COMPSCI 514: Algorithms for Data Science

Cameron Musco

University of Massachusetts Amherst. Fall 2022.
Lecture 18

Logistics

- Problem Set 3 is due Monday at 11:59pm.
- No quiz due Monday.
- I will hold additional office hours on Thursday from 11:30am-12:40pm.

Summary

Last Class

- The Singular Value Decomposition (SVD) and its connection to eigendecomposition of X^TX and XX^T, and low-rank approximation.

This Class: Application of Low-Rank Approximation Beyond Compression

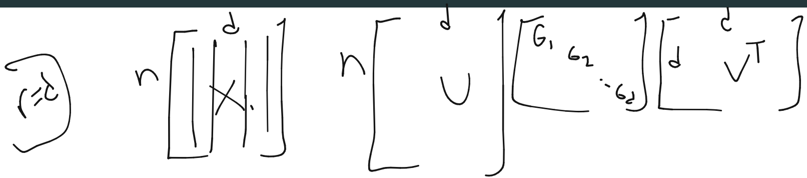
- Low-rank matrix completion (predicting missing measurements using low-rank structure).
- Entity embeddings (e.g., word embeddings, node embeddings).
- Low-rank approximation for non-linear dimensionality reduction.
- ~~Eigendecomposition to partition graphs into clusters.~~

SVD Review

$X=$

- Every $\underrightarrow{\mathbf{X} \in \mathbb{R}^{n \times d}}$ can be written in its SVD as $\mathbf{U\Sigma V}^T$.

SVD Review

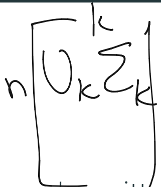


- Every $\mathbf{X} \in \mathbb{R}^{n\times d}$ can be written in its SVD as $\mathbf{U\Sigma V}^T$.

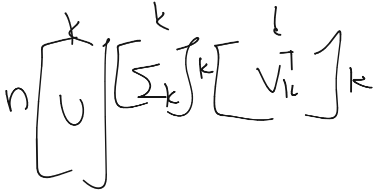
rank(x)=r

- $\mathbf{U} \in \mathbb{R}^{n\times r}$ (orthonormal) contains the eigenvectors of $\underline{\mathbf{XX}^T}$.
  $\mathbf{V} \in \mathbb{R}^{d\times r}$ (orthonormal) contains the eigenvectors of $\underline{\mathbf{X}^\mathbf{T}\mathbf{X}}$.
  $\mathbf{\Sigma} \in \mathbb{R}^{r\times r}$ (diagonal) contains their $\underline{\text{eigenvalues}}$.

SVD Review

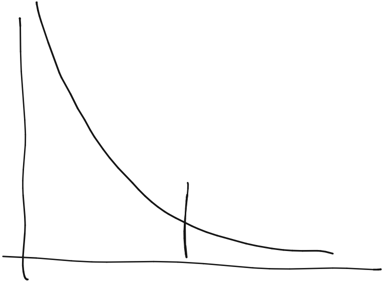


- Every $\mathbf{X} \in \mathbb{R}^{n\times d}$ can be written in its SVD as $\mathbf{U\Sigma V}^T$.
- $\mathbf{U} \in \mathbb{R}^{n\times r}$ (orthonormal) contains the eigenvectors of $\mathbf{XX}^T$.
  $\underline{\mathbf{V} \in \mathbb{R}^{d\times r}}$ (orthonormal) contains the eigenvectors of $\mathbf{X}^\mathsf{T}\mathbf{X}$.
  $\mathbf{\Sigma} \in \mathbb{R}^{r\times r}$ (diagonal) contains their eigenvalues.
- $\underbrace{\mathbf{U}_k\mathbf{U}_k^T}_{\text{nxn proj.}}\mathbf{X} = \underbrace{\mathbf{XV}_k\mathbf{V}_k^T}_{\text{dxd proj}} = \underline{\mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^T} = \underset{\mathbf{B}\text{ s.t. }\operatorname{rank}(\mathbf{B})\leq k}{\arg\min} \|\mathbf{X}-\mathbf{B}\|_F.$



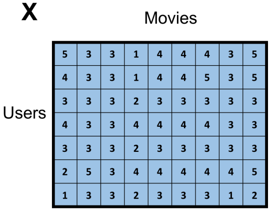
Matrix Completion

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.



X

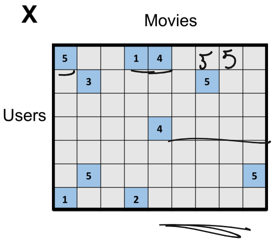
Movies

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

Matrix Completion

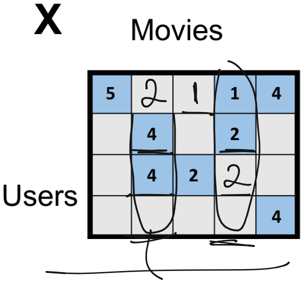
Consider a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ which we cannot fully observe but believe is close to rank-k (i.e., well approximated by a rank k matrix). Classic example: the Netflix prize problem.



5			1	4		5	5	
	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.



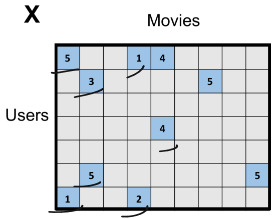
5	2	1	1	4
	4		2	
	4	2	2	
				4

Assume rank($\mathbf{X}$)=1

$$X_2 = 2 \cdot X_4$$

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.

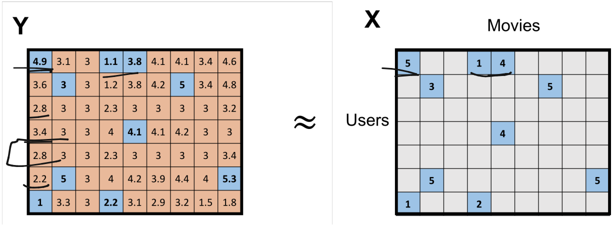


Solve: $$Y = \underset{\mathbf{B}\text{ s.t. }\underline{\operatorname{rank}(\mathbf{B})\leq k}}{\arg\min} \; \sum_{\underline{\text{observed }(j,k)}} \underline{\left[\mathbf{X}_{j,k} - \mathbf{B}_{j,k}\right]^2}$$

$$\min_{\substack{B:\\ \operatorname{rank}(B)\leq k}} \|X - B\|_F^2$$

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

Movies

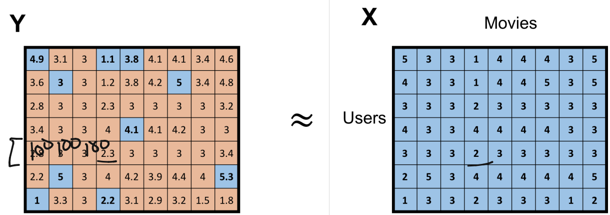
Users

5			1	4				
	3					5		
				4				
	5							5
1			2					

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

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

100 100 100

$\approx$

X

Users \ Movies								
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: $$Y = \underset{\mathbf{B} \text{ s.t. } \operatorname{rank}(\mathbf{B}) \leq k}{\arg\min} \sum_{\text{observed } (j,k)} \left[\mathbf{X}_{j,k} - \mathbf{B}_{j,k}\right]^2$$

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

Entity Embeddings

$\mathbb{R}^d$

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

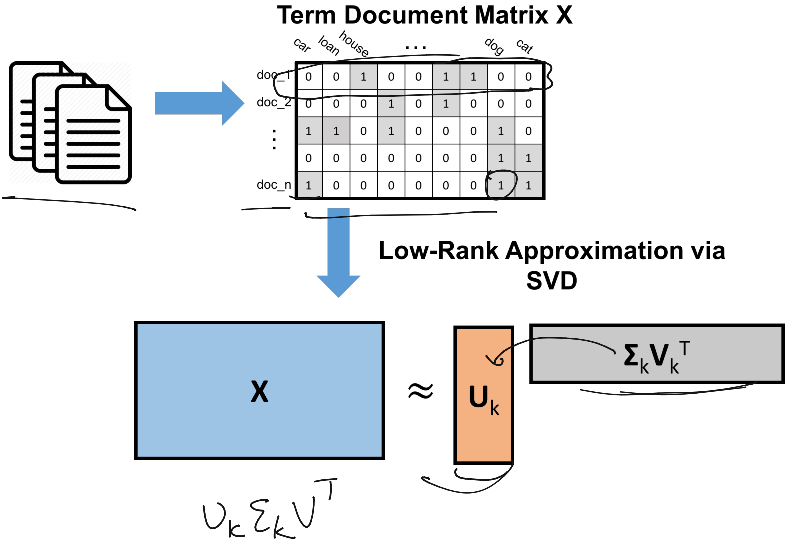
Entity Embeddings

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

Classic Approach: Convert each item into a (very) high-dimensional feature vector and then apply low-rank approximation.

Example: Latent Semantic Analysis



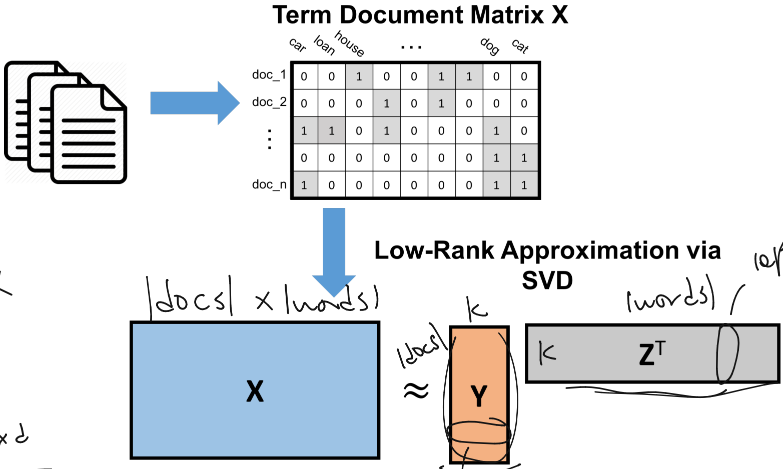
Term Document Matrix X

	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

$U_k \Sigma_k V^T$

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

UU^TX

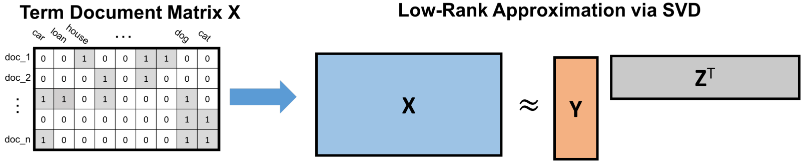
$n \times d$

XVV^T

$XV \leftarrow n \times k$

$\mathbb{R}^k$ represents ~ doc

Example: Latent Semantic Analysis



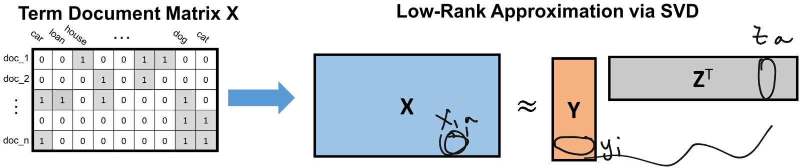
Term Document Matrix X

	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

$$\mathbf{X} \approx \mathbf{Y} \, \mathbf{Z}^T$$

Example: Latent Semantic Analysis



Term Document Matrix X

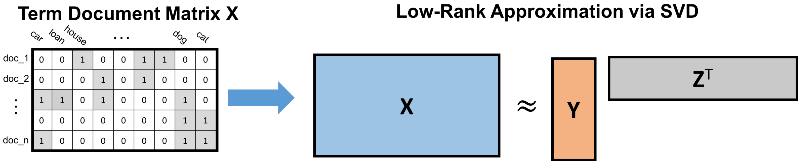
	car	loan	house		…			dog	cat
doc_1	0	0	1	0	0	1	1	0	0
doc_2	0	0	0	1	0	1	0	0	0
⋮	1	1	0	1	0	0	0	1	0
	0	0	0	0	0	0	0	1	1
doc_n	1	0	0	0	0	0	0	1	1

Low-Rank Approximation via SVD

- 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

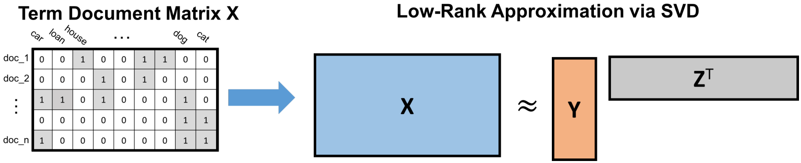
Low-Rank Approximation via SVD

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

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

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

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

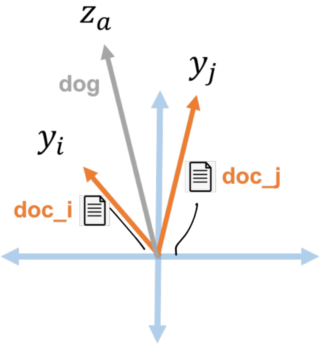
- If the error $\|\mathbf{X} - \mathbf{Y}\mathbf{Z}^T\|_F$ is small, then on average,

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

- 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 $word_a$, $\underbrace{\langle \vec{y}_i, \vec{z}_a \rangle} \approx \underbrace{\langle \vec{y}_j, \vec{z}_a \rangle} \underbrace{\approx 1}$.

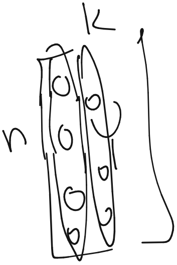
Example: Latent Semantic Analysis

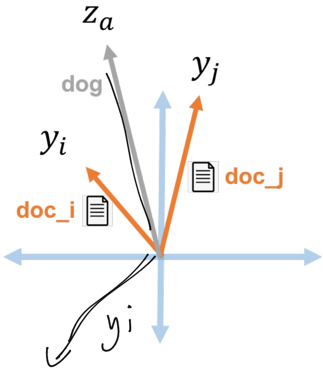
If doc_i and doc_j both contain $word_a$, $\langle \vec{y}_i, \vec{z}_a \rangle \approx \langle \vec{y}_j, \vec{z}_a \rangle \approx 1$



Example: Latent Semantic Analysis

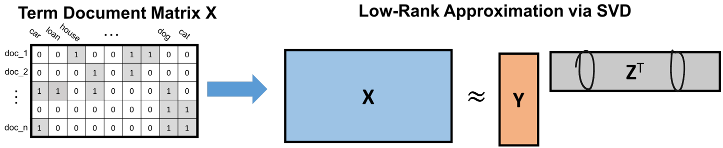
If *doc*$_i$ and *doc*$_j$ both contain *word*$_a$, $\langle \vec{y}_i, \vec{z}_a \rangle \approx \langle \vec{y}_j, \vec{z}_a \rangle \approx 1$





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



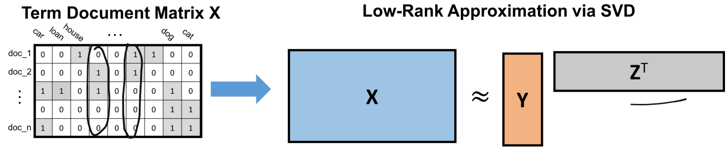
Term Document Matrix X

	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, $\vec{z}_a$ and $\vec{z}_b$ will tend to have high dot product if $word_a$ and $word_b$ appear in many of the same documents.

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 $\underline{\mathbf{Z}^T} = \underline{\mathbf{\Sigma}_k \mathbf{V}_K^T}$.
- The columns of $\mathbf{V}_k$ are equivalently: the top k eigenvectors of $\mathbf{X}^T\mathbf{X}$.

$$\text{words}\overset{\text{docs}}{\begin{bmatrix} X^T \end{bmatrix}} \;\text{docs}\overset{\text{words}}{\begin{bmatrix} X \end{bmatrix}} = \begin{bmatrix} & & (X^TX)_{a,b} \end{bmatrix}$$

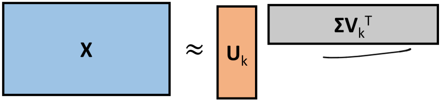
docs that a, b both appear in

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.

word

$$\text{word}\begin{bmatrix} z \end{bmatrix}\begin{bmatrix} z^T \end{bmatrix}$$

- In an SVD decomposition we set $\underline{\mathbf{Z}^T = \mathbf{\Sigma}_k\mathbf{V}_K^T}$.
- The columns of $\mathbf{V}_k$ are equivalently: the top k eigenvectors of $\mathbf{X}^T\mathbf{X}$.

+

SVD $U\Sigma V^T$

- **Claim:** $\mathbf{Z}\mathbf{Z}^T$ is the best rank-k approximation of $\mathbf{X}^T\mathbf{X}$. I.e., $\arg\min_{\text{rank}-k\ \mathbf{B}} \|\mathbf{X}^T\mathbf{X} - \mathbf{B}\|_F$

$$\underline{V_k \Sigma_k^2 V_k^T}$$

$$\text{SVD}\quad X^T X = V\Sigma \underbrace{U^T U}_{I} \Sigma V^T = \underline{V\Sigma^2 V^T}$$

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.

Example: Word Embedding

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

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

Example: Word Embedding

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

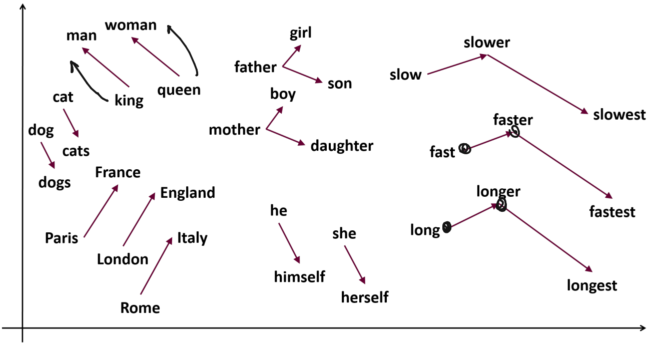
- Embedding is via low-rank approximation of $\mathbf{X}^T\mathbf{X}$: where $(\mathbf{X}^T\mathbf{X})_{a,b}$ is the number of documents that both $word_a$ and $word_b$ appear in.
- Think about $\mathbf{X}^T\mathbf{X}$ as a similarity matrix (gram matrix, kernel matrix) with entry (a, b) being the similarity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

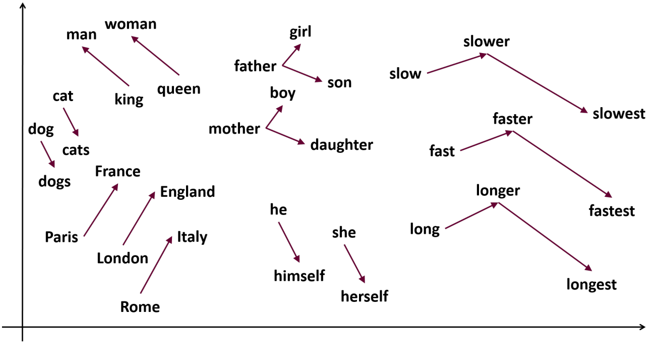
LSA gives a way of embedding words into k-dimensional space.

- Embedding is via low-rank approximation of X^TX: where $(X^TX)_{a,b}$ is the number of documents that both $word_a$ and $word_b$ appear in.
- Think about X^TX 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 X^TX with these different metrics (sometimes appropriately transformed) leads to popular word embedding algorithms: word2vec, GloVe, fastText, etc.

Example: Word Embedding

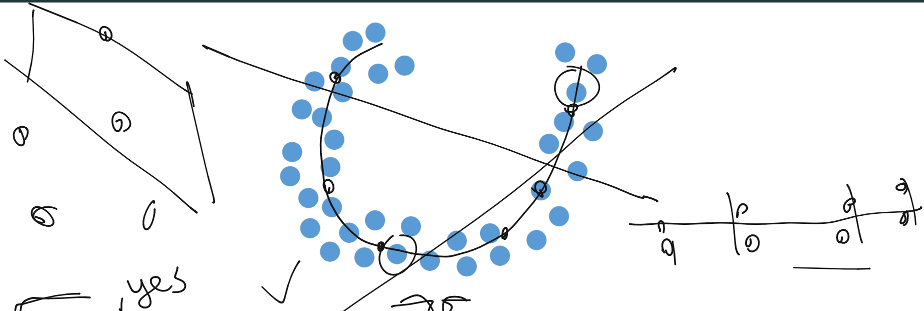


Example: Word Embedding



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

Non-Linear Dimensionality Reduction

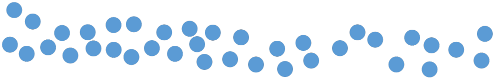


Is this set of points compressible? Does it lie close to a low-dimensional subspace? (A 1-dimensional subspace of $\mathbb{R}^d$.)

✗ no

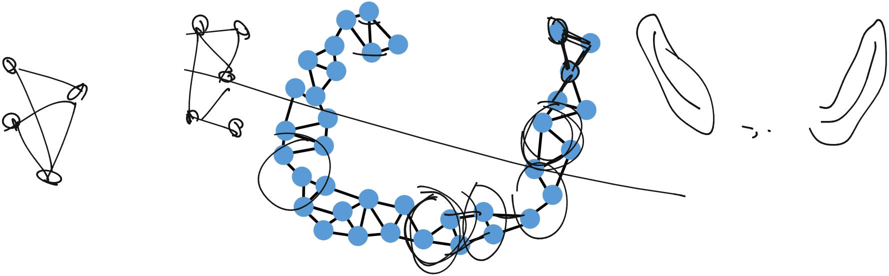
✗✓

Non-Linear Dimensionality Reduction



Is this set of points compressible? Does it lie close to a low-dimensional subspace? (A 1-dimensional subspace of $\mathbb{R}^d$.)

Non-Linear Dimensionality Reduction



Is this set of points compressible? Does it lie close to a low-dimensional subspace? (A 1-dimensional subspace of $\mathbb{R}^d$.)

A common way of automatically identifying this non-linear structure is to connect data points in a graph. E.g., a ~~k~~ r-nearest neighbor graph.

- Connect items to similar items, possibly with higher weight edges when they are more similar.

Linear Algebraic Representation of a Graph

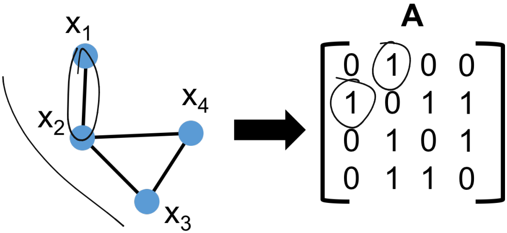
Once we have connected n data points $x_1, \ldots, x_n$ into a graph, we can represent that graph by its (weighted) adjacency matrix.

$$\underline{\mathsf{A} \in \mathbb{R}^{n \times n}} \text{ with } \mathsf{A}_{i,j} = \text{ edge weight between nodes } i \text{ and } j$$

Linear Algebraic Representation of a Graph

Once we have connected n data points $x_1, \ldots, x_n$ into a graph, we can represent that graph by its (weighted) adjacency matrix.

$$\mathrm{A} \in \mathbb{R}^{n \times n} \text{ with } \mathrm{A}_{i,j} = \text{ edge weight between nodes } i \text{ and } j$$

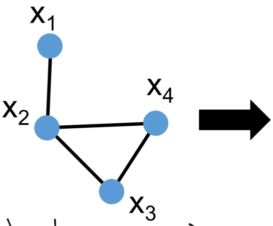


$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Linear Algebraic Representation of a Graph

Once we have connected n data points $x_1, \ldots, x_n$ into a graph, we can represent that graph by its (weighted) adjacency matrix.

$$\mathbf{A} \in \mathbb{R}^{n \times n} \text{ with } \mathbf{A}_{i,j} = \text{ edge weight between nodes } i \text{ and } j$$



$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$(X^T X)_{ab}$ = # docs contain both a, b

In LSA example, when $\mathbf{X}$ is the term-document matrix, $\mathbf{X}^T\mathbf{X}$ is like an adjacency matrix, where $word_a$ and $word_b$ are connected if they appear in at least 1 document together (edge weight is # documents they appear in together).

Adjacency Matrix Eigenvectors

How do we compute an optimal low-rank approximation of $\mathbf{A}$?

- Project onto the top k eigenvectors of $\mathbf{A}^T\mathbf{A} = \mathbf{A}^2$. These are just the eigenvectors of $\mathbf{A}$.

$$Ax = \lambda x$$

$$A^2 x = A(Ax) = A(\lambda x)$$

$$= \lambda(\lambda x)$$

$$= \lambda^2 x$$

$$A(\lambda x) = \lambda \underline{Ax} = \lambda \cdot \lambda x$$

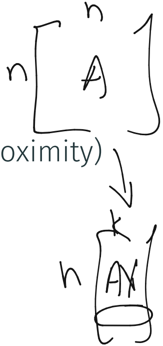
$$= \lambda^2 x$$

Adjacency Matrix Eigenvectors

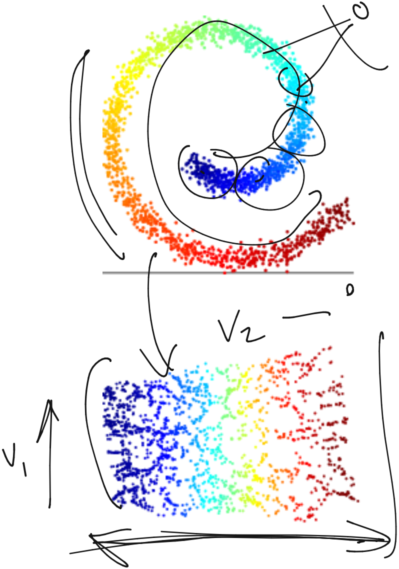
$A^TA = A \cdot A = A^2 \qquad A^T = A$

How do we compute an optimal low-rank approximation of $\mathbf{A}$?

- Project onto the top k eigenvectors of $\mathbf{A}^T\mathbf{A} = \mathbf{A}^2$. These are just the eigenvectors of $\mathbf{A}$.
- $\underline{\mathbf{A} \approx \mathbf{A}\mathbf{V}\mathbf{V}^T}$. The rows $\underline{\text{of } \mathbf{A}\mathbf{V}}$ can be thought of as 'embeddings' for the vertices.
- Similar vertices (close with regards to graph proximity) should have similar embeddings.



Spectral Embedding

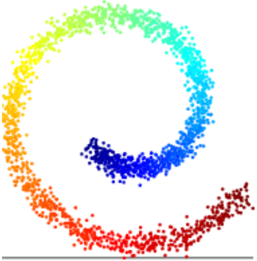


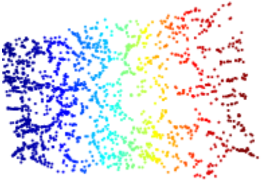
Step 1: Produce a nearest neighbor graph based on your input data in $\mathbb{R}^d$.

Step 2: Apply low-rank approximation to the graph adjacency matrix to produce embeddings in $\mathbb{R}^k$.

Step 3: Work with the data in the embedded space. Where distances represent distances in your original 'non-linear space.'

Spectral Embedding





What other methods do you know for embedding or representing data points with non-linear structure?

Questions?