$$\sigma_i(X)^2 = \lambda_i(X^TX) = \lambda_i(XX^T)$$

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

$$X^TX = V\Sigma^2V^T$$

$$XX^T = U\Sigma^2U^T$$

COMPSCI 514: ALGORITHMS FOR DATA SCIENCE

Cameron Musco

University of Massachusetts Amherst. Fall 2020.
Lecture 17

LOGISTICS

① eigenvalues of X^TX = squared sing. values of X

② = singular values of X^TX

③ any sym. A
singular values of A = absolute values of eigenvals of A.

- Problem Set 3 deadline extended until Monday 10/26, 8pm.
- Week 9 Quiz will now be due Tuesday 10/27, 8pm. ?

symmetric $A = W\Lambda W^T$
what are eigenvalues of A^2?
$= \Lambda^2$

$A^2 = W\Lambda W^T W \Lambda W^T$
$= W\Lambda^2 W^T$

symmetric
$$A = W \Lambda W^T$$
where W is orthonormal

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

$U = W \cdot S$ $S = \text{sign}(\Lambda)$

$\Sigma = |\Lambda|$

$V = W$

SUMMARY

Last Few Classes: Low-Rank Approximation and PCA

$$A = V \Lambda V^T$$
$$= \underline{VS}\,\underline{\Sigma}\,\underline{V^T}$$

where $\Sigma = |\Lambda|$

$S = \text{sign}(\Lambda)$

- Compress data that lies close to a k-dimensional subspace.
- Equivalent to finding a low-rank approximation of the data matrix $\underline{X}$: $X \approx \underbrace{XV}_{n\times 2} V^T$ for orthonormal $V \in \mathbb{R}^{d\times k}$.
- Optimal solution via PCA (eigendecomposition of X^TX or equivalently, SVD of X).
- Singular vectors of X are the eigenvectors of XX^T and X^TX. Singular values squared are the eigenvalues.

d

n $\begin{bmatrix} & \\ & \end{bmatrix}$

$$\Lambda \begin{bmatrix} -6 & & \\ & 2 & \\ & & 3 \end{bmatrix} \quad \begin{bmatrix} 6 & & \\ & 2 & \\ & & 3 \end{bmatrix}$$

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

U, V are orthonormal

Σ is positive diagonal

$XX^T = U\Sigma^2U^T$ — eigendecomp. ✓

$X^TX = V\Sigma^2V^T$

SVD of X^TX?

SVD of X^TX, positive semidefinite matrix

all eigens are +

if I have any symetric A with eigendecomt

$V\Lambda V^T =$

SVD of A?

SUMMARY

Last Few Classes: Low-Rank Approximation and PCA

- Compress data that lies close to a k-dimensional subspace.
- Equivalent to finding a low-rank approximation of the data matrix $\mathbf{X}$: $\mathbf{X} \approx \mathbf{X}\mathbf{V}\mathbf{V}^T$ for orthonormal $\mathbf{V} \in \mathbb{R}^{d \times k}$.
- Optimal solution via PCA (eigendecomposition of $\mathbf{X}^T\mathbf{X}$ or equivalently, SVD of $\mathbf{X}$).
- Singular vectors of $\mathbf{X}$ are the eigenvectors of $\mathbf{X}\mathbf{X}^T$ and $\mathbf{X}^T\mathbf{X}$. Singular values squared are the eigenvalues.

This Class: Applications of low-rank approx. beyond compression.

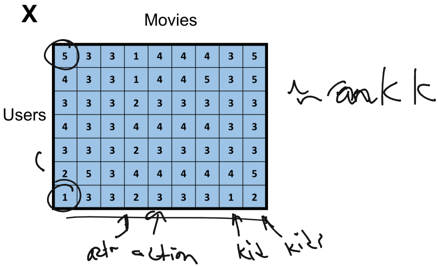
- Matrix completion and collaborative filtering
- Entity embeddings (word embeddings, node embeddings, etc.)
- Low-rank approximation fornon-linear dimensionality reduction.
- Spectral graph theory, spectral clustering.

MATRIX COMPLETION

Consider a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ which we cannot fully observe but believe is close to rank-k (i.e., well approximated by a rank k matrix).

MATRIX COMPLETION

Consider a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ which we cannot fully observe but believe is close to rank-k (i.e., well approximated by a rank k matrix). Classic example: the Netflix prize problem. 1, 2, 3, 4, 5



X

Movies

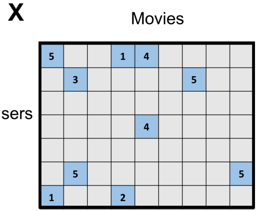
Users

5	3	3	1	4	4	4	3	5
4	3	3	1	4	4	5	3	5
3	3	3	2	3	3	3	3	3
4	3	3	4	4	4	4	3	3
3	3	3	2	3	3	3	3	3
2	5	3	4	4	4	4	4	5
1	3	3	2	3	3	3	1	2

≈ rank k

actn
action
kid
kids

Consider a matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$ which we cannot fully observe but believe is close to rank-k (i.e., well approximated by a rank k matrix). Classic example: the Netflix prize problem.



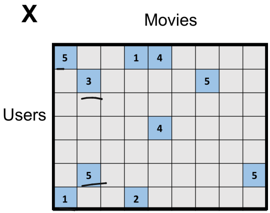
5			1	4				
	3					5		
				4				
	5							5
1			2					

MATRIX COMPLETION

Consider a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ which we cannot fully observe but believe is close to rank-k (i.e., well approximated by a rank k matrix). Classic example: the Netflix prize problem.

LRA : solved by PCA

$$Y = \underset{\text{rank-}B}{\arg\min} \|X - B\|_F^2 = \sum_{i,j} (X_{ij} - B_{ij})^2$$

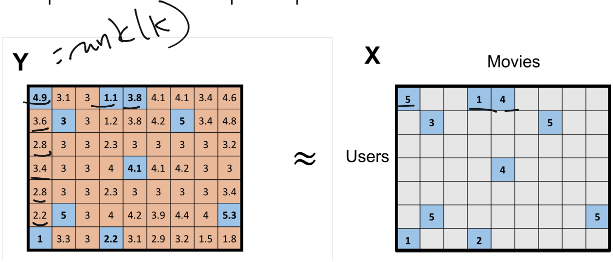


5			1	4				
	3					5		
				4				
	5							5
1			2					

Solve: $Y = \underset{\text{rank }-k\ \mathbf{B}}{\arg\min} \underbrace{\sum_{\text{observed }(j,k)}} \left[\mathbf{X}_{j,k} - \mathbf{B}_{j,k}\right]^2$

MATRIX COMPLETION

Consider a matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$ which we cannot fully observe but believe is close to rank-k (i.e., well approximated by a rank k matrix). Classic example: the Netflix prize problem.



Y

4.9	3.1	3	1.1	3.8	4.1	4.1	3.4	4.6
3.6	3	3	1.2	3.8	4.2	5	3.4	4.8
2.8	3	3	2.3	3	3	3	3	3.2
3.4	3	3	4	4.1	4.1	4.2	3	3
2.8	3	3	2.3	3	3	3	3	3.4
2.2	5	3	4	4.2	3.9	4.4	4	5.3
1	3.3	3	2.2	3.1	2.9	3.2	1.5	1.8

X

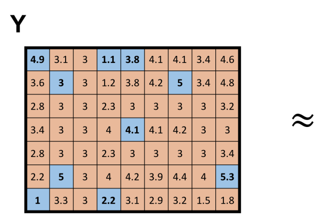
5			1	4				
	3					5		
				4				
	5							5
1			2					

Solve: $Y = \underset{\text{rank}-k\ \mathbf{B}}{\arg\min} \sum_{\text{observed }(j,k)} \left[\mathbf{X}_{j,k} - \mathbf{B}_{j,k}\right]^2$

MATRIX COMPLETION

Consider a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ which we cannot fully observe but believe is close to rank-k (i.e., well approximated by a rank k matrix). Classic example: the Netflix prize problem.

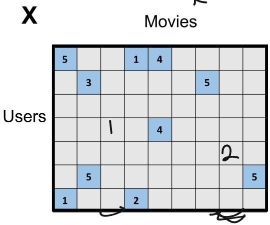
$B = {}_{n}\underbrace{[U]}_{k}[\underline{W}^T] \}k$



Y

4.9	3.1	3	1.1	3.8	4.1	4.1	3.4	4.6
3.6	3	3	1.2	3.8	4.2	5	3.4	4.8
2.8	3	3	2.3	3	3	3	3	3.2
3.4	3	3	4	4.1	4.1	4.2	3	3
2.8	3	3	2.3	3	3	3	3	3.4
2.2	5	3	4	4.2	3.9	4.4	4	5.3
1	3.3	3	2.2	3.1	2.9	3.2	1.5	1.8

$\approx$



X

Movies

Users

5			1	4				
	3					5		
		1		4				
							2	
	5							5
1			2					

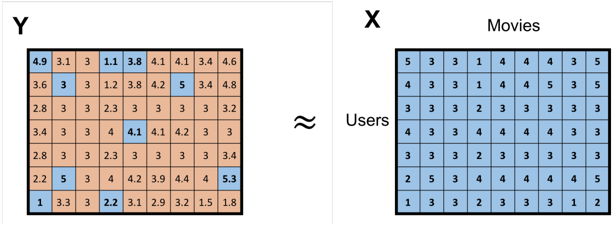
Solve: $$Y = \underset{\text{rank } -k \ \mathbf{B}}{\arg\min} \sum_{\text{observed } (j,k)} \left[\mathbf{X}_{j,k} - \mathbf{B}_{j,k}\right]^2$$

Under certain assumptions, can show that **Y** well approximates **X** on both the observed and (most importantly) unobserved entries.

$y = \arg\min_{z} f(z)$

MATRIX COMPLETION

Consider a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ which we cannot fully observe but believe is close to rank-k (i.e., well approximated by a rank k matrix). Classic example: the Netflix prize problem.



Y

4.9	3.1	3	**1.1**	**3.8**	4.1	4.1	3.4	4.6
3.6	**3**	3	1.2	3.8	4.2	**5**	3.4	4.8
2.8	3	3	2.3	3	3	3	3	3.2
3.4	3	3	4	**4.1**	4.1	4.2	3	3
2.8	3	3	2.3	3	3	3	3	3.4
2.2	**5**	3	4	4.2	3.9	4.4	4	**5.3**
1	3.3	3	**2.2**	3.1	2.9	3.2	1.5	1.8

$\approx$

X

Movies

Users								
5	3	3	1	4	4	4	3	5
4	3	3	1	4	4	5	3	5
3	3	3	2	3	3	3	3	3
4	3	3	4	4	4	4	3	3
3	3	3	2	3	3	3	3	3
2	5	3	4	4	4	4	4	5
1	3	3	2	3	3	3	1	2

Solve: $Y = \underset{\text{rank}-k\ \mathbf{B}}{\arg\min} \sum_{\text{observed } (j,\cancel{k}i)} \left[\mathbf{X}_{j,\cancel{k}i} - \mathbf{B}_{j,\cancel{k}i}\right]^2$

Under certain assumptions, can show that **Y** well approximates **X** on both the observed and (most importantly) unobserved entries.

ENTITY EMBEDDINGS

Dimensionality reduction embeds d-dimensional vectors into k dimensions. But what about when you want to embed objects other than vectors?

ENTITY EMBEDDINGS

Dimensionality reduction embeds d-dimensional vectors into k dimensions. But what about when you want to embed objects other than vectors?

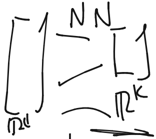
- Documents (for topic-based search and classification)
- Words (to identify synonyms, translations, etc.)
- Nodes in a social network

ENTITY EMBEDDINGS

Dimensionality reduction embeds *d*-dimensional vectors into *k* dimensions. But what about when you want to embed objects other than vectors?

- Documents (for topic-based search and classification)
- Words (to identify synonyms, translations, etc.)
- Nodes in a social network

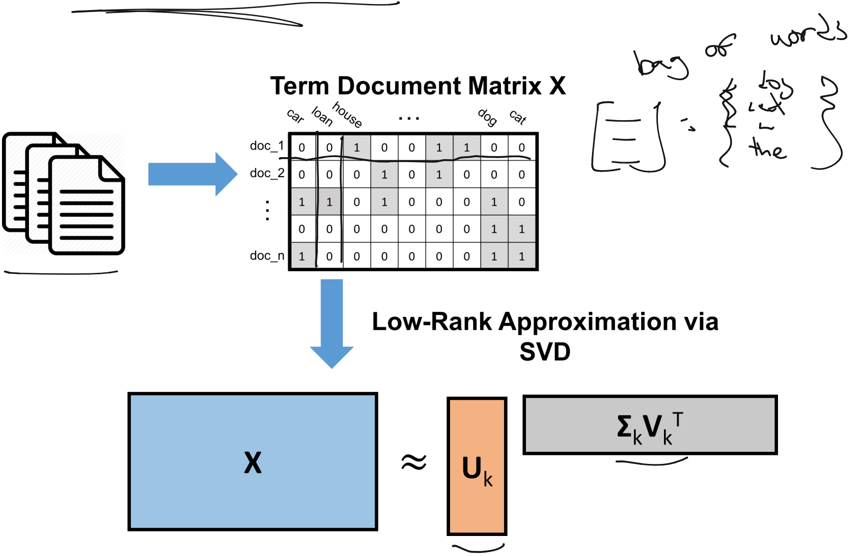
tune with a neural net



Usual Approach: Convert each item into a high-dimensional feature vector and then apply low-rank approximation.

document → high dimensional vector $x \in \mathbb{R}^d$ $\xrightarrow{LRA}$ low dimensional vector $\tilde{x} \in \mathbb{R}^k$

EXAMPLE: LATENT SEMANTIC ANALYSIS



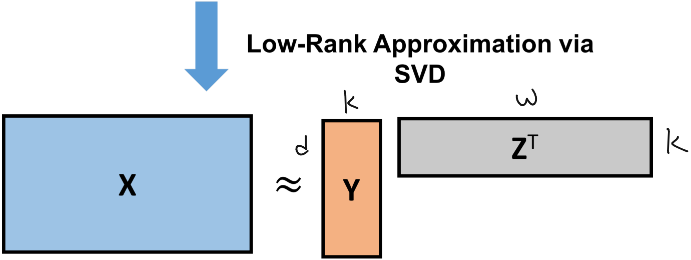
	car	loan	house		...			dog	cat
doc_1	0	0	1	0	0	1	1	0	0
doc_2	0	0	0	1	0	1	0	0	0
⋮	1	1	0	1	0	0	0	1	0
	0	0	0	0	0	0	0	1	1
doc_n	1	0	0	0	0	0	0	1	1

$$X \approx U_k \, \Sigma_k V_k^T$$

EXAMPLE: LATENT SEMANTIC ANALYSIS

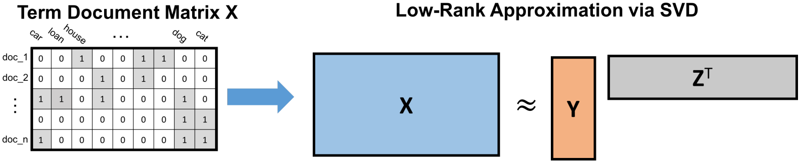
Term Document Matrix X

	car	loan	house		. . .			dog	cat
doc_1	0	0	1	0	0	1	1	0	0
doc_2	0	0	0	1	0	1	0	0	0
⋮	1	1	0	1	0	0	0	1	0
	0	0	0	0	0	0	0	1	1
doc_n	1	0	0	0	0	0	0	1	1



Low-Rank Approximation via SVD

EXAMPLE: LATENT SEMANTIC ANALYSIS

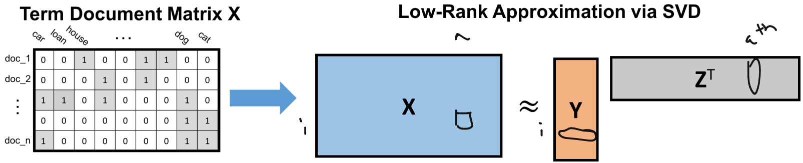


Term Document Matrix X

	car	loan	house		…			dog	cat
doc_1	0	0	1	0	0	1	1	0	0
doc_2	0	0	0	1	0	1	0	0	0
⋮	1	1	0	1	0	0	0	1	0
	0	0	0	0	0	0	0	1	1
doc_n	1	0	0	0	0	0	0	1	1

Low-Rank Approximation via SVD

EXAMPLE: LATENT SEMANTIC ANALYSIS



Term Document Matrix X

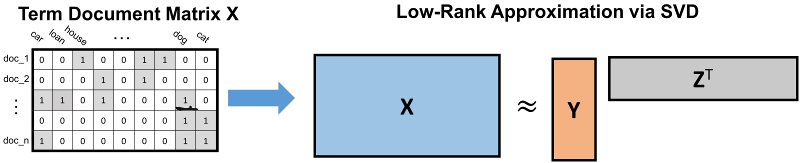
	car	loan	house	…				dog	cat
doc_1	0	0	1	0	0	1	1	0	0
doc_2	0	0	0	1	0	1	0	0	0
⋮	1	1	0	1	0	0	0	1	0
	0	0	0	0	0	0	0	1	1
doc_n	1	0	0	0	0	0	0	1	1

Low-Rank Approximation via SVD

- If the error $\underline{\|X - YZ^T\|}_F$ is small, then on average,

$$\underline{X_{i,a}} \approx \underline{(YZ^T)_{i,a}} = \langle \vec{y}_i, \vec{z}_a \rangle.$$

EXAMPLE: LATENT SEMANTIC ANALYSIS



Term Document Matrix X

	car	loan	house		…			dog	cat
doc_1	0	0	1	0	0	1	1	0	0
doc_2	0	0	0	1	0	1	0	0	0
⋮	1	1	0	1	0	0	0	1	0
	0	0	0	0	0	0	0	1	1
doc_n	1	0	0	0	0	0	0	1	1

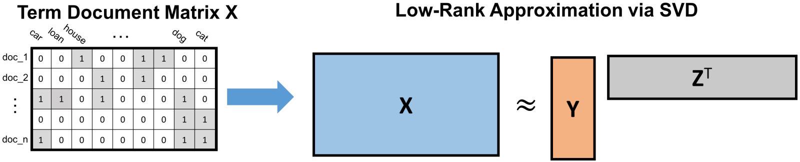
Low-Rank Approximation via SVD

- If the error $\|\mathbf{X} - \mathbf{YZ}^T\|_F$ is small, then on average,

$$\underline{\mathbf{X}_{i,a} \approx (\mathbf{YZ}^T)_{i,a}} = \langle \vec{y}_i, \vec{z}_a \rangle.$$

- I.e., $\underline{\langle \vec{y}_i, \vec{z}_a \rangle} \approx 1$ when doc_i contains $word_a$.

EXAMPLE: LATENT SEMANTIC ANALYSIS



Term Document Matrix X

	car	loan	house		⋯			dog	cat
doc_1	0	0	1	0	0	1	1	0	0
doc_2	0	0	0	1	0	1	0	0	0
⋮	1	1	0	1	0	0	0	1	0
	0	0	0	0	0	0	0	1	1
doc_n	1	0	0	0	0	0	0	1	1

Low-Rank Approximation via SVD

y_i

y_j

- If the error $\|\mathbf{X} - \mathbf{Y}\mathbf{Z}^T\|_F$ is small, then on average,

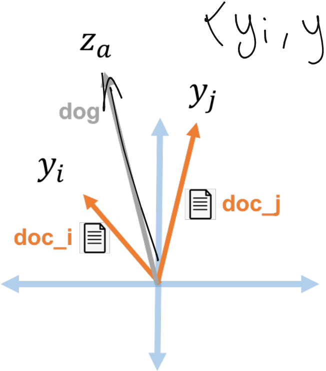
$$\mathbf{X}_{i,a} \approx (\mathbf{Y}\mathbf{Z}^T)_{i,a} = \langle \vec{y}_i, \vec{z}_a \rangle.$$

- I.e., $\langle \vec{y}_i, \vec{z}_a \rangle \approx 1$ when doc_i contains $word_a$.
- If $\underline{doc_i}$ and $\underline{doc_j}$ both contain $word_a$, $\underline{\langle \vec{y}_i, \vec{z}_a \rangle} \approx \underline{\langle \vec{y}_j, \vec{z}_a \rangle} \approx 1$.

EXAMPLE: LATENT SEMANTIC ANALYSIS

If *doc*$_i$ and *doc*$_j$ both contain *word*$_a$, $\langle \vec{y}_i, \vec{z}_a \rangle \approx \langle \vec{y}_j, \vec{z}_a \rangle \approx 1$

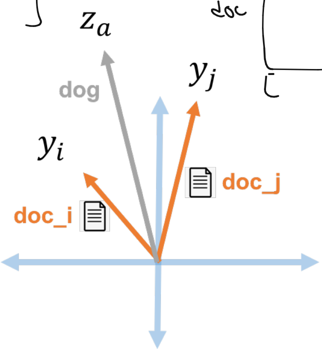
$\langle y_i, y_j \rangle$ is positive large



EXAMPLE: LATENT SEMANTIC ANALYSIS

If doc_i and doc_j both contain $word_a$, $\langle \vec{y}_i, \vec{z}_a \rangle \approx \langle \vec{y}_j, \vec{z}_a \rangle \approx 1$

$X \approx Y [\, Z^T \,]$



term

doc $\begin{bmatrix} \; \end{bmatrix} \approx [Y][Z^T]$

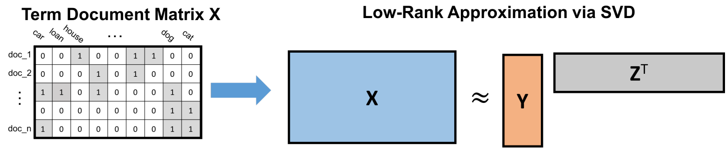
[new document]

$y_n \in \mathbb{R}^k$

$y_n^T Z^T \approx$ [new document]

Another View: Each column of **Y** represents a 'topic'. $\vec{y}_i(j)$ indicates how much doc_i belongs to topic j. $\vec{z}_a(j)$ indicates how much $word_a$ associates with that topic.

EXAMPLE: LATENT SEMANTIC ANALYSIS



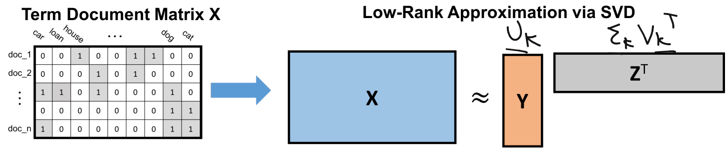
Term Document Matrix X

	car	loan	house		…			dog	cat
doc_1	0	0	1	0	0	1	1	0	0
doc_2	0	0	0	1	0	1	0	0	0
⋮	1	1	0	1	0	0	0	1	0
	0	0	0	0	0	0	0	1	1
doc_n	1	0	0	0	0	0	0	1	1

Low-Rank Approximation via SVD

- Just like with documents, $\underbrace{\vec{z}_a}$ and $\underbrace{\vec{z}_b}$ will tend to have high dot product if *word*$_a$ and *word*$_b$ appear in many of the same documents.

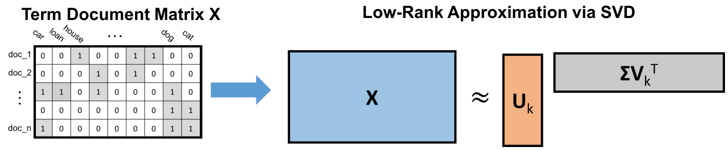
EXAMPLE: LATENT SEMANTIC ANALYSIS



	car	loan	house		…			dog	cat
doc_1	0	0	1	0	0	1	1	0	0
doc_2	0	0	0	1	0	1	0	0	0
⋮	1	1	0	1	0	0	0	1	0
	0	0	0	0	0	0	0	1	1
doc_n	1	0	0	0	0	0	0	1	1

- Just like with documents, $\vec{z}_a$ and $\vec{z}_b$ will tend to have high dot product if *word*$_a$ and *word*$_b$ appear in many of the same documents.
- In an SVD decomposition we set $\mathbf{Z}^T = \mathbf{\Sigma}_k \mathbf{V}_K^T$.
- The columns of $\mathbf{V}_k$ are equivalently: the top k eigenvectors of $\mathbf{X}^T\mathbf{X}$.

EXAMPLE: LATENT SEMANTIC ANALYSIS



Term Document Matrix X

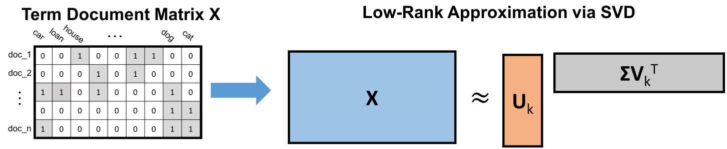
	car	loan	house		…			dog	cat
doc_1	0	0	1	0	0	1	1	0	0
doc_2	0	0	0	1	0	1	0	0	0
⋮	1	1	0	1	0	0	0	1	0
	0	0	0	0	0	0	0	1	1
doc_n	1	0	0	0	0	0	0	1	1

Low-Rank Approximation via SVD

- Just like with documents, $\vec{z}_a$ and $\vec{z}_b$ will tend to have high dot product if $word_a$ and $word_b$ appear in many of the same documents.
- In an SVD decomposition we set $\mathbf{Z}^T = \mathbf{\Sigma}_k \mathbf{V}_K^T$.
- The columns of $\mathbf{V}_k$ are equivalently: the top k eigenvectors of $\mathbf{X}^T\mathbf{X}$. The eigendecomposition of $\underline{\mathbf{X}^T\mathbf{X}}$ is $\mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T$.

$X = U\Sigma V^T \quad V\Sigma U^T U \Sigma V^T = V\Sigma^2 V^T$

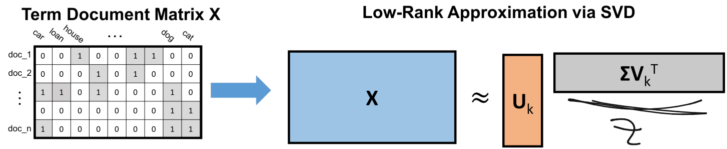
EXAMPLE: LATENT SEMANTIC ANALYSIS



	car	loan	house		…			dog	cat
doc_1	0	0	1	0	0	1	1	0	0
doc_2	0	0	0	1	0	1	0	0	0
⋮	1	1	0	1	0	0	0	1	0
	0	0	0	0	0	0	0	1	1
doc_n	1	0	0	0	0	0	0	1	1

- Just like with documents, $\vec{z}_a$ and $\vec{z}_b$ will tend to have high dot product if $word_a$ and $word_b$ appear in many of the same documents.
- In an SVD decomposition we set $\mathbf{Z}^T = \mathbf{\Sigma}_k \mathbf{V}_K^T$.
- The columns of $\mathbf{V}_k$ are equivalently: the top k eigenvectors of $\mathbf{X}^T\mathbf{X}$. The eigendecomposition of $\mathbf{X}^T\mathbf{X}$ is $\mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T$. SVD X^TX
- What is the best rank-k approximation of $\mathbf{X}^T\mathbf{X}$? I.e. $\arg\min_{\text{rank}-k\ \mathbf{B}} \|\mathbf{X}^T\mathbf{X} - \mathbf{B}\|_F = V_k \Sigma_k^2 V_k^T = Z Z^T$

EXAMPLE: LATENT SEMANTIC ANALYSIS



Term Document Matrix X

	car	loan	house		…			dog	cat
doc_1	0	0	1	0	0	1	1	0	0
doc_2	0	0	0	1	0	1	0	0	0
⋮	1	1	0	1	0	0	0	1	0
	0	0	0	0	0	0	0	1	1
doc_n	1	0	0	0	0	0	0	1	1

Low-Rank Approximation via SVD

- Just like with documents, $\vec{z}_a$ and $\vec{z}_b$ will tend to have high dot product if $word_a$ and $word_b$ appear in many of the same documents.
- In an SVD decomposition we set $\mathbf{Z}^T = \mathbf{\Sigma}_k \mathbf{V}_K^T$.
- The columns of $\mathbf{V}_k$ are equivalently: the top k eigenvectors of $\mathbf{X}^T\mathbf{X}$. The eigendecomposition of $\mathbf{X}^T\mathbf{X}$ is $\mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T$.
- What is the best rank-k approximation of $\mathbf{X}^T\mathbf{X}$? I.e. $\arg\min_{\text{rank}-k\ \mathbf{B}} \|\mathbf{X}^T\mathbf{X} - \mathbf{B}\|_F$
- $\mathbf{X}^T\mathbf{X} = \underline{\mathbf{V}_k \mathbf{\Sigma}_k^2 \mathbf{V}_k^T} = \mathbf{Z}\mathbf{Z}^T$.

EXAMPLE: WORD EMBEDDING

$Z \in {}_{w}\left[\ \right]^{k}$ each row is an embedding of a work

$ZZ^T \approx X^TX$ $w \times w$ matrixe

LSA gives a way of embedding words into *k*-dimensional space.

- Embedding is via low-rank approximation of X^TX: where $\underline{(X^TX)_{a,b}}$ is the number of documents that both $word_a$ and $word_b$ appear in.

$$\begin{array}{c|ccc|} & & \text{docs} & \\ & 1 & 0 & 1 \\ \text{words} & 0 & 1 & 1 \\ & 0 & 1 & 1 \end{array} \quad \begin{array}{|cccc|l} \text{words} & & & & \\ 0 & & & & \\ 1 & & & & \text{doc} \\ 1 & & & & \end{array}$$

$$0 \cdot 0 + 1 \cdot 1 + 1 \cdot 1 = 2$$

$(X^TX)_{ab}$ = # docs that $word_a$ + $word_b$ both appear in

EXAMPLE: WORD EMBEDDING

LSA gives a way of embedding words into k-dimensional space.

- Embedding is via low-rank approximation of $\mathbf{X}^T\mathbf{X}$: where $(\mathbf{X}^T\mathbf{X})_{a,b}$ is the number of documents that both $word_a$ and $word_b$ appear in.
- Think about $\mathbf{X}^T\mathbf{X}$ as a similarity matrix (gram matrix, kernel matrix) with entry (a, b) being the similarity between $word_a$ and $word_b$.

ZZ^T is best LRA to this similarity matrix.

EXAMPLE: WORD EMBEDDING

LSA gives a way of embedding words into k-dimensional space.

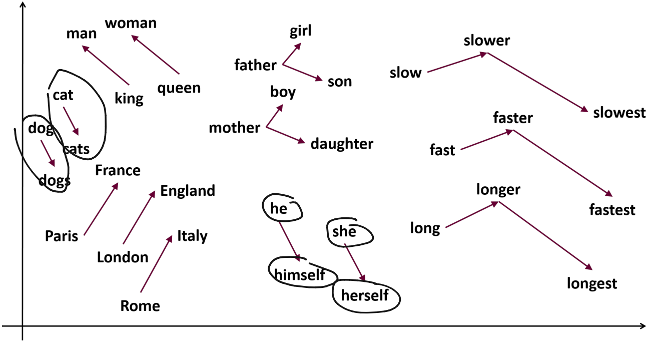
- Embedding is via low-rank approximation of $\mathbf{X}^T\mathbf{X}$: where $(\mathbf{X}^T\mathbf{X})_{a,b}$ is the number of documents that both $word_a$ and $word_b$ appear in.
- Think about $\mathbf{X}^T\mathbf{X}$ as a similarity matrix (gram matrix, kernel matrix) with entry (a, b) being the similarity between $word_a$ and $word_b$.
- Many ways to measure similarity: number of sentences both occur in, number of times both appear in the same window of w words, in similar positions of documents in different languages, etc.

EXAMPLE: WORD EMBEDDING

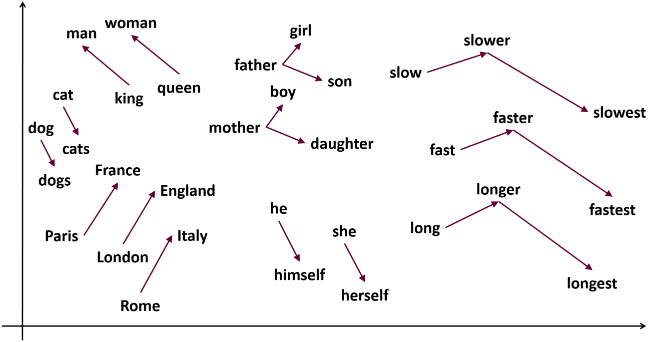
LSA gives a way of embedding words into k-dimensional space.

- Embedding is via low-rank approximation of $\mathbf{X}^T\mathbf{X}$: where $(\mathbf{X}^T\mathbf{X})_{a,b}$ is the number of documents that both $word_a$ and $word_b$ appear in.
- Think about $\mathbf{X}^T\mathbf{X}$ as a similarity matrix (gram matrix, kernel matrix) with entry (a, b) being the similarity between $word_a$ and $word_b$.
- Many ways to measure similarity: number of sentences both occur in, number of times both appear in the same window of w words, in similar positions of documents in different languages, etc.
- Replacing $\mathbf{X}^T\mathbf{X}$ with these different metrics (sometimes appropriately transformed) leads to popular word embedding algorithms: word2vec, GloVe, fastText, etc.

EXAMPLE: WORD EMBEDDING



EXAMPLE: WORD EMBEDDING



Note: word2vec is typically described as a neural-network method, but it is really just low-rank approximation of a specific similarity matrix. *Neural word embedding as implicit matrix factorization*, Levy and Goldberg.