COMPSCI 514: Algorithms for Data Science

Cameron Musco
University of Massachusetts Amherst. Fall 2023.
Lecture 18

Logistics

- Problem Set 3 is due next **Friday at 11:59pm**.
- I made a small change to Problem 1.4: replacing $\sum_{i=1}^{n} \sigma_i(\mathbf{A})^2$ with $\sum_{i=1}^{\text{rank}(\mathbf{A})} \sigma_i(\mathbf{A})^2$. This don't change the solution to the problem, but as we will see will better match the conventions for SVD that I introduce today.
- Linear algebra review session **Monday 2-3pm**. Location TBD.

Summary

Last Class

- Finish up optimal low-rank approximation via eigendecomposition.
- Eigenvalue spectrum as a way of measuring low-rank approximation error.

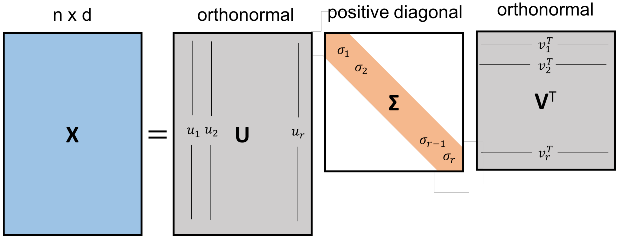
This Class: The SVD and Application of Low-Rank Approximation Beyond Compression

- The Singular Value Decomposition (SVD) and its connection to eigendecomposition and low-rank approximation.
- Low-rank matrix completion (predicting missing measurements using low-rank structure).
- Entity embeddings (e.g., word embeddings, node embeddings).
- Low-rank approximation for non-linear dimensionality reduction.

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

- $\mathbf{U}$ 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).
- $\boldsymbol{\Sigma}$ is diagonal with elements $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$ (singular values).



Connection of the SVD to Eigendecomposition

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 left and right 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 $\vec{u}_1, \ldots, \vec{u}_k$?
Gives exactly the same approximation!

$\mathbf{X} \in \mathbb{R}^{n \times d}$: 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 SVD and Optimal Low-Rank Approximation

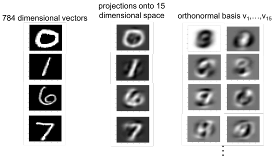
The best low-rank approximation to $\mathbf{X}$:

$\mathbf{X}_k = \arg\min_{\text{rank}-k\ \mathbf{B} \in \mathbb{R}^{n \times d}} \|\mathbf{X} - \mathbf{B}\|_F$ is given by:

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

Correspond 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

n x d orthonormal positive diagonal orthonormal n x d (rank k) orthonorma

The SVD and Optimal Low-Rank Approximation

The best low-rank approximation to $\mathbf{X}$:

$\mathbf{X}_k = \arg\min_{\text{rank}-k\ \mathbf{B} \in \mathbb{R}^{n \times d}} \|\mathbf{X} - \mathbf{B}\|_F$ is given by:

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

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

The SVD and Optimal Low-Rank Approximation

The best low-rank approximation to $\mathbf{X}$:

$\mathbf{X}_k = \arg\min_{\text{rank}-k\ \mathbf{B} \in \mathbb{R}^{n \times d}} \|\mathbf{X} - \mathbf{B}\|_F$ is given by:

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

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

SVD Review

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

Applications of Low-Rank Approximation Beyond 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). Classic example: the Netflix prize problem.

X Movies

Users

5	3	3	1	4	4	4	3	5
4	3	3	1	4	4	5	3	5
3	3	3	2	3	3	3	3	3
4	3	3	4	4	4	4	3	3
3	3	3	2	3	3	3	3	3
2	5	3	4	4	4	4	4	5
1	3	3	2	3	3	3	1	2

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

Under certain assumptions, can show that $\mathbf{Y}$ well approximates $\mathbf{X}$ on both the observed and (most importantly) unobserved entries.

Entity Embeddings

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

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

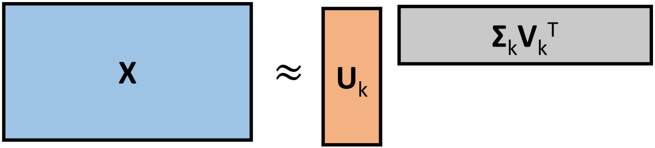
Classic Approach: Convert each item into a (very) high-dimensional feature vector and then apply low-rank approximation.

Example: Latent Semantic Analysis

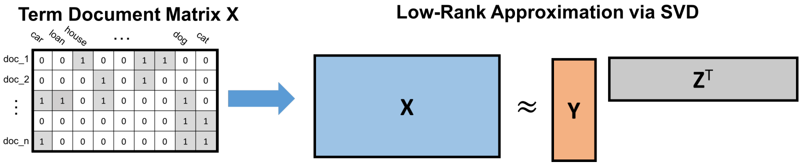
Term Document Matrix X

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

Low-Rank Approximation via SVD



Example: Latent Semantic Analysis



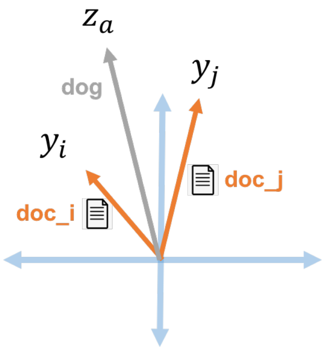
- If the error $\|X - YZ^T\|_F$ is small, then on average,

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

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

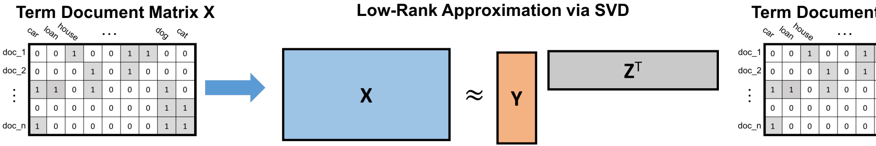
Example: Latent Semantic Analysis

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



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

Example: Latent Semantic Analysis



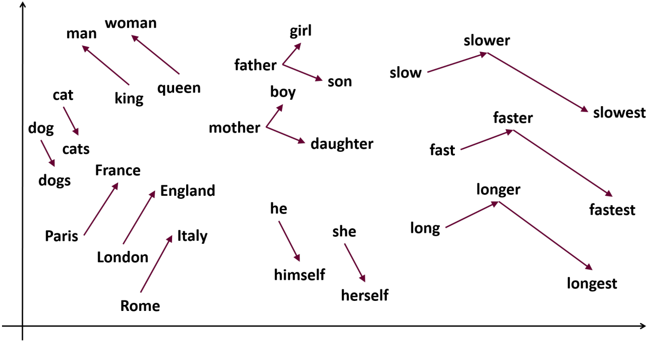
- Just like with documents, $\vec{z}_a$ and $\vec{z}_b$ will tend to have high dot product if $word_a$ and $word_b$ appear in many of the same documents.
- In an SVD decomposition we set $\mathbf{Z}^T = \mathbf{\Sigma}_k \mathbf{V}_K^T$.
- The columns of $\mathbf{V}_k$ are equivalently: the top k eigenvectors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

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.

Example: Word Embedding



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

Questions?