Design : Simple Youtube Recommendation using Simple Math

Problem Definition

In this problem I will attempt to explain "**How to build Simple Youtube Recommendation**" using simple 10th std Mathematics. We want to recommend the videos based on what the user is currently watching?

Business Problem

- We want to recommend more meaningful videos to users based on what they are watching currently.
- We want to increase user engagement on youtube, more time users will spend on youtube, more chances to show ads and generate revenue for youtube.
- Youtube will generate revenue based on ads. If users spend more time on youtube, youtube has a higher chance of showing ads and building revenue.

What Data does Youtube have?

- User history
- Playlist information
- Search history
- Subscribers information
- Video Categories
- Demographic information
- Android, Chrom and other google product information

Youtube may have more information than what we have listed above, for the simplified task at our hand we will narrow down our scope for this task. We will only consider viewing history to build a very basic recommendation system (We will not consider temporal information - we currently don't want to focus on time series problems).

We have set some assumptions for our solution. User tastes change over time, and there are some constraints on how long we should keep user information in our database, so we will only consider the last 7 days of user viewing history for our solution. We will not use any other information to build a recommendation system. Make a note that the state of the art system deployed at Youtube is very advanced (but here we are focusing on very basic recommendation system design that 10th std students can understand.

Formulate the business Problem into Machine Learning Problem

Viewing history sample :
U1 : v1, v2, v69, v100, v1020 ……., vm
U2 : v2, v3, v70, v1040……., vn

U_i has watched these videos - v_j is the id of the video.
Data matrix :

	V_1	V_2	V_3	V_4	V_j		V_k	V_m
U_1	1	1	1		1		1	0
U_2	1	0	0		0		1	0
U_3	1	1	0		1		1	0
U_4	0	1	0		0		1	0
U_i	1	0	1		0		1	0
							1	0
U_n	1	1	0				1	0

We can create a binary data matrix from the User Video viewing history. Where each entry A_i_j represents whether the User i has watched video j or not.

$$A_{ij} = 1 \; ; \; user \; U_i \; has \; watched \; video \; V_j \; else \; 0$$

m = # total number of videos
n = # total number of users

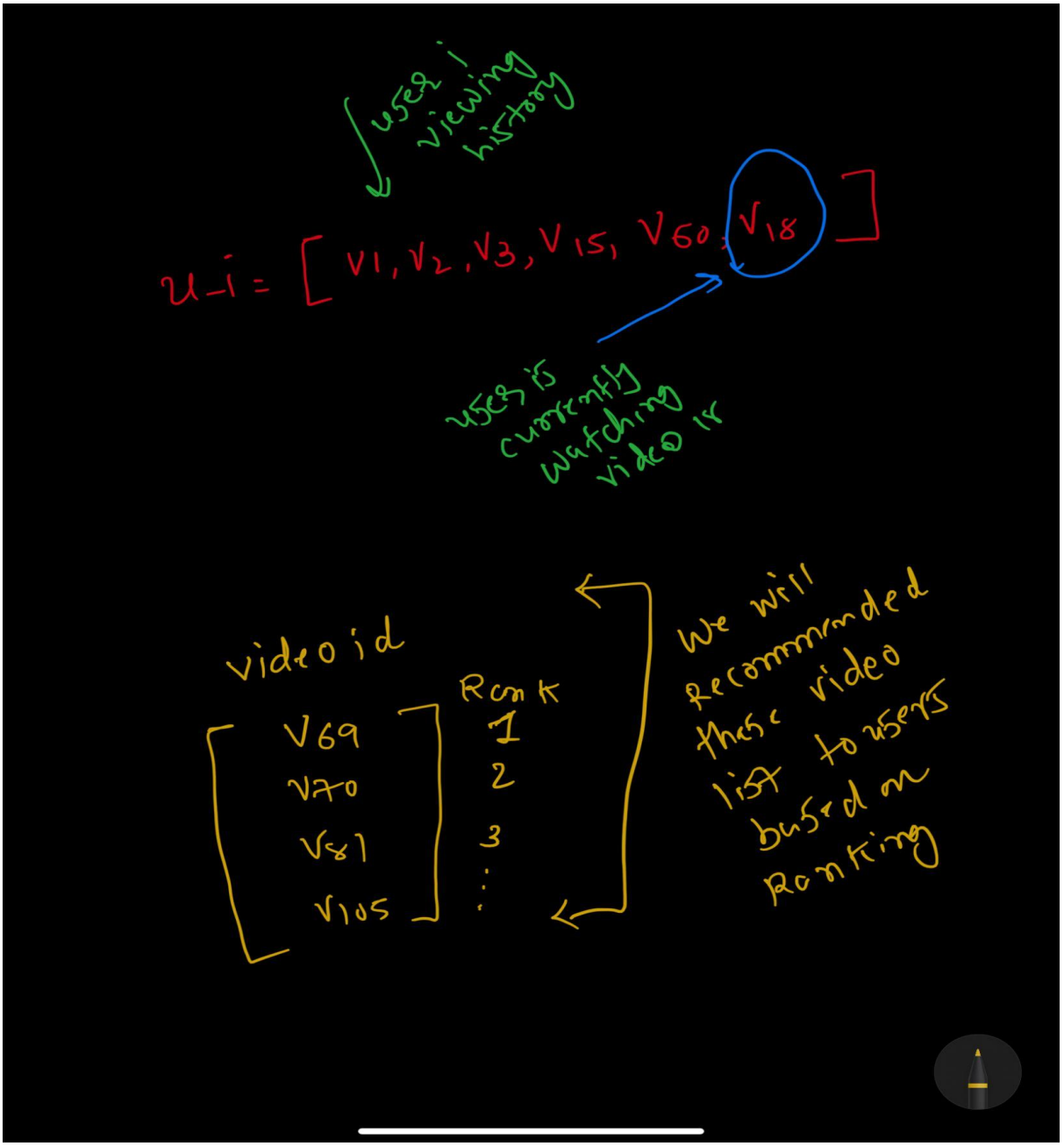
how to formulate a Data Matrix into a machine learning recommendation problem?

U_I = [V_1, V_6,V_10, V_15,V_18]

If User I has watched the above video sequences and he is currently watching Video 18, we want to recommend the video list that user might want to watch next.

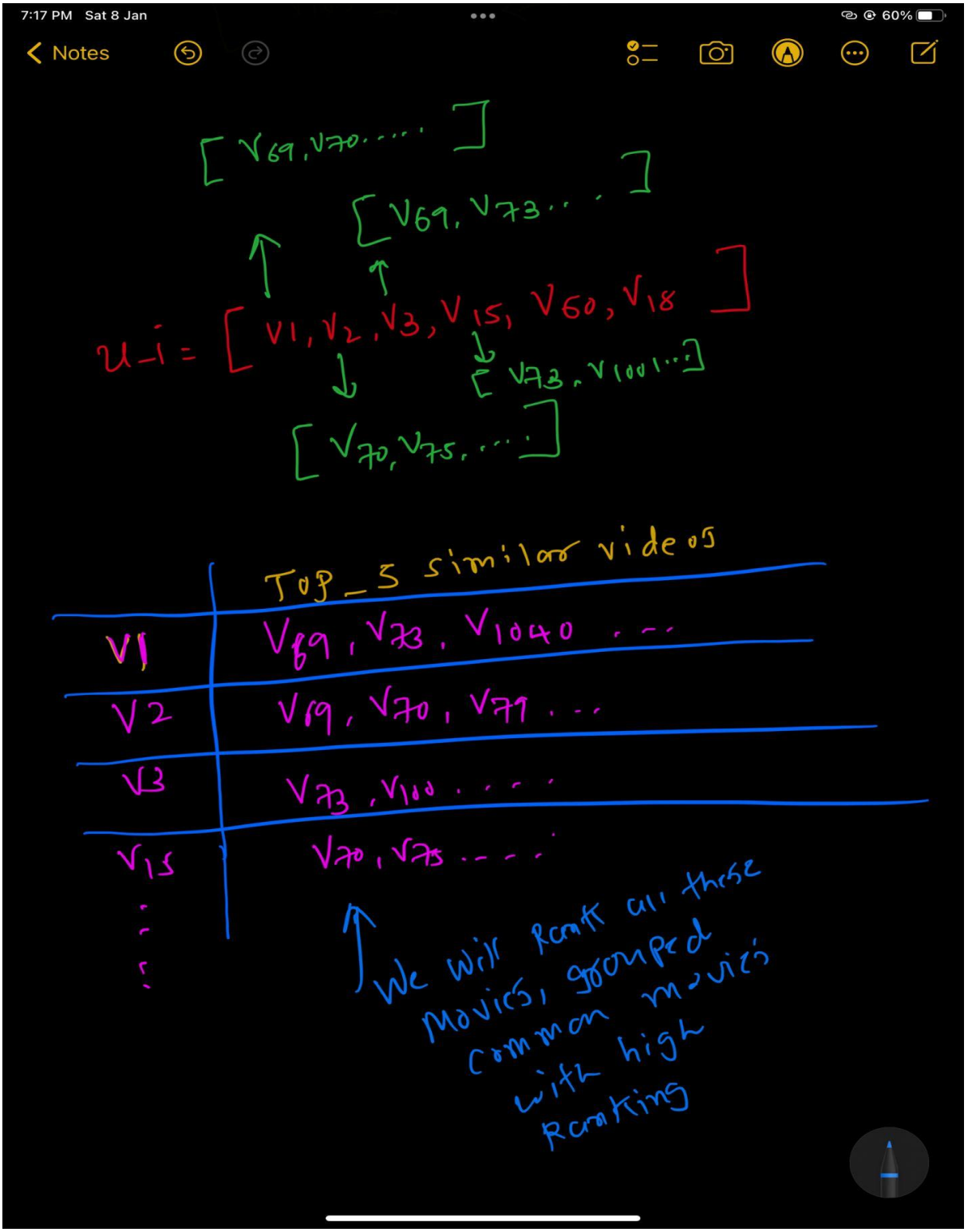
We want to return the list of recommendations in order of how likely the video will be useful for the user based on what he/she is watching currently.

We will transfer this problem to return the rank list of recommended video.



Thinking Like Similarity Problem

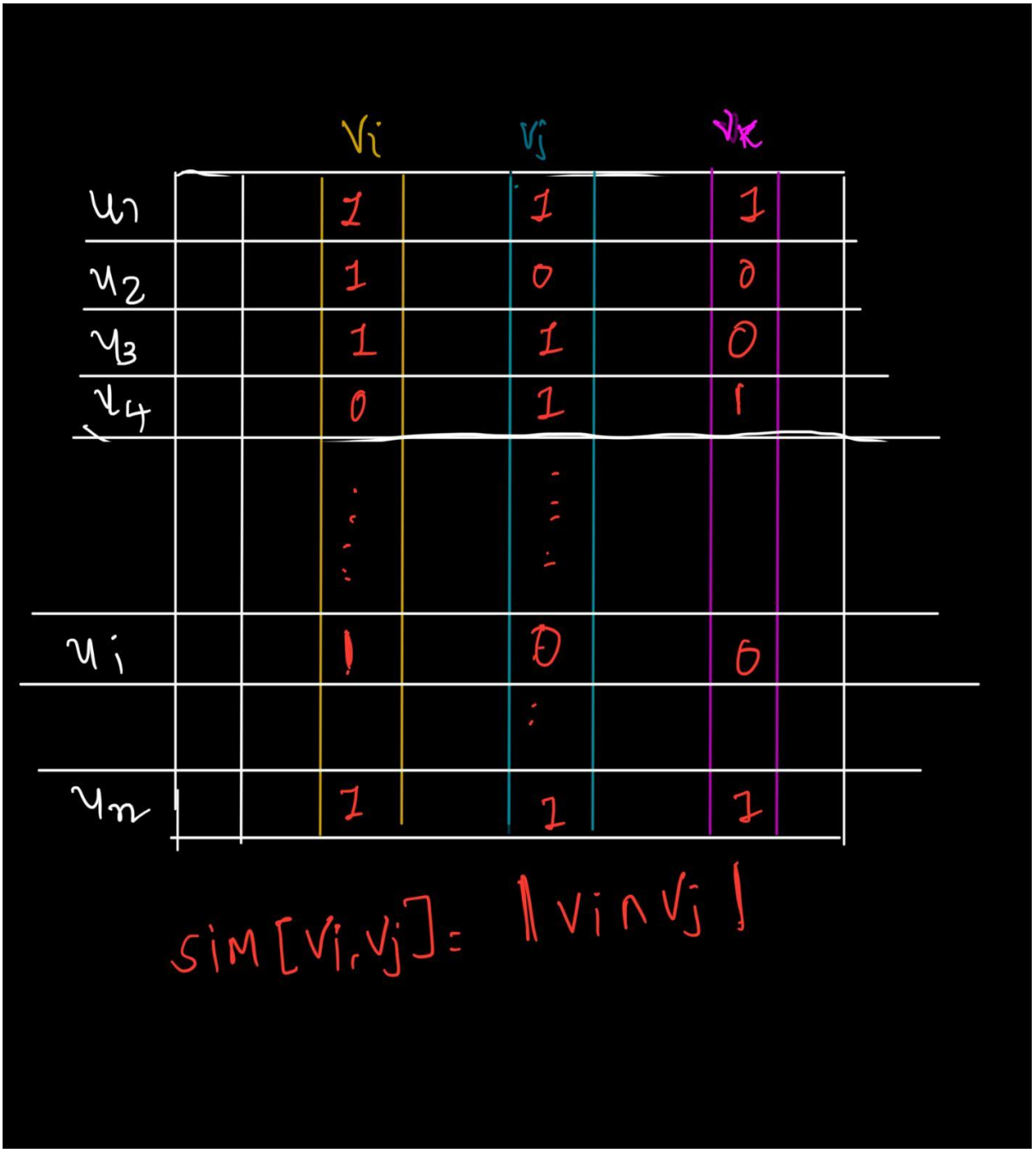
We will consider this problem as a similarity problem. We will recommend all the videos which are similar videos based on what the user is currently watching.



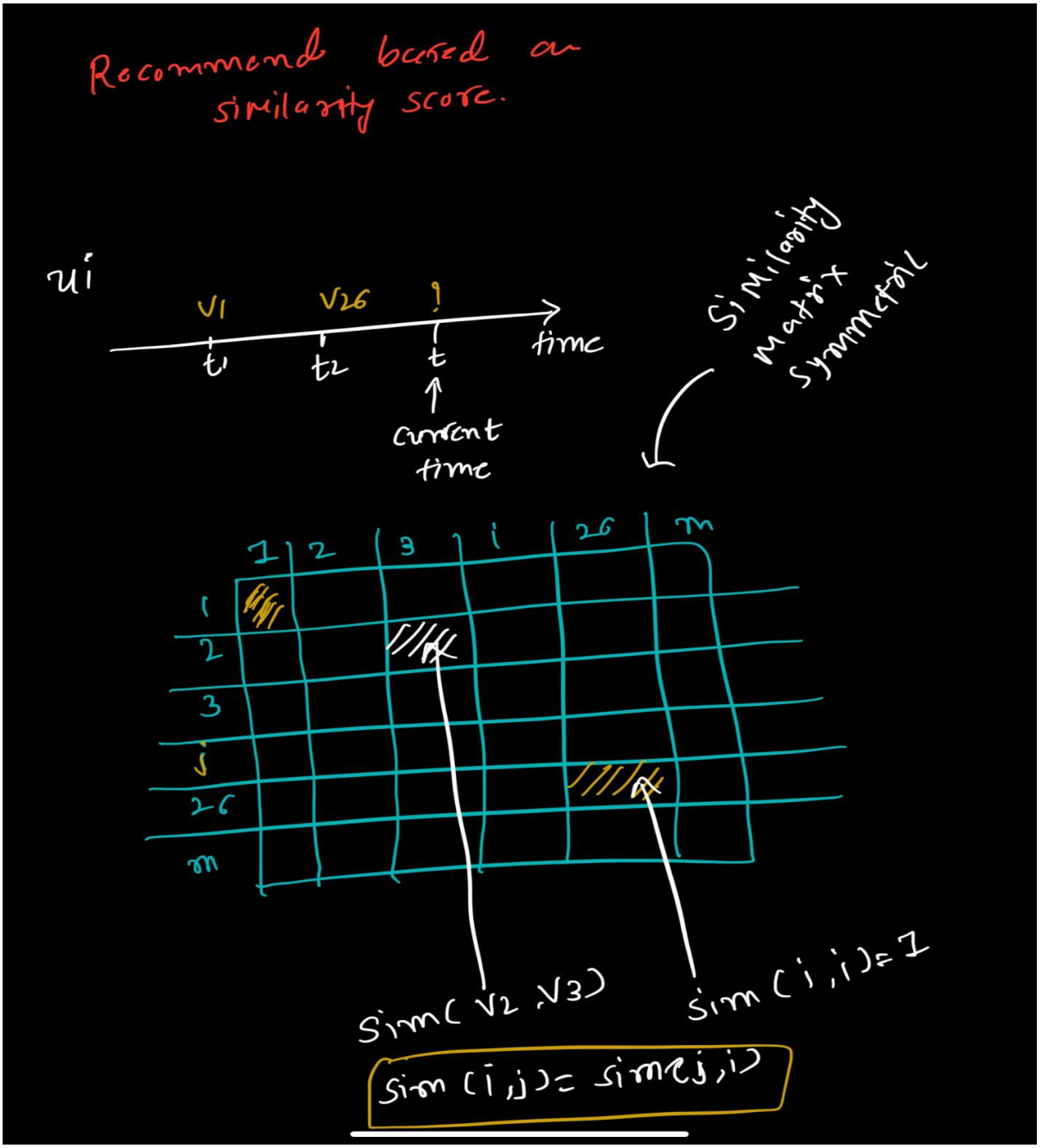
Challenge here is using the Data Matrix. How would we get a similarity score?

We will get top videos similar to what videos users have watched recently and then we will give higher ranking to all videos which are common among the top returned result and then we will return the rank list of videos.

Find Similar Videos based on Data Matrix



- **We can define a similarity score between v_i and v_j as a Dot product between two binary vectors.**
- Mathematically it is a **set intersection** ($|v_i \cap v_j|$) operation between two video vectors.
- v_i and v_j are binary vectors representing which users have watched v_i and which users haven't.
- If **Dot Product** between v_i and v_j $(v_i . v_j)$ is **high** then v_i and v_j are watched by many users which suggests that **they are similar videos** (liked by same kind of users)



	V_i	V_j	V_K
u_1	1	1	1
u_2	1	0	0
u_3	1	1	0
u_4	0	1	1
u_i	1	0	0
u_n	1	1	1

V_i = [1 | 1 | 1 | 0 | ... | 1 ... | 1]

V_j = [1 | 0 | 1 | 1 | ... | 0 | ... | 1]

V_K = [1 | 0 | 0 | 1 | ... | 0 | ... | 1]

$sim[v_j, v_j] = |V_i \cap v_j|| : v_i . v_j$

$sim[v_i, v_j] = 3$

$sim[V_i, V_K] = 2$

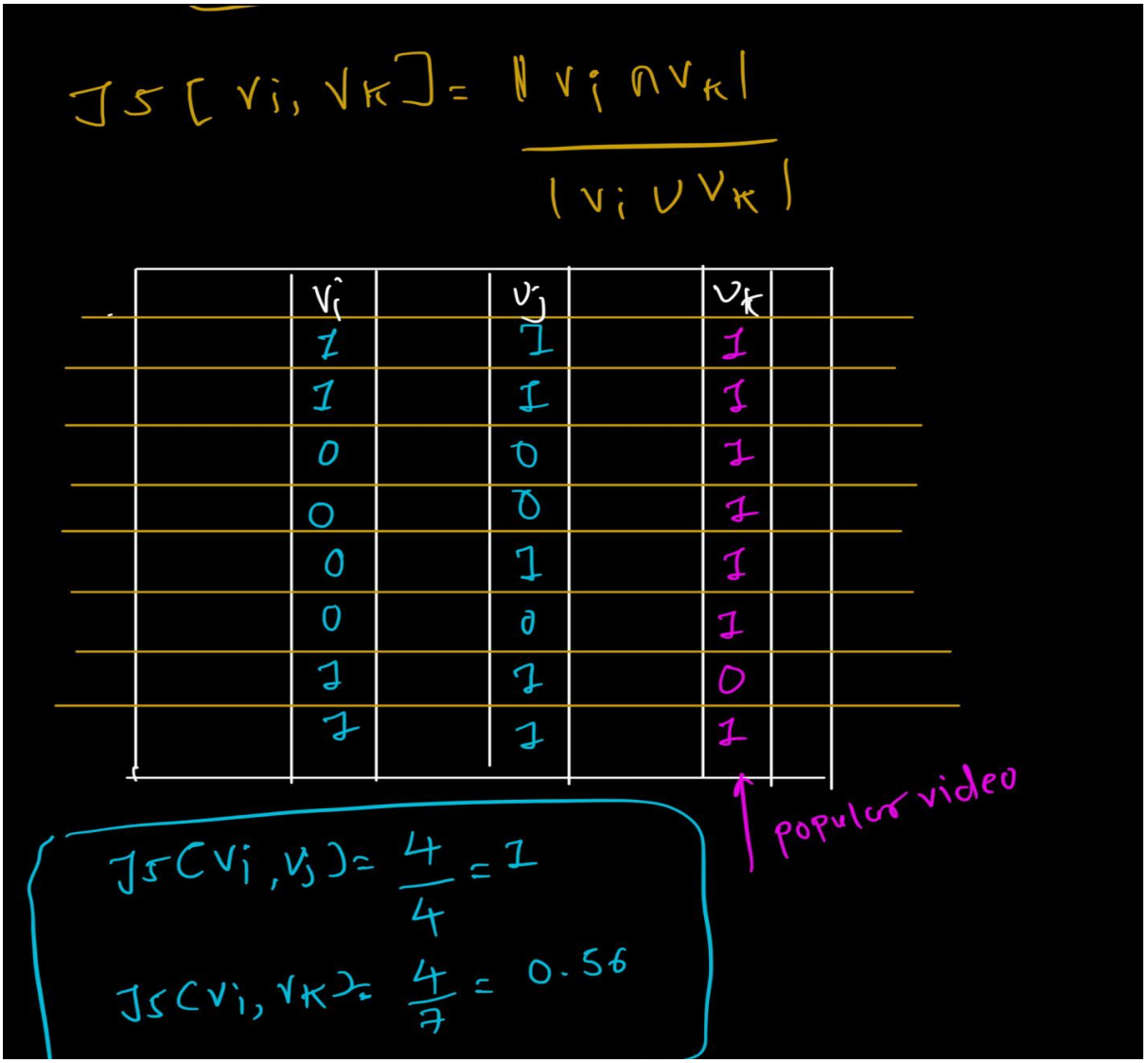
If we Recommend video For V_j then Ranking would be → V_j → Rank 1, V_i → Rank 2

Problem with Dot Product as similarity Score

- If we use **Dot product as a similarity score problem** occurs for **popular or most trending videos.**
- Most users will watch popular and trending videos. If we examine the vector for these videos, many entries will be 1 as it is watched by most users.
- Similarity score will be biased in that case and it will not represent the correct similarity score between two videos.
- Ranking will be affected by this. We would like to recommend videos based on what the user is watching currently.

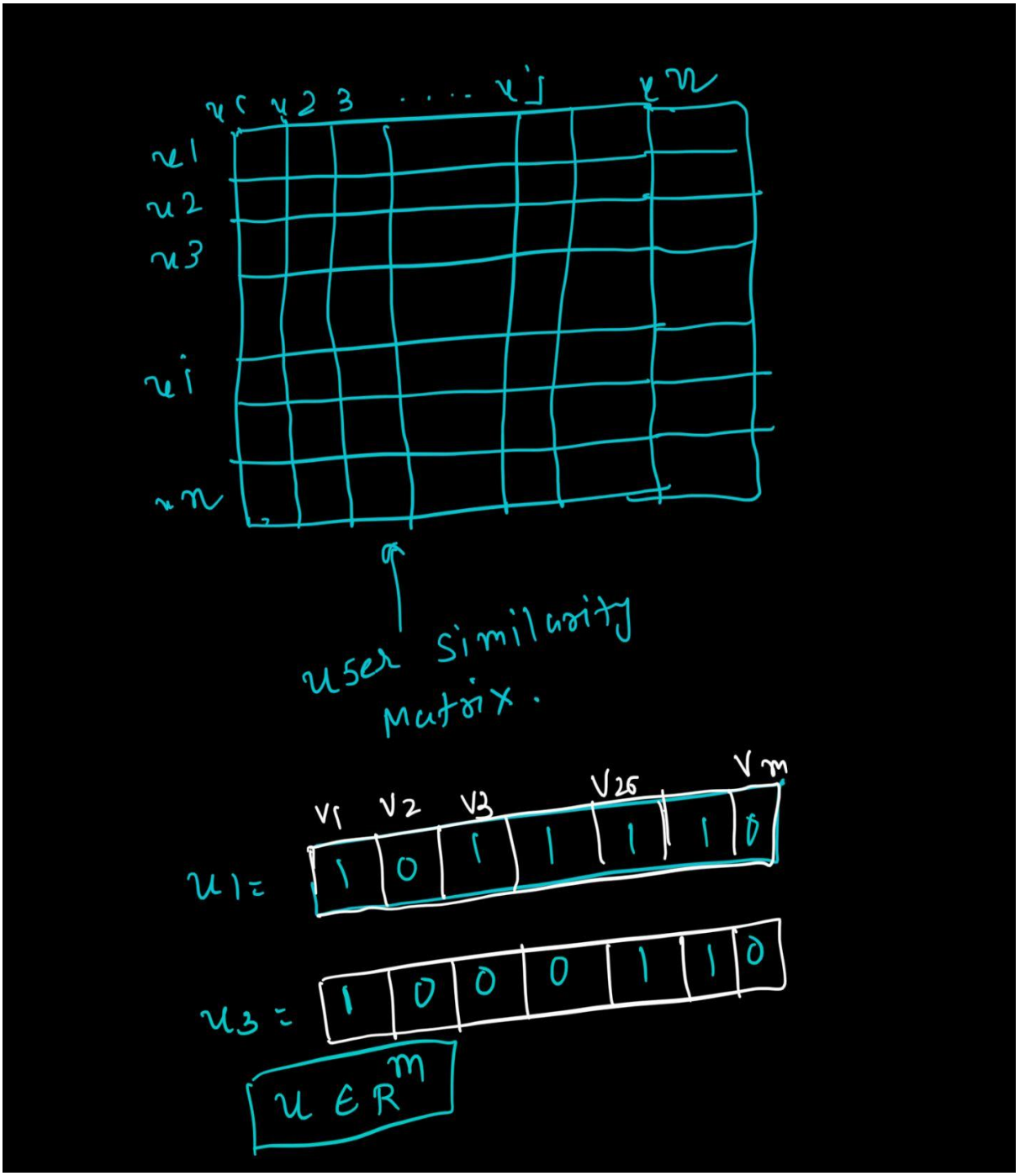
Solution : Jaccard similarity

Jaccard(V_i, V_j) $= |V_i \cap V_j| \div |V_i \cup V_j|$



- In the above example we can clearly see that video k is a very popular video and it is watched by most users. If we use only dot product as a similarity score we would rank video k and video j to the same as dot product will be 4.
    - $V_i . V_j = V_i . V_k = 4$
- Actually video j is very similar to video i, based on how many users watched both the video irrespective of whether it is popular or not.
- If we will use Jaccard similarity then we would rank video j higher than video k.
    - $JS(V_i, V_j) = 4/4 = 1$
    - $JS(V_i, V_k) = 4/7 = 0.56$

Different Approach : User User Similarity matrix



- We can formulate the problem from a user-user similarity. In that case we will generate user-user similarity matrix (R^{n*n}) .
- Each user vector will be in R^m. Each user vector will be a binary vector, where entry will be 1 if User has watched that video, else it will be 0.

Deployment and Productization

- For deployment and productization of the above solution we need to think in terms of **number of users, number of videos, and how many videos are added each week or day.**
- For Youtube we can say we have **1 Billion** (10^9) **users and we have around 10 millions videos (**10^7**) and each day thousands of videos are added to the system.**
- If We have maintained the **similarity matrix for videos** we have data matrix of size ($10^7 * 10^7 = 10^{14}$)
- , if we have maintained the **similarity matrix for users** we have a data matrix of size ($10^9 * 10^9 = 10^{18}$).
- To avoid the complexity of storing a large number of values, we can **store only a few top 100 similar videos or users.**

Other Advanced Approach

- We can use Eigenvalues and vector based approach like SVD and NMF.