Matrix approximation

COMS 4771 Fall 2019

Overview

- Compression using k-means
- Compression using SVD
- Power iteration
- Latent semantic analysis
- Matrix completion

Encoder/decoder interpretation of k-means clustering

- Given data $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n \in \mathbb{R}^d$, we learn a *lossy encoding/decoding (compression) scheme*
- Scheme is represented by $\boldsymbol{c}_1, \dots, \boldsymbol{c}_k \in \mathbb{R}^d$
- Encoder: $\boldsymbol{x} \mapsto \arg\min_j \|\boldsymbol{x} - \boldsymbol{c}_j\|_2^2$
  - Need $\log_2 k$ bits to represent encoding of $\boldsymbol{x}$!
- Decoder: $j \mapsto \boldsymbol{c}_j$

Matrix view

- Let $\boldsymbol{A} = \begin{bmatrix} \leftarrow & \boldsymbol{x}_1^{\mathsf{T}} & \rightarrow \\ & \vdots & \\ \leftarrow & \boldsymbol{x}_n^{\mathsf{T}} & \rightarrow \end{bmatrix} \in \mathbb{R}^{n \times d}$ (forget the $1/\sqrt{n}$ scaling)
- Let $\boldsymbol{C} = \begin{bmatrix} \leftarrow & \boldsymbol{c}_1^{\mathsf{T}} & \rightarrow \\ & \vdots & \\ \leftarrow & \boldsymbol{c}_k^{\mathsf{T}} & \rightarrow \end{bmatrix} \in \mathbb{R}^{k \times d}$
- We try to approximate $\boldsymbol{A}$ with $\boldsymbol{BC}$, where $\boldsymbol{B} \in \{0,1\}^{n \times k}$ is matrix where $B_{i,j} = 1$ iff $\boldsymbol{x}_i$ is assigned to group j.
- The k-means cost is $\|\boldsymbol{A} - \boldsymbol{BC}\|_F^2$
  - Here, $\|\cdot\|_F$ is a matrix norm called *Frobenius norm*, which treats the $n \times d$ matrix as a vector in nd-dimensional Euclidean space
- $\boldsymbol{BC}$ is a particular rank $\leq k$ approximation of the data matrix A

Matrix factorization

- Try to approximate $\boldsymbol{A}$ with $\boldsymbol{BC}$, where $\boldsymbol{B} \in \mathbb{R}^{n \times k}$ and $\boldsymbol{C} \in \mathbb{R}^{k \times d}$, to minimize $\|\boldsymbol{A} - \boldsymbol{BC}\|_F^2$.
  - Think of $\boldsymbol{B}$ as the encodings of the data in $\boldsymbol{A}$
  - At least get "dimension reduction" when $k < d$
  - Maybe not quite as space-efficient as k-means
- **Theorem** (Schmidt, 1907; Eckart-Young, 1936): Optimal solution is given by truncating the *singular value decomposition (SVD)* of $\boldsymbol{A}$

Singular value decomposition

- Every matrix $\boldsymbol{A} \in \mathbb{R}^{n \times d}$—say, with rank r—can be written as

$$\boldsymbol{A} = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^\top$$

  where
  - $\sigma_1 \geq \cdots \geq \sigma_r > 0$ (*singular values*)
  - $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n \in \mathbb{R}^n$ (orthonormal *left singular vectors*)
  - $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n \in \mathbb{R}^d$ (orthonormal *right singular vectors*)
- Can also write as

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\top$$

  where
  - $\boldsymbol{U} = [\boldsymbol{u}_1 | \cdots | \boldsymbol{u}_r] \in \mathbb{R}^{n \times r}$, satisfies $\boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}$
  - $\boldsymbol{S} = \operatorname{diag}(\sigma_1, \ldots, \sigma_r) \in \mathbb{R}^{r \times r}$
  - $\boldsymbol{V} = [\boldsymbol{v}_1 | \cdots | \boldsymbol{v}_r] \in \mathbb{R}^{d \times r}$, satisfies $\boldsymbol{V}^\top \boldsymbol{V} = \boldsymbol{I}$
- This is sometimes called the *thin SVD*, since the matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ are as thin as possible.

Truncated SVD

- Let $\boldsymbol{A}$ have SVD $\boldsymbol{A} = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^\top$ (rank of $\boldsymbol{A}$ is r)
- Truncate at rank k (for any $k \leq r$): *rank-k SVD*

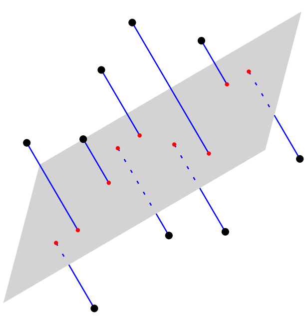
$$\boldsymbol{A}_k := \sum_{i=1}^{k} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^\top$$

- Can write as $\boldsymbol{A}_k := \boldsymbol{U}_k \boldsymbol{S}_k \boldsymbol{V}_k^\top$, where
  - $\boldsymbol{U}_k = [\boldsymbol{u}_1 | \cdots | \boldsymbol{u}_k] \in \mathbb{R}^{n \times k}$, satisfies $\boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}$
  - $\boldsymbol{S}_k = \operatorname{diag}(\sigma_1, \ldots, \sigma_k) \in \mathbb{R}^{k \times k}$
  - $\boldsymbol{V}_k = [\boldsymbol{v}_1 | \cdots | \boldsymbol{v}_k] \in \mathbb{R}^{d \times r}$, satisfies $\boldsymbol{V}^\top \boldsymbol{V} = \boldsymbol{I}$
- **Theorem** (Schmidt/Eckart-Young):

$$\|\boldsymbol{A} - \boldsymbol{A}_k\|_F^2 = \min_{\boldsymbol{M}:\operatorname{rank}(\boldsymbol{M})=k} \|\boldsymbol{A} - \boldsymbol{M}\|_F^2 = \sum_{i=k+1}^{r} \sigma_i^2$$

Encoder/decoder interpretation I

- Encoder: $\boldsymbol{x} \mapsto \boldsymbol{V}_k^\top \boldsymbol{x} \in \mathbb{R}^k$
  - Encoding rows of $\boldsymbol{A}$: $\boldsymbol{A}\boldsymbol{V}_k = \boldsymbol{U}_k \boldsymbol{S}_k$
- Decoder: $\boldsymbol{z} \mapsto \boldsymbol{V}_k \boldsymbol{z} \in \mathbb{R}^d$
  - Decoding rows of $\boldsymbol{U}_k \boldsymbol{S}_k$: $\boldsymbol{U}_k \boldsymbol{S}_k \boldsymbol{V}_k^\top = \boldsymbol{A}_k$
- Same as k-dimensional PCA mapping!
  - $\boldsymbol{A}^\top \boldsymbol{A} = \boldsymbol{V}\boldsymbol{S}^2\boldsymbol{V}^\top$, so eigenvectors of $\boldsymbol{A}^\top \boldsymbol{A}$ are right singular vectors of $\boldsymbol{A}$, non-zero eigenvalues are squares of the singular values
- PCA/SVD finds k-dimensional subspace of smallest sum of squared distances to data points.



Encoder/decoder interpretation II

- Example: OCR data, compare original image to decoding of k-dimensional PCA encoding ($k \in \{1, 10, 50, 200\}$)

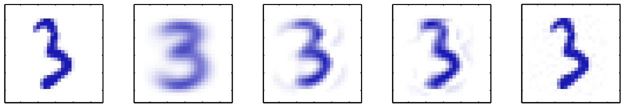


Figure 2: OCR digits, original and decoding of k-dimensional PCA encoding

Computation: power iteration

- How to find $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$?
- Easier problem: how to find $\boldsymbol{v}_1$?
- *Power iteration*:
  - Start with any unit vector $\boldsymbol{u} \in \mathbb{R}^d$
  - Repeat: Multiply $\boldsymbol{u}$ by $\boldsymbol{A}^\top \boldsymbol{A}$ to get new vector $\boldsymbol{u}'$; renormalize back to unit length to replace $\boldsymbol{u}$

Why does power iteration work?

- Ignore renormalization for now
- Then after t iterations, resulting vector is

$$\underbrace{(\boldsymbol{A}^\top \boldsymbol{A}) \cdots (\boldsymbol{A}^\top \boldsymbol{A})}_{t \text{ times}} \boldsymbol{u} = \underbrace{(\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^\top) \cdots (\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^\top)}_{t \text{ times}} \boldsymbol{u} = \boldsymbol{V}\boldsymbol{\Lambda}^t \boldsymbol{V}^\top \boldsymbol{u}$$

- Look at $\boldsymbol{\Lambda}^t$:

$$\boldsymbol{\Lambda}^t = \lambda_1^t \begin{bmatrix} 1 & & & \\ & \left(\frac{\lambda_2}{\lambda_1}\right)^t & & \\ & & \ddots & \\ & & & \left(\frac{\lambda_r}{\lambda_1}\right)^t \end{bmatrix} \approx \lambda_1^t \begin{bmatrix} 1 & & & \\ & 0 & & \\ & & \ddots & \\ & & & 0 \end{bmatrix}$$

  whenever $\lambda_2 < \lambda_1$ and t is large.
- Therefore

$$\boldsymbol{V}\boldsymbol{\Lambda}^t \boldsymbol{V}^\top \approx \lambda_1^t \boldsymbol{v}_1 \boldsymbol{v}_1^\top$$

  and hence

$$\boldsymbol{V}\boldsymbol{\Lambda}^t \boldsymbol{V}^\top u \approx \lambda_1^t \boldsymbol{v}_1 \boldsymbol{v}_1^\top \boldsymbol{u} = (\lambda_1^t \boldsymbol{v}_1^\top \boldsymbol{u}) \boldsymbol{v}_1$$

  which renormalizes back to $\boldsymbol{v}_1$ (assuming $\boldsymbol{v}_1^\top \boldsymbol{u} \neq 0$).

Beyond the top eigenvector

- To get $\boldsymbol{v}_2, \ldots, \boldsymbol{v}_k$:
  - Can *deflate* $\boldsymbol{A}^\top \boldsymbol{A}$: replace $\boldsymbol{A}^\top \boldsymbol{A}$ by $\boldsymbol{A}^\top \boldsymbol{A} - \lambda_1 \boldsymbol{v}_1 \boldsymbol{v}_1^\top$; now $\boldsymbol{v}_2$ is top eigenvector.
  - This only works well for small k, since errors accumulate
- Alternative: to get some $\boldsymbol{U} \in \mathbb{R}^{d \times k}$ with same (similar) range as $\boldsymbol{V}_k$, use *subspace iteration*.

Application: Topic modeling

- Start with n documents, represent using "bag-of-words" count vectors
- Arrange in matrix $\boldsymbol{A} \in \mathbb{R}^{n \times d}$, where d is vocabulary size

	aardvark	abacus	abalone	$\cdots$
doc 1	3	0	0	$\cdots$
doc 2	7	0	4	$\cdots$
doc 3	2	4	0	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	

- Rank k SVD provides an approximate factorization

$$\boldsymbol{A} \approx \boldsymbol{B}\boldsymbol{C}$$

  where $\boldsymbol{B} \in \mathbb{R}^{n \times k}$ and $\boldsymbol{C} \in \mathbb{R}^{k \times d}$
- This use of SVD is called *Latent Semantic Analysis (LSA)*
- Interpret rows of $\boldsymbol{C}$ as "topics"
- $B_{i,t}$ is "weight" of document i on topic t
- If rows of $\boldsymbol{C}$ were probability distributions, could interpret as $C_{t,w}$ as probability that word w appears in topic t

Application: Matrix completion

- Start with ratings of movies given by users
- Arrange in a matrix $\boldsymbol{A} \in \mathbb{R}^{n \times d}$, where $A_{i,j}$ is rating given by user i for movie j.
  - Netflix: $n = 480000$, $d = 18000$; on avg, each user rates 200 movies
  - Most entries of $\boldsymbol{A}$ are unknown
- Idea: Approximate $\boldsymbol{A}$ with low-rank matrix, i.e., find

$$\boldsymbol{U} = \begin{bmatrix} \leftarrow & \boldsymbol{u}_1^{\mathsf{T}} & \rightarrow \\ & \vdots & \\ \leftarrow & \boldsymbol{u}_n^{\mathsf{T}} & \rightarrow \end{bmatrix}, \qquad \boldsymbol{M} = \begin{bmatrix} \uparrow & & \uparrow \\ \boldsymbol{m}_1 & \cdots & \boldsymbol{m}_d \\ \downarrow & & \downarrow \end{bmatrix}$$

  with goal of minimizing $\|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{M}\|_F^2$
- Note: If all entries of $\boldsymbol{A}$ were observed, we could do this with truncated SVD.
- Need to find a low-rank approximation without all of $\boldsymbol{A}$: *(low-rank) matrix completion*

SGD for matrix completion

- Instead, try to minimize

$$\sum_{(i,j)\in\Omega} (A_{i,j} - \boldsymbol{u}_i^{\mathsf{T}}\boldsymbol{m}_j)^2$$

  where Ω indexes the collection of observed entries of $\boldsymbol{A}$.
- Not convex in $(\boldsymbol{U}, \boldsymbol{M})$; still use (stochastic) gradient descent anyway
- For epoch $p = 1, 2, \ldots$:
  - For each $(i,j) \in \Omega$ in random order:

$$\boldsymbol{u}_i := \boldsymbol{u}_i - 2\eta(\boldsymbol{u}_i^{\mathsf{T}}\boldsymbol{m}_j - A_{i,j})\boldsymbol{m}_j$$
$$\boldsymbol{m}_j := \boldsymbol{m}_j - 2\eta(\boldsymbol{u}_i^{\mathsf{T}}\boldsymbol{m}_j - A_{i,j})\boldsymbol{u}_i$$

    (Could also switch or randomize order of updates)
- Can also use regularization and other objectives / optimization algorithms

Feature representations from matrix completion

- Small data set ($n = 6040$ users, $d = 3952$ movies, $|\Omega| = 800000$ ratings)
- Fit $\boldsymbol{U}$ and $\boldsymbol{M}$ by optimizing (regularized) least squares objective
- Are $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_d \in \mathbb{R}^k$ useful feature vectors for movies?
- Some nearest-neighbor pairs $(\boldsymbol{m}_j, \mathrm{NN}(\boldsymbol{m}_j))$:
  - Toy Story (1995), Toy Story 2 (1999)
  - Sense and Sensibility (1995), Emma (1996)
  - Heat (1995), Carlito's Way (1993)
  - The Crow (1994), Blade (1998)
  - Forrest Gump (1994), Dances with Wolves (1990)
  - Mrs. Doubtfire (1993), The Bodyguard (1992)