COMPSCI 514: ALGORITHMS FOR DATA SCIENCE

Andrew McGregor

Lecture 16

Last Class: Low-Rank Approximation, Eigendecomposition, PCA

- For any symmetric square matrix A, we can write $\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$ where columns of $\mathbf{V}$ are orthonormal eigenvectors.
- Can approximate data lying close to in a k-dimensional subspace by projecting data points into that space.
- Can find the best k-dimensional subspace via eigendecomposition applied to $\mathbf{X}^T\mathbf{X}$ (PCA).
- Measuring error in terms of the eigenvalue spectrum.

This Class: SVD and Applications

- SVD and connection to eigenvalue value decomposition.
- Applications of low-rank approximation beyond compression.

SINGULAR VALUE DECOMPOSITION

The Singular Value Decomposition (SVD) generalizes the eigendecomposition to asymmetric (even rectangular) matrices.

SINGULAR VALUE DECOMPOSITION

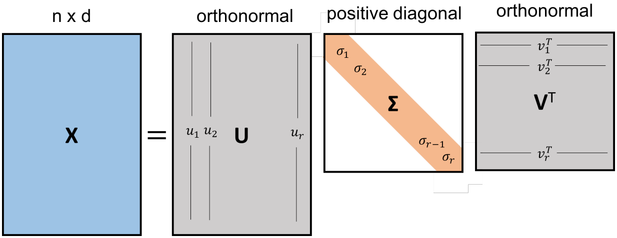
The Singular Value Decomposition (SVD) generalizes the eigendecomposition to asymmetric (even rectangular) matrices. Any matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ with $\text{rank}(\mathbf{X}) = r$ can be written as $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$.

- $\mathbf{U}$ has orthonormal columns $\vec{u}_1, \ldots, \vec{u}_r \in \mathbb{R}^n$ (left singular vectors).
- $\mathbf{V}$ has orthonormal columns $\vec{v}_1, \ldots, \vec{v}_r \in \mathbb{R}^d$ (right singular vectors).
- $\mathbf{\Sigma}$ is diagonal with elements $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$ (singular values).

SINGULAR VALUE DECOMPOSITION

The Singular Value Decomposition (SVD) generalizes the eigendecomposition to asymmetric (even rectangular) matrices. Any matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ with $\text{rank}(\mathbf{X}) = r$ can be written as $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$.

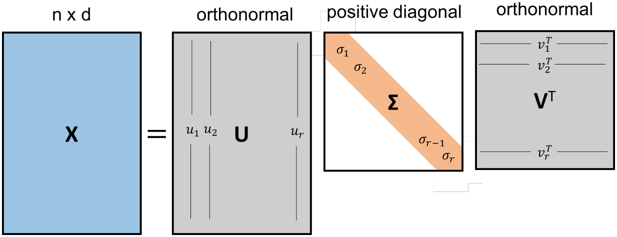
- $\mathbf{U}$ has orthonormal columns $\vec{u}_1, \ldots, \vec{u}_r \in \mathbb{R}^n$ (left singular vectors).
- $\mathbf{V}$ has orthonormal columns $\vec{v}_1, \ldots, \vec{v}_r \in \mathbb{R}^d$ (right singular vectors).
- $\mathbf{\Sigma}$ is diagonal with elements $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$ (singular values).



SINGULAR VALUE DECOMPOSITION

The Singular Value Decomposition (SVD) generalizes the eigendecomposition to asymmetric (even rectangular) matrices. Any matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ with $\text{rank}(\mathbf{X}) = r$ can be written as $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$.

- $\mathbf{U}$ has orthonormal columns $\vec{u}_1, \ldots, \vec{u}_r \in \mathbb{R}^n$ (left singular vectors).
- $\mathbf{V}$ has orthonormal columns $\vec{v}_1, \ldots, \vec{v}_r \in \mathbb{R}^d$ (right singular vectors).
- $\mathbf{\Sigma}$ is diagonal with elements $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$ (singular values).



The 'swiss army knife' of modern linear algebra.

Writing $\mathbf{X} \in \mathbb{R}^{n \times d}$ in its singular value decomposition $\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$:

$$\mathbf{X}^T\mathbf{X} =$$

$\mathbf{X} \in \mathbb{R}^{n \times d}$: data matrix, $\mathbf{U} \in \mathbb{R}^{n \times \text{rank}(\mathbf{X})}$: matrix with orthonormal columns $\vec{u}_1, \vec{u}_2, \ldots$ (left singular vectors), $\mathbf{V} \in \mathbb{R}^{d \times \text{rank}(\mathbf{X})}$: matrix with orthonormal columns $\vec{v}_1, \vec{v}_2, \ldots$ (right singular vectors), $\boldsymbol{\Sigma} \in \mathbb{R}^{\text{rank}(\mathbf{X}) \times \text{rank}(\mathbf{X})}$: positive diagonal matrix containing singular values of $\mathbf{X}$.

Writing $\mathbf{X} \in \mathbb{R}^{n \times d}$ in its singular value decomposition $\mathbf{X} = \mathbf{U\Sigma V}^T$:

$$\mathbf{X}^T\mathbf{X} = \mathbf{V\Sigma U}^T\mathbf{U\Sigma V}^T$$

$\mathbf{X} \in \mathbb{R}^{n \times d}$: data matrix, $\mathbf{U} \in \mathbb{R}^{n \times \text{rank}(\mathbf{X})}$: matrix with orthonormal columns $\vec{u}_1, \vec{u}_2, \ldots$ (left singular vectors), $\mathbf{V} \in \mathbb{R}^{d \times \text{rank}(\mathbf{X})}$: matrix with orthonormal columns $\vec{v}_1, \vec{v}_2, \ldots$ (right singular vectors), $\mathbf{\Sigma} \in \mathbb{R}^{\text{rank}(\mathbf{X}) \times \text{rank}(\mathbf{X})}$: positive diagonal matrix containing singular values of $\mathbf{X}$.

Writing $\mathbf{X} \in \mathbb{R}^{n \times d}$ in its singular value decomposition $\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$:

$$\mathbf{X}^T\mathbf{X} = \mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^T\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T = \mathbf{V}\boldsymbol{\Sigma}^2\mathbf{V}^T$$

$\mathbf{X} \in \mathbb{R}^{n \times d}$: data matrix, $\mathbf{U} \in \mathbb{R}^{n \times \text{rank}(\mathbf{X})}$: matrix with orthonormal columns $\vec{u}_1, \vec{u}_2, \ldots$ (left singular vectors), $\mathbf{V} \in \mathbb{R}^{d \times \text{rank}(\mathbf{X})}$: matrix with orthonormal columns $\vec{v}_1, \vec{v}_2, \ldots$ (right singular vectors), $\boldsymbol{\Sigma} \in \mathbb{R}^{\text{rank}(\mathbf{X}) \times \text{rank}(\mathbf{X})}$: positive diagonal matrix containing singular values of $\mathbf{X}$.

CONNECTION OF THE SVD TO EIGENDECOMPOSITION

Writing $\mathbf{X} \in \mathbb{R}^{n \times d}$ in its singular value decomposition $\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$:

$$\mathbf{X}^T\mathbf{X} = \mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^T\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T = \mathbf{V}\boldsymbol{\Sigma}^2\mathbf{V}^T \text{ (the eigendecomposition)}$$

$\mathbf{X} \in \mathbb{R}^{n \times d}$: data matrix, $\mathbf{U} \in \mathbb{R}^{n \times \text{rank}(\mathbf{X})}$: matrix with orthonormal columns $\vec{u}_1, \vec{u}_2, \ldots$ (left singular vectors), $\mathbf{V} \in \mathbb{R}^{d \times \text{rank}(\mathbf{X})}$: matrix with orthonormal columns $\vec{v}_1, \vec{v}_2, \ldots$ (right singular vectors), $\boldsymbol{\Sigma} \in \mathbb{R}^{\text{rank}(\mathbf{X}) \times \text{rank}(\mathbf{X})}$: positive diagonal matrix containing singular values of $\mathbf{X}$.

CONNECTION OF THE SVD TO EIGENDECOMPOSITION

Writing $\mathbf{X} \in \mathbb{R}^{n \times d}$ in its singular value decomposition $\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$:

$$\mathbf{X}^T\mathbf{X} = \mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^T\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T = \mathbf{V}\boldsymbol{\Sigma}^2\mathbf{V}^T \text{ (the eigendecomposition)}$$

Similarly: $\mathbf{X}\mathbf{X}^T = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T\mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^T = \mathbf{U}\boldsymbol{\Sigma}^2\mathbf{U}^T$.

$\mathbf{X} \in \mathbb{R}^{n \times d}$: data matrix, $\mathbf{U} \in \mathbb{R}^{n \times \text{rank}(\mathbf{X})}$: matrix with orthonormal columns $\vec{u}_1, \vec{u}_2, \ldots$ (left singular vectors), $\mathbf{V} \in \mathbb{R}^{d \times \text{rank}(\mathbf{X})}$: matrix with orthonormal columns $\vec{v}_1, \vec{v}_2, \ldots$ (right singular vectors), $\boldsymbol{\Sigma} \in \mathbb{R}^{\text{rank}(\mathbf{X}) \times \text{rank}(\mathbf{X})}$: positive diagonal matrix containing singular values of $\mathbf{X}$.

CONNECTION OF THE SVD TO EIGENDECOMPOSITION

Writing $\mathbf{X} \in \mathbb{R}^{n \times d}$ in its singular value decomposition $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$:

$$\mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{\Sigma}\mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T \text{ (the eigendecomposition)}$$

Similarly: $\mathbf{X}\mathbf{X}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{V}\mathbf{\Sigma}\mathbf{U}^T = \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^T$.

The right and left singular vectors are the eigenvectors of the covariance matrix $\mathbf{X}^T\mathbf{X}$ and the gram matrix $\mathbf{X}\mathbf{X}^T$ respectively.

$\mathbf{X} \in \mathbb{R}^{n \times d}$: data matrix, $\mathbf{U} \in \mathbb{R}^{n \times \text{rank}(\mathbf{X})}$: matrix with orthonormal columns $\vec{u}_1, \vec{u}_2, \ldots$ (left singular vectors), $\mathbf{V} \in \mathbb{R}^{d \times \text{rank}(\mathbf{X})}$: matrix with orthonormal columns $\vec{v}_1, \vec{v}_2, \ldots$ (right singular vectors), $\mathbf{\Sigma} \in \mathbb{R}^{\text{rank}(\mathbf{X}) \times \text{rank}(\mathbf{X})}$: positive diagonal matrix containing singular values of $\mathbf{X}$.

CONNECTION OF THE SVD TO EIGENDECOMPOSITION

Writing $\mathbf{X} \in \mathbb{R}^{n \times d}$ in its singular value decomposition $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$:

$$\mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{\Sigma}\mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T \text{ (the eigendecomposition)}$$

Similarly: $\mathbf{X}\mathbf{X}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{V}\mathbf{\Sigma}\mathbf{U}^T = \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^T$.

The right and left singular vectors are the eigenvectors of the covariance matrix $\mathbf{X}^T\mathbf{X}$ and the gram matrix $\mathbf{X}\mathbf{X}^T$ respectively.

So, letting $\mathbf{V}_k \in \mathbb{R}^{d \times k}$ have columns equal to $\vec{v}_1, \ldots, \vec{v}_k$, we know that $\mathbf{X}\mathbf{V}_k\mathbf{V}_k^T$ is the best rank-k approximation to $\mathbf{X}$ (given by PCA).

$\mathbf{X} \in \mathbb{R}^{n \times d}$: data matrix, $\mathbf{U} \in \mathbb{R}^{n \times \text{rank}(\mathbf{X})}$: matrix with orthonormal columns $\vec{u}_1, \vec{u}_2, \ldots$ (left singular vectors), $\mathbf{V} \in \mathbb{R}^{d \times \text{rank}(\mathbf{X})}$: matrix with orthonormal columns $\vec{v}_1, \vec{v}_2, \ldots$ (right singular vectors), $\mathbf{\Sigma} \in \mathbb{R}^{\text{rank}(\mathbf{X}) \times \text{rank}(\mathbf{X})}$: positive diagonal matrix containing singular values of $\mathbf{X}$.

CONNECTION OF THE SVD TO EIGENDECOMPOSITION

Writing $\mathbf{X} \in \mathbb{R}^{n \times d}$ in its singular value decomposition $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$:

$$\mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{\Sigma}\mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T \text{ (the eigendecomposition)}$$

Similarly: $\mathbf{X}\mathbf{X}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{V}\mathbf{\Sigma}\mathbf{U}^T = \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^T$.

The right and left singular vectors are the eigenvectors of the covariance matrix $\mathbf{X}^T\mathbf{X}$ and the gram matrix $\mathbf{X}\mathbf{X}^T$ respectively.

So, letting $\mathbf{V}_k \in \mathbb{R}^{d \times k}$ have columns equal to $\vec{v}_1, \ldots, \vec{v}_k$, we know that $\mathbf{X}\mathbf{V}_k\mathbf{V}_k^T$ is the best rank-k approximation to $\mathbf{X}$ (given by PCA).

What about $\mathbf{U}_k\mathbf{U}_k^T\mathbf{X}$ where $\mathbf{U}_k \in \mathbb{R}^{n \times k}$ has columns equal to $\vec{u}_1, \ldots, \vec{u}_k$?

$\mathbf{X} \in \mathbb{R}^{n \times d}$: data matrix, $\mathbf{U} \in \mathbb{R}^{n \times \text{rank}(\mathbf{X})}$: matrix with orthonormal columns $\vec{u}_1, \vec{u}_2, \ldots$ (left singular vectors), $\mathbf{V} \in \mathbb{R}^{d \times \text{rank}(\mathbf{X})}$: matrix with orthonormal columns $\vec{v}_1, \vec{v}_2, \ldots$ (right singular vectors), $\mathbf{\Sigma} \in \mathbb{R}^{\text{rank}(\mathbf{X}) \times \text{rank}(\mathbf{X})}$: positive diagonal matrix containing singular values of $\mathbf{X}$.

Writing $\mathbf{X} \in \mathbb{R}^{n \times d}$ in its singular value decomposition $\mathbf{X} = \mathbf{U \Sigma V}^T$:

$$\mathbf{X}^T\mathbf{X} = \mathbf{V \Sigma U}^T \mathbf{U \Sigma V}^T = \mathbf{V \Sigma}^2 \mathbf{V}^T \text{ (the eigendecomposition)}$$

Similarly: $\mathbf{X}\mathbf{X}^T = \mathbf{U \Sigma V}^T \mathbf{V \Sigma U}^T = \mathbf{U \Sigma}^2 \mathbf{U}^T$.

The right and left singular vectors are the eigenvectors of the covariance matrix $\mathbf{X}^T\mathbf{X}$ and the gram matrix $\mathbf{X}\mathbf{X}^T$ respectively.

So, letting $\mathbf{V}_k \in \mathbb{R}^{d \times k}$ have columns equal to $\vec{v}_1, \ldots, \vec{v}_k$, we know that $\mathbf{X}\mathbf{V}_k\mathbf{V}_k^T$ is the best rank-k approximation to $\mathbf{X}$ (given by PCA).

What about $\mathbf{U}_k\mathbf{U}_k^T\mathbf{X}$ where $\mathbf{U}_k \in \mathbb{R}^{n \times k}$ has columns equal to $\vec{u}_1, \ldots, \vec{u}_k$?

Exercise: $\mathbf{U}_k\mathbf{U}_k^T\mathbf{X} = \mathbf{X}\mathbf{V}_k\mathbf{V}_k^T = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^T$

> $\mathbf{X} \in \mathbb{R}^{n \times d}$: data matrix, $\mathbf{U} \in \mathbb{R}^{n \times \text{rank}(\mathbf{X})}$: matrix with orthonormal columns $\vec{u}_1, \vec{u}_2, \ldots$ (left singular vectors), $\mathbf{V} \in \mathbb{R}^{d \times \text{rank}(\mathbf{X})}$: matrix with orthonormal columns $\vec{v}_1, \vec{v}_2, \ldots$ (right singular vectors), $\mathbf{\Sigma} \in \mathbb{R}^{\text{rank}(\mathbf{X}) \times \text{rank}(\mathbf{X})}$: positive diagonal matrix containing singular values of $\mathbf{X}$.

The best low-rank approximation to $\mathbf{X}$, i.e.,

$$\mathbf{X}_k = \underset{\text{rank}-k\ \mathbf{B} \in \mathbb{R}^{n \times d}}{\arg\min} \|\mathbf{X} - \mathbf{B}\|_F$$

is given by $\mathbf{X}_k = \mathbf{X}\mathbf{V}_k\mathbf{V}_k^T = \mathbf{U}_k\mathbf{U}_k^T\mathbf{X} = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^T$

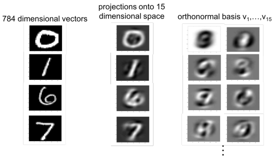
The best low-rank approximation to $\mathbf{X}$, i.e.,

$$\mathbf{X}_k = \underset{\text{rank}-k\ \mathbf{B} \in \mathbb{R}^{n \times d}}{\arg\min} \|\mathbf{X} - \mathbf{B}\|_F$$

is given by $\mathbf{X}_k = \mathbf{X}\mathbf{V}_k\mathbf{V}_k^T = \mathbf{U}_k\mathbf{U}_k^T\mathbf{X} = \mathbf{U}_k\boldsymbol{\Sigma}_k\mathbf{V}_k^T$

Corresponds to projecting the rows (data points) onto the span of $\mathbf{V}_k$ or the columns (features) onto the span of $\mathbf{U}_k$

Row (data point) compression



Column (feature) compression

10000* bathrooms+ 10* (sq. ft.) ≈ list price

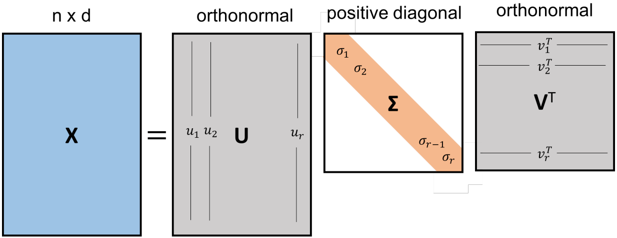
	bedrooms	bathrooms	sq.ft.	floors	list price	sale price
home 1	2	2	1800	2	200,000	195,000
home 2	4	2.5	2700	1	300,000	310,000
⋮	⋮	⋮	⋮	⋮	⋮	⋮
home n	5	3.5	3600	3	450,000	450,000

The best low-rank approximation to $\mathbf{X}$, i.e.,

$$\mathbf{X}_k = \underset{\text{rank}-k\ \mathbf{B} \in \mathbb{R}^{n \times d}}{\arg\min} \|\mathbf{X} - \mathbf{B}\|_F$$

is given by $\mathbf{X}_k = \mathbf{X}\mathbf{V}_k\mathbf{V}_k^T = \mathbf{U}_k\mathbf{U}_k^T\mathbf{X} = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^T$

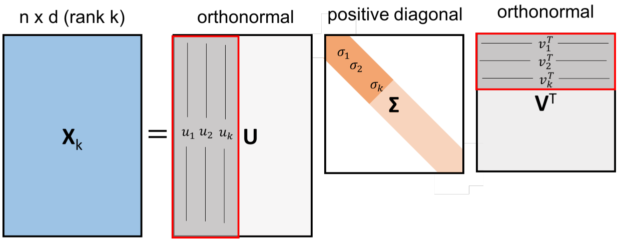
Corresponds to projecting the rows (data points) onto the span of $\mathbf{V}_k$ or the columns (features) onto the span of $\mathbf{U}_k$



The best low-rank approximation to $\mathbf{X}$, i.e.,

$$\mathbf{X}_k = \underset{\text{rank}-k\ \mathbf{B}\in\mathbb{R}^{n\times d}}{\arg\min} \|\mathbf{X}-\mathbf{B}\|_F$$

is given by $\mathbf{X}_k = \mathbf{X}\mathbf{V}_k\mathbf{V}_k^T = \mathbf{U}_k\mathbf{U}_k^T\mathbf{X} = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^T$
Corresponds to projecting the rows (data points) onto the span of $\mathbf{V}_k$ or the columns (features) onto the span of $\mathbf{U}_k$

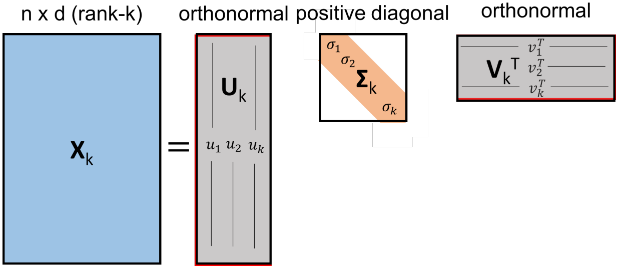


The best low-rank approximation to $\mathbf{X}$, i.e.,

$$\mathbf{X}_k = \underset{\text{rank}-k\ \mathbf{B}\in\mathbb{R}^{n\times d}}{\arg\min} \|\mathbf{X}-\mathbf{B}\|_F$$

is given by $\mathbf{X}_k = \mathbf{X}\mathbf{V}_k\mathbf{V}_k^T = \mathbf{U}_k\mathbf{U}_k^T\mathbf{X} = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^T$

Corresponds to projecting the rows (data points) onto the span of $\mathbf{V}_k$ or the columns (features) onto the span of $\mathbf{U}_k$



BASIC IDEA TO PROVE EXISTENCE OF SVD

- Let $\vec{v}_1, \vec{v}_2, \ldots, \in \mathbb{R}^d$ be orthonormal eigenvectors of $\mathbf{X}^T\mathbf{X}$.

BASIC IDEA TO PROVE EXISTENCE OF SVD

- Let $\vec{v}_1, \vec{v}_2, \ldots, \in \mathbb{R}^d$ be orthonormal eigenvectors of $\mathbf{X}^T\mathbf{X}$.
- Let $\sigma_i = \|\mathbf{X}\vec{v}_i\|_2$ and define unit vector $\vec{u}_i = \frac{\mathbf{X}\vec{v}_i}{\sigma_i}$.

BASIC IDEA TO PROVE EXISTENCE OF SVD

- Let $\vec{v}_1, \vec{v}_2, \ldots, \in \mathbb{R}^d$ be orthonormal eigenvectors of $\mathbf{X}^T\mathbf{X}$.
- Let $\sigma_i = \|\mathbf{X}\vec{v}_i\|_2$ and define unit vector $\vec{u}_i = \frac{\mathbf{X}\vec{v}_i}{\sigma_i}$.
- Exercise: Show $\vec{u}_1, \vec{u}_2, \ldots$ are orthonormal.

- Let $\vec{v}_1, \vec{v}_2, \ldots, \in \mathbb{R}^d$ be orthonormal eigenvectors of $\mathbf{X}^T\mathbf{X}$.
- Let $\sigma_i = \|\mathbf{X}\vec{v}_i\|_2$ and define unit vector $\vec{u}_i = \frac{\mathbf{X}\vec{v}_i}{\sigma_i}$.
- Exercise: Show $\vec{u}_1, \vec{u}_2, \ldots$ are orthonormal.
- This establishes that $\mathbf{XV} = \mathbf{U\Sigma}$ and that $\mathbf{V}$ and $\mathbf{U}$ have the required properties.

BASIC IDEA TO PROVE EXISTENCE OF SVD

- Let $\vec{v}_1, \vec{v}_2, \ldots, \in \mathbb{R}^d$ be orthonormal eigenvectors of $\mathbf{X}^T\mathbf{X}$.
- Let $\sigma_i = \|\mathbf{X}\vec{v}_i\|_2$ and define unit vector $\vec{u}_i = \frac{\mathbf{X}\vec{v}_i}{\sigma_i}$.
- Exercise: Show $\vec{u}_1, \vec{u}_2, \ldots$ are orthonormal.
- This establishes that $\mathbf{XV} = \mathbf{U\Sigma}$ and that $\mathbf{V}$ and $\mathbf{U}$ have the required properties.
- To see rest of the details, see `https://math.mit.edu/classes/18.095/2016IAP/lec2/SVD_Notes.pdf`

APPLICATIONS OF LOW-RANK APPROXIMATION

Rest of Class: Examples of how low-rank approximation is applied in a variety of data science applications.

Rest of Class: Examples of how low-rank approximation is applied in a variety of data science applications.

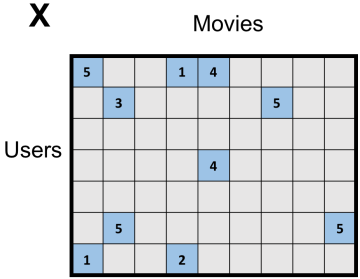
- Used for many reasons other than dimensionality reduction/data compression.

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).

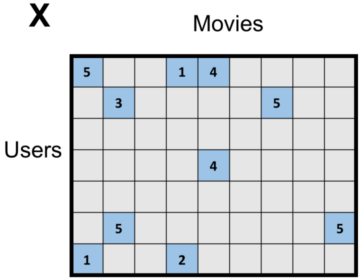
MATRIX COMPLETION

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.



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

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

Movies

Users

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

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

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

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

ENTITY EMBEDDINGS

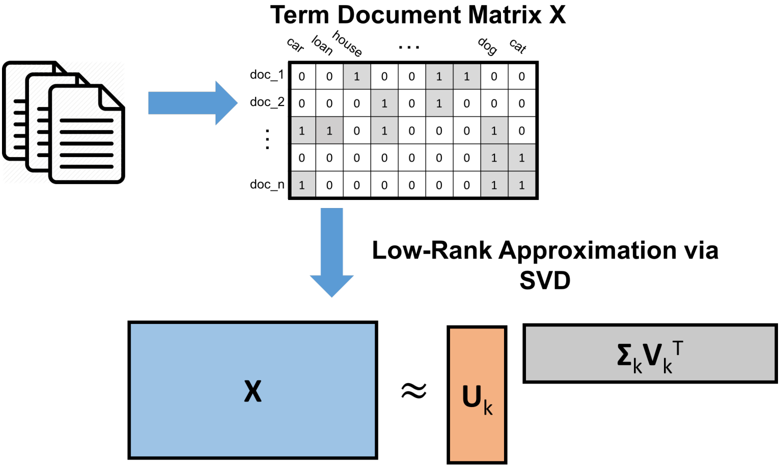
Dimensionality reduction embeds d-dimensional vectors into $k \ll d$ dimensions. But what about when you want to embed objects other than vectors?

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

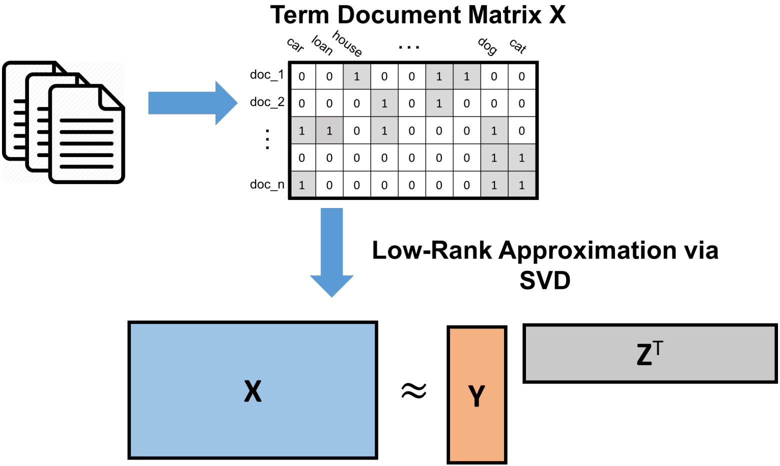
Dimensionality reduction embeds d-dimensional vectors into $k \ll d$ dimensions. But what about when you want to embed objects other than vectors?

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

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

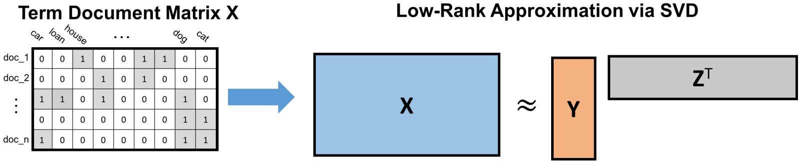


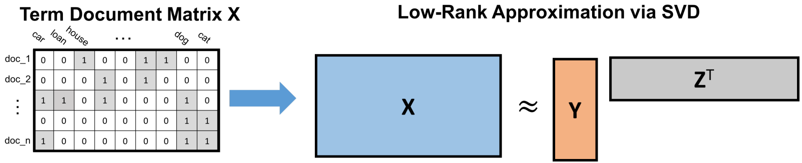
	car	loan	house	...				dog	cat
doc_1	0	0	1	0	0	1	1	0	0
doc_2	0	0	0	1	0	1	0	0	0
⋮	1	1	0	1	0	0	0	1	0
	0	0	0	0	0	0	0	1	1
doc_n	1	0	0	0	0	0	0	1	1



	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

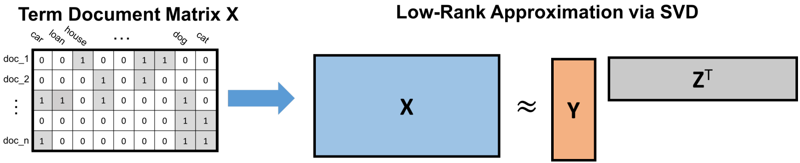
EXAMPLE: LATENT SEMANTIC ANALYSIS





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

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



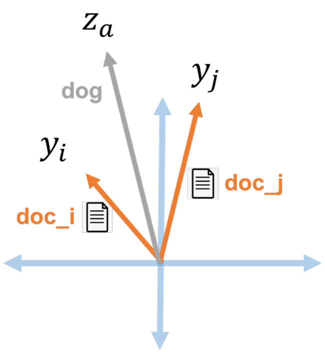
- If the error $\|\mathbf{X} - \mathbf{Y}\mathbf{Z}^T\|_F$ is small, then on average,

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

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

EXAMPLE: LATENT SEMANTIC ANALYSIS

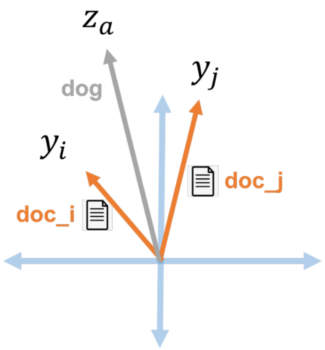
If doc_i and doc_j both contain $word_a$, $\langle \vec{y}_i, \vec{z}_a \rangle \approx \langle \vec{y}_j, \vec{z}_a \rangle \approx 1$ If doc_i and doc_j both don't contain $word_a$, $\langle \vec{y}_i, \vec{z}_a \rangle \approx \langle \vec{y}_j, \vec{z}_a \rangle \approx 0$



Since this applies for all words, documents with that involve a similar set of words tend to have high dot product with each other.

EXAMPLE: LATENT SEMANTIC ANALYSIS

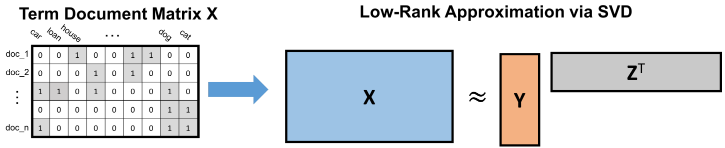
If doc_i and doc_j both contain $word_a$, $\langle \vec{y}_i, \vec{z}_a \rangle \approx \langle \vec{y}_j, \vec{z}_a \rangle \approx 1$ If doc_i and doc_j both don't contain $word_a$, $\langle \vec{y}_i, \vec{z}_a \rangle \approx \langle \vec{y}_j, \vec{z}_a \rangle \approx 0$



Since this applies for all words, documents with that involve a similar set of words tend to have high dot product with each other.

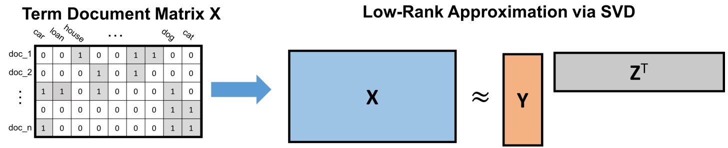
Another View: Column of **Y** represent 'topics'. $\vec{y}_i(j)$ indicates how much doc_i belongs to topic j. $\vec{z}_a(j)$ indicates how much $word_a$ associates with that topic.

EXAMPLE: LATENT SEMANTIC ANALYSIS



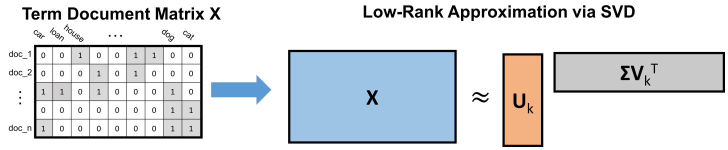
- Just like with documents, $\vec{z}_a$ and $\vec{z}_b$ will tend to have high dot product if $word_a$ and $word_b$ appear in many of the same documents.

EXAMPLE: LATENT SEMANTIC ANALYSIS



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

EXAMPLE: LATENT SEMANTIC ANALYSIS



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

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 $\mathbf{X}^T\mathbf{X}$: 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 $word_b$.

EXAMPLE: WORD EMBEDDING

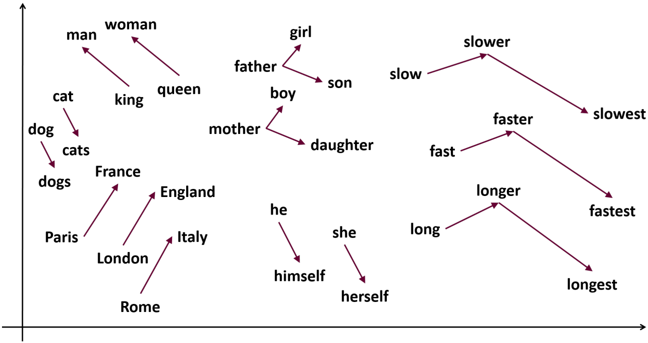
LSA gives a way of embedding words into k-dimensional space.

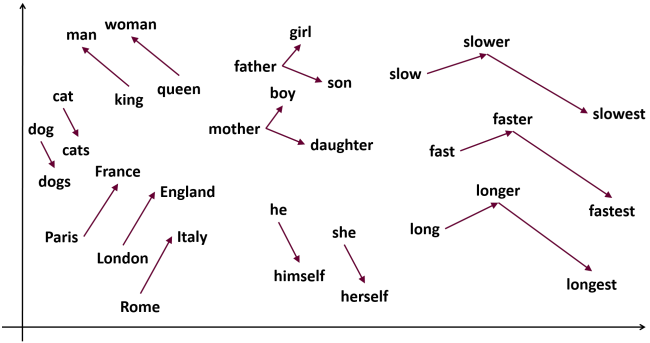
- Embedding is via low-rank approximation of $\mathbf{X}^T\mathbf{X}$: where $(\mathbf{X}^T\mathbf{X})_{a,b}$ is the number of documents that both $word_a$ and $word_b$ appear in.
- Think about $\mathbf{X}^T\mathbf{X}$ as a similarity matrix (gram matrix, kernel matrix) with entry (a, b) being the similarity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

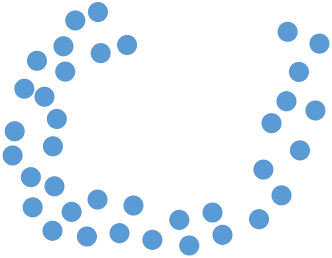
- Embedding is via low-rank approximation of $\mathbf{X}^T\mathbf{X}$: where $(\mathbf{X}^T\mathbf{X})_{a,b}$ is the number of documents that both $word_a$ and $word_b$ appear in.
- Think about $\mathbf{X}^T\mathbf{X}$ as a similarity matrix (gram matrix, kernel matrix) with entry (a, b) being the similarity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


man
woman
king
queen
cat
cats
dog
dogs
France
Paris
England
London
Italy
Rome
father
girl
son
mother
boy
daughter
he
himself
she
herself
slow
slower
slowest
fast
faster
fastest
long
longer
longest

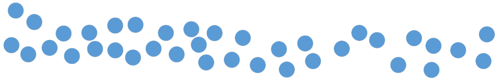


Note: word2vec is typically described as a neural-network method, but it is really just low-rank approximation of a specific similarity matrix. *Neural word embedding as implicit matrix factorization*, Levy and Goldberg.

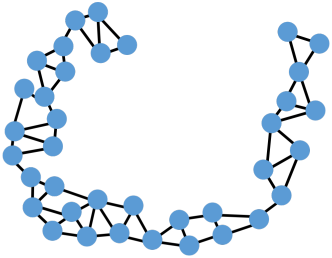
GRAPH EMBEDDINGS



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



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



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-nearest neighbor graph.

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

LINEAR ALGEBRAIC REPRESENTATION OF A GRAPH

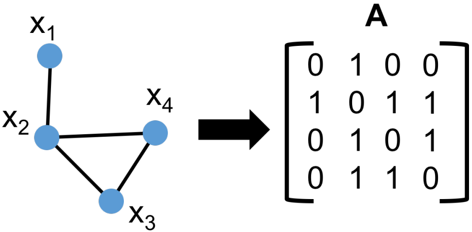
Once we have connected n data points $x_1, \ldots, x_n$ into a graph, we can represent that graph by its (weighted) adjacency matrix.

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

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.

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



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

ADJACENCY MATRIX EIGENVECTORS

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

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

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

- Project onto the top k eigenvectors of $\mathbf{A}^T\mathbf{A} = \mathbf{A}^2$. (Note these are just the eigenvectors of $\mathbf{A}$).
    1. $\mathbf{A} \approx \mathbf{A}\mathbf{V}_k\mathbf{V}_k^T$ where $\mathbf{V}_k$ is the matrix with the top k eigenvectors as columns.

ADJACENCY MATRIX EIGENVECTORS

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

- Project onto the top k eigenvectors of $\mathbf{A}^T\mathbf{A} = \mathbf{A}^2$. (Note these are just the eigenvectors of $\mathbf{A}$).
  1. $\mathbf{A} \approx \mathbf{A}\mathbf{V}_k\mathbf{V}_k^T$ where $\mathbf{V}_k$ is the matrix with the top k eigenvectors as columns.
  2. Rows of $\mathbf{A}\mathbf{V}_k$ are an embedding of the nodes into $\mathbb{R}^k$.

ADJACENCY MATRIX EIGENVECTORS

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

- Project onto the top k eigenvectors of $\mathbf{A}^T\mathbf{A} = \mathbf{A}^2$. (Note these are just the eigenvectors of $\mathbf{A}$).
  1. $\mathbf{A} \approx \mathbf{A}\mathbf{V}_k\mathbf{V}_k^T$ where $\mathbf{V}_k$ is the matrix with the top k eigenvectors as columns.
  2. Rows of $\mathbf{A}\mathbf{V}_k$ are an embedding of the nodes into $\mathbb{R}^k$.
- Similar vertices (close with regards to graph proximity) should have similar embeddings since

$$\|(\mathbf{A})_i - (\mathbf{A})_j\|_2 \approx \|(\mathbf{A}\mathbf{V}_k\mathbf{V}_k^T)_i - (\mathbf{A}\mathbf{V}_k\mathbf{V}_k^T)_j\|_2 = \|(\mathbf{A}\mathbf{V}_k)_i - (\mathbf{A}\mathbf{V}_k)_j\|_2$$

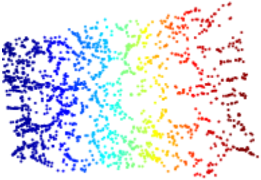
ADJACENCY MATRIX EIGENVECTORS

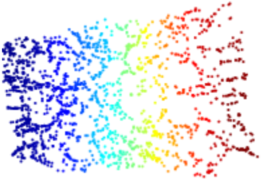
How do we compute an optimal low-rank approximation of $\mathbf{A}$?

- Project onto the top k eigenvectors of $\mathbf{A}^T\mathbf{A} = \mathbf{A}^2$. (Note these are just the eigenvectors of $\mathbf{A}$).
  1. $\mathbf{A} \approx \mathbf{A}\mathbf{V}_k\mathbf{V}_k^T$ where $\mathbf{V}_k$ is the matrix with the top k eigenvectors as columns.
  2. Rows of $\mathbf{A}\mathbf{V}_k$ are an embedding of the nodes into $\mathbb{R}^k$.
- Similar vertices (close with regards to graph proximity) should have similar embeddings since

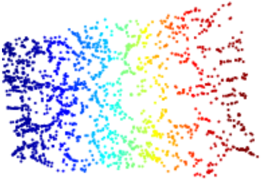
$$\|(\mathbf{A})_i - (\mathbf{A})_j\|_2 \approx \|(\mathbf{A}\mathbf{V}_k\mathbf{V}_k^T)_i - (\mathbf{A}\mathbf{V}_k\mathbf{V}_k^T)_j\|_2 = \|(\mathbf{A}\mathbf{V}_k)_i - (\mathbf{A}\mathbf{V}_k)_j\|_2$$

  where we showed the equality in Lecture 14.



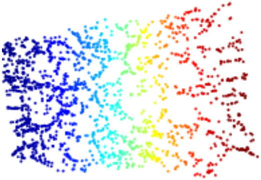


Step 1: Produce a nearest neighbor graph based on your input data in $\mathbb{R}^d$.



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 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 approximate distances in your original 'non-linear space.'