COMPSCI 514: ALGORITHMS FOR DATA SCIENCE

Cameron Musco

University of Massachusetts Amherst. Fall 2019.

Lecture 14

LOGISTICS

- Midterm grades are on Moodle.
- Average was 32.67, median 33, standard deviation 6.8
- Come to office hours if you would like to see your exam/discuss solutions.

Last Few Weeks: Low-Rank Approximation and PCA

Last Few Weeks: 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 $\mathbf{X}$: $\mathbf{X} \approx \mathbf{X}\mathbf{V}\mathbf{V}^T$.
- Optimal solution via PCA (eigendecomposition of $\mathbf{X}^T\mathbf{X}$ or equivalently, SVD of $\mathbf{X}$).

Last Few Weeks: 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 $\mathbf{X}$: $\mathbf{X} \approx \mathbf{X}\mathbf{V}\mathbf{V}^T$.
- Optimal solution via PCA (eigendecomposition of $\mathbf{X}^T\mathbf{X}$ or equivalently, SVD of $\mathbf{X}$).

This Class: Non-linear dimensionality reduction.

Last Few Weeks: 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 $\mathbf{X}$: $\mathbf{X} \approx \mathbf{X}\mathbf{V}\mathbf{V}^T$.
- Optimal solution via PCA (eigendecomposition of $\mathbf{X}^T\mathbf{X}$ or equivalently, SVD of $\mathbf{X}$).

This Class: Non-linear dimensionality reduction.

- How do we compress data that does not lie close to a k-dimensional subspace?
- Spectral methods (SVD and eigendecomposition) are still key techniques in this setting.
- Spectral graph theory, spectral clustering.

End of Last Class: Embedding objects other than vectors into Euclidean space.

End of Last Class: Embedding objects other than vectors into Euclidean space.

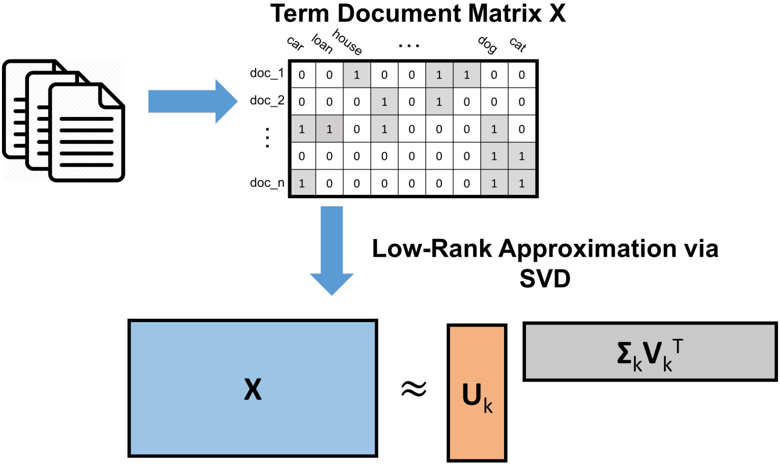
- Documents (for topic-based search and classification)
- Words (to identify synonyms, translations, etc.)
- Nodes in a social network

End of Last Class: Embedding objects other than vectors into Euclidean space.

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

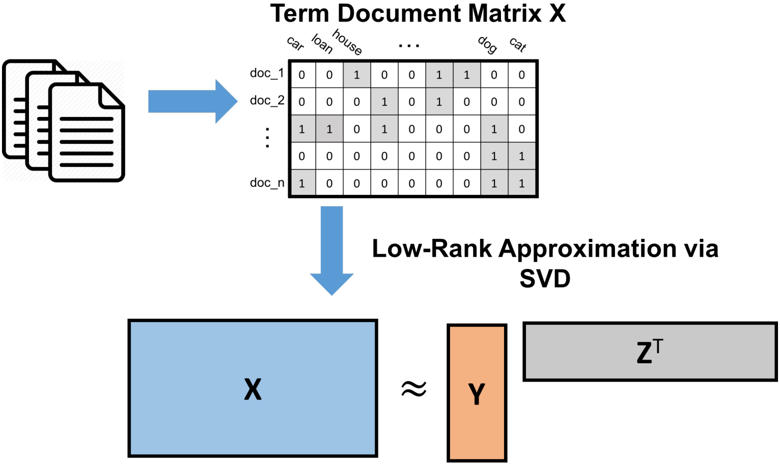
Usual Approach: Convert each item into a high-dimensional feature vector and then apply low-rank approximation

EXAMPLE: LATENT SEMANTIC ANALYSIS

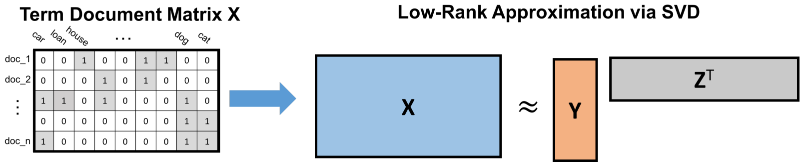


	car	loan	house		...			dog	cat
doc_1	0	0	1	0	0	1	1	0	0
doc_2	0	0	0	1	0	1	0	0	0
⋮	1	1	0	1	0	0	0	1	0
	0	0	0	0	0	0	0	1	1
doc_n	1	0	0	0	0	0	0	1	1

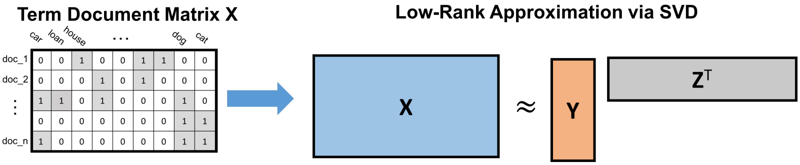
EXAMPLE: LATENT SEMANTIC ANALYSIS



EXAMPLE: LATENT SEMANTIC ANALYSIS



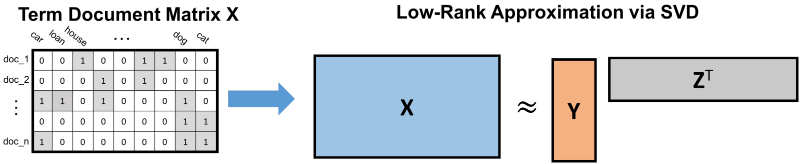
EXAMPLE: LATENT SEMANTIC ANALYSIS



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

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

EXAMPLE: LATENT SEMANTIC ANALYSIS

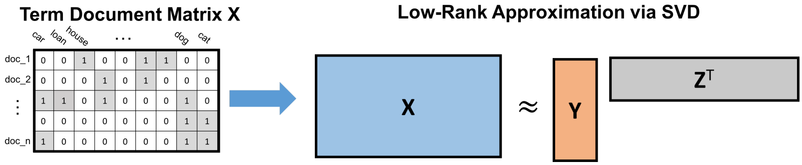


- If the error $\|\mathrm{X} - \mathrm{YZ}^T\|_F$ is small, then on average,

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

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

EXAMPLE: LATENT SEMANTIC ANALYSIS

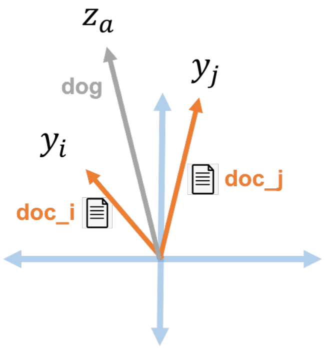


- If the error $\|\mathbf{X} - \mathbf{Y}\mathbf{Z}^T\|_F$ is small, then on average,

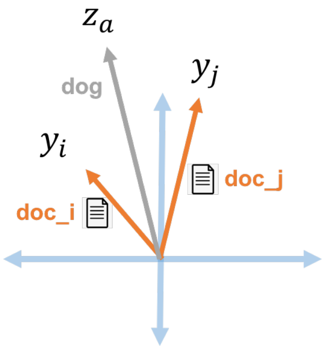
$$\mathbf{X}_{i,a} \approx (\mathbf{Y}\mathbf{Z}^T)_{i,a} = \langle \vec{y}_i, \vec{z}_a \rangle.$$

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

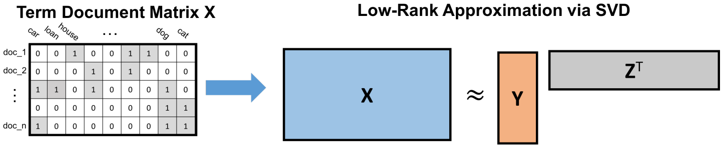
If doc_i and doc_j both contain $word_a$, $\langle \vec{y}_i, \vec{z}_a \rangle \approx \langle \vec{y}_j, \vec{z}_a \rangle = 1$



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

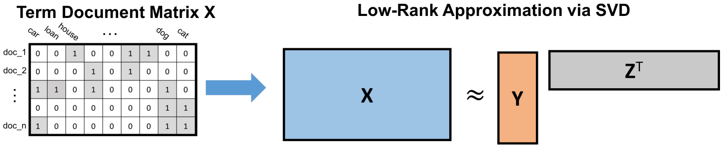


Another View: Each column of **Y** represents a 'topic'. $\vec{y}_i(j)$ indicates how much doc_i belongs to topic j. $\vec{z}_a(j)$ indicates how much $word_a$ associates with that topic.



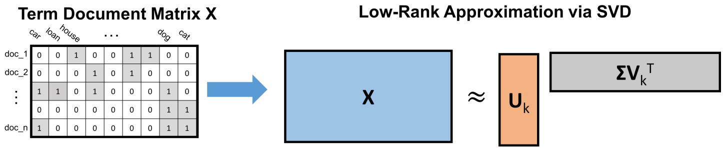
- Just like with documents, $\vec{z}_a$ and $\vec{z}_b$ will tend to have high dot product if $word_i$ and $word_j$ appear in many of the same documents.

EXAMPLE: LATENT SEMANTIC ANALYSIS

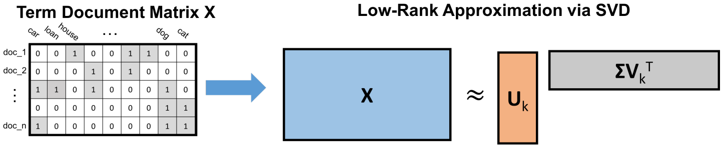


- Just like with documents, $\vec{z}_a$ and $\vec{z}_b$ will tend to have high dot product if $word_i$ and $word_j$ appear in many of the same documents.
- In an SVD decomposition we set $\mathbf{Z} = \mathbf{\Sigma}_k \mathbf{V}_K^T$.

EXAMPLE: LATENT SEMANTIC ANALYSIS

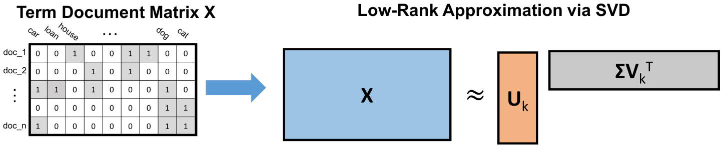


- Just like with documents, $\vec{z}_a$ and $\vec{z}_b$ will tend to have high dot product if $word_i$ and $word_j$ appear in many of the same documents.
- In an SVD decomposition we set $\mathbf{Z} = \mathbf{\Sigma}_k \mathbf{V}_K^T$.
- The columns of $\mathbf{V}_k$ are equivalently: the top k eigenvectors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



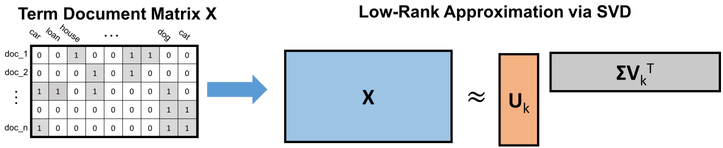
- Just like with documents, $\vec{z}_a$ and $\vec{z}_b$ will tend to have high dot product if $word_i$ and $word_j$ appear in many of the same documents.
- In an SVD decomposition we set $\mathbf{Z} = \mathbf{\Sigma}_k \mathbf{V}_K^T$.
- The columns of $\mathbf{V}_k$ are equivalently: the top k eigenvectors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

EXAMPLE: LATENT SEMANTIC ANALYSIS



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

EXAMPLE: LATENT SEMANTIC ANALYSIS



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

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

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

- Think about $\mathbf{X}\mathbf{X}^T$ as a similarity matrix (gram matrix, kernel matrix) with entry (a, b) being the similarity between $word_a$ and $word_b$.

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

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

- Think about $\mathbf{X}\mathbf{X}^T$ 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 time both appear in the same window of w words, in similar positions of documents in different languages, etc.

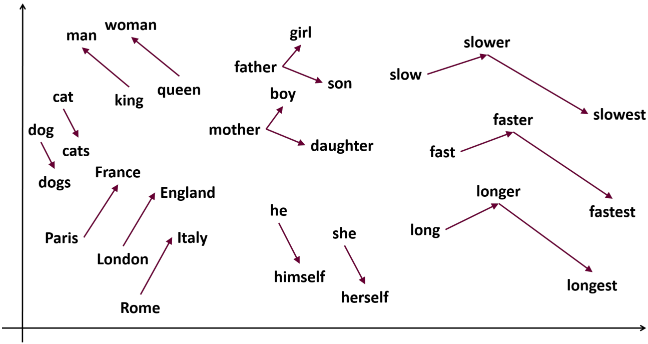
EXAMPLE: WORD EMBEDDING

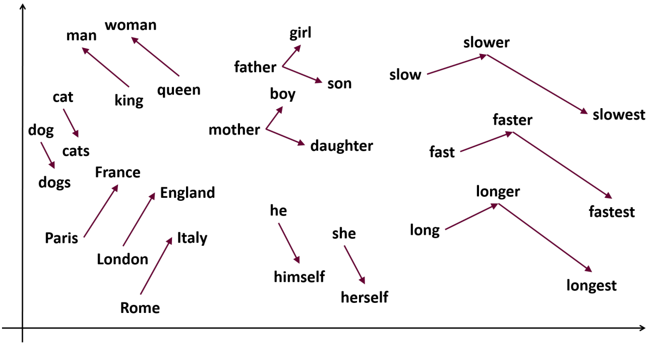
LSA gives a way of embedding words into k-dimensional space.

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

- Think about $\mathbf{X}\mathbf{X}^T$ 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 time both appear in the same window of w words, in similar positions of documents in different languages, etc.
- Replacing $\mathbf{X}\mathbf{X}^T$ with these different metrics (sometimes appropriately transformed) leads to popular word embedding algorithms: word2vec, GloVe, fastTest, etc.

EXAMPLE: WORD EMBEDDING





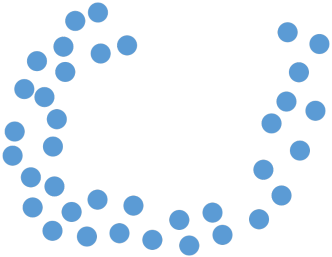
Note: word2vec is typically described as a neural-network method, but it is really just low-rank approximation of a specific similarity matrix. *Neural word embedding as implicit matrix factorization*, Levy and Goldberg.

A common way of encoding similarity is via a graph. E.g., a k-nearest neighbor graph.

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

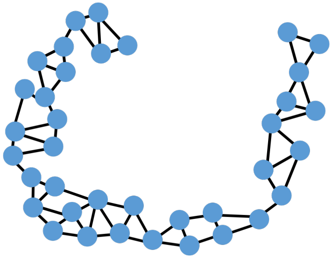
A common way of encoding similarity is via a graph. E.g., a k-nearest neighbor graph.

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



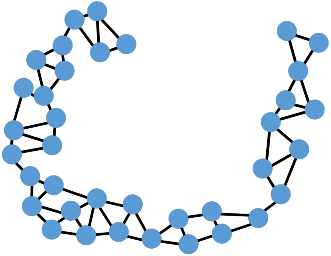
A common way of encoding similarity is via a graph. E.g., a k-nearest neighbor graph.

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



A common way of encoding similarity is via a graph. E.g., a k-nearest neighbor graph.

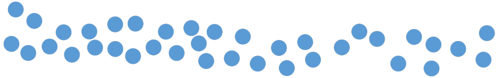
- Connect items to similar items, possibly with higher weight edges when they are more similar.



Is this set of points compressible? Does it lie close to a low-dimensional subspace?

A common way of encoding similarity is via a graph. E.g., a k-nearest neighbor graph.

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



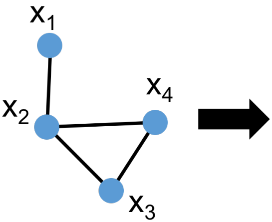
Is this set of points compressible? Does it lie close to a low-dimensional subspace?

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.

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

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.

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

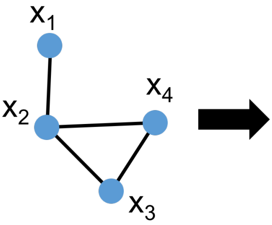


$$\begin{array}{c} \mathbf{A} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

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.

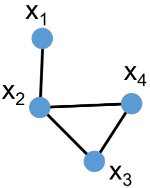
$$\mathbf{A} \in \mathbb{R}^{n \times n} \text{ with } \mathbf{A}_{i,j} = \text{ edge weight between nodes } i \text{ and } j$$



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

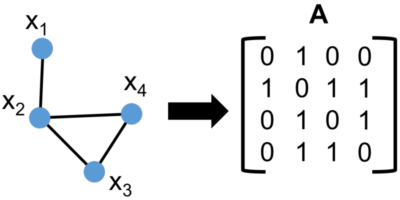
In LSA example, when $\mathbf{X}$ is the term-document matrix, $\mathbf{X}^T\mathbf{X}$ is like an adjacency matrix, where $word_a$ and $word_b$ are connected if they appear in at least 1 document together (edge weight is # documents they appear in together).

NORMALIZED ADJACENCY MATRIX

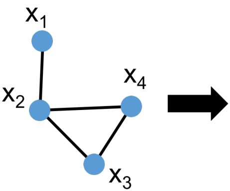


$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

What is the sum of entries in the i^{th} column of A?



What is the sum of entries in the i^{th} column of A? The (weighted) degree of vertex i.

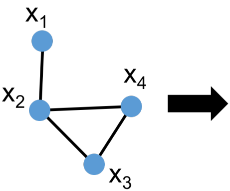


$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

What is the sum of entries in the i^{th} column of A? The (weighted) degree of vertex i.

Often, $\mathbf{A}$ is normalized as $\bar{\mathbf{A}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ where $\mathbf{D}$ is the degree matrix.

NORMALIZED ADJACENCY MATRIX



$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \qquad \mathbf{D} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

What is the sum of entries in the i^{th} column of A? The (weighted) degree of vertex i.

Often, $\mathbf{A}$ is normalized as $\bar{\mathbf{A}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ where $\mathbf{D}$ is the degree matrix.

$$\overset{\bar{\mathbf{A}}}{\begin{bmatrix} 0 & \frac{1}{\sqrt{3}} & 0 & 0 \\ \frac{1}{\sqrt{3}} & 0 & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ 0 & \frac{1}{\sqrt{6}} & 0 & \frac{1}{2} \\ 0 & \frac{1}{\sqrt{3}} & \frac{1}{2} & 0 \end{bmatrix}} = \overset{\mathbf{D}^{-1/2}}{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}} \overset{\mathbf{A}}{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}} \overset{\mathbf{D}^{-1/2}}{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}}$$

What is the sum of entries in the i^{th} column of A? The (weighted) degree of vertex i.

Often, $\mathbf{A}$ is normalized as $\bar{\mathbf{A}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ where $\mathbf{D}$ is the degree matrix.

$$\overset{\bar{\mathbf{A}}}{\begin{bmatrix} 0 & \frac{1}{\sqrt{3}} & 0 & 0 \\ \frac{1}{\sqrt{3}} & 0 & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ 0 & \frac{1}{\sqrt{6}} & 0 & \frac{1}{2} \\ 0 & \frac{1}{\sqrt{3}} & \frac{1}{2} & 0 \end{bmatrix}} = \overset{\mathbf{D}^{-1/2}}{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}} \overset{\mathbf{A}}{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}} \overset{\mathbf{D}^{-1/2}}{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}}$$

What is the sum of entries in the i^{th} column of A? The (weighted) degree of vertex i.

Often, $\mathbf{A}$ is normalized as $\bar{\mathbf{A}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ where $\mathbf{D}$ is the degree matrix.

Spectral graph theory is the field of representing graphs as matrices and applying linear algebraic techniques.

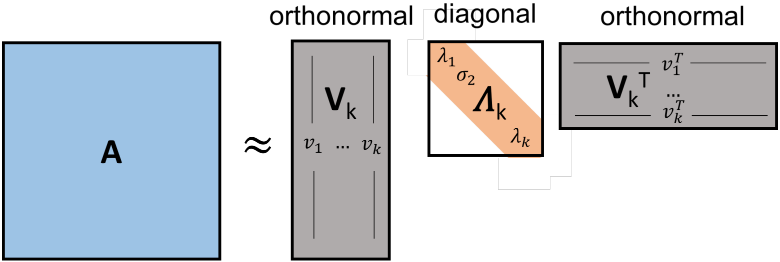
How do we compute an optimal low-rank approximation of $\mathbf{A}$?

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

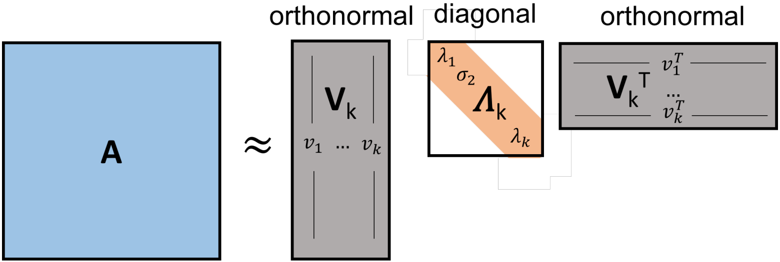
- Project onto the top k eigenvectors of $\mathbf{A}^T\mathbf{A} = \mathbf{A}^2$.

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

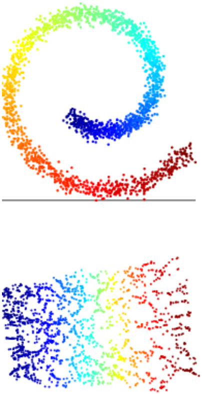
- Project onto the top k eigenvectors of $\mathbf{A}^T\mathbf{A} = \mathbf{A}^2$. These are just the eigenvectors of $\mathbf{A}$.



orthonormal
diagonal
orthonormal
A
≈
V_k
v_1 … v_k
λ_1
σ_2
Λ_k
λ_k
V_k^T
v_1^T
…
v_k^T



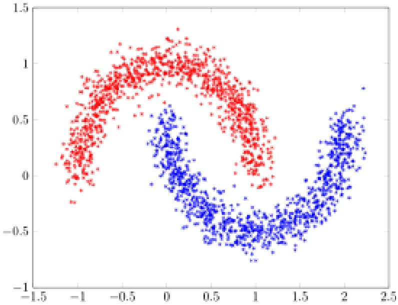
- Similar vertices (close with regards to graph proximity) should have similar embeddings. I.e., $\mathrm{V}_k(i)$ should be similar to $\mathrm{V}_k(j)$.



A very common task aside from just embedding points via graph based similarity and SVD, is to partition or cluster vertices based on this similarity.

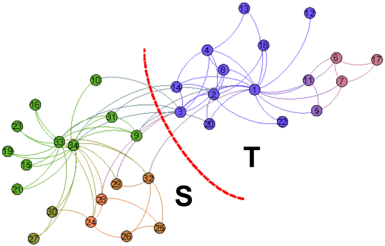
A very common task aside from just embedding points via graph based similarity and SVD, is to partition or cluster vertices based on this similarity.

Non-linearly separable data.



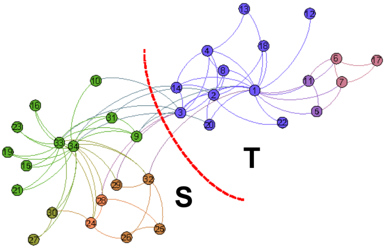
A very common task aside from just embedding points via graph based similarity and SVD, is to partition or cluster vertices based on this similarity.

Community detection in naturally occuring networks.



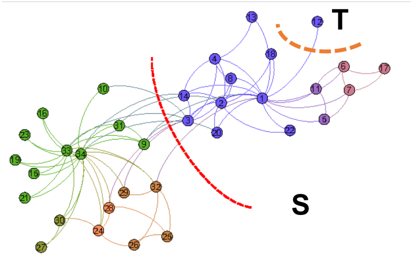
(a) Zachary Karate Club Graph

Simple Idea: Partition clusters along minimum cut in graph.



(a) Zachary Karate Club Graph

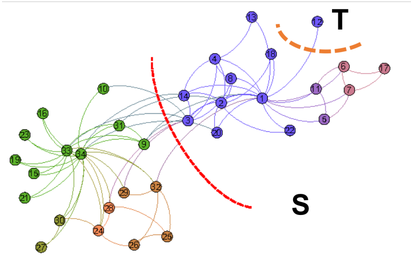
Simple Idea: Partition clusters along minimum cut in graph.



(a) Zachary Karate Club Graph

Small cuts are often not informative.

Simple Idea: Partition clusters along minimum cut in graph.

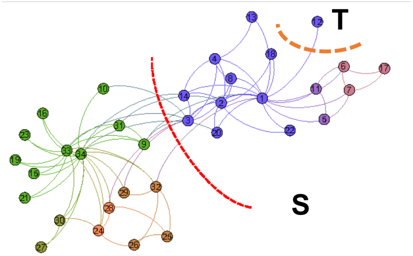


(a) Zachary Karate Club Graph

Small cuts are often not informative.

Solution: Encourage cuts that separate large sections of the graph.

Simple Idea: Partition clusters along minimum cut in graph.



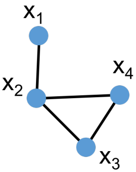
(a) Zachary Karate Club Graph

Small cuts are often not informative.

Solution: Encourage cuts that separate large sections of the graph.

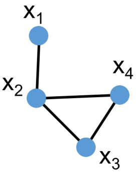
- Let $\vec{v} \in \mathbb{R}^n$ represent a cut: $\vec{v}(i) = 1$ if $i \in S$ and $\vec{v}(i) = -1$ if $i \in T$. Want $\vec{v}$ to have roughly equal numbers of 1s and -1s. I.e., $\vec{v}^T\vec{1} \approx 0$.

For a graph with adjacency matrix **A** and degree matrix **D**, $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is the graph Laplacian.



$$\underset{\mathbf{D}}{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}} - \underset{\mathbf{A}}{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}} = \underset{\mathbf{L}}{\begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 3 & -1 & -1 \\ 0 & -1 & 2 & -1 \\ 0 & -1 & -1 & 2 \end{bmatrix}}$$

For a graph with adjacency matrix $\mathbf{A}$ and degree matrix $\mathbf{D}$, $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is the graph Laplacian.



$$\overset{\mathbf{D}}{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}} - \overset{\mathbf{A}}{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}} = \overset{\mathbf{L}}{\begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 3 & -1 & -1 \\ 0 & -1 & 2 & -1 \\ 0 & -1 & -1 & 2 \end{bmatrix}}$$

For any vector $\vec{v}$,

$$\vec{v}^T L \vec{v} = \vec{v}^T D \vec{v} - \vec{v}^T A \vec{v} = \sum_{i=1}^{n} d(i)\vec{v}(i)^2 - \sum_{i=1}^{n}\sum_{j=1}^{n} A(i,j) \cdot v(i) \cdot v(j)$$

For a cut indicator vector $\vec{v} \in \{-1, 1\}^n$ with $\vec{v}(i) = -1$ for $i \in S$ and $\vec{v}(i) = 1$ for $i \in T$:

$$\vec{v}^T L \vec{V} = \sum_{(i,j) \in E} (\vec{v}(i) - \vec{v}(j))^2 = 4 \cdot cut(S, T).$$

So minimizing $\vec{v}^T L \vec{v}$ corresponds to minimizing the cut size.

For a cut indicator vector $\vec{v} \in \{-1, 1\}^n$ with $\vec{v}(i) = -1$ for $i \in S$ and $\vec{v}(i) = 1$ for $i \in T$:

$$\vec{v}^T L \vec{V} = \sum_{(i,j) \in E} (\vec{v}(i) - \vec{v}(j))^2 = 4 \cdot cut(S, T).$$

So minimizing $\vec{v}^T L \vec{v}$ corresponds to minimizing the cut size.

$$\arg\min_{v \in \{-1,1\}^n} \vec{v}^T L \vec{V}$$

For a cut indicator vector $\vec{v} \in \{-1, 1\}^n$ with $\vec{v}(i) = -1$ for $i \in S$ and $\vec{v}(i) = 1$ for $i \in T$:

$$\vec{v}^T L \vec{V} = \sum_{(i,j)\in E} (\vec{v}(i) - \vec{v}(j))^2 = 4 \cdot cut(S, T).$$

So minimizing $\vec{v}^T L \vec{v}$ corresponds to minimizing the cut size.

$$\underset{v \in \mathbb{R}^d \text{ with } \|\vec{v}\|=1}{\arg\min} \quad \vec{v}^T L \vec{V}$$

For a cut indicator vector $\vec{v} \in \{-1, 1\}^n$ with $\vec{v}(i) = -1$ for $i \in S$ and $\vec{v}(i) = 1$ for $i \in T$:

$$\vec{v}^T L \vec{V} = \sum_{(i,j) \in E} (\vec{v}(i) - \vec{v}(j))^2 = 4 \cdot cut(S, T).$$

So minimizing $\vec{v}^T L \vec{v}$ corresponds to minimizing the cut size.

$$\arg\min_{v \in \mathbb{R}^d \text{ with } \|\vec{v}\|=1} \vec{v}^T L \vec{V}$$

By the Courant-Fischer theorem, $\vec{v}$ is the smallest eigenvector of $L = D - A$.

We have:

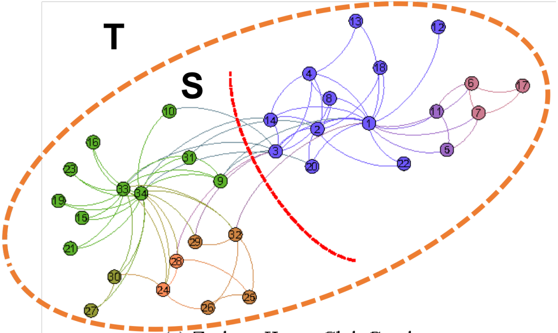
$$\vec{v}_n = \frac{1}{\sqrt{n}} \cdot \vec{1} = \underset{v \in \mathbb{R}^d \text{with } \|\vec{v}\|=1}{\arg\min} \quad \vec{v}^T L \vec{V}$$

with $\vec{v}_n^T L \vec{v}_n = 0$.

We have:

$$\vec{v}_n = \frac{1}{\sqrt{n}} \cdot \vec{1} = \underset{v \in \mathbb{R}^d \text{with } \|\vec{v}\|=1}{\arg\min} \quad \vec{v}^T L \vec{V}$$

with $\vec{v}_n^T L \vec{v}_n = 0$.



(a) Zachary Karate Club Graph

By Courant-Fischer, second small eigenvector is obtained greedily:

$$\vec{v}_1 = \underset{v \in \mathbb{R}^d \text{with } \|\vec{v}\|=1}{\arg\min} \vec{v}^T L \vec{V}$$

$$\vec{v}_2 = \underset{v \in \mathbb{R}^d \text{with } \|\vec{v}\|=1,\ \vec{v}_2^T \vec{v}_1 = 0}{\arg\min} \vec{v}^T L \vec{V}$$

By Courant-Fischer, second small eigenvector is obtained greedily:

$$\vec{v}_1 = \operatorname*{arg\,min}_{v \in \mathbb{R}^d \text{with } \|\vec{v}\|=1} \vec{v}^T L \vec{V}$$

$$\vec{v}_2 = \operatorname*{arg\,min}_{v \in \mathbb{R}^d \text{with } \|\vec{v}\|=1,\ \vec{v}_2^T \vec{v}_1 = 0} \vec{v}^T L \vec{V}$$

If $\vec{v}_2$ were binary $\{-1, 1\}^d$, orthogonality condition ensures that there are an equal number of vertices on each side of the cut.

By Courant-Fischer, second small eigenvector is obtained greedily:

$$\vec{v}_1 = \underset{v \in \mathbb{R}^d \text{with } \|\vec{v}\|=1}{\arg\min} \vec{v}^T L \vec{V}$$

$$\vec{v}_2 = \underset{v \in \mathbb{R}^d \text{with } \|\vec{v}\|=1,\ \vec{v}_2^T \vec{v}_1 = 0}{\arg\min} \vec{v}^T L \vec{V}$$

If $\vec{v}_2$ were binary $\{-1, 1\}^d$, orthogonality condition ensures that there are an equal number of vertices on each side of the cut. When $\vec{v}_2 \in \mathbb{R}^d$, enforces a 'relaxed' version of this constraint.

CUTTING WITH THE SECOND LAPLACIAN EIGENVECTOR

Find a good partition of the graph by computing

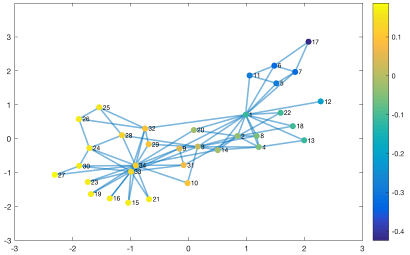
$$\vec{v}_2 = \operatorname*{arg\,min}_{v \in \mathbb{R}^d \text{with } \|\vec{v}\|=1,\ \vec{v}_2^T \vec{1}=0} \vec{v}^T L \vec{V}$$

Set S to be all nodes with $\vec{v}_2(i) < 0$, T to be all with $\vec{v}_2(i) \geq 0$.

Find a good partition of the graph by computing

$$\vec{v}_2 = \underset{v \in \mathbb{R}^d \text{with } \|\vec{v}\|=1, \ \vec{v}_2^T \vec{1}=0}{\arg\min} \vec{v}^T L \vec{v}$$

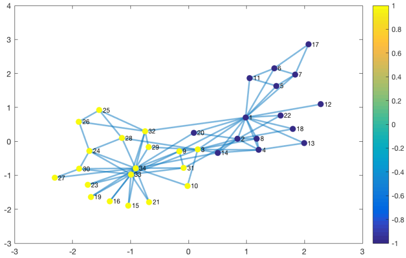
Set S to be all nodes with $\vec{v}_2(i) < 0$, T to be all with $\vec{v}_2(i) \geq 0$.



Find a good partition of the graph by computing

$$\vec{v}_2 = \underset{v \in \mathbb{R}^d \text{with } \|\vec{v}\|=1,\ \vec{v}_2^T \vec{1}=0}{\arg\min} \vec{v}^T L \vec{V}$$

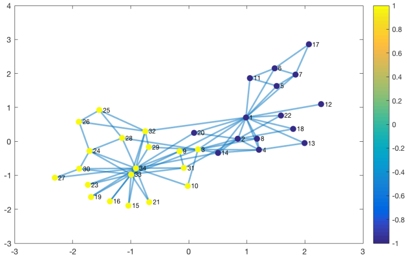
Set S to be all nodes with $\vec{v}_2(i) < 0$, T to be all with $\vec{v}_2(i) \geq 0$.



Find a good partition of the graph by computing

$$\vec{v}_2 = \underset{v \in \mathbb{R}^d \text{with } \|\vec{v}\|=1,\ \vec{v}_2^T \vec{1}=0}{\arg\min} \vec{v}^T L \vec{V}$$

Set S to be all nodes with $\vec{v}_2(i) < 0$, T to be all with $\vec{v}_2(i) \geq 0$.



The Shi-Malik normalized cuts algorithm is a commonly used variance on this approach, using the normalize Laplacian $D^{-1/2} L D^{-1/2}$.

The smallest eigenvectors of $\mathbf{L} = \mathbf{D} - \mathbf{A}$ give the orthogonal 'functions' that are smoothest over the graph. I.e., minimize

$$\vec{v}^T L \vec{v} = \sum_{(i,j) \in E} [\vec{v}(i) - \vec{v}(j)]^2.$$

The smallest eigenvectors of $\mathtt{L} = \mathtt{D} - \mathtt{A}$ give the orthogonal 'functions' that are smoothest over the graph. I.e., minimize

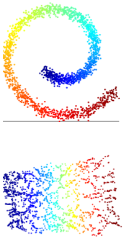
$$\vec{v}^T L \vec{v} = \sum_{(i,j) \in E} [\vec{v}(i) - \vec{v}(j)]^2.$$

Embedding points with coordinates given by $[\vec{v}_{n-1}(j), \vec{v}_{n-2}(j), \ldots, \vec{v}_{n-k}(j)]$ ensures that coordinates connected by edges have minimum Euclidean distance.

The smallest eigenvectors of $\mathbf{L} = \mathbf{D} - \mathbf{A}$ give the orthogonal 'functions' that are smoothest over the graph. I.e., minimize

$$\vec{v}^T L \vec{v} = \sum_{(i,j) \in E} [\vec{v}(i) - \vec{v}(j)]^2.$$

Embedding points with coordinates given by $[\vec{v}_{n-1}(j), \vec{v}_{n-2}(j), \ldots, \vec{v}_{n-k}(j)]$ ensures that coordinates connected by edges have minimum Euclidean distance.

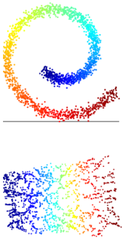


LAPLACIAN EMBEDDING

The smallest eigenvectors of $\mathbf{L} = \mathbf{D} - \mathbf{A}$ give the orthogonal 'functions' that are smoothest over the graph. I.e., minimize

$$\vec{v}^T L \vec{v} = \sum_{(i,j) \in E} [\vec{v}(i) - \vec{v}(j)]^2.$$

Embedding points with coordinates given by $[\vec{v}_{n-1}(j), \vec{v}_{n-2}(j), \ldots, \vec{v}_{n-k}(j)]$ ensures that coordinates connected by edges have minimum Euclidean distance.



- Laplacian Eigenmaps
- Locally linear embedding
- Isomap
- Etc...

Questions?