STA 141C - Big Data & High Performance Statistical Computing **Spring 2022**

Lecture 11: Singular value decomposition

Lecturer: Bo Y.-C. Ning *February 10, 2022*

Disclaimer: *My notes may contain errors, distribution outside this class is allowed only with the permission of the Instructor.*

Announcement

- HW2 due next Wednesday

Last time

- PageRank problem
- Iterative method

Today

- Singular value decomposition (SVD)
- Principal component analysis (PCA)

1 Review of singular value decomposition (SVD)

For a rectangular matrix $A \in \mathbb{R}^{m \times n}$, let $p = \min\{m, n\}$, then we have the SVD

$$A = U\Sigma V',$$

where $U = (u_1, \dots, u_m)$ and $V = (v_1, \dots, v_n)$ are orthogonal matrices and $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_p)$ is a diagonal matrix such that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$. σ_is are called the *singular values*, u_is are the left singular vectors and v_is are the right singular vectors.

The matrix Σ is not a square matrix, one can define thin SVD, which factorizes A as

$$A = U_n \Sigma_n V' = \sum_{i=1}^{n} \sigma_i u_i v_i',$$

where $U_n \in \mathbb{R}^{m \times n}$, $U_n' U_n = I_n$, $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_n)$. This is for $m > n$, if $m < n$, then we let $V \in \mathbb{R}^{m \times n}$,

The following properties are useful: for $\sigma(A) = (\sigma_1, \dots, \sigma_p)'$, the rank of A is the number of nonzero singular values denoted as $\|\sigma(A)\|_0$. The Frobenius norm of A, $\|A\|_F = (\sum_{i=1}^{p} \sigma_i^2)^{1/2} = \|\sigma(A)\|_2$, and the spectrum norm of A, $\|A\|_2 = \sigma_1 = \|\sigma(A)\|_\infty$. Using the fact that U, V are both orthogonal matrices

$$A'A = V\Sigma U'U\Sigma V' = V\Sigma^2 V',$$
$$AA' = U\Sigma V'V\Sigma U' = U\Sigma^2 U'$$

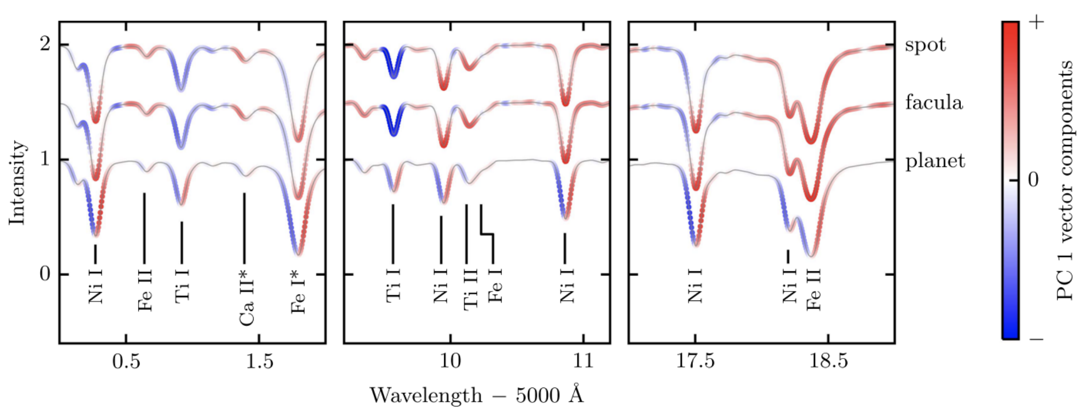
Last, the eigen-decomposition for a real symmetric matrix is $B = W\Lambda W'$, where $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, which is the SVD of B.

2 Applications for SVD

1. Principal component analysis and dimension deduction.

Let $X \in \mathbb{R}^{n\times p}$ be a centered data matrix, perform SVD on $X = U\Sigma V'$. The linear combinations $\tilde{x}_i = Xv_i$ are the principal components (PCs) with variance σ_i^2.

Dimension deduction: reduce dimensionality p to $q \le p$ and use the first few PCs $\tilde{x}_i, \ldots, \tilde{x}_q$ in downstream analysis. Used in medical studies, astronomy, etc.



Davis et al. (2017). *Insights on the Spectral Signatures of Stellar Activity and Planets from PCA*. The Astrophysical Journal.

2. Low rank approximation in image/data compression.

Goal: find Y, $\min_{\mathrm{rank}(X)=r}\|X - Y\|_F^2$. By Eckart-Young theorem, $Y = \sum_{i=1}^r \sigma_i u_i' v_i$ with optimal value $\sum_{i=1}^r \sigma_i^2$, where (σ_i, u_i, v_i) are singular values and vectors of X.

Gene Golub's 2691×598 picture requires $2691 \times 598 \times 6 = 9,655,308$ bytes (RGB 16 bit per channel). Rank 120 approximation requires $120 \times (2691 + 598) \times 6 = 2,368,080$ bytes. Rank 50 approximation requires $50 \times (2691 + 598) \times 6 = 986,700$ bytes. Rank 12 approximation requires $12 \times (2691 + 598) \times 8 = 236,808$ bytes.

3. Ridge regression by SVD.

In ridge regression, we minimize

$$\|y - X\beta\|_2^2 + \lambda\|\beta\|_2^2$$

If we obtain SVD of X such that $X = U\Sigma V$, then the equation is

$$(\Sigma^2 + \lambda I_p)V'\beta = \Sigma U' y.$$

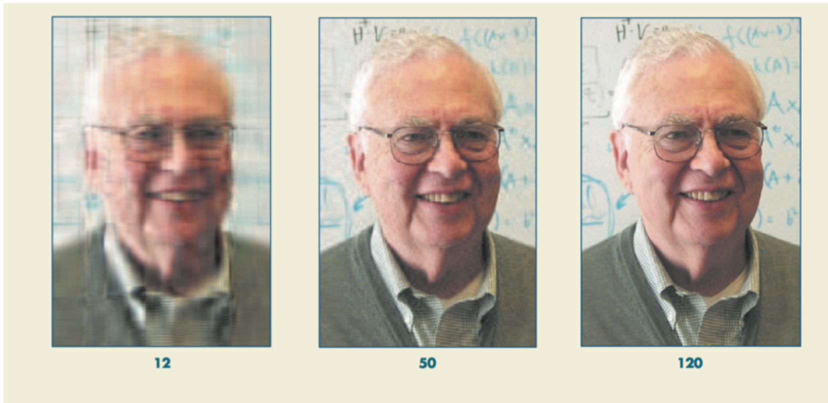


Figure 2. Rank 12, 50, and 120 approximations to a rank 598 color photo of Gene Golub.

We get

$$\hat{\beta}_\lambda = \sum_{j=1}^{r} \frac{\sigma_i u_i' y}{\sigma_i^2 + \lambda} v_i, \ \ r = \text{rank}(X).$$

It is clear that $\hat{\beta}_\lambda \to \beta_{OLS}$ as $\lambda \to 0$ and $\|\hat{\beta}_\lambda\|_2$ is monotone decreasing as $\lambda \to \infty$.

3 Method for computing SVD: Power method

To start, let's assume $A \in \mathbb{R}^{n \times n}$ is a symmetric and p.s.d. matrix, the power method for obtaining the largest eigenvalue is given as:

1) Choose an initial guess of $q^{(0)}$ (non-zero);

2) Repeat $k = 1, \dots, K$,

$$\begin{aligned} z^{(k)} &= Aq^{(k-1)} \\ q^{(k)} &= \frac{z^{(k)}}{\|z^{(k)}\|_2}; \end{aligned}$$

3) Output: $\lambda_1 \leftarrow {q^{(K)}}' A q^{(K)}$.

4 Why the power method works?

Let's understand how the power method works. Before that, we need to recall a few facts:

- The eigenvalue v_i attached to i-th eigenvalue λ_i has the relation $Av_i = \lambda_i v_i$
- Given A, $A = \sum_{i=1}^n \lambda_i v_i v_i'$, where $\lambda_1 \geq \cdots \geq \lambda_n \geq 0$ and $\langle v_i, v_j \rangle = 0$ for $i \neq j$
- $A^k = \sum_{i=1}^n \lambda_i^k v_i v_i'$, why?

By inspecting the algorithm, we have

$$q^{(k)} = \frac{A^k q^{(0)}}{\|A^k q^{(0)}\|_2}.$$

Now given an initial guess of $q^{(0)}$ of unit Euclidean norm, it is possible to express

$$q^{(0)} = \alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_n v_n,$$

for $\alpha_1, \dots, \alpha_n$ are scalars. By the relation $Av_i = \lambda_i v_i$,

$$A^k q^{(0)} = \alpha_1 \lambda_1^k \left(v_1 + \sum_{j=2}^{n} \frac{\alpha_j \lambda_j^k}{\alpha_1 \lambda_1^k} v_j \right)$$

For simplicity, let's denote $y^{(k)} = \sum_{j=2}^{n} \frac{\alpha_j \lambda_j^k}{\alpha_1 \lambda_1^k} v_j$, note that $y^{(k)} \to 0$ as $k \to \infty$ as long as $\lambda_1 > \lambda_2 \geq \cdots \geq \lambda_n$ then

$$q^{(k)} = \frac{A^k q^{(0)}}{\|A^k q^{(0)}\|_2} = \frac{\alpha_1 \lambda_1^k (v_1 + y^{(k)})}{\|\alpha_1 \lambda_1^k (v_1 + y^{(k)})\|_2} \to v_1, \quad \text{as } k \to \infty$$

In practice, k will never goes to ∞, the algorithm will stop as some K when $\min\{\|q^{(K)} - q^{(K-1)}\|_2, \|-q^{(K)} - q^{(K-1)}\|_2\} \leq \epsilon$ for some small ϵ.

The output $q^{(K)}$ is a close approximation of v_1, the leading eigenvector. How to obtain the leading eigenvalue λ_1? (Hint: using $Av_1 = \lambda_1 v_1$).

A few comments:

- The power method works well if $\lambda_1 > \lambda_2$. It converges slowly if $\lambda_1/\lambda_2 \approx 1$.
- The convergence speed of the power method is proportional to $(\lambda_2/\lambda_1)^k$, the ratio between λ_2 and λ_1
- For a general matrix $A^{n\times p}$, we can apply the power method to $A'A$ or AA' instead. Then the output is the absolute value of λ_1.
- How to get $\lambda_2, \cdots, \lambda_n$?
- Eigen-decomposition is implemented in LAPACK, see `eigen()` in R and `np.linalg.eig` in numpy.

The power method is the most basic algorithm for SVD. There are other methods such as

- Inverse power method for
  finding the eigenvalue of smallest absolute value (replace A with A^{-1} in the power method);
- QR algorithm for symmetric eigen-decomposition (takes $4n^3/3$ for eigenvalues and $8n^3/3$ for eigenvector)
- "Golub-Kahan-Reinsch" algorithm (Section 8.6 of Golub and Van Loan); used in `svd` function in R ($4m^2n + 8mn^2 + 9n^3$ flops for an $m > n$ matrix)
- Jacobi methods (Section 8.5 of Golub and Van Loan) (suitable for parallel computing).

Concluding remarks on numerical linear algebra:

- Numerical linear algebra forms the building blocks of most computation we do. Most lines of our code are numerical linear algebra.

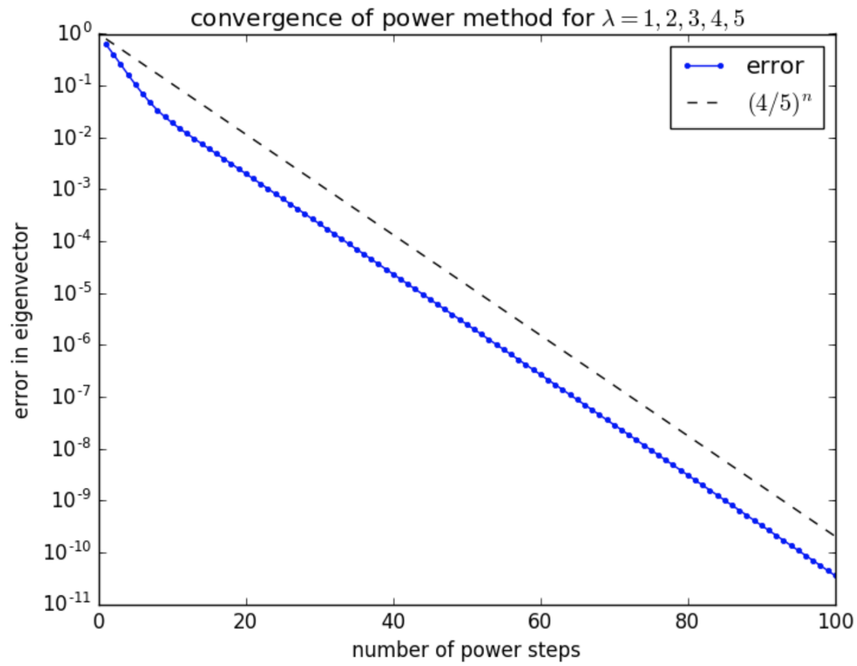


Figure 11.1: Convergence speed of the power method.
[Source: `https://web.mit.edu/18.06/www/Spring17/Power-Method.pdf`.]

- Be flop and memory aware! The form of a mathematical expression and the way the expression should be evaluated in actual practice may be quite different.
- Be alert to problem structure and make educated choice of software/algorithm — the structure should be exploited whenever solving a problem.
- Do not write your own matrix computation routines unless for good reason. Utilize BLAS and LAPACK as much as possible!
- In contrast, for optimization, often we need to devise problem specific optimization routines, or even "mix and match" them.