Matrix Factorization and SVD

- Rank and Rank Factorization of a Matrix
- Low Rank Approximation
- Singular Value Decomposition
- Low Rank Approximation from Truncated SVD
- SVD Applications
  - Recommendation System
  - Latent Semantic Analysis
  - Data Denoising
- PCA and SVD

Imdad ullah Khan

Rank Factorization of Matrices

Rank of a matrix

For an $n \times m$ matrix A

Column Rank of A, col-rank(A) is the maximum number of linearly independent columns of A

Row Rank of A, row-rank(A) is the maximum number of linearly independent rows of A

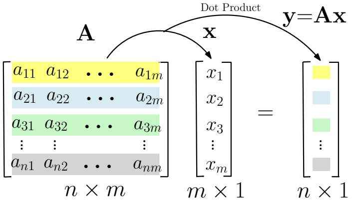
$$rank(A) := \text{col-rank}(A) = \text{row-rank}(A)$$

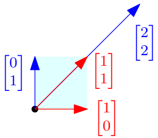
Rank of a matrix

For an $n \times m$ matrix A

Looking at A as a linear transformation i.e. $A : \mathbb{R}^m \mapsto \mathbb{R}^n$

$rank(A)$ is the true dimensionality of the range (output) space of A





columns of $\begin{bmatrix}1 & 2\\1 & 2\end{bmatrix}$ are linearly dependent

If $rank(A) = k$, then output vectors live in a k-d subspace

Rank of a matrix

Another definition of rank (aka decomposition rank)

An $n \times m$ matrix A has

Rank-0 if all its entries are 0

Rank-1 if it is outer product of an $n \times 1$ and an $m \times 1$ vector, $A = \mathbf{u}\mathbf{v}^T$

$$A = \mathbf{u}\mathbf{v}^T = \begin{bmatrix} | \\ \mathbf{u} \\ | \end{bmatrix} \begin{bmatrix} — & \mathbf{v}^T & — \end{bmatrix} = \begin{bmatrix} | & | & \dots & | \\ v_1\mathbf{u} & v_2\mathbf{u} & \dots & v_m\mathbf{u} \\ | & | & \dots & | \end{bmatrix} = \begin{bmatrix} — & u_1\mathbf{v}^T & — \\ — & u_2\mathbf{v}^T & — \\ \vdots & \ddots & \vdots \\ — & u_n\mathbf{v}^T & — \end{bmatrix}$$

Rank-2 if it is non-trivial sum of two rank-1 matrices $A = \mathbf{u}\mathbf{v}^T + \mathbf{w}\mathbf{x}^T$

$$A = \mathbf{u}\mathbf{v}^T + \mathbf{w}\mathbf{x}^T = \begin{bmatrix} | & | & \dots & | \\ v_1\mathbf{u} + x_1\mathbf{w} & v_2\mathbf{u} + x_2\mathbf{w} & \dots & v_m\mathbf{u} + x_m\mathbf{w} \\ | & | & \dots & | \end{bmatrix} = \begin{bmatrix} — & u_1\mathbf{v}^T + w_1\mathbf{x}^T & — \\ — & u_2\mathbf{v}^T + w_2\mathbf{x}^T & — \\ \vdots & \ddots & \vdots \\ — & u_n\mathbf{v}^T + w_n\mathbf{x}^T & — \end{bmatrix} = \begin{bmatrix} | & | \\ \mathbf{u} & \mathbf{w} \\ | & | \end{bmatrix} \begin{bmatrix} — & \mathbf{v}^T & — \\ — & \mathbf{x}^T & — \end{bmatrix}$$

Rank-k if it is sum of k rank-1 matrices and cannot be written as sum of $k-1$ or fewer rank-1 matrices

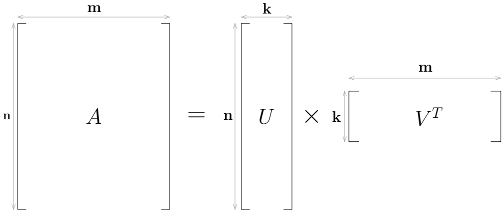
Rank Factorization of a matrix

An $n \times m$ matrix A has rank-k if A can be "factored into" the product of a

- $(n \times k)$ matrix U and ▷ tall and skinny
- $(k \times n)$ matrix V^T ▷ short and long

$$A = UV^T$$

- A cannot be factored into $n \times (k-1)$ and $(k-1) \times m$ matrices



- columns of U are the columns of the rank-1 factors $\mathbf{u}_i$'s
- rows of V^T are the rows of the rank-1 factors $\mathbf{v}_i$'s

All definitions of rank are equivalent - each implies the other

Rank Factorization of a matrix

- $n \times n$ matrix A is "full rank" if it has rank n
  - It uniquely maps $n \times 1$ vectors to $n \times 1$ vectors
  - A is a "bijection", A is invertible
- If $rank(A) < n$, then A is a singular matrix (rank deficient matrix)
  - The resulting dimensionality is $\leq n - 1$
  - Cannot get pre-images from images
  - A is not invertible
- There cannot be any inverse for a non-square matrix

Low Rank Structure in Data

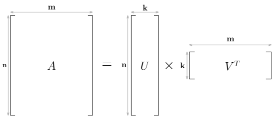
Low rank data matrix

A: a $n \times m$ data matrix ▷ rows: data points – columns: features

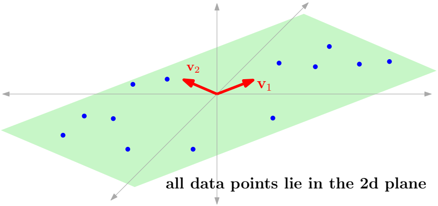
If A has rank k, then $A = UV^T$ ▷ $|U| = n \times k$ $|V| = k \times m$

Each row (data point) of A can be represented as a linear combination of $\mathbf{v}_1, \ldots, \mathbf{v}_k$

$\mathbf{a}_i = \sum_{j=1}^{k} u_{ij}\mathbf{v}_j$, u_{ij} are projection lengths of $\mathbf{a}_i$ on $\mathbf{v}_j$



Geometrically, all data lie in a k-d subspace (spanned by $\mathbf{v}_1, \ldots, \mathbf{v}_k$)



all data points lie in the 2d plane

Data Compression

Space reqtt for A: $n \times m$

Store the matrix U and V

Space reqtt: $k(n + m)$

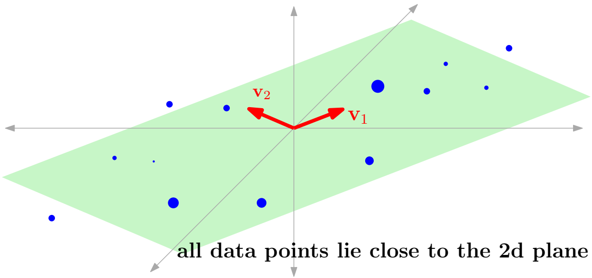
Low rank approximation

Data may not be in a k-d subspace,

but it may be lying 'close by' to a low dimensional subspace

We say data is approximately low rank

May not get $A = UV^T$ – would like to find U and V so $A \simeq UV^T$



Low rank approximation

May not get $A = UV^T$ – would like to find U and V so $A \simeq UV^T$

Need a goodness measure to assess $A \simeq UV^T$

$$\sum_{i=1}^{n} \|\mathbf{a}_i - \sum_{j=1}^{k} u_{ij}\mathbf{v}_j\|^2 =: \|A - UV^T\|_F^2$$

For a matrix M, $\|M\|_F = \sqrt{\sum_{i,j} M_{ij}^2}$ is the Frobenius norm of M

The optimization problem of finding the best low rank approximation for A

$$\underset{V \in \mathbb{R}^{k\times m},\, U \in \mathbb{R}^{n\times k}}{\arg\min} \|A - UV^T\|_F^2$$

Why expect low rank structure

Data is not necessarily described by the attributes in which it is measured

I am going to show you two examples with dependencies between columns

These examples are adapted from real-world data

Why expect low rank structure

Housing Data

ID	Beds	Baths	Living sq-ft	Lot sq-ft	Floors	Garage Cars	List Price	Sale Price
1	1	1	870	1100	1	0	31630	31544
2	1	1	1080	1400	1	0	35920	35916
3	2	1	1250	1500	1	0	48250	48025
4	2	1	1285	1550	1	0	48965	48738
5	2	2	1460	1800	2	1	67540	67633
6	3	2	1560	1800	1	0	68440	68763
7	3	2	1630	1900	2	1	79870	79533
8	3	2	2050	2500	2	1	88450	88054
9	3	2.5	2120	2600	2	2	102380	102576
10	4	2	2360	2800	2	1	103640	103892
11	4	2.5	2500	3000	2	1	109000	109523
12	4	2.5	2570	3100	2	1	110430	110393
13	4	3	2710	3300	3	2	125790	125945
14	5	2	2880	3400	2	2	133120	133503
15	5	2.5	2880	3400	3	2	135620	136124
16	5	2.5	3300	4000	3	2	144200	144365
17	5	3	3650	4500	3	2	153850	154444
18	5	3	3720	4600	3	3	165280	165439

Why expect low rank structure

Housing Data : Rank of this matrix is not 8 Some linear dependencies are shown (there may be others including non-linear)

$$\textbf{List-Price} = 10k\times \textbf{ bed } + 5k\times\textbf{baths } + 9\times \textbf{ liv-sqFT } + 8\times \textbf{ Lot } + 10k\times \textbf{ Cars}$$

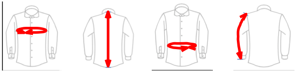
ID	Beds	Baths	Living sq-ft	Lot sq-ft	Floors	Garage Cars	List Price	Sale Price
1	1	1	870	1100	1	0	31630	31544
2	1	1	1080	1400	1	0	35920	35916
3	2	1	1250	1500	1	0	48250	48025
4	2	1	1285	1550	1	0	48965	48738
5	2	2	1460	1800	2	1	67540	67633
6	3	2	1560	1800	1	0	68440	68763
7	3	2	1630	1900	2	1	79870	79533
8	3	2	2050	2500	2	1	88450	88054
9	3	2.5	2120	2600	2	2	102380	102576
10	4	2	2360	2800	2	1	103640	103892
11	4	2.5	2500	3000	2	1	109000	109523
12	4	2.5	2570	3100	2	1	110430	110393
13	4	3	2710	3300	3	2	125790	125945
14	5	2	2880	3400	2	2	133120	133503
15	5	2.5	2880	3400	3	2	135620	136124
16	5	2.5	3300	4000	3	2	144200	144365
17	5	3	3650	4500	3	2	153850	154444
18	5	3	3720	4600	3	3	165280	165439

$$\textbf{Sale Price} = (1 \pm 0.02) \times \textbf{ List Price}$$

Why expect low rank structure

Shirt Dimension Many measurements (chest and waist circumferences, sleeve and back lengths) for shirt

In market shirts are marked with collar measurement only

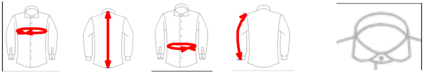


Chest	Back	Waist	Sleeve
104	81	98	67
107	81	100	67
110	82	102	67
113	82	104	67
116	83	106	68
120	83	110	68
124	84	114	68
128	84	118	68
132	85	122	68
136	85	126	68

Why expect low rank structure

Shirt Dimension The collar feature is a linear combination of other features. The data actually lies in a one dimensional space

$$\textbf{Collar} = 0.44\times\ \textbf{Chest} + 0.015\times\ \textbf{Back} - 0.2\times\ \textbf{Waist} + 0.153\times\ \textbf{Sleeve}$$



Chest	Back	Waist	Sleeve	Collar
104	81	98	67	37
107	81	100	67	38
110	82	102	67	39
113	82	104	67	40
116	83	106	68	41
120	83	110	68	42
124	84	114	68	43
128	84	118	68	44
132	85	122	68	45
136	85	126	68	46

Singular Value Decomposition

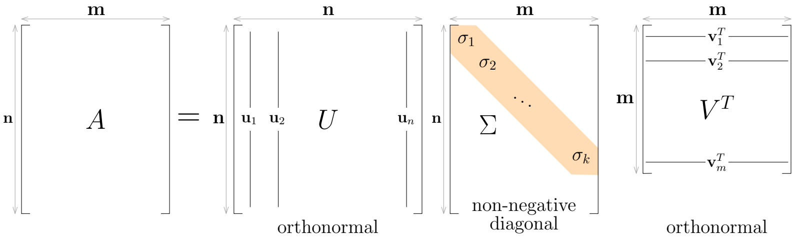
Singular Value Decomposition

Theorem

Any $n \times m$ matrix can be written as a product of three matrices

$$A \;=\; U\Sigma V^T$$

- *U is a $n \times n$ orthogonal matrix (columns are orthonormal)*
- *V is a $m \times m$ orthogonal matrix*
- *Σ is a $n \times m$ diagonal matrix, with non-negative entries and entries at the main diagonal are sorted from highest value to lowest*

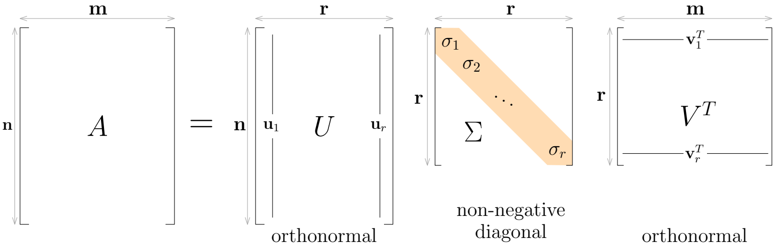


Singular Value Decomposition

Theorem (Compact SVD)

Any $n \times m$ matrix with rank $r \leq \min\{m, n\}$ can be written as a product of three matrices, $A = U\Sigma V^T$

- *U is a $n \times r$ orthogonal matrix (columns are orthonormal)*
- *V is a $r \times r$ orthogonal matrix*
- *Σ is a $r \times m$ diagonal matrix, with non-negative entries and entries at the main diagonal are sorted from highest value to lowest*

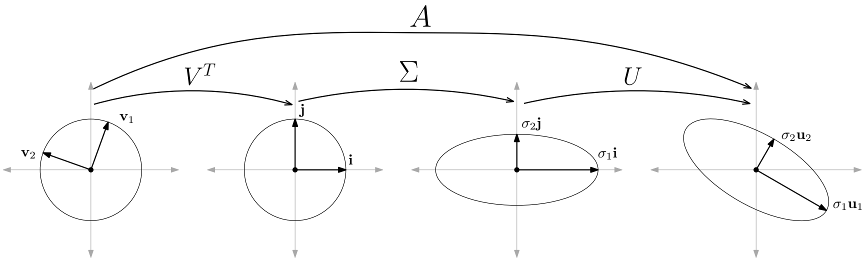


Singular Value Decomposition

$$\text{SVD:} \qquad A = U\Sigma V^T$$

- U is orthogonal – its columns are called left singular vectors
- V is orthogonal – its columns are called right singular vectors
- Diagonal entries of Σ are called singular values

Any transformation is a rotation followed by scaling followed by a rotation



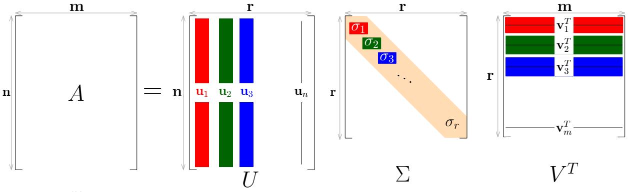
Rank k Approximation from SVD

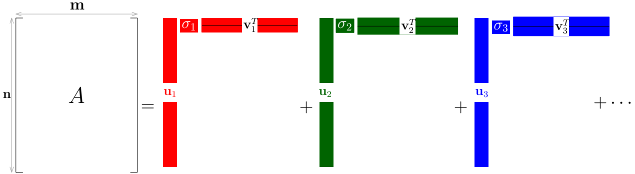
Spectral decomposition of a matrix

From SVD of $A \in \mathbb{R}^{n\times m}$ we can get spectral decomposition of A

i.e. Express A as linear combination of r rank-1 matrices (outer products of singular vectors) – coefficients are the corresponding singular values

$$A = U\Sigma V^T \quad \Leftrightarrow \quad A = \sum_{\ell=1}^{\min\{m,n\}} \sigma_\ell \, \mathbf{u}_\ell \circ \mathbf{v}_\ell^T$$



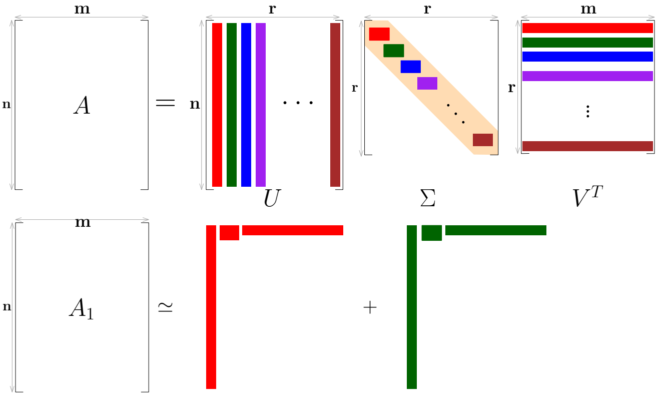


Truncated SVD

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

$$A_k = \sum_{\ell=1}^{k} \sigma_\ell \mathbf{u}_\ell \circ \mathbf{v}_\ell^T + \cancel{\sum_{\ell=k+1}^{r} \sigma_\ell \mathbf{u}_\ell \circ \mathbf{v}_\ell^T}$$

$$A = \sum_{\ell=1}^{r} \sigma_\ell \; \mathbf{u}_\ell \circ \mathbf{v}_\ell^T$$

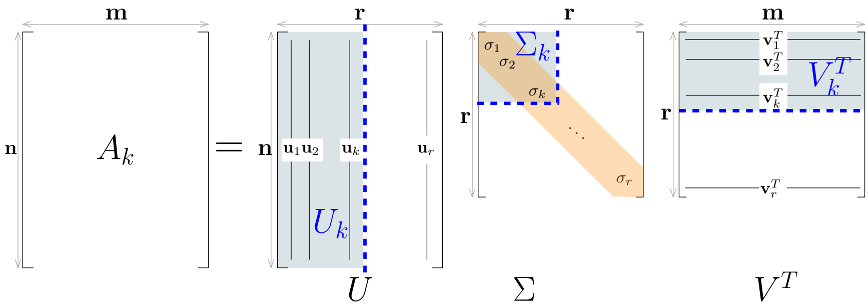


Set to 0 (truncate) the last $r - k$ singular values (σ_{k+1} to σ_r)

Truncated SVD

- $U_k \in \mathbb{R}^{n \times k}$: the first k left singular vectors (the first k columns of U)
- $\Sigma_k \in \mathbb{R}^{k \times k}$: the first k singular values
- V_k^T be the first k right singular vectors

$$A_k = \sum_{\ell=1}^{k} \sigma_\ell \mathbf{u}_\ell \circ \mathbf{v}_\ell^T + \cancel{\sum_{\ell=k+1}^{r} \sigma_\ell \mathbf{u}_\ell \circ \mathbf{v}_\ell^T} = U_k \Sigma_k V_k^T$$



Rank-k approximation from Truncated SVD

The optimization problem of finding the best low rank approximation for A

$$\underset{V \in \mathbb{R}^{k \times m},\ U \in \mathbb{R}^{n \times k}}{\arg\min} \|A - UV^T\|_F^2$$

Theorem

A_k is the best rank-k approximation to A, i.e. it is the solution to the above optimization problem

More formally,

- the U in the above problem would be $U_k\sqrt{\Sigma_k}$
- and V would be $\sqrt{\Sigma_k}V_k^T$

If k is not part of input, then k can be chosen as we discussed for number of principal components (scree plot, elbow method etc.)

Applications of SVD

SVD Applications

Matrix Completion - extrapolate missing values of a matrix

Interpretation of SVD

Approximate A in terms of k "concepts" or "latent factors"

Singular vectors U_k and V_k^T are numeric representations of rows and columns of concepts

Singular values Σ_k measure strength of these concepts

- ith row of U represents a data item (ith row of A) as a linear combination of the rows of concepts (with coefficients Σ)
- jth column of V^T represents a dimension (jth columns of A) as a linear combination of the columns of concepts (with coefficient Σ)

SVD Application: Recommenders

Recall the recommendation system problem

Given a matrix R – users (rows) ratings for items (columns), predict $R(i,j)$

	p_1	p_2	p_3						p_j									p_m
u_1	1		2	1		4			2		3	2		5				2
u_2		1				2		1			2			1				3
u_3		1	1	2				1							1		2	
				3		2			5			2			3	4		
		1			2									5				
u_i			3	2	1		4	5	$\boxed{?}$	1		3	1		2			1
			4													4		
			5			1										5		
		1		4							1	3		5		1	2	
u_n			3		1	1		2	1					4				5

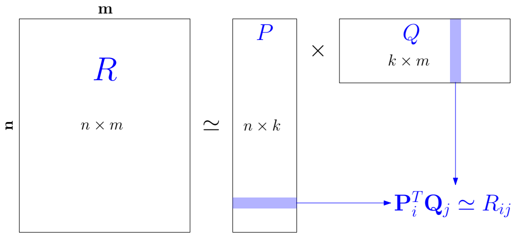
SVD Application: Recommenders

- Given $n \times m$ matrix R For $k \ll m, n$, Find
- $n \times k$ matrix P and $k \times m$ matrix Q such that

$$R = PQ$$

Generally, for very small k, we seek

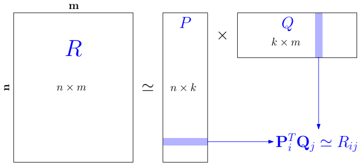
$$R \simeq PQ$$



SVD Application: Recommenders

- Given $n \times m$ matrix R For $k \ll m, n$, Find
- $n \times k$ matrix P and $k \times m$ matrix Q such that

$$R \simeq PQ$$



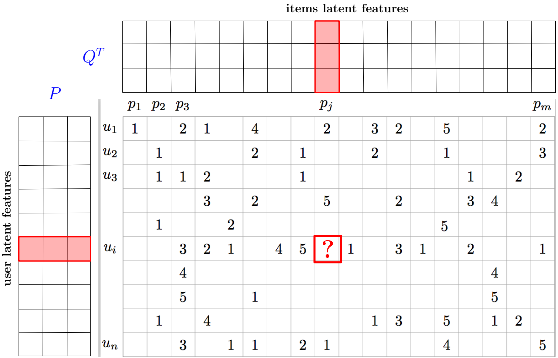
This is a classic optimization problem can be solved as

$$\min_{\substack{P \in \mathbb{R}^{n \times k} \\ Q \in \mathbb{R}^{m \times k}}} \sum_{(i,j)} \left(R_{ij} - P_i Q_j^T \right)^2 + \underbrace{\lambda \left(\|P\|_F^2 + \|Q\|_F^2 \right)}_{\substack{\text{regularization term} \\ \text{avoids overfitting}}}$$

SVD Application: Recommenders

Matrix Factorization for Recommenders $\quad R \simeq PQ$

- P : k-dim representation of users in a latent feature space $\mathbb{R}^k$
- Q : k-dim representation of items latent feature space
- $P_i Q_j^T$: interaction between user i and item j – approximation of R_{ij}



SVD Application: Recommenders

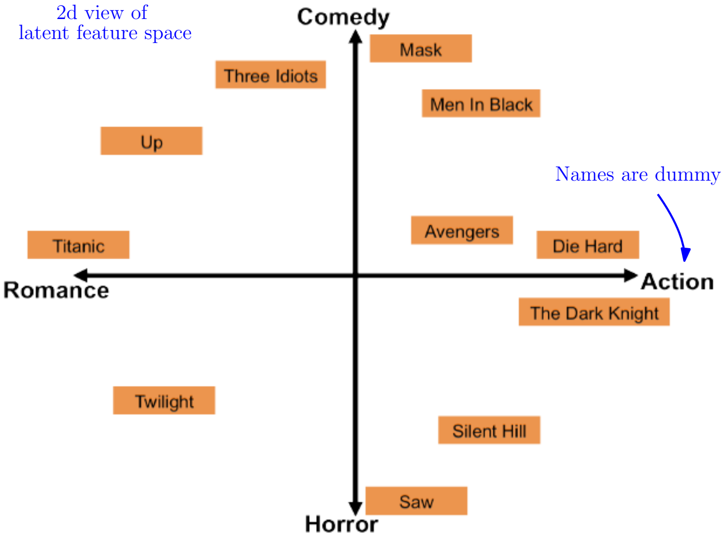


diagram adapted from Cho-Jui Hsieh @ UCLA

SVD Application: Recommenders

Users and movies mapped to latent feature space

Points are coordinates of users and movies (2-dim P and Q)

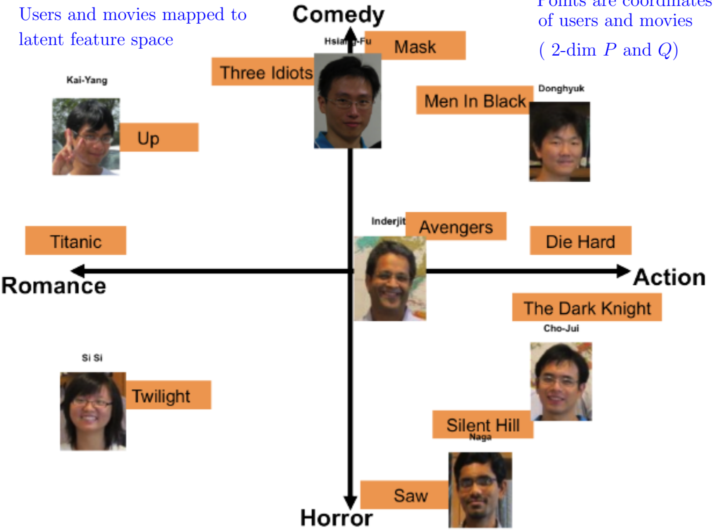
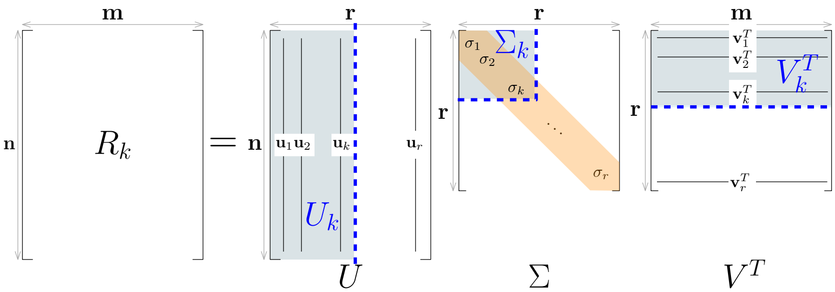
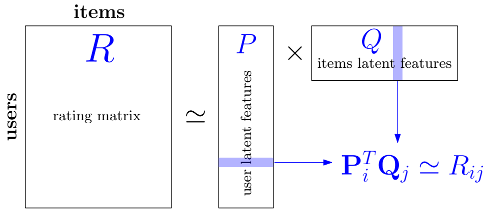


diagram adapted from Cho-Jui Hsieh @ UCLA

SVD Application: Recommenders

Using SVD to get $R = PQ$





$$\mathbf{P} = U_k \sqrt{\Sigma_k}$$

$$\mathbf{Q} = \sqrt{\Sigma_k}\, V_k^T$$

SVD Application: Recommenders

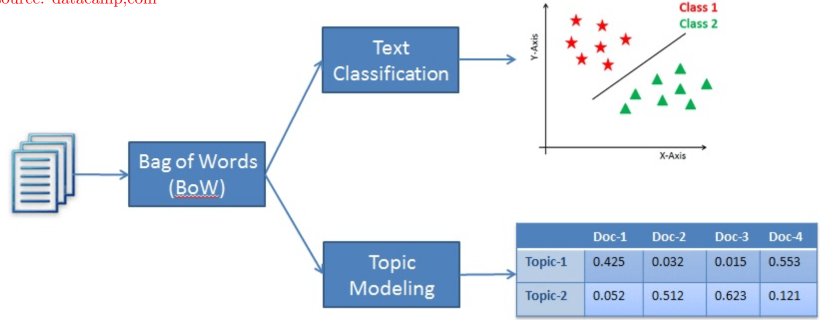
- SVD is not the best approach to factorize rating matrix
- Typically R will have many values missing
- SVD will adjust U, Σ and V to the 0's or any default values
- One can try other default values
  - matrix average, row averages, column averages, ANOVA
- SVD performs good if R is close to rank-k and has few missing values

SVD Application

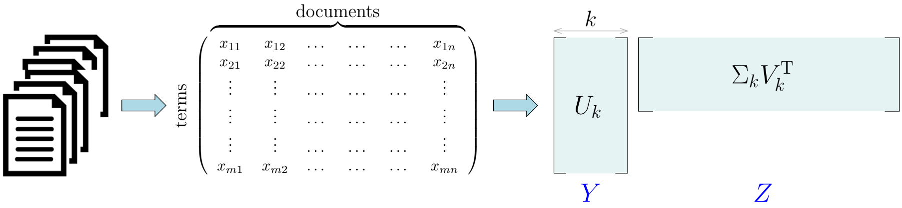
Classic data science applications of of SVD are

- Latent Semantic Analysis – the basis of Word Embedding (particularly word2vec)
- Latent Semantic Indexing (when used in information retrieval). LSA is widely used in many text analytics applications

source: datacamp,com



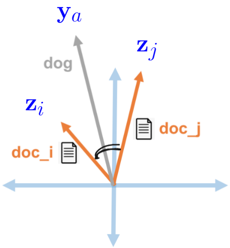
SVD Application: LSA



Term-Document Incidence Matrix, X Low rank approximation of X via SVD

- $\|X - YZ^{\mathrm{T}}\|_F$ is minimum (amongst rank k matrices)
- On average every entry $X_{ij} \simeq \mathbf{y}ai\mathbf{z}_i^{\mathrm{T}}$
- $\mathbf{y}_a\mathbf{z}_i^{\mathrm{T}} \simeq 1$ when doc_i contains $term_a$
- $\mathbf{y}_a$ and $\mathbf{z}_i$ is the ath row of Y and ith columns of Z^{T}

SVD Application: LSA

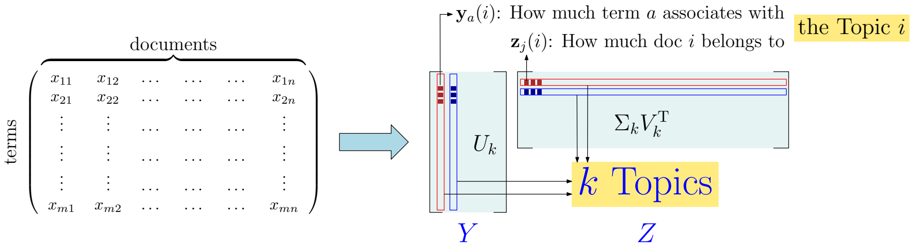


- doc_i and doc_j both contains $term_a \implies \mathbf{y}_a\mathbf{z}_i^{\mathrm{T}} \simeq \mathbf{y}_a\mathbf{z}_j^{\mathrm{T}} \simeq 1$
- $\mathbf{z}_i$ and $\mathbf{z}_j$ both have high dot product with $\mathbf{y}_a$ (low cosine distance)
- If doc_i and doc_j contain many terms, in common, they will have small angle between them (high dot-product)
- If terms a and b appear in many common documents, then $\mathbf{y}_a$ and $\mathbf{y}_b$ will have higher dot products

SVD Application: LSA

Alternative Interpretation:

- Y and Z represent k abstract concepts (latent factors)
- the ath row of Y represent term a as linear combin. of the k concepts
- Columns of Z^T represent docs as linear combin. of the k concepts



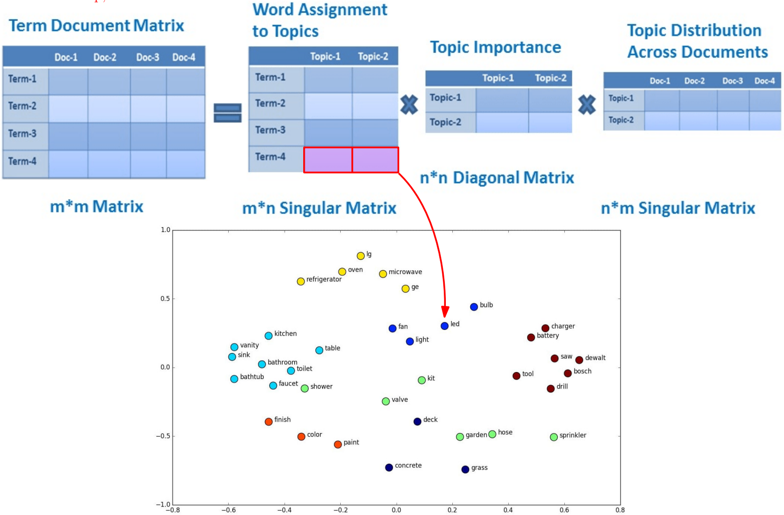
Term-Document Incidence Matrix, X

Low rank approximation of X via SVD

SVD Application: LSA

LSA embeds terms into the k-d space (rows of Y are the representation)

source: datacamp,com



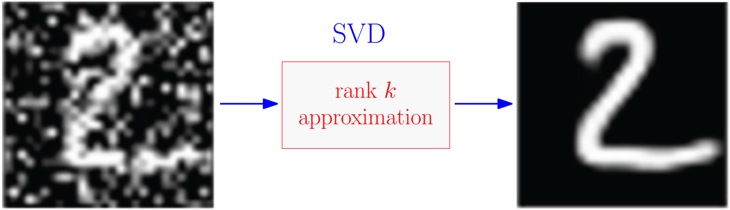
SVD Application: Data denoising

Data denoising

If true underlying data in A is low-rank, Truncated SVD of A might throw out a significant amount of noise and little ground truth data (the signal)

The resulting approximate data might be a cleaner, more informative and better version of A

Especially, if singular values have a good elbow structure, (smaller singular values will more likely correspond to the added noise in data)



Noisy image

Denoised image

SVD and Eigendecomposition

SVD and eigen-decomposition are related but there are differences

- Not every matrix has an eigen-decomposition (not even every square matrix). Any matrix (even rectangular) has an SVD
- In eigen-decomposition $A = X\Lambda X^{-1}$, that is, the eigen-basis is not always orthogonal. The basis of singular vectors is always orthogonal
- In SVD we have two singular-spaces (right and left)
- Computing the SVD of a matrix is more numerically stable

SVD and Eigendecomposition

For $n \times n$ real symmetric real matrix A

$$A \;=\; U\Sigma V^T \qquad\qquad A \;=\; X\Lambda X^{-1}$$

In this case we must have the following

- U, V, X are orthonormal matrices
- Λ and Σ are diagonal matrices with values in decreasing orders (eigenvalues and singular values, respectively)
- U and V are the left and right singular matrices of A, respectively
- X are eigenvectors of A

SVD and Eigendecomposition

PCA using SVD

$$A^{\mathrm{T}}A = (U\Sigma V^{\mathrm{T}})^{\mathrm{T}}U\Sigma V^{\mathrm{T}} = (V^{\mathrm{T}})^{\mathrm{T}}\Sigma^{\mathrm{T}}U^{\mathrm{T}}U\Sigma V^{\mathrm{T}} = V\Sigma\Sigma V^{T} = V\Sigma^2 V^{\mathrm{T}}$$

- V contains eigenvectors of $C = A^T A$
- Note we get it directly from $A = U\Sigma V^T$ without having to explicitly compute the covariance matrix.
- For large dataset and large dimensions computing C is already computationally expensive
- So SVD provides another way of computing the principal components
- Recall that principal components are eigenvectors of $A^T A$ (if A is the data matrix with each data point as a row and each dimension as a column). Eigenvalues are just the square roots of the singular values