Final Assignment Report

On

Single Value Decomposition vs CUR Decomposition

CS F469: Course Assignment

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI (Rajasthan)

Hyderabad Campus

(Sept 2016)

Mudit Pandey 2014A7PS017H

Sagar Gupta 2014A7PS030H

Introduction

We are implementing TASK-2 which is SVD VS CUR Decomposition

The dataset used: Film Trust Data Set
Link: http://www.librec.net/datasets.html

The link to the code Repository is as follows:
https://github.com/sgmonusg/Recommender-System

Data structures used

We have used the numpy library for python. Thus the 'user vs rating' matrix is stored using numpy matrix data structure and numpy ND array. Using the numpy library allows us to efficiently perform operations on matrices such as matrix multiplication, transpose, etc.

Python dictionaries and lists are also used for manipulating the data.

Implementation

Implementation of SVD

We want decompose the matrix A into such a form:-

$$A_{[m \times n]} = U_{[m \times r]} \Sigma_{[r \times r]} (V_{[n \times r]})^T$$

The method which was followed is as follows :(As given in the slides)

To compute U:
1. We calculate AA^T
2. Next we find the Eigen values and corresponding Eigen vectors of AA^T
3. These Eigenvectors become column vectors in a matrix ordered by the size of the corresponding Eigenvalue.
4. Finally, we have to convert this matrix into an orthogonal matrix which we do by applying the Gram-Schmidt orthonormalization process to the column vectors.

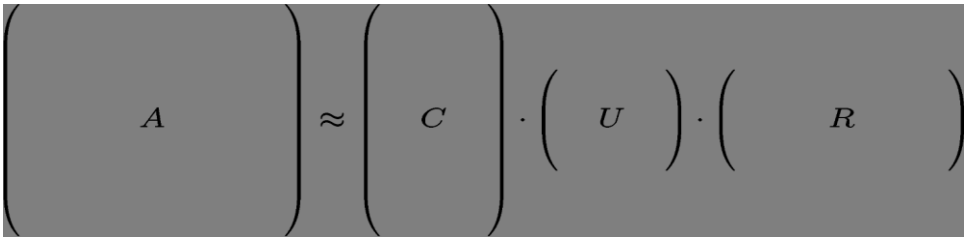
To compute V:
Instead of AA^T we start with A^TA and continue steps as above.

To compute Σ:
Σ is basically a diagonal matrix of the Eigen values obtained above ordered in decreasing order of their magnitude.

Note:-For computing the Eigen vectors and Eigen values, the inbuilt functions in numpy were used.

Implementation of CUR

Given a matrix A,
We wish to decompose the matrix A as follows:



The following method was followed for obtaining the CUR decomposition :(As given in the lecture slides)

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 $P(j)$
5. Compute $\mathbf{C}_d(:,i) = \mathbf{A}(:,j)/\sqrt{cP(j)}$

The above procedure was performed to obtain the C and R matrices.

Note:-In the above procedure, line 4 i.e. picking a column/row (in case of R) was achieved by creating a cumulative probability distribution of row/columns. Next, a random number was generated between 0 and 1. Depending on this randomly generated value, a column/row was chosen.

For Calculating U:

Let **W** be the "intersection" of sampled columns **C** and rows **R**

Let SVD of **W** = **X Z** $\mathbf{Y}^T$

Then: U = $\mathbf{W}^+$ = **Y** $\mathbf{Z}^+$ $\mathbf{X}^T$ which is the Moore-Penrose pseudo-inverse of matrix W

Note:-numpy has an inbuilt function which calculates the Moore-Penrose pseudo inverse of a matrix. This function was used for calculating U.

numpy.linalg.pinv

Observations and Results

Comparison between SVD and CUR

Frobenius Errors:

No. of elements in matrix	SVD Frobenius Error	CUR Frobenius Error
5x5	7.14	5.18
10x10	34.099	11.26
20x20	96.86	28.39
50x50	613.660	93.123
140x140	4804.901	363.418
200x200	9975.530	429.877
230x230	13191.22	454.973
270x270	18099.01	454.28
400x400	39700.084	968.846
500x500	62330.103	1405.79
760x760	144427.13	1279.964
1000x1000	251354.834102	1452.348

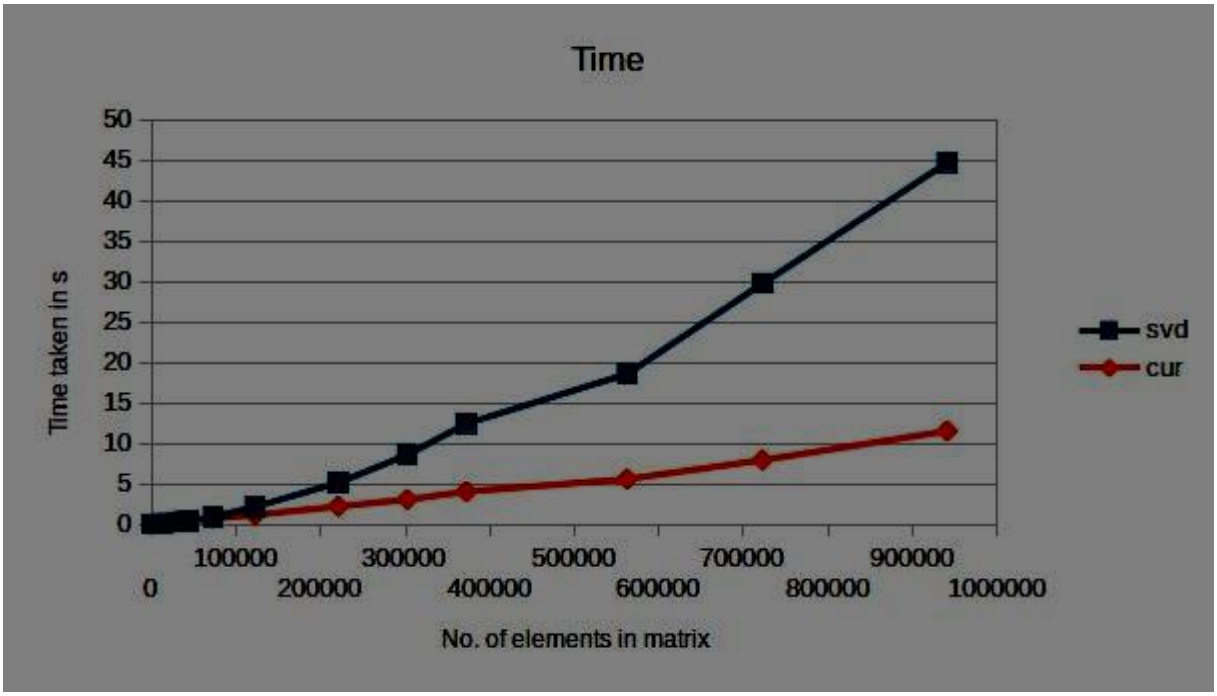
Space:

Since two matrices in the SVD decomposition are big and dense,

As compared to CUR (where in only U matrix is dense, although small), the space occupied by the SVD is more when compared to CUR decomposition of the same matrix. We observe red that for two files containing the SVD decomposition and CUR decomposition of a given matrix, the file containing SVD decomposition was larger in size as compared to the file containing the CUR decomposition.(For a sample test case the size of SVD file was 1.9 KB whereas the CUR file was 1.5KB)

Time:

No. of elements	SVD(in s)	CUR(in s)
10x10	0.000737	0.001464
30x30	0.0017	0.008532
110x110	0.060	0.106
170x170	0.229	0.273
190x190	0.314	0.333
210x210	0.437	0.406
270x270	0.968	0.732
350x350	2.164	1.18
470x470	5.154	2.213
550x550	8.599	3.061
610x610	12.432	4.050
710x710	18.579	5.597
850x850	29.764	7.94
970x970	44.689	11.525



From the above data, we can deduce that SVD performs better than CUR in terms of time for smaller values. However, after a certain threshold, its performance drastically drops as compared to CUR algorithm.

Thus SVD is useful for smaller number of 'concepts' that connect the rows and columns together but is worse off for more number of concepts.

Note: - For CUR, the accuracy increases when the number of sampled rows and the number of sampled columns is increased.
Here we have set the number of sampled rows and the number of sampled columns as half of the number of rows or columns in the original matrix.

Future Research

We would like to create an application based on this technique in the field of Information Retrieval by building a recommender System based on SVD and CUR.

References

- Tensor-CUR Decompositions For Tensor-Based Data by Michael W. Mahoney.
- The BellKor Solution to the Netflix Grand Prize by Yehuda Koren.
- infolab.stanford.edu/~ullman/mmds/ch9.pdf