Last Class: Low-Rank Approximation, Eigendecomposition, PCA

- 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 $X^T X$ (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.
The Singular Value Decomposition (SVD) generalizes the eigendecomposition to asymmetric (even rectangular) matrices.

\[ X = U \Sigma V^T \]

- \( U \) has orthonormal columns \( \vec{u}_1, \ldots, \vec{u}_r \in \mathbb{R}^n \) (left singular vectors).
- \( V \) has orthonormal columns \( \vec{v}_1, \ldots, \vec{v}_r \in \mathbb{R}^d \) (right singular vectors).
- \( \Sigma \) 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.
The Singular Value Decomposition (SVD) generalizes the eigendecomposition to asymmetric (even rectangular) matrices. Any matrix $X \in \mathbb{R}^{n \times d}$ with $\text{rank}(X) = r$ can be written as $X = U \Sigma V^T$.

- $U$ has orthonormal columns $\vec{u}_1, \ldots, \vec{u}_r \in \mathbb{R}^n$ (left singular vectors).
- $V$ has orthonormal columns $\vec{v}_1, \ldots, \vec{v}_r \in \mathbb{R}^d$ (right singular vectors).
- $\Sigma$ is diagonal with elements $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$ (singular values).
The Singular Value Decomposition (SVD) generalizes the eigendecomposition to asymmetric (even rectangular) matrices. Any matrix $X \in \mathbb{R}^{n \times d}$ with $\text{rank}(X) = r$ can be written as $X = U \Sigma V^T$.

- $U$ has orthonormal columns $\vec{u}_1, \ldots, \vec{u}_r \in \mathbb{R}^n$ (left singular vectors).
- $V$ has orthonormal columns $\vec{v}_1, \ldots, \vec{v}_r \in \mathbb{R}^d$ (right singular vectors).
- $\Sigma$ 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 $X \in \mathbb{R}^{n \times d}$ with $\text{rank}(X) = r$ can be written as $X = U\Sigma V^T$.

- $U$ has orthonormal columns $\vec{u}_1, \ldots, \vec{u}_r \in \mathbb{R}^n$ (left singular vectors).
- $V$ has orthonormal columns $\vec{v}_1, \ldots, \vec{v}_r \in \mathbb{R}^d$ (right singular vectors).
- $\Sigma$ 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 $X \in \mathbb{R}^{n \times d}$ in its singular value decomposition $X = U\Sigma V^T$:

$$X^T X =$$

\[ X \in \mathbb{R}^{n \times d}: \text{data matrix}, \quad U \in \mathbb{R}^{n \times \text{rank}(X)}: \text{matrix with orthonormal columns} \quad \vec{u}_1, \vec{u}_2, \ldots \text{ (left singular vectors)}, \quad V \in \mathbb{R}^{d \times \text{rank}(X)}: \text{matrix with orthonormal columns} \quad \vec{v}_1, \vec{v}_2, \ldots \text{ (right singular vectors)}, \quad \Sigma \in \mathbb{R}^{\text{rank}(X) \times \text{rank}(X)}: \text{positive diagonal matrix containing singular values of } X. \]
Writing \( X \in \mathbb{R}^{n \times d} \) in its singular value decomposition \( X = U \Sigma V^T \):

\[
X^T X = V \Sigma U^T U \Sigma V^T
\]

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

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

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

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

(the eigendecomposition)

**X** ∈ **R**^{n×d}: data matrix, **U** ∈ **R**^{n× rank(X)}: matrix with orthonormal columns  
  $\vec{u}_1, \vec{u}_2, \ldots$ (left singular vectors),  
**V** ∈ **R**^{d× rank(X)}: matrix with orthonormal columns  
  $\vec{v}_1, \vec{v}_2, \ldots$ (right singular vectors),  
**Σ** ∈ **R**^{rank(X)× rank(X)}: positive diagonal matrix containing singular values of **X**.
Writing $X \in \mathbb{R}^{n \times d}$ in its singular value decomposition $X = U \Sigma V^T$:

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

(the eigendecomposition)

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

---

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

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

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

The right and left singular vectors are the eigenvectors of the covariance matrix $X^T X$ and the gram matrix $XX^T$ respectively.
Writing \( X \in \mathbb{R}^{n \times d} \) in its singular value decomposition \( X = U \Sigma V^T \):

\[
X^T X = V \Sigma U^T U \Sigma V^T = V \Sigma^2 V^T \quad \text{(the eigendecomposition)}
\]

Similarly: \( XX^T = U \Sigma V^T V \Sigma U^T = U \Sigma^2 U^T \).

The right and left singular vectors are the eigenvectors of the covariance matrix \( X^T X \) and the gram matrix \( XX^T \) respectively.

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

---

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

\[
X^T X = V \Sigma U^T U \Sigma V^T = V \Sigma^2 V^T \quad \text{(the eigendecomposition)}
\]

Similarly: \( X X^T = U \Sigma V^T V \Sigma U^T = U \Sigma^2 U^T \).

The right and left singular vectors are the eigenvectors of the covariance matrix \( X^T X \) and the gram matrix \( XX^T \) respectively.

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

What about \( U_k U_k^T X \) where \( U_k \in \mathbb{R}^{n \times k} \) has columns equal to \( \vec{u}_1, \ldots, \vec{u}_k \)?
Writing $X \in \mathbb{R}^{n \times d}$ in its singular value decomposition $X = U \Sigma V^T$:

$$X^T X = V \Sigma U^T U \Sigma V^T = V \Sigma^2 V^T$$ (the eigendecomposition)

Similarly: $XX^T = U \Sigma V^T V \Sigma U^T = U \Sigma^2 U^T$.

The right and left singular vectors are the eigenvectors of the covariance matrix $X^T X$ and the gram matrix $XX^T$ respectively.

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

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

**Exercise:** $U_k U_k^T X = XV_k V_k^T = U_k \Sigma_k V_k^T$

---

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

$$X_k = \arg \min_{\text{rank } - k} \min_{B \in \mathbb{R}^{n \times d}} \|X - B\|_F$$

is given by:

$$X_k = XV_k V_k^T = U_k U_k^T X = U_k \Sigma_k V_k^T$$
The best low-rank approximation to $X$:

$$X_k = \arg \min_{\text{rank} - k} \|X - B\|_F$$

is given by:

$$X_k = XV_kV_k^T = U_kU_k^TX = U_k\Sigma_kV_k^T$$

Correspond to projecting the rows (data points) onto the span of $V_k$ or the columns (features) onto the span of $U_k$. 

### Row (data point) compression

784 dimensional vectors

projections onto 15 dimensional space

orthonormal basis $v_1, \ldots, v_{15}$

### Column (feature) compression

- Home 1: 2 bedrooms, 1800 sq. ft., 2 floors, 200,000
- Home 2: 4 bedrooms, 2700 sq. ft., 1 floor, 300,000
- .
- .
- Home n: 5 bedrooms, 3600 sq. ft., 3 floors, 450,000

10000* bathrooms + 100* (sq. ft.) = list price
The best low-rank approximation to $X$:

$$X_k = \arg \min_{B \in \mathbb{R}^{n \times d}} \|X - B\|_F$$

is given by:

$$X_k = XV_k V_k^T = U_k U_k^T X = U_k \Sigma_k V_k^T$$

Correspond to projecting the rows (data points) onto the span of $V_k$ or the columns (features) onto the span of $U_k$. 

The best low-rank approximation to $X$:

$$X_k = \arg \min_{\text{rank} - k} \{ B \in \mathbb{R}^{n \times d} \mid \| X - B \|_F \}$$

is given by:

$$X_k = XV_kV_k^T = U_kU_k^TX = U_k\Sigma_kV_k^T$$

Correspond to projecting the rows (data points) onto the span of $V_k$ or the columns (features) onto the span of $U_k$. 
The best low-rank approximation to $X$:

$$X_k = \arg \min_{\text{rank} - k} \| X - B \|_F$$

is given by:

$$X_k = X V_k V_k^T = U_k U_k^T X = U_k \Sigma_k V_k^T$$

Correspond to projecting the rows (data points) onto the span of $V_k$ or the columns (features) onto the span of $U_k$. 

\[ \text{Diagram showing $X_k$, $U_k$, $\Sigma_k$, and $V_k$} \]
• Let \( \vec{v}_1, \vec{v}_2, \ldots, \in \mathbb{R}^d \) be orthonormal eigenvectors of \( X^T X \).
• Let $\vec{v}_1, \vec{v}_2, \ldots, \in \mathbb{R}^d$ be orthonormal eigenvectors of $X^T X$.
• Let $\sigma_i = \|X \vec{v}_i\|_2$ and define unit vector $\vec{u}_i = \frac{X \vec{v}_i}{\sigma_i}$.

Exercise: Show $\vec{u}_1, \vec{u}_2, \ldots$ are orthonormal.

This establishes that $X V = U \Sigma$ and that $V$ and $U$ have the required properties.

To see rest of these details, see https://math.mit.edu/classes/18.095/2016IAP/lec2/SVD_Notes.pdf
BASIC IDEA TO PROVE EXISTENCE OF SVD

- Let \( \vec{v}_1, \vec{v}_2, \ldots, \in \mathbb{R}^d \) be orthonormal eigenvectors of \( X^TX \).
- Let \( \sigma_i = \|X\vec{v}_i\|_2 \) and define unit vector \( \vec{u}_i = \frac{X\vec{v}_i}{\sigma_i} \).
- Exercise: Show \( \vec{u}_1, \vec{u}_2, \ldots \) are orthonormal.
Basic Idea to Prove Existence of SVD

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

$$\mathbf{X}$$

Users

Movies

$$\begin{array}{ccc}
5 & 1 & 4 \\
3 & 4 & 5 \\
5 & 2 & 5 \\
1 & & \\
\end{array}$$

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

$$\mathbf{Y} = \arg \min_{\text{rank} - k} \sum_{\text{observed } (j,k)} \| \mathbf{X}_{j,k} - \mathbf{B}_{j,k} \|^2$$

Under certain assumptions, can show that $\mathbf{Y}$ well approximates $\mathbf{X}$ 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?
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.
EXAMPLE: LATENT SEMANTIC ANALYSIS

Term Document Matrix X

Low-Rank Approximation via SVD

\[ \Sigma_k V_k^T \]
EXAMPLE: LATENT SEMANTIC ANALYSIS

Term Document Matrix X

\[
\begin{array}{cccccc}
\text{car} & \text{loan} & \text{house} & \cdots & \text{dog} & \text{cat} \\
\text{doc}_1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
\text{doc}_2 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
\vdots & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
\text{doc}_n & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
\end{array}
\]

Low-Rank Approximation via SVD

\[ X \approx Y \]

\[ Z^T \]
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., $\langle \vec{y}_i, \vec{z}_a \rangle \approx 1$ when doc$_i$ contains word $a$.

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

**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 \( \|X - YZ^T\|_F \) is small, then on average,

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

- I.e., \( \langle \vec{y}_i, \vec{z}_a \rangle \approx 1 \) when \( \text{doc}_i \) contains \( \text{word}_a \).
If the error $\|X - YZ^T\|_F$ is small, then on average,

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

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\(_a\), $\langle \vec{y}_i, \vec{z}_a \rangle \approx \langle \vec{y}_j, \vec{z}_a \rangle \approx 1$. 
If $doc_i$ and $doc_j$ both contain $word_a$, $\langle \vec{y}_i, \vec{z}_a \rangle \approx \langle \vec{y}_j, \vec{z}_a \rangle \approx 1$
EXAMPLE: LATENT SEMANTIC ANALYSIS

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

**Another View:** Each column of \( \mathbf{Y} \) represents a ‘topic’. \( \vec{y}_i(j) \) indicates how much \( \text{doc}_i \) belongs to topic \( j \). \( \vec{z}_a(j) \) indicates how much \( \text{word}_a \) associates with that topic.
• Just like with documents, $\vec{z}_a$ and $\vec{z}_b$ will tend to have high dot product if $\text{word}_a$ and $\text{word}_b$ appear in many of the same documents.
• Just like with documents, $\vec{z}_a$ and $\vec{z}_b$ will tend to have high dot product if \textit{word}_a and \textit{word}_b appear in many of the same documents.

• In an SVD decomposition we set $Z^T = \Sigma_k V_k^T$.

• The columns of $V_k$ are equivalently: the top $k$ eigenvectors of $X^TX$. 

\begin{align*}
\text{Term Document Matrix } X \\
\begin{array}{cccccc}
\text{doc}_1 & \text{dog} & \text{cat} & \text{bird} & \text{fish} & \text{cat} \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
\end{array}
\end{align*}

\begin{align*}
\text{Low-Rank Approximation via SVD} \\
X \approx Y \\
\end{align*}
• 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 $Z^T = \Sigma_k V_k^T$.

• The columns of $V_k$ are equivalently: the top $k$ eigenvectors of $X^T X$. The eigendecomposition of $X^T X$ is $X^T X = V \Sigma^2 V^T$. 

**EXAMPLE: LATENT SEMANTIC ANALYSIS**
Just like with documents, $\vec{z}_a$ and $\vec{z}_b$ will tend to have high dot product if $\text{word}_a$ and $\text{word}_b$ appear in many of the same documents.

In an SVD decomposition we set $Z^T = \Sigma_k V_k^T$.

The columns of $V_k$ are equivalently: the top $k$ eigenvectors of $X^T X$. The eigendecomposition of $X^T X$ is $X^T X = V \Sigma^2 V^T$.

The best rank-$k$ approximation of $X^T X$ is

$$\arg\min_{\text{rank} - k \ B} \|X^T X - B\|_F = V_k \Sigma_k^2 V_k^T = ZZ^T$$
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.
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 $word_b$. 
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 $\text{word}_a$ and $\text{word}_b$ appear in.
- Think about $X^TX$ as a similarity matrix (gram matrix, kernel matrix) with entry $(a, b)$ being the similarity between $\text{word}_a$ and $\text{word}_b$.
- 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.
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.
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.
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.
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.
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.

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

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

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

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

$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$
What is the sum of entries in the $i^{th}$ column of $A$? The (weighted) degree of vertex $i$.

Often, $A$ is normalized as $\bar{A} = D^{-1/2}AD^{-1/2}$ where $D$ is the degree matrix.
What is the sum of entries in the $i^{th}$ column of $A$? The (weighted) degree of vertex $i$.

Often, $A$ is normalized as $\tilde{A} = D^{-1/2}AD^{-1/2}$ where $D$ is the degree matrix.
What is the sum of entries in the \( i^{th} \) column of \( A \)? The (weighted) degree of vertex \( i \).

Often, \( A \) is normalized as \( \tilde{A} = D^{-1/2} A D^{-1/2} \) where \( D \) is the degree matrix.
What is the sum of entries in the $i^{th}$ column of $A$? The (weighted) degree of vertex $i$.

Often, $A$ is normalized as $\tilde{A} = D^{-1/2}AD^{-1/2}$ where $D$ 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 $A$?

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

Similar vertices (close with regards to graph proximity) should have similar embeddings.
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 $\mathbf{A}$.

- Similar vertices (close with regards to graph proximity) should have similar embeddings.
SPECTRAL EMBEDDING