance of getting an outlier. So, what kind of ah preprocessing is necessary matrix will be within the range of the data that you are expecting it to be. However, there is some situation here what is the situation? Situation is there can be bias from the user side or from the item side. So, what is this bias? Now, look at the concept of user bias we are going to discuss about this user bias many more times as well, but here I am just trying to introduce it.

So, what is user bias? As an user when you try giving rating all the time your rating may not be consistent as your ah thought process. For example, suppose you are not very happy today and you are watching a movie. So, probably you will give a bad rating and somehow you are ah otherwise happy and you are watching a movie. So, even if the movie is not a great movie you can because your mood is elevated you will be giving a better value. So, such kind of biases are possible.

So, similarly this bias this is just one example depending on place, depending on time then bias can be always there. So, similarly there can be bias related to the item. So, what is this bias? So, sometimes you see that some items are very highly rated. So, even you personally feel that that item is not ah very good item because others everybody else is telling that item is a good item probably you will try giving a better value. And many times this fact is exploited by the attackers of the recommender system.

So, what do they do? They use many fake ratings using ah using some ah fake reviewers just to escalate the rating. So, as a result even if the movie is not so good because of thisfake ratings suddenly the average rating increase. And if the average rating increase ah genuine user genuine viewer becomes little biased anyway. So, whatever may be the case so there can be many methods to remove this user and item bias. So, first of all we can subtract ah from each entry the average rating of user i followed by average entry of item i in that particular.



$$p_{rj} = \sum_{k=1}^{d} u_{rk}v_{kj} = \sum_{k\neq s} u_{rk}v_{kj} + xv_{sj}$$

$$\Rightarrow (m_{rj} - p_{rj})^2 = \left(m_{rj} - \sum_{k\neq s} u_{rk}v_{kj} - xv_{sj}\right)^2$$

$$\Rightarrow \sum_{j}\left(m_{rj} - \sum_{k\neq s} u_{rk}v_{kj} - xv_{sj}\right)^2$$

$$\Rightarrow \sum_{j} -2v_{sj}\left(m_{rj} - \sum_{k\neq s} u_{rk}v_{kj} - xv_{sj}\right) = 0$$

$$\Rightarrow \quad x = \frac{\sum_j v_{sj}\left(m_{rj} - \sum_{k\neq s} u_{rk}v_{kj}\right)}{\sum_j v_{sj}^2}$$

$$\overset{\textbf{U}}{\begin{bmatrix} x & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}} \times \overset{\textbf{V}}{\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}} = \overset{\textbf{P}}{\begin{bmatrix} x+1 & x+1 & x+1 & x+1 & x+1 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}}$$

So, when we have certain rating m i j it is a part of the ith row and jth column maybe. So, what is ith row? ith row is for user jth row is for the item. So, ah with respect to this user there will be certain this user must have rated many items. So, there will be some average value let us call it m i bar maybe with respect to him there can be some average value let us say m j bar maybe. Now, combining ah while pre processing this you may subtract this first subtract my second this one or you may think of subtracting this first and this one second.

So, in the process the rating value may get modified and you can even think of adding these two and subtract them from from because you will be ah iteratively doing this it may so happen if you follow any of these methods the it may not give consistent result, but it is ok. Now, next comes initializing u and v as such there is no rule for initializing u and v why we have to initialize u and v come back to your original problem we do not we cannot really determine there are blank entries. So, there are blank entries. So, we cannot determine. So, therefore, we have to start with certain random value and keep on updating it as we go ahead.

So, what kind of random value you will be filling up here? In general you can put any random value, but in general it is a good choice that all these values have the same value and many times ah you have to ah you have to many times you sometimes you give the average rating of the entire matrix M. And if you have already normalized M then probably you will be getting all these initial values to be 0, but this with this particular example that we are doing for this particular matrix. So, let us just randomly initialize this u matrix and v matrix and after multiplying you get this one, but this one is not same as that of your original matrix M. So, what where is the difference? So, you can now find the difference. So, how do we find the difference? Element wise element wise element wise you will be finding the difference.

$$\begin{bmatrix} 2.6 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} y & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2.6y+1 & 3.6 & 3.6 & 3.6 & 3.6 \\ y+1 & 2 & 2 & 2 & 2 \\ y+1 & 2 & 2 & 2 & 2 \\ y+1 & 2 & 2 & 2 & 2 \\ y+1 & 2 & 2 & 2 & 2 \end{bmatrix}$$

$$y = \frac{\sum_i u_{ir}\left(m_{is} - \sum_{k\neq r} u_{ik}v_{ks}\right)}{\sum_i u_{ir}^2}$$

So, this first element 5 and what was the value? Value was 2 5 minus 2 2 minus 2 and so on. So, you will be finding the difference. So, next task is you are supposed to optimize this value and you will be sorry you will be you will be following certain optimization procedure. So, that this u and v which you initially have chosen as certain random values should be slowly updating and the error will keep on getting minimized slowly. So, to start with we can choose any random element.

So, let us choose this random element x. So, let us say this particular value we would like to update. So, if you would like to update this particular value and try ah replace it with certain x and the value of x we are supposed to find multiply these two and this resulting matrix is this one. So, when the resulting matrix is this one what do we do next? This is the resulting matrix and this was the original that I was telling you this is your original rating matrix and as I told you told you you randomized it with this once and you got this two and you subtract subtracted element wise 5 minus 2 then 2 minus 2 then 4 minus 2 then 4 minus 2 again then 3 minus 2. So, if we take element wise subtract the values element wise and take the square and add them together you get this root mean square error.

So, from this root mean square error our aim is to keep on reducing this error as much as possible. So, what has to what has to be done? We have to now replace it with x and once we replace it x this is our new matrix. Now, what is the value of x? Value of x is that which minimizes the error function. Now, what is the error function here? The error function here is element wise element you will be subtracting and taking the square. So, for the first row 5 minus x plus 1 2 minus x plus 1 and so on.

So, you did it along with this you will be subtracting others as well. So, others only this was the first row. So, others will be 7 plus 6 plus 23 plus 21 etcetera anyway. So, now, from this function how do I get this value? This value I will be getting using first order condition which means I will be taking let us say this function is error function I will be taking the error function is let us say this RMSE is a function of x. So, I will be taking d f x by d x and set it to 0 ok.

So, you can try if you take this function you take with respect to f x. So, what will happen? So, first term 2 into 5 minus x plus 1 then this is only x. So, that is minus 1 and so on. So, this I am not going to do it for you do it and find out the value of x turns out to be this and this x you now replace here. So, this x was determined from where from solving this optimization problem you replace it here.

So, after you replace it here then you keep on multiplying and get this updated value. So, once we have this updated value then what next? This is not my that true minimum our aim is to still reduce further how do I reduce? Choose another value. So, particularly just for demonstration purpose let us choose this value to be updated something from the v vector v transpose vector. So, accordingly when you multiply here some row was getting affected here some column is getting affected column is getting affected this column is getting affected ok. Then accordingly you construct certain RMSE which becomes a function of y and again you use first order condition which means you differentiate with respect to y and get the value of y.

Once you get the value of y you replace it. So, one iteration next iteration is over then you choose another. So, this way you can proceed. So, what is that generic procedure that you must follow? So, to choose any arbitrary element which can be either x or y we are supposed to minimize the error in each iteration. So, this is the process to do so.

To start with we started with one element x. So, what was this element? This element was one element which belongs to rth row and sth column. So, in this particular example what is r what is s? r is 1 s is also 1. So, rth first row first column. So, in my original matrix original matrix let us say this particular value this particular value is m certain rs ok. So, so let us do it now to get these elements this values what you have to do? This row with this column this column this column you have to multiply to get it ok.

So, that is what is being done here. So, first you multiply so which means individual element whatever was there into x and corresponding v value you are getting. Now, from m of course, here we have used simple rj. So, from m rj you subtract this p rj. What is p rj? p rj is this this.

So, square of that turns out to be this. So, this p rj is computed here. So, as a result this becomes your error with respect to the sorry this becomes error with respect to the first term. Now, if this error has occurred in this entire row. So, we have added them all remaining values we do not have to be thinking about because they will become constant. So, you when you take the differentiation they will anyway automatically go.

So, you take the differentiation set it to 0 when you set it to 0 then take move x to this side and you get this formula. So, this is the error update formula for x. So, similarly moving ahead you can choose any value y and develop one such error update formula. And this using this x and y you can iterate you can iterate taking see once you took x here second time you took y here next time you choose some x here some y here. So, you can in a some either in a systematic order or in some random order you change and you update.

But, remember at one go you are not going to make it as make it quite close to the original matrix. I mean the of course, the available entries of the original matrix. So, how many times you will be iterating it depends the error limit that you keep. So, you keep on updating keep on updating and at some point of time you meet whatever is the target epsilon difference you have decided and you stop.

And here is one interesting fact. In fact, from the very first lecture I have been talking about this Netflix challenge. In fact, all these matrix factorizations based approaches which are responsible for building this model based specifically latent factor based models have started from this Netflix challenge. And because it was a very important driver I thought of mentioning it once again. Thank you. These are my references and today's lecture specifically is from this one.

So, we wind up and stop now. So, these are my concluding remarks. Singular value decomposition can identify latent features both for users and items in one go. UVD composition simplifies SVD and violates orthonormality assumption. So, UV matrix can be viewed as latent factor matrices for users and items respectively. Now, when matrix M is not fully specified some of the entries are blank UV can be computed by minimizing the error with respect to specific entries in M.

With this process missing entries in M can be computed at one go. So, this can be treated as a matrix completion problem and we have to learn the value of u and v. With this we wind up this lecture. Thank you.