COMPSCI 514: ALGORITHMS FOR DATA SCIENCE

Andrew McGregor
Lecture 17
Last Class: Low-Rank Approximation, Eigendecomposition, PCA

• For any symmetric square matrix $A$, we can write $A = \mathbf{V} \mathbf{\Lambda} \mathbf{V}^T$ 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.
The Singular Value Decomposition (SVD) generalizes the eigendecomposition to asymmetric (even rectangular) matrices.
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 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).

The ‘swiss army knife’ of modern linear algebra.
Writing $X \in \mathbb{R}^{n \times d}$ in its singular value decomposition $X = U \Sigma V^T$:

$$X^T X =$$

---

**Definitions:**
- $X \in \mathbb{R}^{n \times d}$: 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 $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$:

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

| $\mathbf{X} \in \mathbb{R}^{n \times d}$: 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 \( X = U \Sigma V^T \):

\[
X^T X = V \Sigma U^T U \Sigma V^T = V \Sigma^2 V^T
\]

**Definitions:**
- \( X \in \mathbb{R}^{n \times d} \): 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 $X = U \Sigma V^T$:

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

(the eigendecomposition)

**X ∈ \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 $X = U \Sigma V^T$:

$$X^T X = V \Sigma U^T U \Sigma V^T = V \Sigma^2 V^T$$ (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), $\Sigma \in \mathbb{R}^{\text{rank}(X) \times \text{rank}(X)}$: positive diagonal matrix containing singular values of $X$. 
Writing $X \in \mathbb{R}^{n \times d}$ in its singular value decomposition $X = U \Sigma V^T$:

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

---

$X \in \mathbb{R}^{n \times d}$: 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 $X = U\Sigma V^T$:

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

(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 $X^TX$ 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 $XV_kV_k^T$ is the best rank-$k$ approximation to $X$ (given by PCA).

---

**X ∈ \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 $X = U \Sigma V^T$:

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

(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).

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

---

**X ∈ \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 $X = U \Sigma V^T$:

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

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

---

**Table:**

- $X \in \mathbb{R}^{n \times d}$: 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 $X$. 

---

3
The best low-rank approximation to $X$, i.e.,

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

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 $X$, i.e.,

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

is given by $X_k = XV_kV_k^T = U_kU_k^TX = U_k\Sigma_kV_k^T$

Corresponds 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 $X$, i.e.,

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

is given by $X_k = XV_k V_k^T = U_k U_k^T X = U_k \Sigma_k V_k^T$

Corresponds 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 $\mathbf{X}$, i.e.,

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

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 $X$, i.e.,

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

is given by $X_k = XV_kV_k^T = U_k U_k^T X = U_k \Sigma_k V_k^T$

Corresponds to projecting the rows (data points) onto the span of $V_k$ or the columns (features) onto the span of $U_k$.
• 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 $X^TX$.

• Let $\sigma_i = \|X\vec{v}_i\|_2$ and define unit vector $\vec{u}_i = \frac{X\vec{v}_i}{\sigma_i}$. 
BASIC IDEA TO PROVE EXISTENCE OF SVD

• Let \( \vec{v}_1, \vec{v}_2, \ldots, \in \mathbb{R}^d \) be orthonormal eigenvectors of \( X^TX \).
• 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.
• 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.
• This establishes that $XV = U\Sigma$ and that $V$ and $U$ have the required properties.
• 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 $XV = U\Sigma$ and that $V$ and $U$ have the required properties.
• To see rest of the 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.
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.

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

$$Y = \text{arg min}_{\text{rank} - k B} \sum_{\text{observed} (j, k)} (X_{j, k} - B_{j, k})^2$$

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?

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.
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**

<table>
<thead>
<tr>
<th></th>
<th>car</th>
<th>loan</th>
<th>house</th>
<th>...</th>
<th>dog</th>
<th>cat</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc_1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>doc_2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>doc_n</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Low-Rank Approximation via SVD**

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

Term Document Matrix $X$

Low-Rank Approximation via SVD

$X \approx YZ^T$
If the error \( \| X - YZ^T \|_F \) is small, then on average, 
\[ X_i, a \approx (YZ^T)_i, a = \langle \vec{y}_i, \vec{z}_a \rangle. \]

\[ I.e., \quad \langle \vec{y}_i, \vec{z}_a \rangle \approx 1 \text{ when } \text{doc}_i \text{ contains word } a. \]

If \( \text{doc}_i \) and \( \text{doc}_j \) 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,

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

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

![Term Document Matrix X](image)

![Low-Rank Approximation via SVD](image)
EXAMPLE: LATENT SEMANTIC ANALYSIS

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

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

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

**Term Document Matrix X**

<table>
<thead>
<tr>
<th></th>
<th>cat</th>
<th>dog</th>
<th>...</th>
<th>cat</th>
<th>dog</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc_1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>doc_2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>doc_n</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Low-Rank Approximation via SVD**

\[ X \approx Y \]

\[ Z^T \]
• 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 $Z^T = \sum_k V_k^T$ where columns of $V_k$ are the top $k$ eigenvectors of $X^T X$. 
• 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 $Z^T = \Sigma_k V_K^T$ where columns of $V_k$ are the top $k$ eigenvectors of $X^T X$. 
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.
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 $\text{word}_a$ and $\text{word}_b$ 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 $\text{word}_a$ 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.

- Replacing $\mathbf{X}^T \mathbf{X}$ 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 $\text{word}_a$ 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 $X^T X$: where $(X^T X)_{a,b}$ is the number of documents that both word$\_a$ and word$\_b$ appear in.
- Think about $X^T X$ as a similarity matrix (gram matrix, kernel matrix) with entry $(a, b)$ 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 $X^T X$ 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 a 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.