CS F469 - Information Retrieval

Assignment 2

Design Document

SVD vs. CUR Decomposition

G V Sandeep	2014A7PS106H
Kushagra Agrawal	2014AAPS334H
Snehal Wadhwani	2014A7PS430H

BITS Pilani
Hyderabad Campus

1 Introduction

1.1 Aim

To implement SVD and CUR Matrix decomposition algorithms on a given user-item ratings data and compare the efficiency of the two based on space, running time, accuracy etc.

1.2 Scope

This document describes the implementation details of the SVD and CUR decomposititon and also attempts to compare the two based on the running time, accuracy etc.

1.3 Definitions, Acronyms and Abbreviations

- SVD (Singular Value Decomposition) - A matrix approximation which gives a set of 3 matrices that when multiplied together closely approximate the given matrix.
- CUR - A matrix approximation which gives a set of 3 matrices that when multiplied together closely approximate the given matrix and uses SVD decomposition in one of its intermediate steps.

2 Design Overview

2.1 Problem Statement

This task is aimed at implementing both the SVD and the CUR Matrix Decomposition Algorithms and comparing the efficiency of both these approaches.

2.2 Technologies Used

- Programming Languages : Python v2.7.2
- Data Set : a text file

2.3 Data Description

- Corpus : Film Trust
- File Name : ratings.txt
- URL : `http://www.librec.net/datasets.html`
- 35497 item ratings with format: userId, movieId, movie rating
- Dimension : 1508 x 2071

This data is then stored in the `numpy.matrix` datatype.

2.4 Application Operation

SVD

The function takes a matrix and then initially calls the `eigen_pairs()` function. This function then calculates the eigenvalues and corresponding eigenvectors. The square root of the eigenvalues form the diagonal elements of the sigma matrix. The corresponding normalized eigenvectors of *ratings*ratings*^{T} form the columns of the U matrix. The corresponding normalized eigenvectors of *ratings*^{T}**ratings* form the rows of the V matrix. The multiplication of U,sigma and V gives us back an approximation of the original matrix.

There are two variants of the SVD implementation submitted in the assignment. One which returns the SVD with the eigenvalues and eigenvectors returned by the `numpy.linalg.eig()` function and other is a version which iterates through all the possible eigenvectors which keep the U and V matrices column and row orthonormal respectively. After this we return the matrix with least possible reconstruction error.

CUR

In this method we choose some (i.e. equal to rank of the ratings matrix, which in this case is 556) random columns and rows based on the probability function which we calculate by dividing the length of the row by the total length of the matrix (sum of squares). The rows are also normalized so as to compensate the fact that they are sometimes chosen repeatedly. Then we form the *intersection matrix (say, W)* on the basis of the selected random rows and columns and take its *Moore-Penrose Psuedoinverse (say, W+)* which is found in the following manner :

$$W = XZY^T \tag{1}$$

$$U = W+ = YZ + X^T \tag{2}$$

Where Z+ is reciprocals of non-zero singular values (i.e. $Z+_{ii} = 1/Z_{ii}$) We then multiply C,U (W+) and R matrices to obtain an approximation of the original matrix.
A 'seed' has been given to the randomized algorithm so that the output remains constant for comparison purposes.

2.5 Efficiency Comparison

	SVD No Iteration	**SVD Multiple Iterations**	**CUR**
Running Time	66.1 seconds	18350 seconds	24 seconds
Frobenius Error	1067.3	575.1	816 (seed dependant)

Observations

- SVD, not being dependant on a randomized algorithm gives a decomposition with a constant, definite accuracy which is optimized to the highest value.
- CUR takes a much lesser running time since the only time consuming step in CUR is the calculation of psuedo-inverse of a small, dense matrix (achieved using SVD) whereas SVD is rather time consuming because of calculation of eigenvalues and eigenvectors of a large matrix (ratings).
- Based on observations of data structures, it can be said that CUR is more optimized regarding memory usage as CUR results into a decomposition of two large, sparse matrices (C and R) and one small dense matrix (W or W+) whereas SVD gives two large, dense matrices (U and V) and one small sparse matrix (sigma).

2.6 Depecdencies

- Numpy : A python package for data storage and matrix operations.
- Time : A python package for calculating the running time of the program.