Factorization

Let's say we have a number 42

Questions

Can we decompose this number into product of it's another smaller numbers?

Answers

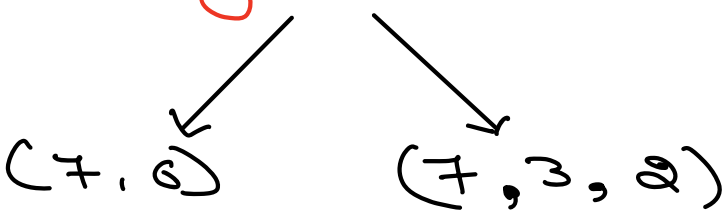
Yes! Ex:

① $42 = 7 \times 6$

② $42 = 7 \times 3 \times 2$

Definitions

These smaller numbers that we used for decomposing the larger Number are called factors



You Already knew this! Right?

Can we do the same for a matrix ?

i.e.

Can we decompose a large matrix into products of smaller Matrix ?

if you said : Yes 

Let's see how and Why ?

$$A_{n \times m} \Rightarrow U_{n \times d} \text{ and } I_{m \times d}$$

such that

$$A_{n \times m} = U_{n \times d} \times I^{\text{Transpose}}_{m \times d}$$

Note : d is a hyper-parameter and is also known as Rank or latent / Hidden dimension

Well this Looks good ! but Why ?

Answers Let's go back to Recommendation System.

Remember the data was represented using SPARSE MATRIX :

$A_{n \times m}$: where $n, m \rightarrow$ O(millions)

For example, Let's say

$n = 10{,}000{,}000$ (Users) (10^7)

$m = 10{,}000{,}000$ (items) (10^7)

This makes our matrix with size

$n \times m$

$10^7 \times 10^7 \Rightarrow 10^{14}$ Records

Now, Not all of these 10^{14} records !! are useful, infact most of these are sparse i.e. 0 or null

You know where we are going with this Right?

Can we decompose this huge matrix?

Lets pick a random value of

$d = 100$

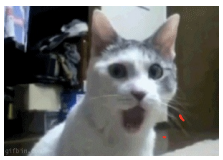
$$A_{n \times m} = U_{n \times d} \cdot I_{d \times m}$$

$$10^{14} \text{ records} = 10^7 \times 100 + 100 \times 10^7$$

$$10^9 + 10^9 \Rightarrow 2 \times 10^9$$

i.e. a reduction of:

$$\frac{10^{14}}{2\times10^{9}} = \frac{1}{2}10^{5}$$!!



i.e. Lots of memory and compute power saved to process and store same amount of information

---

So, How do we find these U and I matrix such that $A = U \cdot I^{Transpose}$

$$P_{n\times m} \Rightarrow U_{n\times d} \cdot I_{d\times m}$$

(Predicted Rating Matrix)

Hypothesis $\Rightarrow A_{ij} \approx P_{ij}$

For ex:

The Rating User 2 will give to item 3 can be calculated as

$$P_{23} = U_2 \cdot I_3 \approx A_{23}$$

5) From above Hypothesis, we can create Loss functions to minimize

$$A_{n \times m} \quad P_{n \times m}$$

↓

MSE

$$\sum_{i,j} (A_{ij} - U_i \cdot I_j)^2$$

I_j → output from predicted interaction

S.t. $A_{ij} \neq$ Null

Note: We calculate Loss for only those cells where user has interacted with items

Now, we can write our optimization equation as

Optimize $\underset{B_i, C_j}{\text{arg min}} \sum_{i,j} \left(A_{ij} - U_i I_j \right)^2$

S.t. $A_{ij} \neq$ Null

* We can optimize above equation now with

Method ①

SGD to find value of U_i and I_j

Step 1: Initialize matrix U and Matrix I randomly

Step 2: Take derivative of loop and update U and I

Remember: We calculate loss only for values where user has given some ratings

---

Method 2:

Co-ordinate Descent Algorithm

also known as

Alternate Least Square

① Random initialization

② Calculate Loss

③ We fix value U for some iteration and update I such that loss Reduce

④ After few iteration freeze U and update I

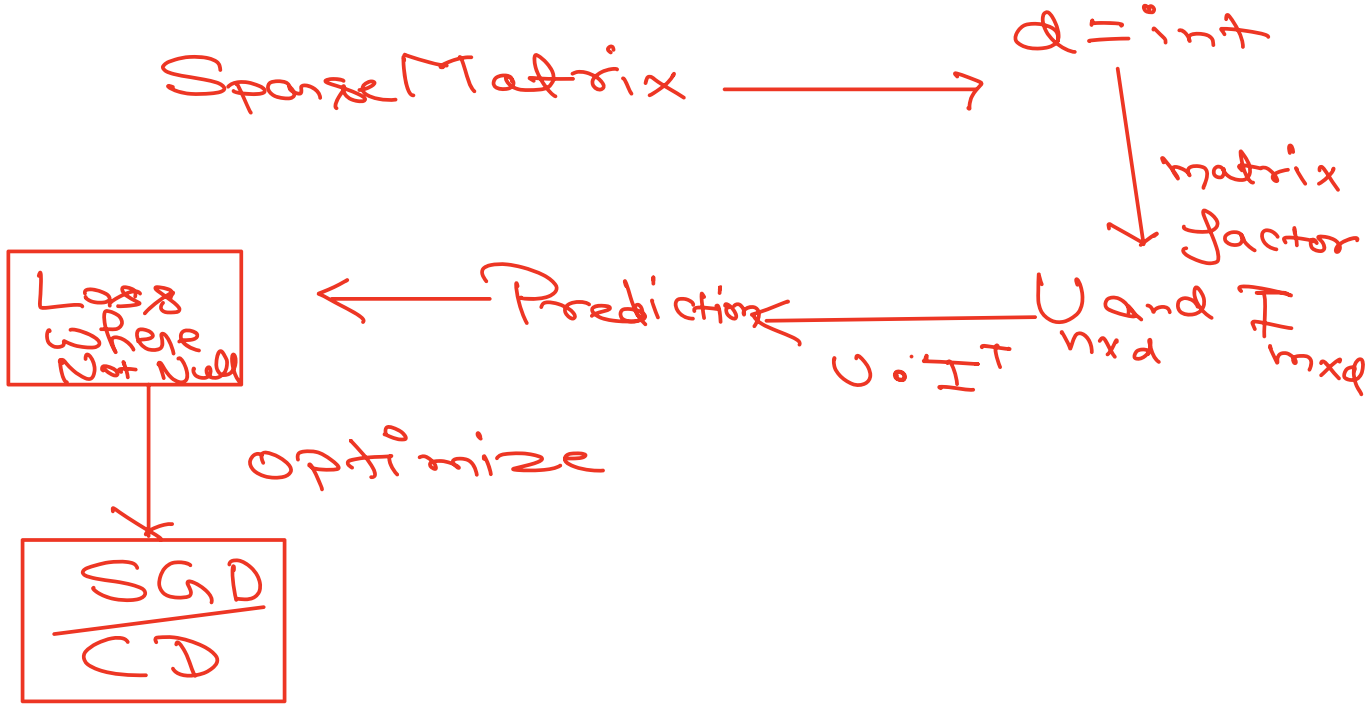
⑤ Repeat until Convergence

Potential Drawbacks with Matrix Factorization

Cold Start Problem

A new user or user that never rates

Overall Flow



For now, Let's go to a challenging Exercise to understand How prediction works for a given Interaction matrix after it's decomposed into Factors

Let's say we are given Interaction matrix A, and we have found U and I matrix as following:

$A_{n \times m} =$

	m_1	m_2	m_3
u_1	2	4	5
u_2	3	1	2
u_3	?	?	5
u_4	1	2	1

$U_{n \times 2}$

u_1	0.8	0.5
u_2	0.2	0.9
u_3	1.2	0.8
u_4	1.1	0.7

$I^T_{2 \times m}$

m_1	m_2	m_3
2.3	0.7	0.5
1.5	2.1	3.1

Where the values stored in matrix A are Ratings given by User$_i$ to movie j.

Note:
?: represents movies, user haven't seen yet.

Questions

1. What is the value of Latent dimension?
2. Calculate what Rating the user U_3 will provide to movie m_1 and m_2 respectively?
3. Which movie among the two shall be recommended to the user U_3 first?

Hint : $A = U \cdot I^T$

Solution: Fill the values of ? Here:

2.3	0.7	0.5
1.5	2.1	3.1

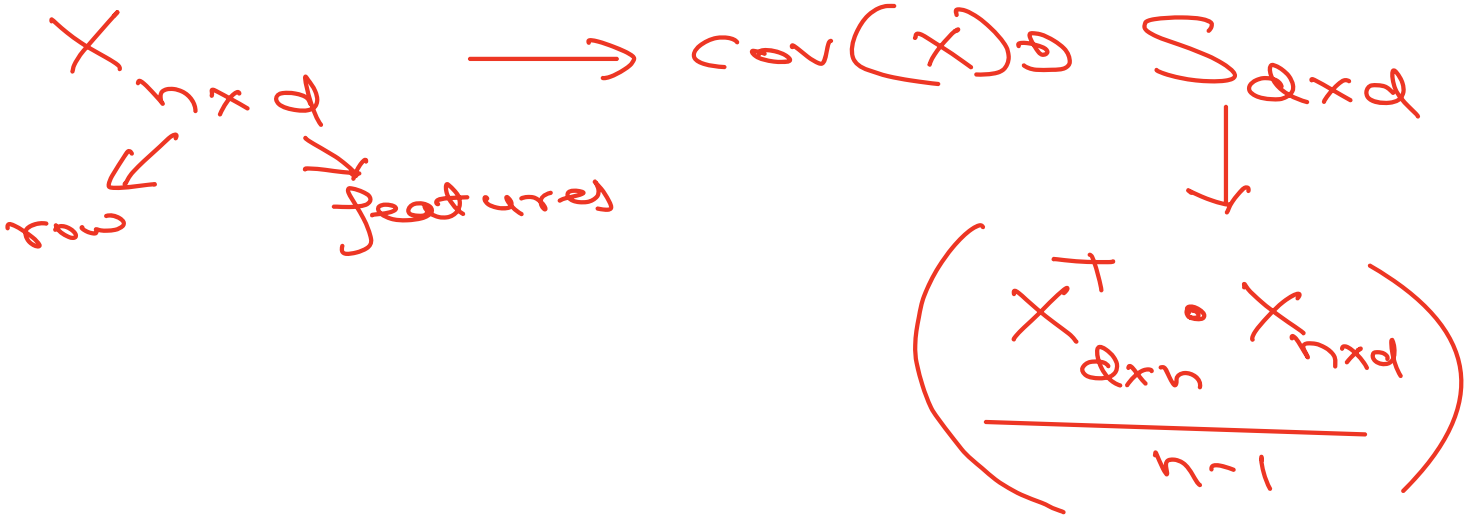
U_3

0.8	0.5
0.2	0.9
1.2	0.8
1.1	0.7

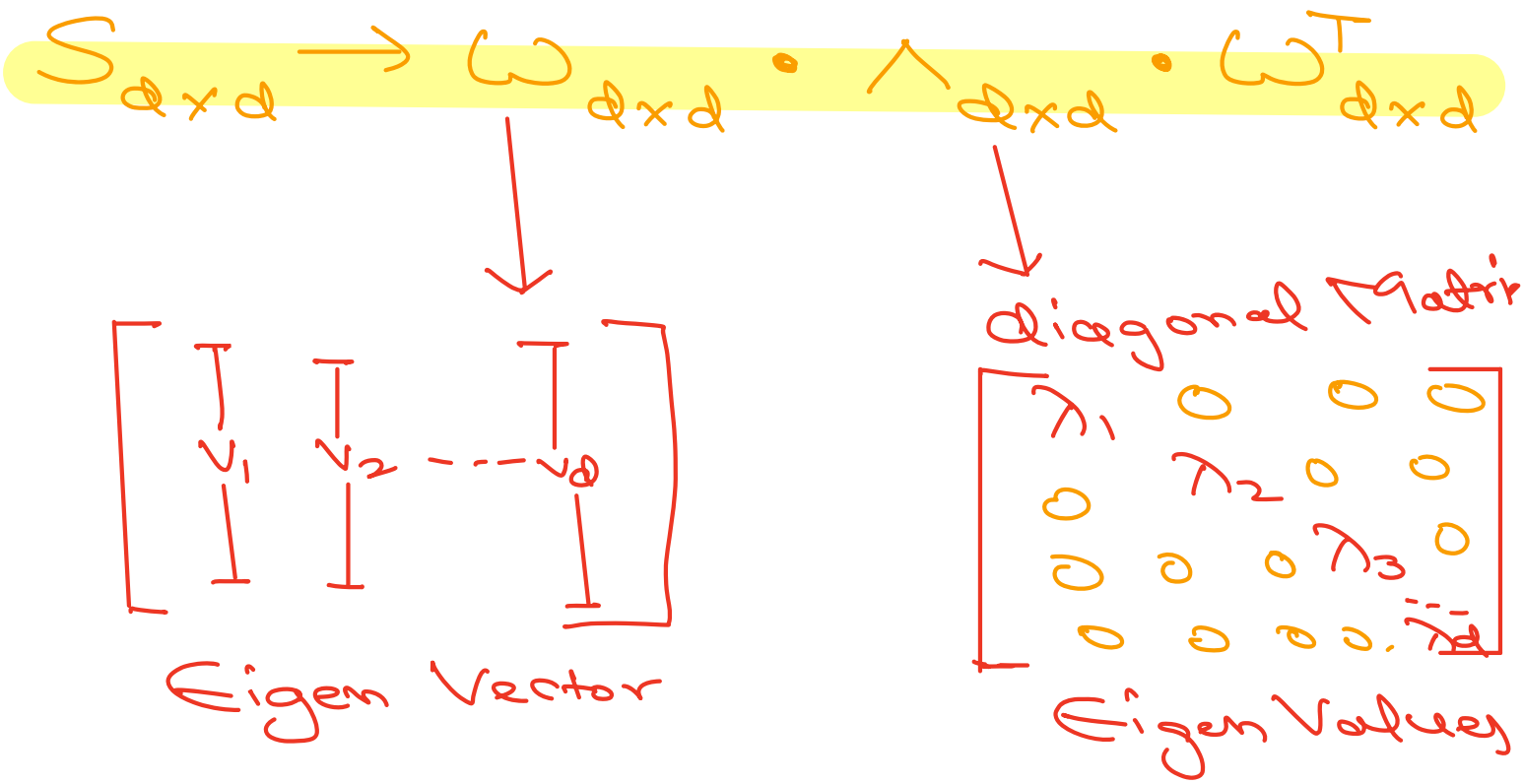
?	?	

$$\text{Arg max} \begin{pmatrix} R_1 & R_2 \\ m_1 & m_2 \end{pmatrix}$$

MF in PCA



E.V.D : Matrix Needs to Square



* Special case of MF
  * Constraint 1 : V's are Orthogonal's
  * Constraint 2 : Λ is diagonal

SVD

- MF: For Non-Square Matrices