Singular Value Decomposition (SVD)

Mahbuba Perveen

September 22, 2020
CSE-5400

Properties

- A powerful technique to solve singular (or nearly singular) equations/matrices
- Capable of diagnosing if a solution obtained through other methods doesn't give satisfactory result
    - Sometimes SVD may even solve it, or give a better solution
- Method of choice for solving most linear least-squares problem
    - Problems where solutions of overdetermined systems are approximated by minimizing the sum of the squares of the residuals made in the results of every single equation
- It's based on the following theorem of Linear Algebra:

$$\mathbf{A}_{M \times N} = \mathbf{U}_{M \times N}\, \mathbf{W}_{N \times N}\, \mathbf{V}^{\mathbf{T}}_{N \times N}$$

Where, **U** is a column-orthogonal matrix, **W** is a diagonal matrix with singular values and **V** is an orthogonal matrix.

Properties (cont.)

$$\mathbf{A}_{M \times N} = \mathbf{U}_{M \times N}\, \mathbf{W}_{N \times N}\, \mathbf{V}^{\mathbf{T}}_{N \times N}$$

- $M > N$:
  - Overdetermined system: More equations than unknowns
  - No exact solution (SVD can get “least-squares” solution using pseudo-inverse)
- $M < N$:
  - Undetermined system: Fewer equations than unknowns
  - N x M dimensional family of solutions (“solve” on SVD.h gets the shortest solution)
- $M = N$:
  - Equal number of equations and unknowns

Properties (cont.)

$$\mathbf{A}_{M \times N} = \mathbf{U}_{M \times N} \mathbf{W}_{N \times N} \mathbf{V}^{\mathbf{T}}_{N \times N}$$

- Matrix $\mathbf{V}_{N \times N}$ is orthogonal:
  - Its *columns* are orthonormal: $V^T.V = I$
  - Its *rows* are orthonormal: $V.V^T = I$
  - If k=n(diagon), □=1; else □=0

$$\sum_{j=0}^{N-1} V_{jk} V_{jn} = \delta_{kn} \qquad \begin{matrix} 0 \le k \le N-1 \\ 0 \le n \le N-1 \end{matrix}$$

- Matrix $\mathbf{U}_{M \times N}$
  - When M ≥ N, column-orthogonal: $U^T.U = I$
  - When M < N:
    - W_j for j = M,… , N-1 are all zero
    - Corresponding columns of U are also zero
    - If k=n(diagon), □=1; else □=0

$$\sum_{i=0}^{M-1} U_{ik} U_{in} = \delta_{kn} \qquad \begin{matrix} 0 \le k \le N-1 \\ 0 \le n \le N-1 \end{matrix}$$

SVD Outline

SVD has two major steps:

1) Reduce the initial matrix to *bidiagonal* form using *Householder transformation*
2) *Diagonalize* the resulting matrix using *QR transformation*

$$\underset{\text{Initial matrix}}{\begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{bmatrix}} \rightarrow \underset{\text{Bidiagonal form}}{\begin{bmatrix} x & x & & & \\ & x & x & & \\ & & x & x & \\ & & & x & x \\ & & & & x \end{bmatrix}} \rightarrow \underset{\text{Diagonal form}}{\begin{bmatrix} x & & & & \\ & x & & & \\ & & x & & \\ & & & x & \\ & & & & x \end{bmatrix}}$$

Householder Transformation

A Householder matrix is defined as:

$$\mathbf{H = I - 2ww^T}$$

where **w** is a unit vector with $|w|^2 = 1$

It has the following properties:

$$H = H^T \qquad H^{-1} = H^T \qquad H^2 = I$$

If H is multiplied with another matrix, it results in a new matrix with 0s in a selected row/column, based on the value chosen for w.

Applying Householder

We get a bidiagonal matrix by applying Householder transformation on it successively.

$$P_1 \underset{\mathrm{M}}{\begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{bmatrix}} \rightarrow \underset{\mathrm{M}_1}{\begin{bmatrix} x & x & x & x & x \\ & x & x & x & x \\ & x & x & x & x \\ & x & x & x & x \\ & x & x & x & x \end{bmatrix}} S_1 \rightarrow P_2 \underset{\mathrm{M}_2}{\begin{bmatrix} x & x & & & \\ & x & x & x & x \\ & x & x & x & x \\ & x & x & x & x \\ & x & x & x & x \end{bmatrix}} \rightarrow$$

$$\underset{\mathrm{M}_3}{\begin{bmatrix} x & x & & & \\ & x & x & x & x \\ & & x & x & x \\ & & x & x & x \\ & & x & x & x \end{bmatrix}} S_2 \rightarrow \ldots\ldots \rightarrow P_m \underset{\mathrm{M}_m}{\begin{bmatrix} x & x & & & \\ & x & x & x & x \\ & & x & x & x \\ & & & & x \\ & & & & x \end{bmatrix}} \rightarrow \underset{\mathrm{M}_\mathrm{n}}{\begin{bmatrix} x & x & & & \\ & x & x & & \\ & & x & x & x \\ & & & x & x \\ & & & & x \end{bmatrix}} S_n \rightarrow \underset{\mathrm{B}}{\begin{bmatrix} x & x & & & \\ & x & x & & \\ & & x & x & \\ & & & x & x \\ & & & & x \end{bmatrix}}$$

Householder (cont.)

We can see:

$\mathbf{P_1A = A_1}$ -> $\mathbf{A_1S_1 = A_2}$ -> $\mathbf{P_2A_2 = A_3}$ -> $\mathbf{A_3S_3 = A_4}$ -> …

… -> $\mathbf{A_NS_N = B}$ [If $M > N$, $P_MA_M = B$]

We can also write it in terms of A:

$$\mathbf{A = P_1^TA_1 = P_1^TA_2S_1^T = P_1^TP_2^TA_3S_1^T = P_1^T \dots P_M^TBS_N^T \dots S_1^T = P_1 \dots P_MBS_N \dots S_1}$$

[Because $H^T = H$]

The QR Algorithm

After transforming the initial matrix in bidiagonal form, which results in the following decomposition:

A = PBS with $\mathbf{P = P_1 \ldots P_N}$ and $\mathbf{S = S_N \ldots S_1}$

The next step takes **B** and converts it to the final diagonal form using successive QR transformations.

QR Decomposition

The QR decomposition is defined as:

$$\mathbf{A} = \mathbf{QR}$$

Where, **Q** is an orthogonal matrix and **R** is an upper triangular matrix.

It has the following property:

If $RQ = A_1$, then we can decompose it into $A_1 = Q_1R_1$, $R_1Q_1 = A_2$, …

In practice, after enough decompositions, A_x converges to the desired SVD diagonal matrix **W**.

QR (cont.)

Because Q is *orthogonal*, we can redefine $\mathbf{A_x}$ in terms of $\mathbf{Q_{x-1}}$ and $\mathbf{A_{x-1}}$ only

$\mathbf{R_{x-1}Q_{x-1} = A_x}$ -> $\mathbf{Q_{x-1}R_{x-1}Q_{x-1} = Q_{x-1}A_x}$ -> $\mathbf{Q^T{}_{x-1}Q_{x-1}R_{x-1}Q_{x-1} = A_x}$ -> $\mathbf{Q^T{}_{x-1}A_{x-1}Q_{x-1} = A_x}$

Therefore, $\mathbf{A_{x-1} = Q^T{}_{x-1}A_xQ_{x-1}}$

Starting with $A_0 = A$, we can describe the entire decomposition of W as:

$$\mathbf{A_0 = Q_0{}^TA_1Q_0 = Q_0{}^TQ_1{}^TA_2Q_1Q_0 = \ldots = Q_0{}^TQ_1{}^T \ldots Q_w{}^TWQ_w \ldots Q_1Q_0}$$

SVD Algorithm

Combining all the steps it looks like the following:

- Using Householder transformation: $\mathbf{A} = \mathbf{PBS}$ [Step 1]
- Using QR decomposition: $B = Q_0^T Q_1^T \ldots Q_w^T W Q_w \cdots Q_1 Q_0$ [Step 2]
- Substituting Step 2 into 1:

$$\mathbf{A} = P Q_0^T Q_1^T \ldots Q_w^T W Q_w \cdots Q_1 Q_0 S$$

- With U being derived from: $U = P Q_0^T Q_1^T \ldots Q_w^T$
- And V^T being derived from: $V^T = Q_w \cdots Q_1 Q_0 S$

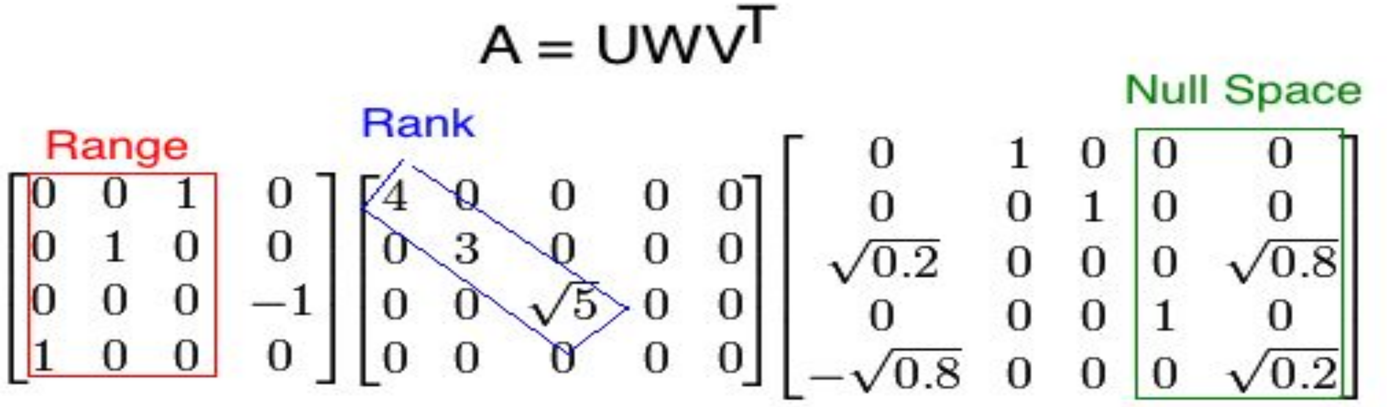
Results in the final SVD: $\mathbf{A} = \mathbf{UWV^T}$

Applications of SVD

- Calculation of the (pseudo) inverse
- Solving a set of homogeneous linear equations, i.e., Ax = b
- The rank of matrix A can be calculated from SVD by the number of nonzero singular values
- The range of a matrix A is the left singular vectors of U corresponding to the non-zero singular values
- The null space of matrix A is the right singular vectors of V corresponding to the zeroed singular values

Applications (cont.)

$$\begin{bmatrix} 1\ 0\ 0\ 0\ 2 \\ 0\ 0\ 3\ 0\ 0 \\ 0\ 0\ 0\ 0\ 0 \\ 0\ 4\ 0\ 0\ 0 \end{bmatrix} =$$



$$A = UWV^T$$

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 4 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & \sqrt{5} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \sqrt{0.2} & 0 & 0 & 0 & \sqrt{0.8} \\ 0 & 0 & 0 & 1 & 0 \\ -\sqrt{0.8} & 0 & 0 & 0 & \sqrt{0.2} \end{bmatrix}$$

Applications (cont.)

- SVD can tell how close a square matrix A is to be singular
- The ratio of the largest singular value to the smallest singular value tells how close a matrix is to be singular:

$$A = U \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \sigma_k \end{bmatrix} V^{\mathsf{T}} \qquad c = \frac{\sigma_1}{\sigma_k}$$

- A is singular if **c** is infinite
- A is ill-conditioned if **c** is too large (machine dependent)

Applications (cont.)

- SVD can be used in data fitting problem
- It can be used in image processing
  - Noise reduction
  - Image smoothing
  - Image compression
- It can be used for noise reduction in signal processing as well
  - If matrix A represents a noisy signal, then we can consider the small singular values as noises and discard those.
  - Thus the rank-k matrix $\mathbf{A}_k$ represents a filtered signal with less noise

References

- Golub & Van Loan – Matrix Computations; 3rd Edition, 1996
- Golub & Kahan – Calculating the Singular Values and Pseudo-Inverse of a Matrix; SIAM Journal for Numerical Analysis; Vol. 2, #2; 1965
- An Example of QR Decomposition, Che-Rung Lee, November 19, 2008
- Press, William H., and William T. Vetterling. Numerical Recipes. Cambridge Univ. Press, 2007.