

# Recommender Systems || Lecture 4

## Summer 2022

Dr. Karthik Mohan

Univ. of Washington, Seattle

July 6, 2022

# Logistics

- 1 Conceptual Assignment 1 due this Saturday

# Logistics

- 1 Conceptual Assignment 1 due this Saturday
- 2 Programming 2 will be assigned shortly

SVD Matrix Factorization  
↳ MovieLens

# Logistics

- 1 Conceptual Assignment 1 due this Saturday
- 2 Programming 2 will be assigned shortly
- 3 **Guest Lecture Monday:** By Director of Data Science at Google on practical applications of recommender systems

# Logistics

- 1 Conceptual Assignment 1 due this Saturday
- 2 Programming 2 will be assigned shortly
- 3 **Guest Lecture Monday:** By Director of Data Science at Google on practical applications of recommender systems
- 4 Next Wednesday Lecture: Maybe pre-recorded due to travel

# Logistics

- 1 Conceptual Assignment 1 due this Saturday
- 2 Programming 2 will be assigned shortly
- 3 **Guest Lecture Monday:** By Director of Data Science at Google on practical applications of recommender systems
- 4 Next Wednesday Lecture: Maybe pre-recorded due to travel
- 5 4 conceptual + programming assignments + in-depth project instead of 6 assignments + 1 mini project - Sounds good?

# Project

- 1 Proposal based projects

# Project

- 1 Proposal based projects
- 2 Project proposal due next week - Guidelines will be shared shortly  
(Scope / Timeline)



# Project

- ① Proposal based projects
- ② Project proposal due next week - Guidelines will be shared shortly
- ③ Will have 2-3 checkpoints on the way to final submission

# Project

- ① Proposal based projects
- ② Project proposal due next week - Guidelines will be shared shortly
- ③ Will have 2-3 checkpoints on the way to final submission
- ④ **Goal:** Gain depth in a topic of choice within recommender systems

# Project

- 1 Proposal based projects
- 2 Project proposal due next week - Guidelines will be shared shortly
- 3 Will have 2-3 checkpoints on the way to final submission
- 4 **Goal:** Gain depth in a topic of choice within recommender systems
- 5 **Your Submission:** Github code check in, White paper submission  
and Final project presentation

↳ Conference papers

# Project

- 1 Proposal based projects
- 2 Project proposal due next week - Guidelines will be shared shortly
- 3 Will have 2-3 checkpoints on the way to final submission
- 4 **Goal:** Gain depth in a topic of choice within recommender systems
- 5 **Your Submission:** Github code check in, White paper submission and Final project presentation
- 6 **Ideally:** Will add to your resume and something you can talk about in depth in a Machine Learning interview!

# Last two Lectures

- 1 Recommender baseline based on Cosine Similarity ✓
- 2 Breakout discussions on real use cases ✓
- 3 Distance vs similarity metrics ✓
- 4 Behavioral vs Content based recommendations } ✓
- 5 Lecture 3 was super short since we wanted to have breakout sessions (hopefully today!)

# Today's Lecture

- 1 Bag of words model for content based recommendations
- 2 SVD and Matrix Factorization methods



# Week by Week Break Down (Tentative)

Week	Lecture Material	Assignment
1	Intro to Recommender Systems	Amazon case study
2	Recommender System Baselines	Shopify case study
3	<u>Matrix Factorization methods</u>	<u>Home Depot</u> case study
4	Matrix Factorization methods	Twitter case study
5	Deep Learning based recommendations	Final Project
6	ML Pipeline for Recommender systems	Final Project
7	Real-time Recommendations	Final Project
8	Diversity and Relevance	Final Project
9	Scaling Recommender systems	Final Project

# Simple Baseline Recommender

## Cosine Similarity

Let's say you want to recommend products from some categories (e.g. groceries, music and fitness to customers) in a way that is personalized and so they find it relevant. Here's a simple baseline model:

- 1 **Representation:** Find a way to represent your product by a fixed length vector of some dimension  $d$ .  $d$  here usually captures the number of relevant "features".



# Simple Baseline Recommender

## Cosine Similarity

Let's say you want to recommend products from some categories (e.g. groceries, music and fitness to customers) in a way that is personalized and so they find it relevant. Here's a simple baseline model:

- 1 **Representation:** Find a way to represent your product by a fixed length vector of some dimension  $d$ .  $d$  here usually captures the number of relevant "features".
- 2 **Cosine Similarity:** Let's say you have a reference product  $p$  and candidate products to recommend  $p_1, p_2, \dots$ . Evaluate the cosine similarity between product  $p$  and all the other products.

# Simple Baseline Recommender

## Cosine Similarity

Let's say you want to recommend products from some categories (e.g. groceries, music and fitness to customers) in a way that is personalized and so they find it relevant. Here's a simple baseline model:

- ① **Representation:** Find a way to represent your product by a fixed length vector of some dimension  $d$ .  $d$  here usually captures the number of relevant “features”.
- ② **Cosine Similarity:** Let's say you have a reference product  $p$  and candidate products to recommend  $p_1, p_2, \dots$ . Evaluate the cosine similarity between product  $p$  and all the other products.
- ③ **Rank:** Rank the candidates for recommendation based on cosine similarity scores (highest to lowest).

# Simple Baseline Recommender

## Cosine Similarity

Let's say you want to recommend products from some categories (e.g. groceries, music and fitness to customers) in a way that is personalized and so they find it relevant. Here's a simple baseline model:

① **Representation:** Find a way to represent your product by a fixed length vector of some dimension  $d$ .  $d$  here usually captures the number of relevant "features".

→ Generation

② **Cosine Similarity:** Let's say you have a reference product  $p$  and candidate products to recommend  $p_1, p_2, \dots$ . Evaluate the cosine similarity between product  $p$  and all the other products.

③ **Rank:** Rank the candidates for recommendation based on cosine similarity scores (highest to lowest).

→ Filter

④ **Truncate:** Pick top  $k$  from the ranked list to recommend. Where  $k$  is usually decided by specifications of your recommender widget.

# Behavioral vs Content Based Recommendations

- ① **Behavioral:** Recommendations on understanding customer behaviors based on data (purchases, views and other data).  
**Collaborative Filtering** is a modeling approach in recommendations that is solely dependent on behavioral data.  
*Customers who bought this also bought this..* is an example of a behavioral recommender.
- ② **Content:** Recommenders that use content information on items or products are called content based recommenders.  
*Items similar to what you purchased . . .* is an example of this.
- ③ **Hybrid:** Often times, a mixture of behavioral and content based data leads to better overall recommender widgets.

# Content based recommendations

- 1 Use features (or content) from the products to make recommendations.

Related to items you've viewed [See more](#)



# Content based recommendations

- 1 Use features (or content) from the products to make recommendations.
- 2 This includes product categories and product brand

Related to items you've viewed [See more](#)



# Content based recommendations

- 1 Use features (or content) from the products to make recommendations.
- 2 This includes product categories and product brand
- 3 Anything else?

Related to items you've viewed [See more](#)



# Breakout discussion (from previous lecture)

## New product on the block

Your team is launching new products and product categories to the massive product line of Sambazon. You are the new data scientist hired to surface these new products to customers so your team can drive the sales for the launched products. As of now, there is no easy way for a customer to see the new products, as it isn't yet popular and also because the new products have very little sales to be surfaced by other widgets that exist on the Sambazon recommendations pipeline. How would you leverage what you know about the new product line including any product descriptions and meta data to drive up the sales for the products. Assume the new products are in a product category that a significant percentage of Sambazon customers have purchased from in the past. Come up with a clear step-by-step model/algorithm/approach and also discuss how you will evaluate your recommender model?



# Content Recommendations Baseline: Bag of Words Representation

## Bag of Words

When we have a dictionary of words and any sentence represented as a binary vector of the dictionary length, where a 1 represents a word present in the sentence and 0 if not - This is a bag of words representation.

# Content Recommendations Baseline: Bag of Words Representation

## Bag of Words

When we have a dictionary of words and any sentence represented as a binary vector of the dictionary length, where a 1 represents a word present in the sentence and 0 if not - This is a bag of words representation.

## Bag of Words Example

Let the dictionary of words be  $\{hello, bye, later, today\}$  and a sentence be "Hello, how are you doing today?" What would be the bag of words representation for the sentence (minus the punctuations)?

$\rightarrow [1 \ 0 \ 0 \ 1] \in \mathbb{R}^4 \rightarrow 4 \text{ dim sep. Bow for sentence!}$

# ICE #1

## Bag of words for content representation

Let a dictionary consist of the following words:

{*fitness, women, sports, clothing, clothes, bags, wear*}.

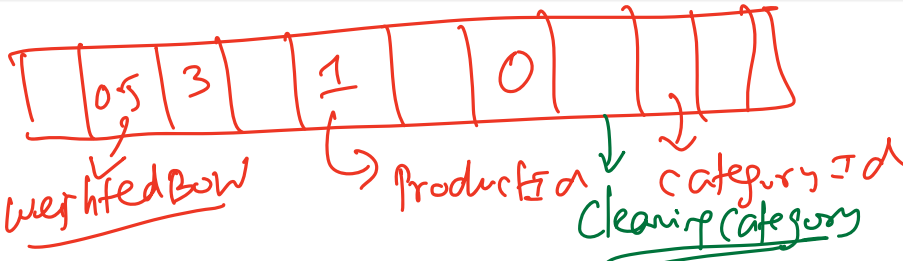
Assume there is a new Sambazon product that just came out today with the following one line description: “The *Avo fitness wear series 9* brings to you fitness clothing exclusively for women athletes and health conscious women looking to up their game.” The dimension of the bag of words representation vector and the number of non-zero elements in the representation are respectively:

- ① 6 and 3
- ② 6 and 4
- ③ 7 and 4
- ④ 8 and 3

# Breakout discussion #1

## Bag of words (BOW) for behavioral representation

Let's say we have purchase history data on customers - Specifically which product was purchased when by which customer. Assume we do not have any content data available on the products (e.g. description, brand, category, etc). We still want to represent each customer by a vector. Think of a bag of words (BOW) representation that can give a fixed length vector representation for each customer, specifically based on behavioral data. What would be the pros/cons of your representation?



- Filters by popularity
- clusters
- available categories

# Breakout discussion #2

## BOW for Fashion

Let's say you have fashion products line where what's fashionable last year may not be fashionable today. You still want to leverage past purchase data to recommend fashion clothing to customers. Given that you have the purchase and view history data of fashion products for each customer for the past one year - Can you think of an effective BOW representation to represent a customer? What would be the length of the representation? How about when we want to have an effective BOW representation for a product - How would you design this? How do you know a representation is effective or not?

tested on  
Text for BERT → 5 different tasks  
↓  
(Language Model)

# BOW summary

- 1 BOW gives a starter representation for any model

# BOW summary

- 1 BOW gives a starter representation for any model
- 2 BOW data representation enables us to get started with baseline recommenders both for content and behavioral models

→ Must DL models begin with BOW in layer!  
→ Deep Learning Model

# BOW summary

- ① BOW gives a starter representation for any model
- ② BOW data representation enables us to get started with baseline recommenders both for content and behavioral models
- ③ For many data sets, BOW can lead to heavily sparse representations (lots of zeros) making the representation unwieldy

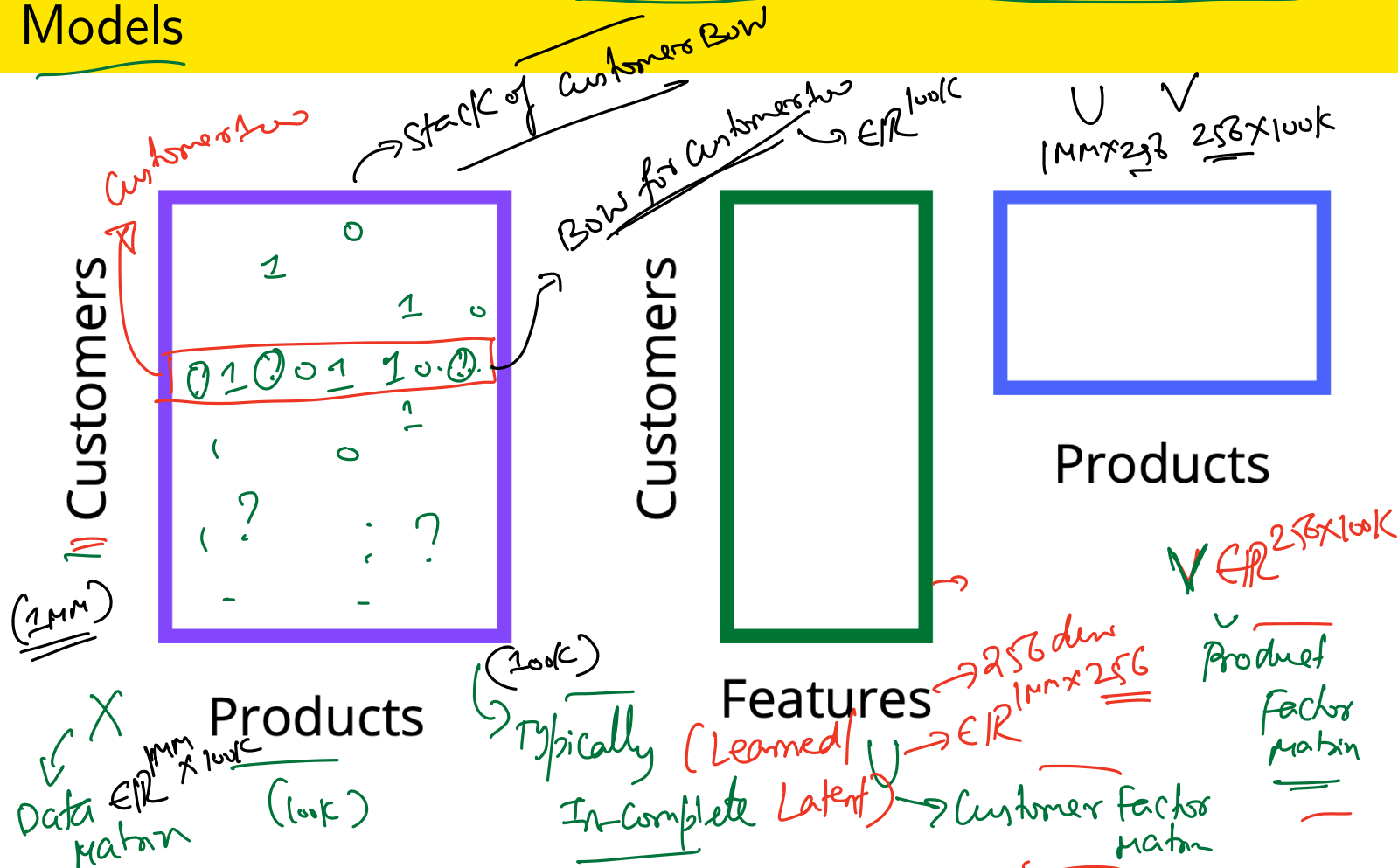


# BOW summary

- 1 BOW gives a starter representation for any model
- 2 BOW data representation enables us to get started with baseline recommenders both for content and behavioral models
- 3 For many data sets, BOW can lead to heavily sparse representations (lots of zeros) making the representation unwieldy
- 4 In Deep Learning models, even if BOW is a starter representation, the DL models learn more refined, data driven, learned and compact representations for both customers and products (e.g. 256 dim instead of 100k dim representation)

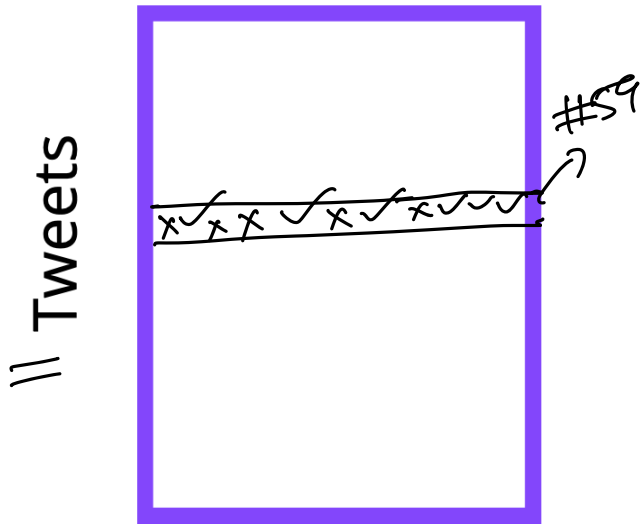
Price to pay:- # weights learned is huge!!  
100k x 256 ≈ 25MM!

# Matrix Factorization: Refined Behavioral Recommendation Models



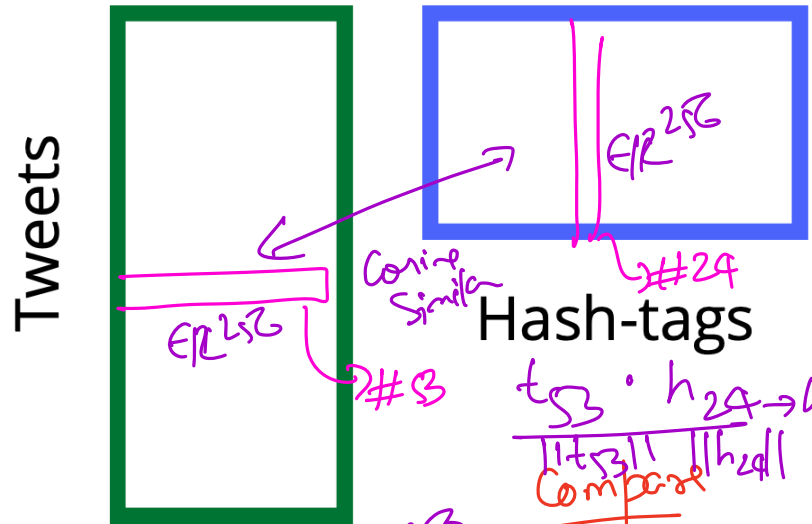
# Matrix Factorization: Refined Behavioral Recommendation Models

Twitter's Recs



Hash-tags

Tweet: - "I enjoyed hikeup today"  
 → #hike #hikeup



Features

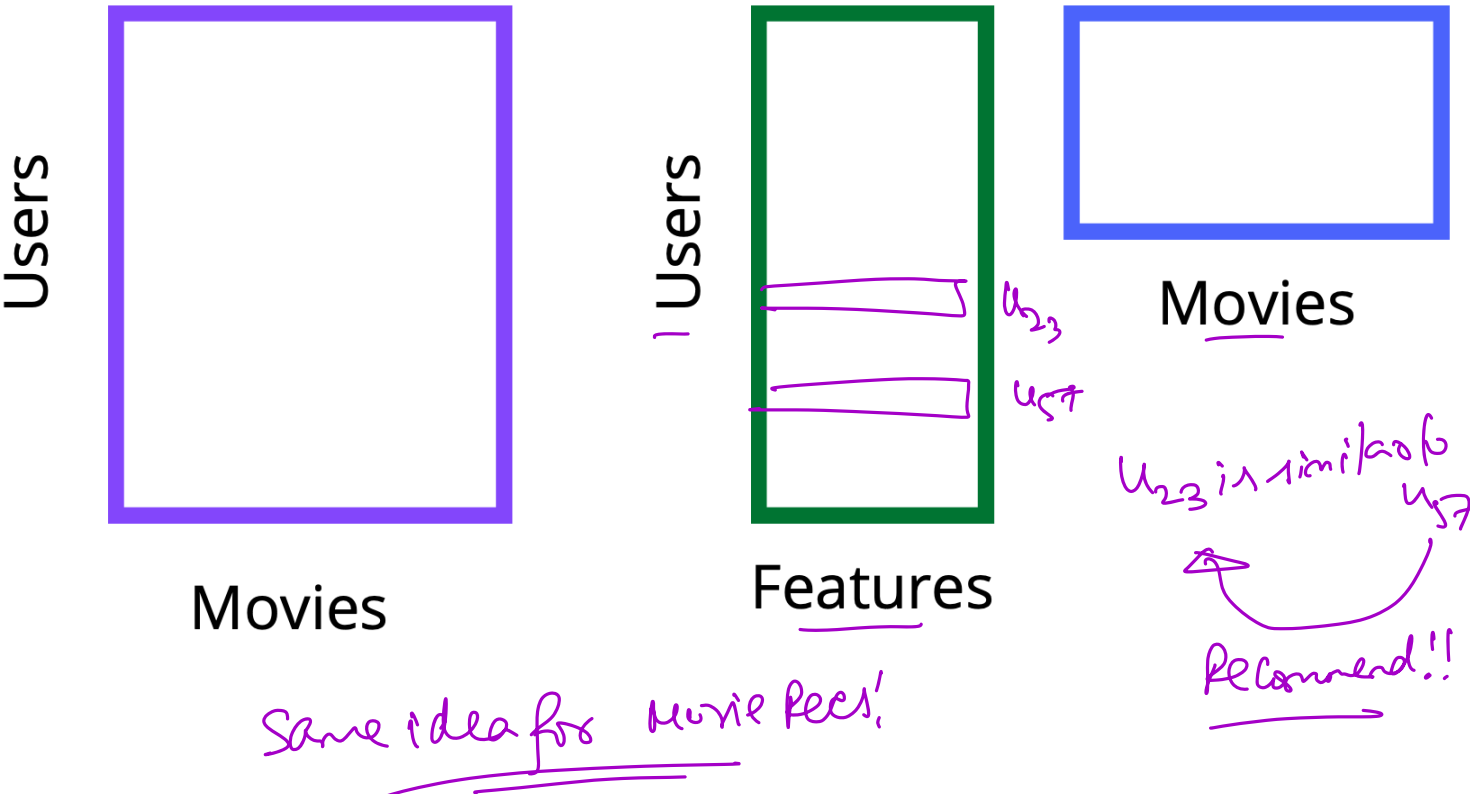
Hash-tags

$$t_{53} \cdot h_{24} \rightarrow \text{Cosine Similarity}$$

Compare

1. Tweet 1 vs Tweet 59
2. Tweet 23 vs Hash-tag 57
2. Hash-tag 24 vs Hash-tag 117

# Matrix Factorization: Refined Behavioral Recommendation Models

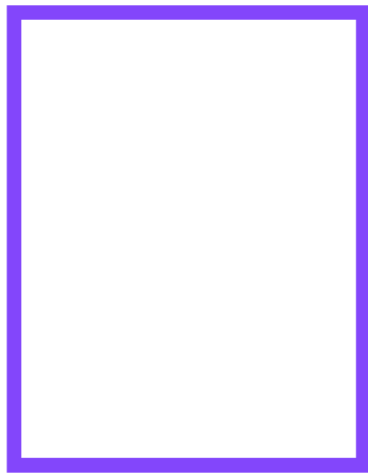


# Matrix Factorization: SVD - Baseline Model

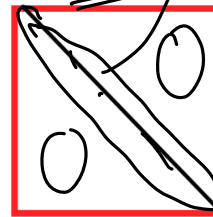
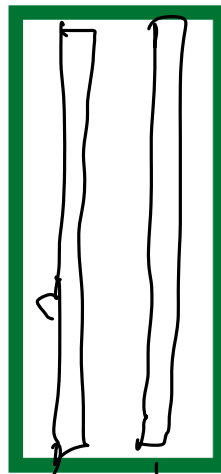
Complexity of SVD:  
 $O(mn^2)$

S - Singular  
V - value  
D - Decomposition

Square  $\rightarrow$  Singular values  $\geq 0$



=



X  $n \times d$

$u_3$   $U$   $u_7$   
 $n \times n$   
Orthogonal

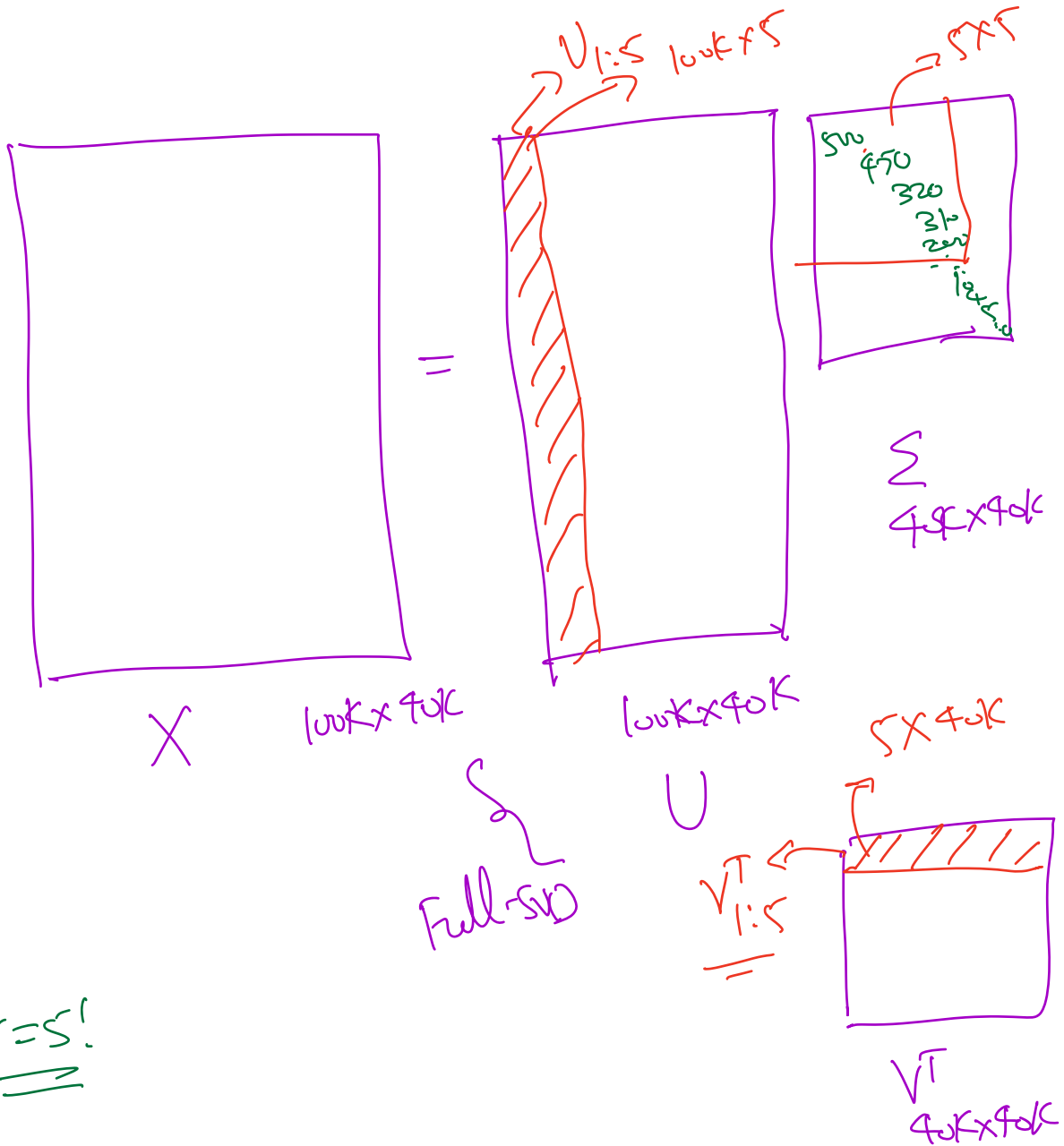
$\Sigma_{r \times r}$   
(diagonal matrix)

$V^T$   
 $r \times d$

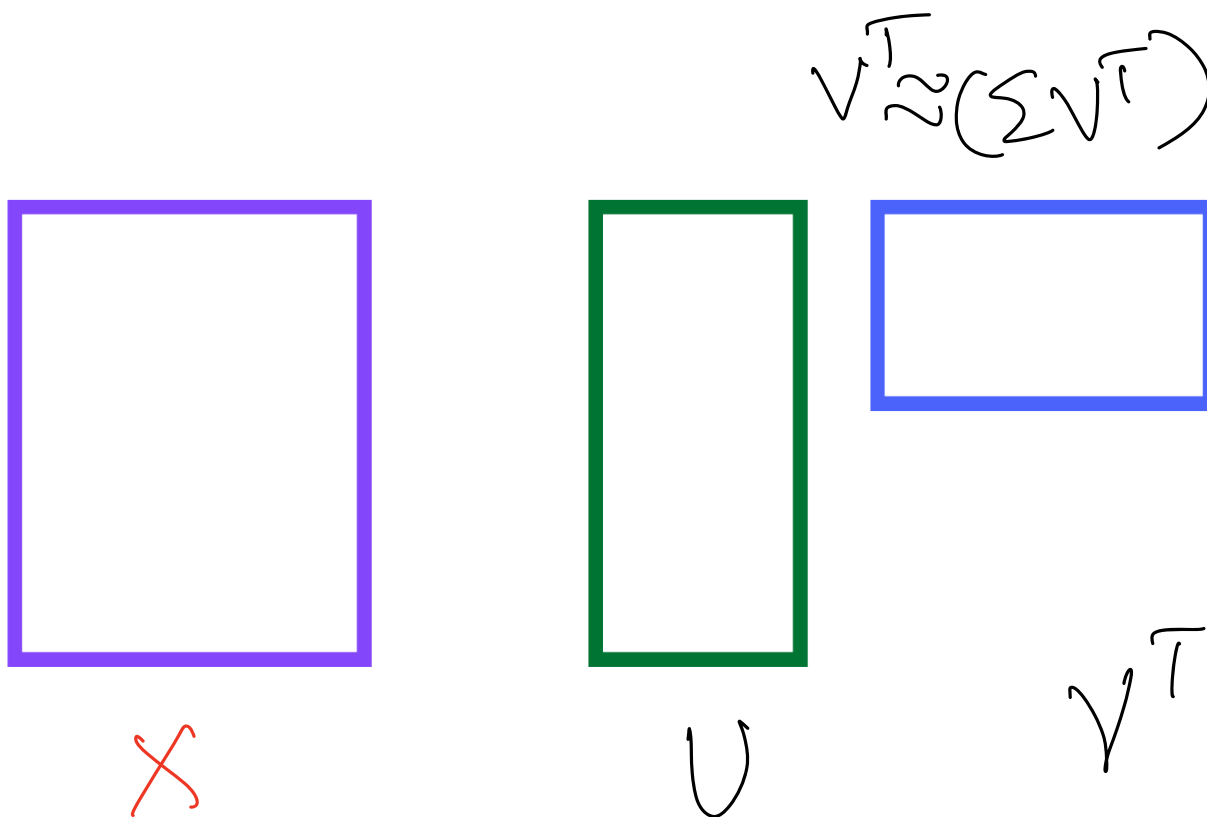
$r \ll n$  and  $d$   
 $\rightarrow$  reduced SVD Truncated SVD

$u_3 \cdot u_7 = 0!$  (orthogonal)  
 $\|u_3\|_2 = 1$   $\|u_7\|_2 = 1$

# Full SVD $\Rightarrow$ Truncated SVD

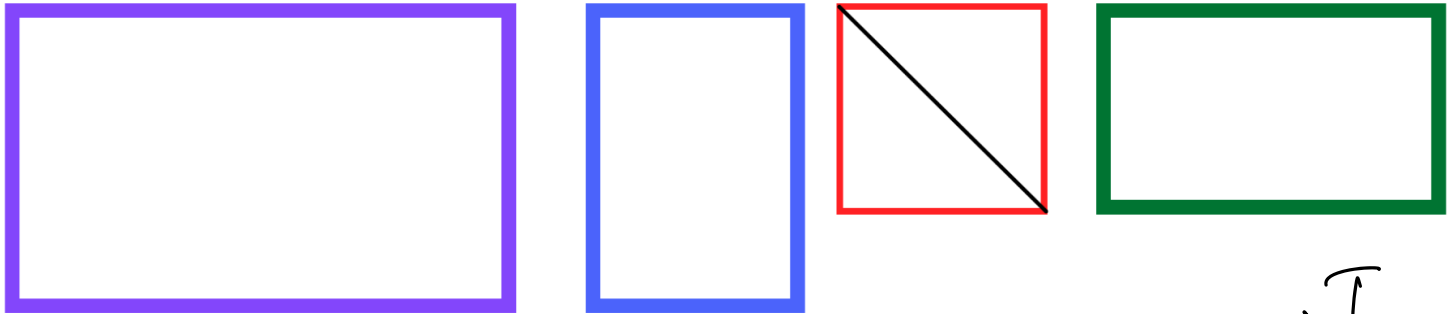


# Matrix Factorization: SVD - Baseline Model



# Matrix Factorization: SVD - Baseline Model

$$X = U \Sigma V^T \quad | \quad X^T = V \Sigma U^T$$



$X^T$

$V$

$\Sigma$

$U^T$

Transposing  $X \Rightarrow$  Reversed factors!!



# SVD vs Eigen Decomposition

1. SVD is fundamental

2. SVD on  $X X^T \rightarrow$  Eigen factors!

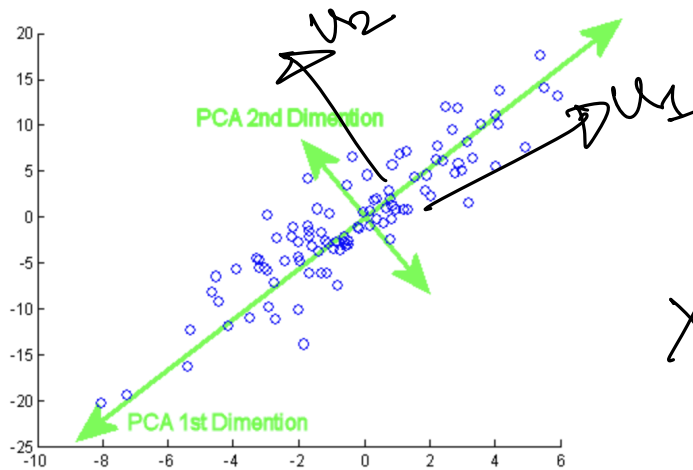
Eigen-values can be negative!

3. Eigen-Decomp only applies to  
square matrices

Eigen-faces / Google it!

# PCA and SVD

SVD factors are orthogonal

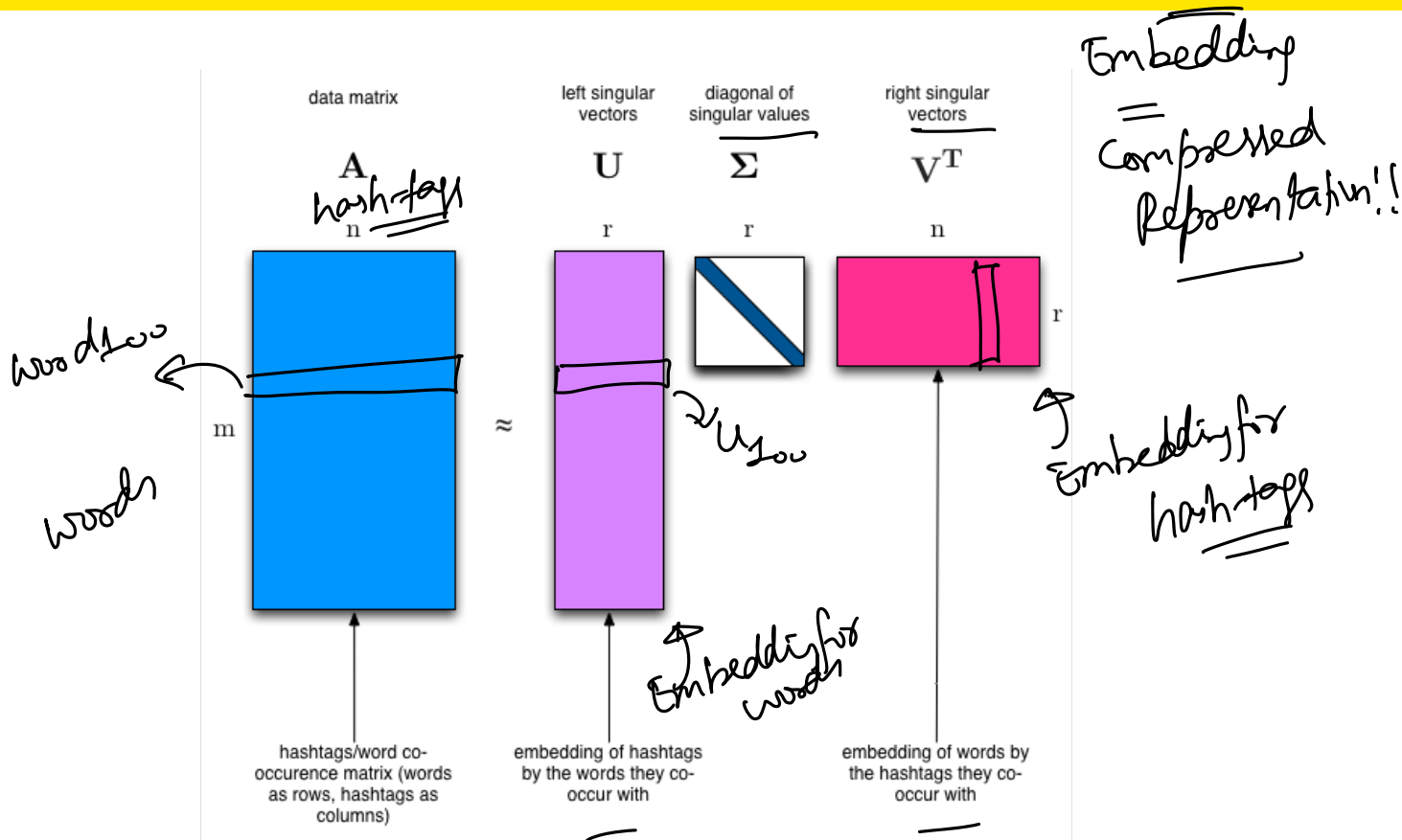


$$X = U \Sigma V^T$$

↓

$u_1$  - First singular vector  $\rightarrow$  Direction of max. variance!  
 $u_2$  - Second  $\perp$  Eigen-vectors!

# Matrix Factorization: SVD for Tweet embeddings and recommendations



# Similarity Search using Embeddings

## Breakout discussion #3

Assume that you would like to build a tweet recommender system to your customer base. You have two models for recommendations: a) First model does similarity search on the “raw customer-tweet views matrix”. I.e. for any tweet a customer has viewed, you search for similar tweets based on cosine similarity using the raw BOW representation of the tweet. Assume the matrix on an average has 1000 non-zeros per row and 10000 non-zeros per column

b) You use the factors coming off of a SVD model with 250 embedded dimensions to do similarity search

## Similarity Search using Embeddings (continued)

By what factor would method b) be faster on an average than method a) assuming you were finding top 10 tweets to recommend for each of a thousand reference tweets. Also assume speed is measured based on number of computational flops (as compared to an actual code run comparison).

- a) 30
- b) 40
- c) 50
- d) 100

# BOW vs Learned Representations

- 1) BOW - Starting point
  - 2) Learned representations (GVD) DL
- Data driven & are better for similarity search!

# Breakout discussion #4





## Handling Missing Values

When we have a matrix of customers and their product purchases - It's straightforward to apply SVD and get customer embeddings and product embeddings from the factored matrices. This enables us to search for similar products given a reference product, or similar customers given a reference customer. However, there is a catch here. The number of products in a big retail company may number in hundreds of thousands and the number of customers might number in millions. So we are looking at  $1MM \times 100k$  matrix or larger. Do we have a value for every cell in this matrix? The answer is no, as customers only buy a tiny selection of the catalog. And conversely, a new product may not have enough purchases. If a matrix is incomplete, how would we perform an SVD to get factors? Discuss pros and cons of your approaches in this case.

# Missing Value Imputations

↓ ↓

Movies

				
Bob	4	?	?	4
Alice	?	5	4	?
Joe	?	5	?	?
Sam	5	?	?	?

*Joe like the movie (The Wolf??)*

*Impute by column*

*Impute by row*

- 1) Imputing based on average ratings
- 2) Impute 0s (Typically doesn't work)



# Limitations of SVD

- 1 PCA is based on SVD for dimensionality reduction. And we know PCA is not robust to outliers

# Limitations of SVD

- ① PCA is based on SVD for dimensionality reduction. And we know PCA is not robust to outliers
- ② SVD is a nice baseline however it is very specific - Requires factors  $U, V$  to be orthogonal and is not flexible to regularization or other considerations.

# Limitations of SVD

- ① PCA is based on SVD for dimensionality reduction. And we know PCA is not robust to outliers
- ② SVD is a nice baseline however it is very specific - Requires factors  $U$ ,  $V$  to be orthogonal and is not flexible to regularization or other considerations.
- ③ Requires behavioral data - Can't handle hybrid formulations that include content as well

# Limitations of SVD

- 1 PCA is based on SVD for dimensionality reduction. And we know PCA is not robust to outliers
- 2 SVD is a nice baseline however it is very specific - Requires factors  $U, V$  to be orthogonal and is not flexible to regularization or other considerations.
- 3 Requires behavioral data - Can't handle hybrid formulations that include content as well
- 4 **Solution and Extension?** Matrix Factorization!

↳ Matrix Completion!

# Matrix Factorization Methods

## Matrix Factorization Problem

Let  $X \in \mathcal{R}^{m \times n}$  be a data matrix, say for ratings of users vs movies. Rows represent users and columns represent movies, and entries represent the ratings. Then, mathematically, the matrix factorization problem is defined as:

$$\min_{U, V} \frac{1}{2} \|X - UV\|_F^2$$

Don't have to be orthogonal

where,  $U \in \mathcal{R}^{m \times r}$  and  $r$  is the dimension corresponding to low-dimensional factors ( $U$  and  $V$ ).

# Matrix Factorization Methods

## Matrix Factorization Problem

Let  $X \in \mathcal{R}^{m \times n}$  be a data matrix, say for ratings of users vs movies. Rows represent users and columns represent movies, and entries represent the ratings. Then, mathematically, the matrix factorization problem is defined as:

$$\min_{U, V} \frac{1}{2} \|X - UV\|_F^2$$

where,  $U \in \mathcal{R}^{m \times r}$  and  $r$  is the dimension corresponding to low-dimensional factors ( $U$  and  $V$ ).

## Quick question

Based on the set up above, What's the dimension of the matrix  $V$ ?

# Matrix Factorization vs Matrix Completion

## Missing ratings

When we have missing ratings in  $X$ , instead of filling it in, we can actually only use the ratings we know to learn the factors  $U$  and  $V$ . This formulation is called the **matrix completion** problem.

# Matrix Factorization vs Matrix Completion

## Missing ratings

When we have missing ratings in  $X$ , instead of filling it in, we can actually only use the ratings we know to learn the factors  $U$  and  $V$ . This formulation is called the **matrix completion** problem.

## Matrix Completion

$$\min_{U, V} \frac{1}{2} \sum_{(i,j) \in R} (X_{ij} - U_{i,\cdot}^T V_{\cdot,j})^2,$$

$$\frac{1}{2} \sum_{(i,j)} (X_{ij} - U_{i,\cdot}^T V_{\cdot,j})^2$$

↳ All entries for matrix factor.

where  $R$  is the set of tuples  $(i, j)$  for which a rating is known apriori. How does this compare with matrix factorization with imputation?



# Matrix Factorization Extensions

Flexible Formulation - Can add regularization!

$$\min_{U, V} \frac{1}{2} \|X - UV\|_F^2 + \lambda \|U\|_F^2 + \lambda \|V\|_F^2$$

# Matrix Factorization Extensions

Flexible Formulation - Can add regularization!

$$\min_{U,V} \frac{1}{2} \|X - UV\|_F^2 + \lambda \|U\|_F^2 + \lambda \|V\|_F^2$$

Reduced rank matrix factorization

Above formulation with the regularization is known to reduce the dimensionality of the factors obtained.

# Alternating Minimization Method for Matrix Completion

# Next Time

CS40 → Stochastic Gradient Descent!

- Gradient Descent for Matrix Factorization

# Next Time

- Gradient Descent for Matrix Factorization
- Applications of Matrix Factorization

# Next Time

- Gradient Descent for Matrix Factorization
- Applications of Matrix Factorization
- Twitter case study