Lec-11 Recommender Systems-5

- key Ideas from Netflix competition
- feature Engineering using MF
- Truncated SVD
- PCA for image data (Eigenfaces)

Netflix Solution

$$\underset{n \times m}{X} = \underset{n \times d}{U} \; \underset{d \times m}{V^T} \quad \rightarrow \quad \text{Loss} = \sum_{i=1}^{n} \sum_{j=1}^{m} \left(A_{ij} - \bar{U}_i^T \bar{V}_j \right)^2$$

$\forall i,j \in$ observed

$$\tilde{A}_{ij} = \bar{U}_i^T \bar{V}_j$$ } only getting info from the user item interaction

Other factors we can include!

1. overall average rating on the platform → Platform bias
2. Average rating of an item on the platform → Item bias
3. Average rating given by a user → User bias

Eg. Netflix → Overall average rating of all items = 3.7 = μ

item → Titanic → item rating bias = $b_i = 0.5$

→ Average rating for titanic = 3.7 + 0.5 = 4.2

user → Ram → user rating bias = $b_u = -0.3$

⇒ Baseline prediction of Titanic by Ram

$$= 3.7 + 0.5 - 0.3$$

$$= 3.9$$

$\hat{r}_{ij}$ = predicted rating given by i^{th} users to j^{th} item

$\hat{r}_{ij} = \mu + b_i + b_u$ } Baseline model

⇒ After incorporating user item interaction through MF (Baseline + MF)

$$\hat{r}_{ij} = \hat{A}_{ij} = \mu + b_i + b_u + \bar{U}_i^T \bar{V}_j$$

$A_{ij} = r_{ij}$ = actual rating observed

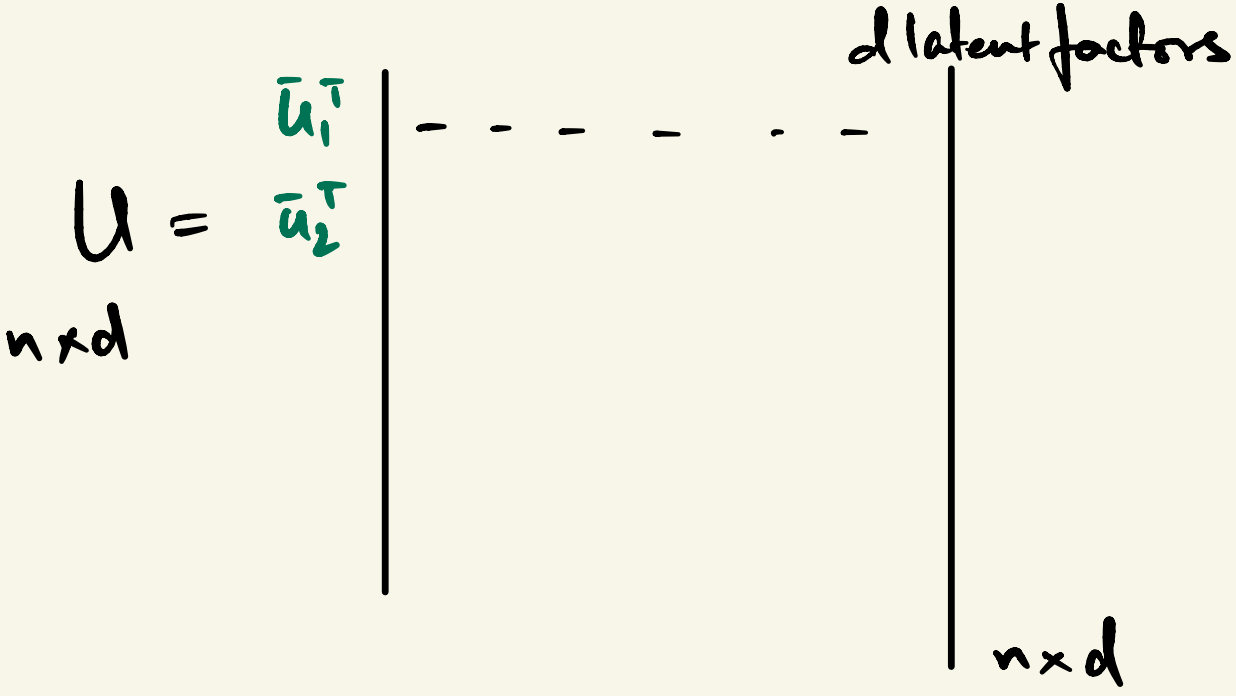
$$\text{New loss} = \sum_{i=1}^{n} \sum_{j=1}^{m} \left(r_{ij} - \underbrace{\left(\mu + b_i + b_u + \bar{u}_i^T \bar{v}_j \right)}_{\hat{r}_{ij}} \right)^2$$

Regularization

$d \rightarrow$ hyperparameter

↳ # of latent factors

↳ d becomes high → risk of overfit



$$\text{Prev loss} = \sum\sum (r_{ij} - \bar{u}_i^T \bar{v}_j)^2$$

loss with Regularization

$$= \sum\sum \left[(r_{ij} - \bar{u}_i^T \bar{v}_j)^2 + \underbrace{\lambda (\|\bar{u}_i\|^2 + \|\bar{v}_j\|^2)}_{\text{Regularization term}} \right]$$

update rule

$$\Rightarrow \bar{u}_i^{t+1} = \bar{u}_i^t - \eta \nabla_{\bar{u}_i} L$$

$$\Rightarrow \bar{v}_j^{t+1} = \bar{v}_j^t - \eta \nabla_{\bar{v}_j} L$$

$$e_{ij} = r_{ij} - \bar{u}_i^T \bar{v}_j$$

$$\frac{\partial L}{\partial x} = \frac{\partial}{\partial x}(r - xy)^2$$

$$= 2(r - xy) \times -y$$

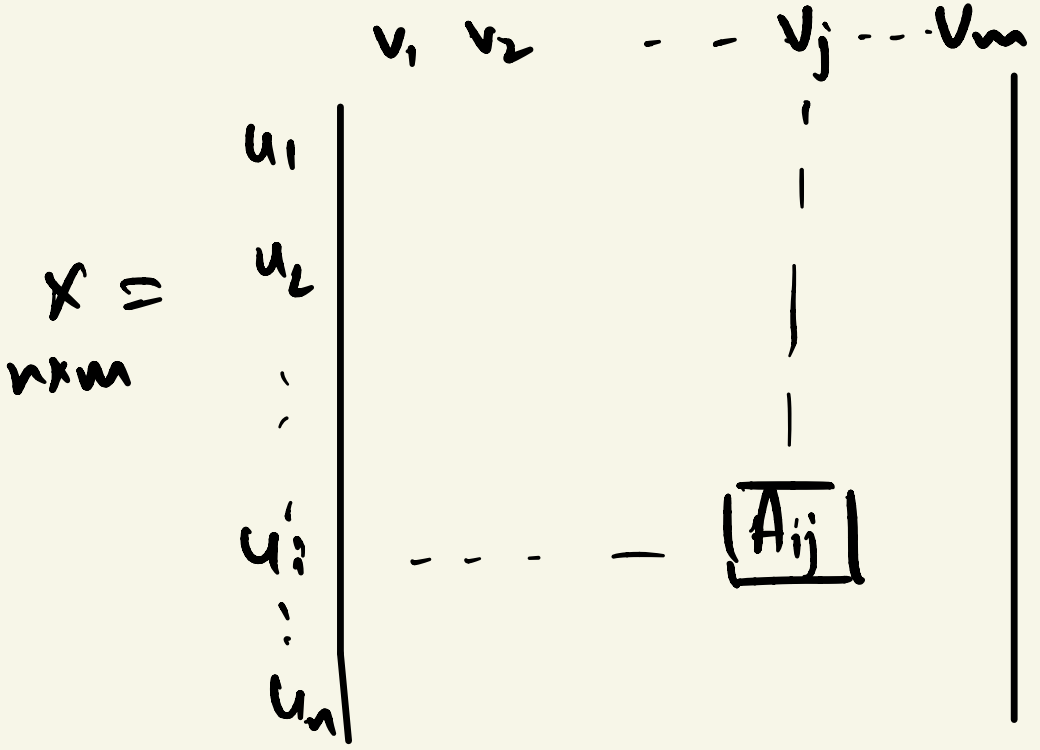
$$\Rightarrow \bar{u}_i^{t+1} = \bar{u}_i^t - \eta \left[2 e_{ij} (-v_j) + 2 \lambda \bar{u}_i \right]$$

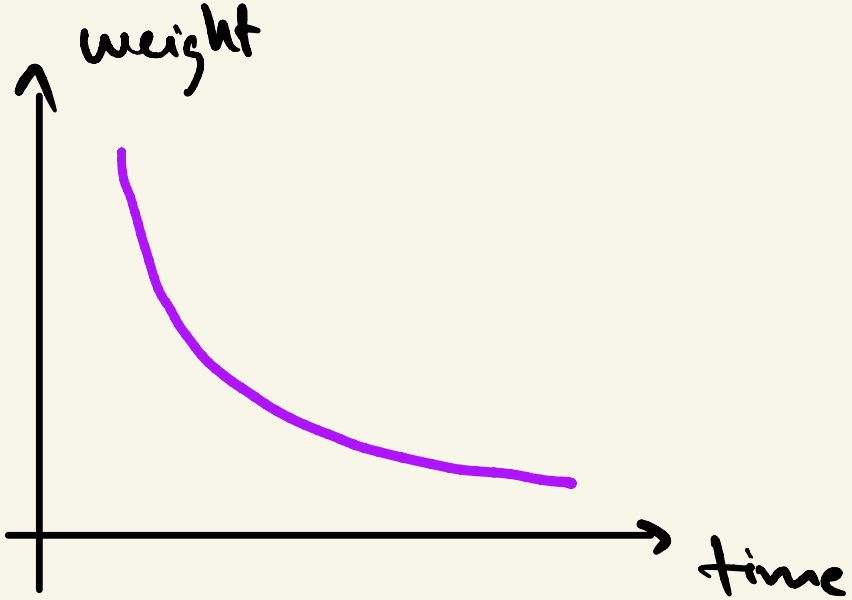
$$\Rightarrow \bar{v}_j^{t+1} = \bar{v}_j^t - \eta \left[2 e_{ij} (-\bar{u}_i) + 2 \lambda \bar{v}_j \right]$$

Temporal Dynamics

$$\hat{\sigma}_{ij}(t) = \mu(t) + b_i(t) + b_u(t) + \bar{u}_i(t)^T \bar{v}_j$$

↳ Biases + Regularization + temporal dynamics





$\rightarrow w_{ij} = e^{-kt}$ (exponentially decreasing)

HW) Implicit feedback

Youtube

U_{10} is currently watching V_{100}

* Recency based recommendation
  - ↳ k nearest neighbors of V_{100} (item - item similarity)
* Recommendations based on historical taste

NOTE: Everything is precomputed offline at regular intervals

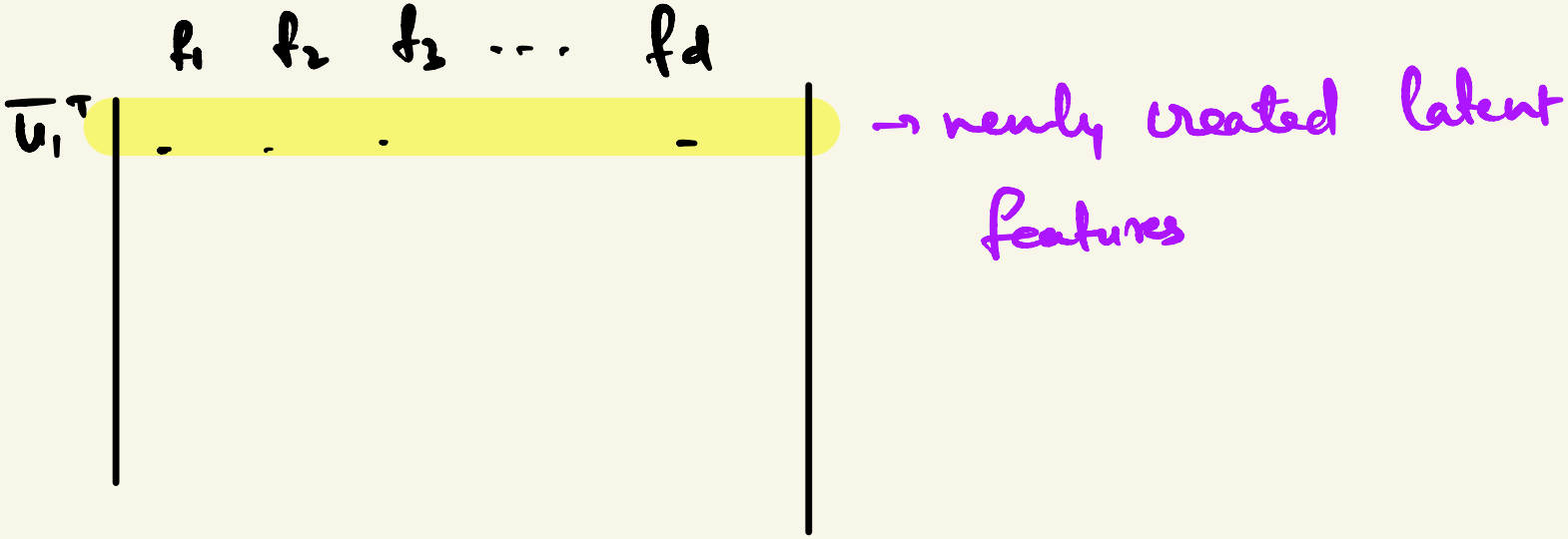
MF : feature engineering

Q) Can we use MF to create features for users and items?

A) Yes.

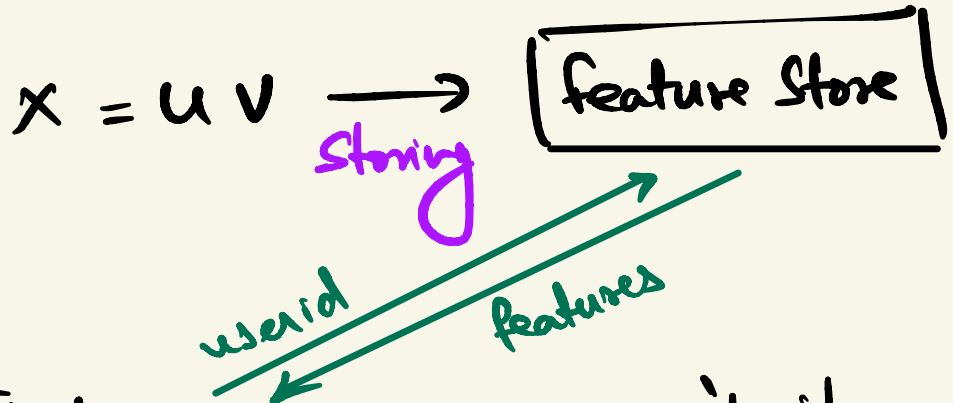
$$X = \begin{bmatrix} & \\ & \end{bmatrix} = U\,V$$

$U =$



→ newly created latent features

for every $\bar{u}_i$ and $\bar{v}_j$ we can store these latent features.



Task 1: User - user similarity

Task 2: Combine these features with user/Item metadata and create Regression / Classification models

Task 3: UMAP/TSNE to visualize user/item clusters and eventually do clustering

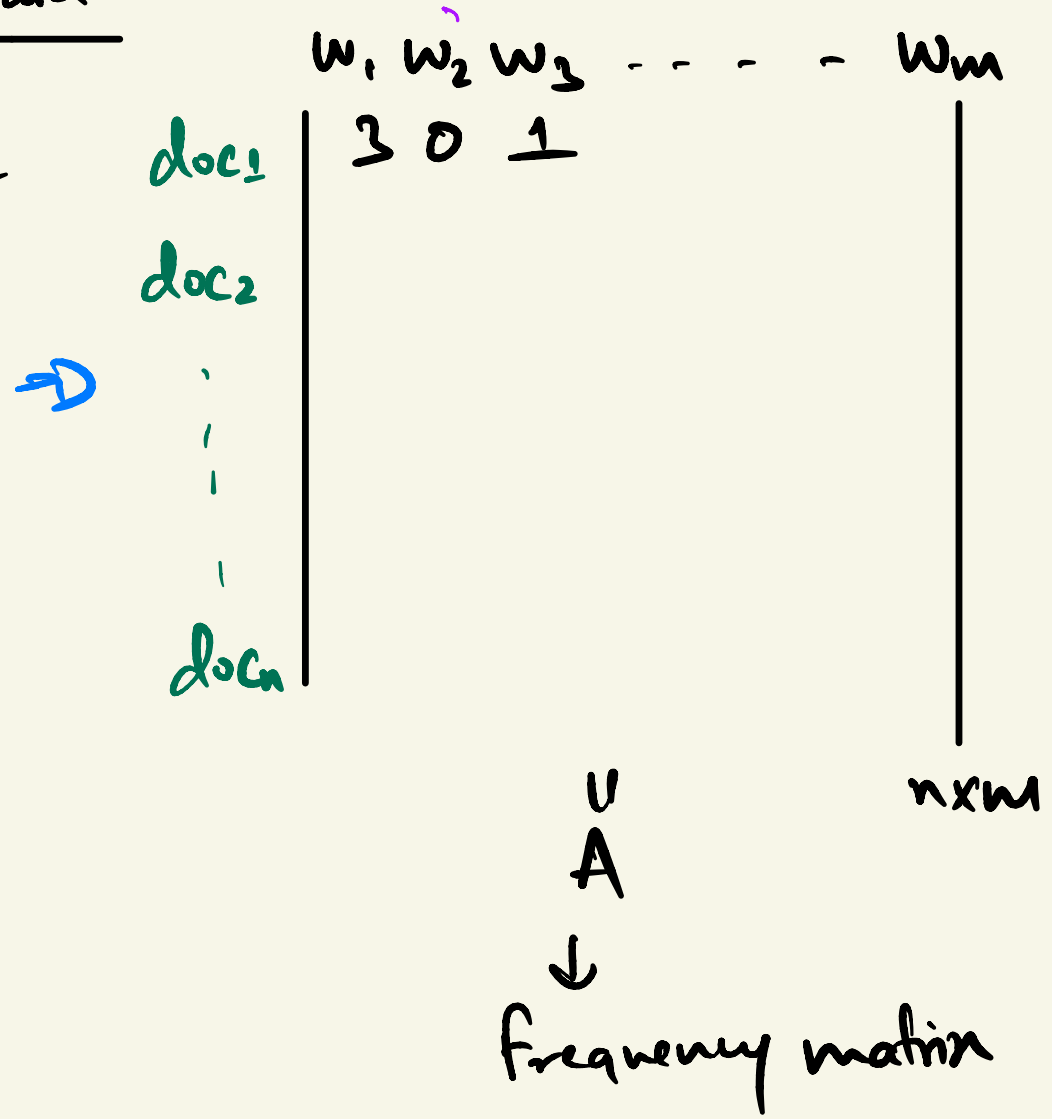
Mf : feature engineering for text data

$doc_1 = w_1\ w_3\ w_{100}\ w_5\ w_{35}$

$doc_2 = - - -$

$\vdots$

$doc_n = - - - -$



A = doc - word freq. matrix

$A = B \; C$

$n \times m \quad n \times d \quad d \times m$

word embeddings

$$\begin{bmatrix} | & | & & | \\ \bar{c}_1 & \bar{c}_2 & \cdots & \bar{c}_m \\ | & | & & | \end{bmatrix}$$

Doc embeddings

$$\begin{bmatrix} \cdots \bar{b}_1^T \cdots \\ \cdots \bar{b}_2^T \cdots \\ \\ \end{bmatrix}$$

Feature store

Usecases:

① find similar documents, using doc embeddings

- Quora, Medium
- Blogs
- Amazon/flipkart, kindle, Any news platform

② find similar words

- Search
- Grammarly
- chatbots

Co-occurence matrix

doc 1 : w_1 w_{100} w_{15} w_{65} w_3 w_{10}

Threshold = 2 (if there are 2 or less words in b/w 2 words, then are considered as co-occuring)

Q) which words are co-occuring with w_1?

A) w_{100}, w_{15}, w_{65}

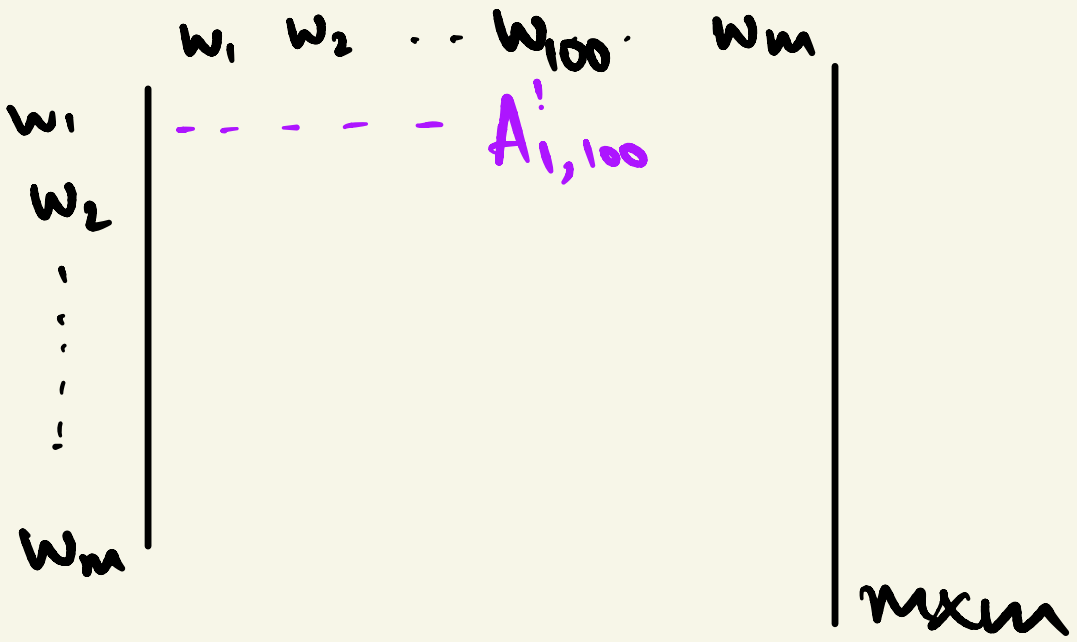
$doc_1 = w_1\ w_3\ w_{100}\ w_5\ w_{35}$

$doc_2 = \ - \ - \ -$

$\vdots$

$doc_n = \ - \ - \ - \ -$

$\Rightarrow X =$



$A_{1,100}$ = how many times w_1 and w_{100} co-occured in our sample space

X = co-occurance matrix

$X = BC \longrightarrow$ word embeddings $\begin{bmatrix} | & | & & | \\ \bar{c}_1 & \bar{c}_2 & \cdots & \bar{c}_n \\ | & | & & | \end{bmatrix}$

$\downarrow$ word emb

$$\begin{bmatrix} - & b_1^T & - & - \\ - & b_2^T & - & - \\ & & & \end{bmatrix}$$

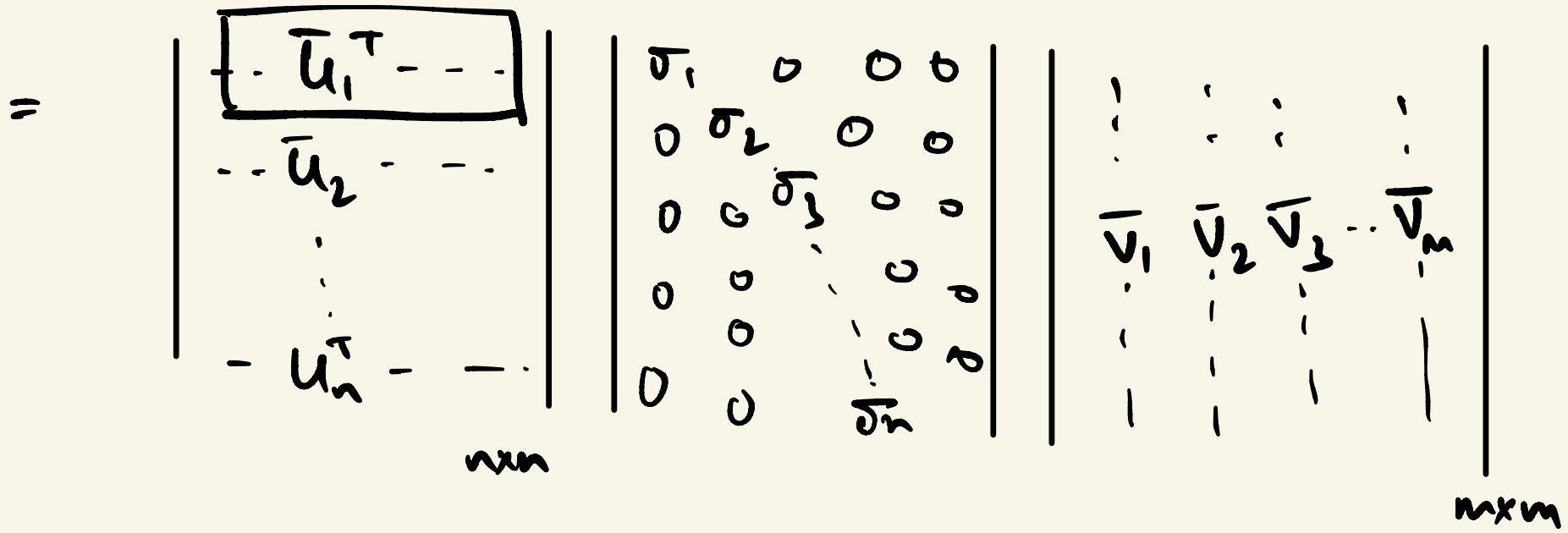
eg.

model $\nearrow$ fashion

model $\searrow$ linear Regression

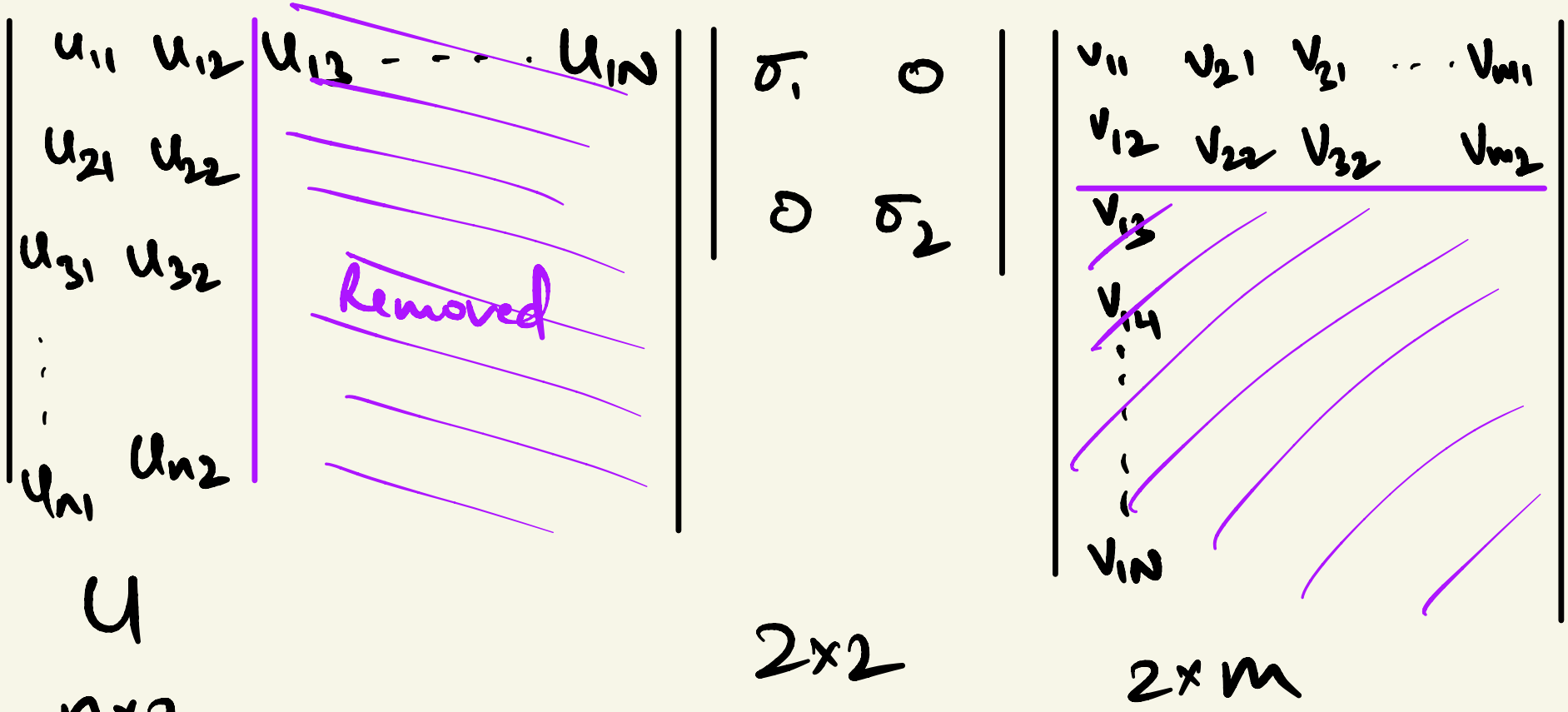
Truncated SVD

$$X_{n \times m} = U_{n \times n} \; \Sigma_{n \times m} \; V^T_{m \times m}$$



n×n n m×m

Truncated SVD $K=2$



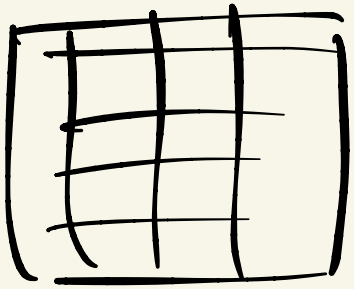
U $n \times 2$ $\quad$ 2×2 $\quad$ $2 \times m$

$$\underset{n \times m}{\hat{X}} = \underset{n \times k}{U_k} \; \underset{k \times k}{\Sigma_k} \; \underset{k \times m}{V_k^T}$$

Use cases:

① Recommender Systems: → Data Completion
↳ feature engineering

② Image compression:

$I =$ 

1990 × 1040

2 mil

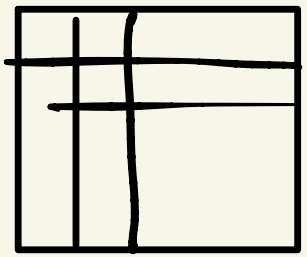
$\rightarrow I = U_k \Sigma_k V_k^T$

eg. $k = 10$

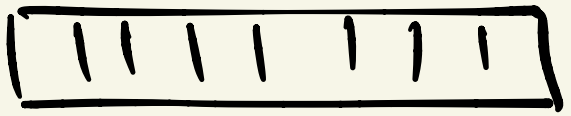
$I = U_k \quad \Sigma_k \quad V_k$

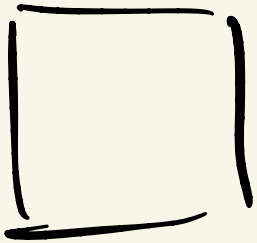
$1990 \times 10 \quad 10 \times 10 \quad 10 \times 1040 \quad \approx 30K$

PCA for Images (facial Recognition)

$I_1 =$ 

flatten →

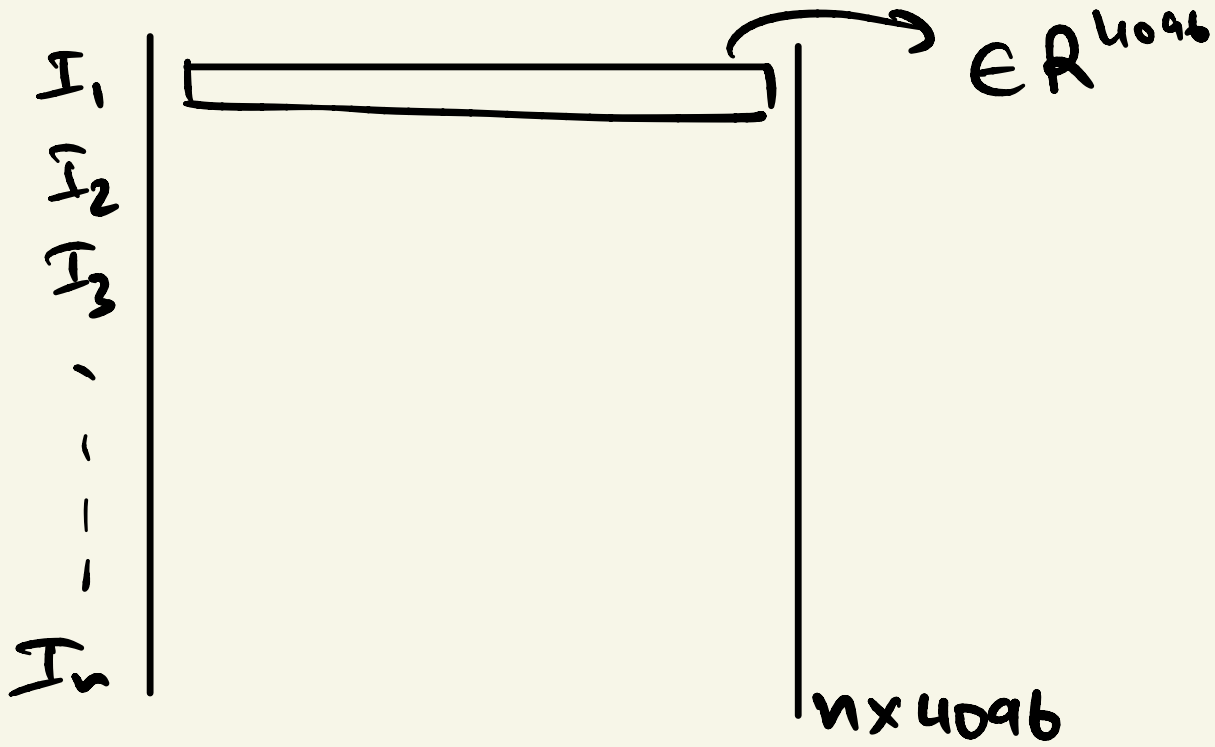


$I_2 =$ 

flatten →

$I_n =$ 

flatten →



$X =$ Image matrix

$$X \Rightarrow \underset{n \times m}{} \quad Cov(X) = \underset{m \times m}{S} \Rightarrow S = \underset{m \times m}{W} \Lambda \underset{m \times m}{W^T}$$

$m \times m$

$n \times m$

$m = 4096$

$$\begin{vmatrix} \vdots & \vdots & \vdots & & \vdots \\ \bar{w}_1 & \bar{w}_2 & \bar{w}_3 & \cdots & \bar{w}_{4096} \\ \vdots & \vdots & \vdots & & \vdots \end{vmatrix}$$

$\bar{w}_1, \bar{w}_2, \ldots \bar{w}_m \in R^m \quad (R^{4096})$

↓

each eigenvector is an image of 64x64 dimensions.

eigenfaces

⇒ we can represent any image in our dataset as a linear combination of these eigenvectors.

Choose top 5 eigenvectors

$I_1 = \lambda_1 \bar{w}_1 + \lambda_2 \bar{w}_2 + \lambda_3 \bar{w}_3 + \lambda_4 \bar{w}_4 + \lambda_5 \bar{w}_5$