e-PGPathshala
Subject : Computer Science
Paper: Data Analytics
Module No 30: CS/DA/30 - Dimensionality Reduction: SVD case study & CUR
Quadrant 1 – e-text

1.1 REVIEW OF THE KEY IDEAS of SVD

- The SVD factors A into UΣV T, with r singular values σ1 ≥ . . . ≥ σr > 0.
- The numbers σ2 1, . . ., σ2 r are the nonzero eigenvalues of AAT and ATA.
- The orthonormal columns of U and V are eigenvectors of AAT and ATA.
- Those columns hold orthonormal bases for the four fundamental subspaces of A.
- Those bases diagonalize the matrix: Avi = σiui for i ≤ r. This is AV = UΣ. 6. A = σ1u1vT 1 + · · · + σrurvT r and σ1 is the maximum of the ratio ||Ax|| / ||x||.

1.2 Learning Objectives

- Understand SVD using a Case Study
- Learn the process of CUR approximation technique
- Know the performance of SVD vs CUR

1.3 Case Study:

the query is *gold silver truck* and the "collection" consists of just three "documents":

- d1: *Shipment of gold damaged in a fire.*
- d2: *Delivery of silver arrived in a silver truck.*
- d3: *Shipment of gold arrived in a truck.*

Terms		d1	d2	d3		q
a		1	1	1		0
arrived		0	1	1		0
damaged		1	0	0		0
delivery		0	1	0		0
fire		1	0	0		0
gold	**A =**	1	0	1	**q =**	1
in		1	1	1		0
of		1	1	1		0
shipment		1	0	1		0
silver		0	2	0		1
truck		0	1	1		1

1.3.1 Dimensionality Reduction: Computing U_k, S_k, V_k and V_k^T

$$U = \begin{bmatrix} -0.4201 & 0.0748 & -0.0460 \\ -0.2995 & -0.2001 & 0.4078 \\ -0.1206 & 0.2749 & -0.4538 \\ -0.1576 & -0.3046 & -0.2006 \\ -0.1206 & 0.2749 & -0.4538 \\ -0.2626 & 0.3794 & 0.1547 \\ -0.4201 & 0.0748 & -0.0460 \\ -0.4201 & 0.0748 & -0.0460 \\ -0.2626 & 0.3794 & 0.1547 \\ -0.3151 & -0.6093 & -0.4013 \\ -0.2995 & -0.2001 & 0.4078 \end{bmatrix} \quad S = \begin{bmatrix} 4.0989 & 0.0000 & 0.0000 \\ 0.0000 & 2.3616 & 0.0000 \\ 0.0000 & 0.0000 & 1.2737 \end{bmatrix}$$

$$V = \begin{bmatrix} -0.4945 & 0.6492 & -0.5780 \\ -0.6458 & -0.7194 & -0.2556 \\ -0.5817 & 0.2469 & 0.7750 \end{bmatrix} \quad V^T = \begin{bmatrix} -0.4945 & -0.6458 & -0.5817 \\ 0.6492 & -0.7194 & 0.2469 \\ -0.5780 & -0.2556 & 0.7750 \end{bmatrix}$$

$$k = 2$$

$$U \approx U_k = \begin{bmatrix} -0.4201 & 0.0748 \\ -0.2995 & -0.2001 \\ -0.1206 & 0.2749 \\ -0.1576 & -0.3046 \\ -0.1206 & 0.2749 \\ -0.2626 & 0.3794 \\ -0.4201 & 0.0748 \\ -0.4201 & 0.0748 \\ -0.2626 & 0.3794 \\ -0.3151 & -0.6093 \\ -0.2995 & -0.2001 \end{bmatrix} \quad S \approx S_k = \begin{bmatrix} 4.0989 & 0.0000 \\ 0.0000 & 2.3616 \end{bmatrix}$$

$$V \approx V_k = \begin{bmatrix} -0.4945 & 0.6492 \\ -0.6458 & -0.7194 \\ -0.5817 & 0.2469 \end{bmatrix} \quad V^T \approx V_k^T = \begin{bmatrix} -0.4945 & 0.6458 & -0.5817 \\ 0.6492 & -0.7194 & 0.2469 \end{bmatrix}$$

1.3.2 Incorporating the Query and Ranking the Documents:

$$d = d^T U_k S_k^{-1}$$

$$q = q^T U_k S_k^{-1}$$

$$sim(q, d) = sim(q^T U_k S_k^{-1}, d^T U_k S_k^{-1})$$

d1(-0.4945, 0.6492)
d2(-0.6458, -0.7194)
d3(-0.5817, 0.2469)

$$q = q^T U_k S_k^{-1}$$

$k = 2$

$$q = \begin{bmatrix}0&0&0&0&0&1&0&0&0&1&1\end{bmatrix} \begin{bmatrix} -0.4201 & 0.0748 \\ -0.2995 & -0.2001 \\ -0.1206 & 0.2749 \\ -0.1576 & -0.3046 \\ -0.1206 & 0.2749 \\ -0.2626 & 0.3794 \\ -0.4201 & 0.0748 \\ -0.4201 & 0.0748 \\ -0.2626 & 0.3794 \\ -0.3151 & -0.6093 \\ -0.2995 & -0.2001 \end{bmatrix} \begin{bmatrix} \frac{1}{4.0989} & 0.0000 \\ 0.0000 & \frac{1}{2.3616} \end{bmatrix}$$

$$q = \begin{bmatrix}-0.2140 & -0.1821\end{bmatrix}$$

1.3.3 Cosine similarities

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. The cosine of 0° is 1, and it is less than 1 for any other angle.

$$sim(q, d) = \frac{q \bullet d}{|q||d|}$$

$$sim(q, d_1) = \frac{(-0.2140)(-0.4945) + (-0.1821)(0.6492)}{\sqrt{(-0.2140)^2 + (-0.1821)^2}\sqrt{(-0.4945)^2 + (0.6492)^2}} = -0.0541$$

$$sim(q, d_2) = \frac{(-0.2140)(-0.6458) + (-0.1821)(-0.7194)}{\sqrt{(-0.2140)^2 + (-0.1821)^2}\sqrt{(-0.6458)^2 + (-0.7194)^2}} = 0.9910$$

$$sim(q, d_3) = \frac{(-0.2140)(-0.5817) + (-0.1821)(0.2469)}{\sqrt{(-0.2140)^2 + (-0.1821)^2}\sqrt{(-0.5817)^2 + (0.2469)^2}} = 0.4478$$

Ranking documents in descending order

$$d_2 > d_3 > d_1$$

1.4 CUR Decomposition

In large-data applications, it is normal for the matrix M being decomposed to be very sparse; that is, most entries are 0. For example, a matrix representing many documents (as rows) and the words they contain (as columns) will be sparse, because most words are not present in most documents. Similarly, a matrix of customers and products will be sparse because most people do not buy most products. We cannot deal with dense matrices that have millions or billions of rows and/or columns. However, with SVD, even if M is sparse, U and V will be dense.4 Since Σ is diagonal, it will be sparse, but Σ is usually much smaller than U and V , so its sparseness does not help.

Consider another r another approach to decomposition, called CUR-decomposition. The merit of this approach lies in the fact that if M is sparse, then the two large matrices (called C and R for "columns" and "rows") analogous to U and V in SVD are also sparse. Only the matrix in the middle (analogous to Σ in SVD) is dense, but this matrix is small so the density does not hurt too much. Unlike SVD, which gives an exact decomposition as long as the parameter r is taken to be at least as great as the rank of the matrix M, CUR-decomposition is an approximation no matter how large we make r. There is a theory that guarantees convergence to M as r gets larger, but typically you have to make r so large to get, say within 1% that the method becomes impractical. Nevertheless, decomposition with a relatively small value of r has a good probability of being a useful and accurate decomposition.

Definition of CUR

Let M be a matrix of m rows and n columns. Pick a target number of "concepts" r to be used in the decomposition. A CUR-decomposition of M is a randomly chosen set of r columns of M, which form the m × r matrix C, and a randomly chosen set of r rows of M, which form the r × n matrix R. There is also an r × r matrix U that is constructed from C and R as follows:

1. Let W be the r × r matrix that is the intersection of the chosen columns of C and the chosen rows of R. That is, the element in row i and column j of W is the element of M whose column is the j^{th} column of C and whose row is the i^{th} row of R.

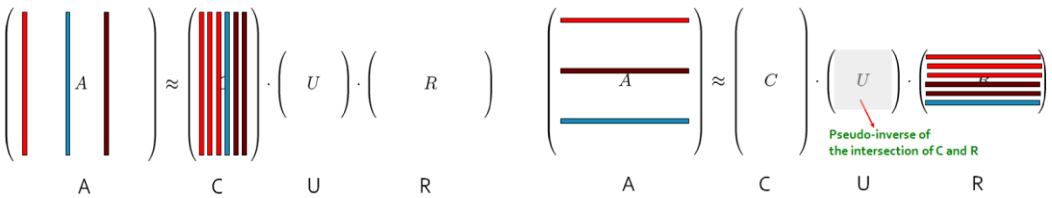
2. Compute the SVD of W; say W = XΣY T.

3. Compute Σ+, the Moore-Penrose pseudo inverse of the diagonal matrix Σ. That is, if the i^{th} diagonal element of Σ is σ 6= 0, then replace it by 1/σ. But if the i^{th} element is 0, leave it as 0.

4. Let U = Y (Σ+) 2XT.

Choosing rows and columns properly recall that the choice of rows and columns is random. However, this choice must be biased so that the more important rows and columns have a better chance of being picked. The measure of importance we must use is the square of the Frobenius norm, that is, the sum of the squares of the elements of the row or column. Let $f = \sum_{i,j} m^2_{ij}$, the square of the Frobenius norm of M. Then each time we select a row, the probability pi with which we select row i is P j m2 ij/f. Each time we select a column, the probability qj with which we select column j is P i $m^2_{ij/f}$

- Goal: Express A as a product of matrices C,U,R Make $\|A-C\cdot U\cdot R\|_F$ small
- "Constraints" on C and R:



How it Works:

- Sampling columns (similarly for rows):

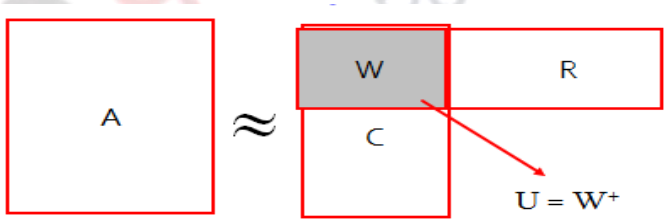
Input: matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, sample size c
Output: $\mathbf{C}_d \in \mathbb{R}^{m \times c}$

1. for $x = 1 : n$ [column distribution]
2. $P(x) = \sum_i \mathbf{A}(i,x)^2 / \sum_{i,j} \mathbf{A}(i,j)^2$
3. for $i = 1 : c$ [sample columns]
4. Pick $j \in 1 : n$ based on distribution $P(x)$
5. Compute $\mathbf{C}_d(:,i) = \mathbf{A}(:,j) / \sqrt{cP(j)}$

- Note this is a randomized algorithm, same column can be sampled more than once

Computing U

- Let W be the "intersection" of sampled columns C and rows R
  - Let SVD of W = X Z Y^T
  - Then: U = W^+ = Y Z^+ X^T
  - Z^+: reciprocals of non-zero singular values: Z^+_{ii} =1/ Z_{ii}
- W^+ is the "pseudoinverse"



Why pseudoinverse works?

A pseudoinverse is a matrix inverse-like object that may be defined for a complex matrix, even if it is not necessarily square. For any given complex matrix, it is possible to define many possible pseudoinverses.

- W = X Z Y then W^{-1} = X^{-1} Z^{-1} Y^{-1}
- Due to orthonomality
- X^{-1}=X^T and Y^{-1}=Y^T

- Since Z is diagonal $Z^{-1} = 1/Z_{ii}$
- Thus, if W is nonsingular, pseudoinverse is the true inverse

1.5 CUR DECOMPOSITION

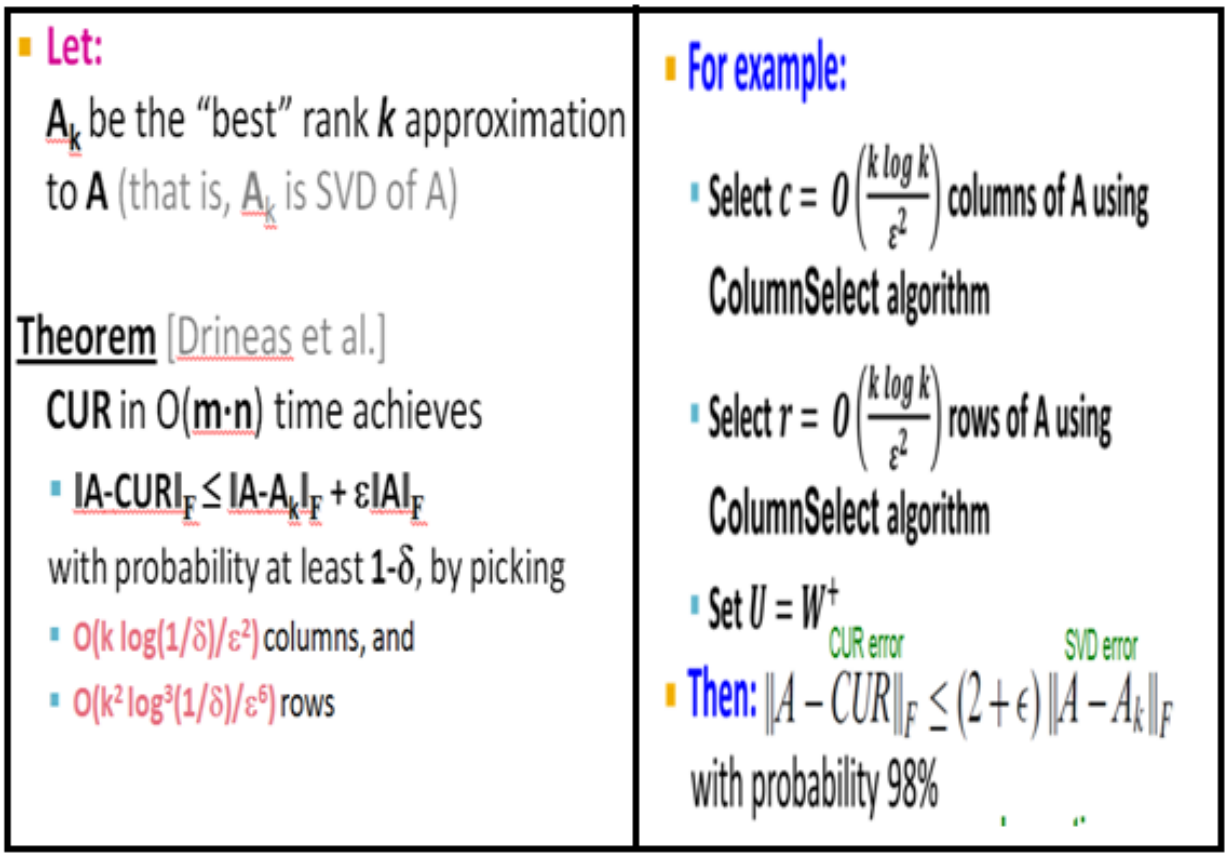
Having selected each of the columns of M, we scale each column by dividing its elements by the square root of the expected number of times this column would be picked. That is, we divide the elements of the jth column of M, if it is selected, by √rqj . The scaled column of M becomes a column of C. Rows of M are selected for R in the analogous way. For each row of R we select from the rows of M, choosing row i with probability pi . Recall pi is the sum of the squares of the elements of the ith row divided by the sum of the squares of all the elements of M. We then scale each chosen row by dividing by √rpi if it is the ith row of M that was chosen.

Pathshala पाठशाला A Gateway to All Post Graduate Courses

The Complete CUR Decomposition

We now have a method to select randomly the three component matrices C, U, and R. Their product will approximate the original matrix M. As we mentioned at the beginning of the discussion, the approximation is only formally guaranteed to be close when very large numbers of rows and columns are selected. However, the intuition is that by selecting rows and columns that tend to have high "importance" (i.e., high Frobenius norm), we are extracting

Provably good approx. to SVD



1.6 Open problems

- **Optimal CUR**: Can we find relative-error CUR algorithms selecting the optimal number of columns and rows, together with a matrix U with optimal rank?
- **Input-sparsity-time CUR**: Can we find relative-error CUR algorithms running in input-sparsity-time (nnz(A) time)?
- **Deterministic CUR**: Can we find relative-error CUR algorithms that are deterministic and run in poly time?

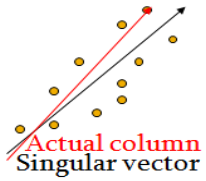
Contributions

- Optimal CUR: First optimal CUR algorithms.

- Input-sparsity-time CUR: First CUR algorithm with running time proportional to the non-zero entries of A.
- Deterministic CUR: First deterministic algorithm for CUR that runs in polynomial time.

Pros & Cons

- **Easy interpretation**
  - Since the basis vectors are actual columns and rows
- **Sparse basis**
  - Since the basis vectors are actual columns and rows
- **Duplicate columns and rows**
  - Columns of large norms will be sampled many times



Solution

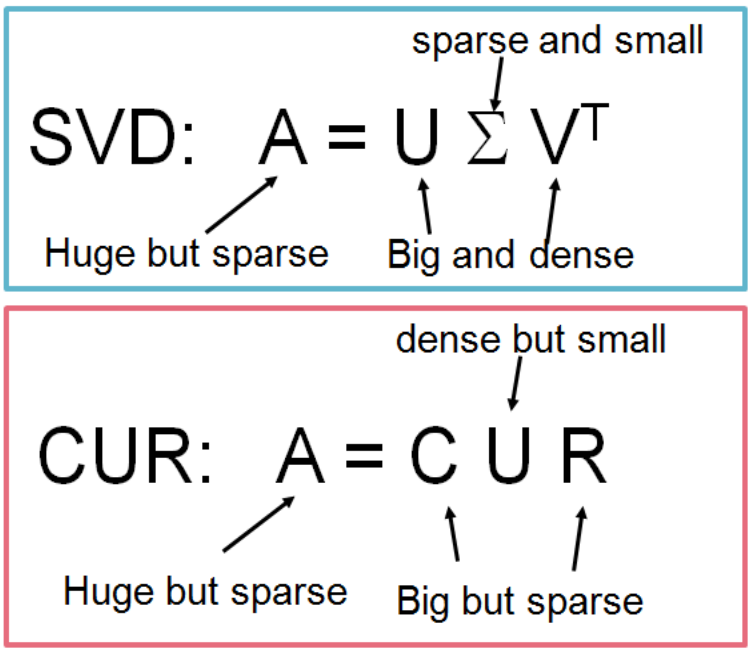
- **If we want to get rid of the duplicates:**
  - Scale (multiply) the columns/rows by the square root of the number of duplicates

Three CUR Data Applications

- **Human Genetics:** DNA SNP Data • Biological Goal: Evaluate intra- and inter-population tag-SNP transferability.
- **Medical Imaging:** Hyper spectral Image Data • Medical Goal: Compress the data, without sacrificing classification quality.
- **Recommendation Systems:** Customer Preference Data • Business Goal: Reconstruct the data, to make high-quality recommendations

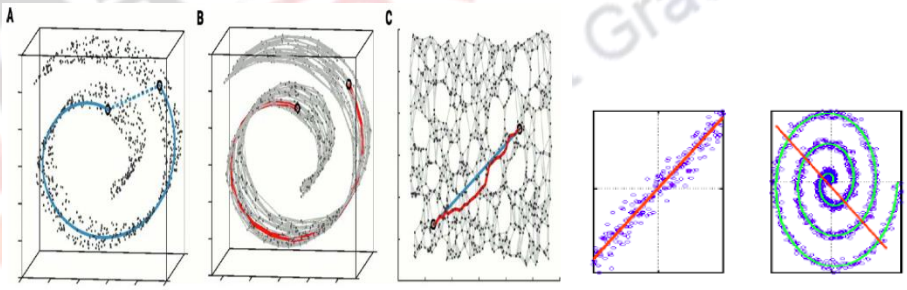
1.7 SVD vs. CUR

SVD has a lesser reconstruction error than CUR but the time taken and space occupied by SVD is far more than that of CUR. So, we can say that both the techniques are useful but the choice o0f the technique that is deem fit for a scenario depends on the characteristics of the data set. For a larger dataset, it is better to go ahead with CUR so as to reduce the space and time overhead. But in the case where every recommendation should be accurate, it's better to use SVD since the reconstruction error is minimum



What about linearity assumption?

- **SVD is limited to linear projections:**
  - Lower-dimensional linear projection that preserves Euclidean distances
- Non-linear methods: **Isomap**
  - Data lies on a nonlinear low-dim curve aka manifold
    - Use the distance as measured along the manifold
- **How?**
    - Build adjacency graph
    - Geodesic distance is graph distance
    - SVD/PCA the graph pair wise distance matrix



Summary:

- Dimensionality reduction involves mathematical approximations
- Dimensionality reductions techniques have their own advantages and drawbacks
- process of CUR approximation technique
- performance of SVD vs CUR