CS6220: Data-sparse Matrix Computations
Lecture 14: CUR factorization and leverage scores

Lecturer: Anil Damle
Scribers: Kun Dong, Johan Björck and Sujit Rao

October 17, 2017

1 The CUR Factorization

Last time, we decided to expand our view from standard factorizations to the CUR factorization, which is more interpretable. If the matrix A comes from data, then the singular vectors associated to the largest singular value can be interpreted as telling us what data point or feature is most important. But this is sometimes a linear combination of features or data points, which we can't necessarily interpret as being associated to an actual feature or data point.

Thus we would like to build a factorization where $A \approx CUR$, where C is a subset of the columns of A, R is a subset of the rows of A, and U is just some matrix. In contrast, in the SVD we choose orthogonal matrices for C and R and force U to be diagonal. This would give more interpretable results on what features or rows in the data are most significant, as opposed to what linear combinations are most significant.

In order to build the factorization, we need to figure out which columns of A are most dominant in a rank-k approximation of A. Let A_k be the best rank$-k$ factorization of A.

How do we pick the columns and rows? We will first discuss columns, and apply the same algorithm for rows. We would like a small number of columns with good approximation properties. Let $A = U\Sigma V^T$ be the SVD. Let u_i and v_i be the columns of U and V. We can write a column in terms of the SVD as

$$A_{:,j} = \sum_{i=1}^{n} (\sigma_i u_i) v_i(j).$$

If we look at the rank-k approximation, we care about columns which have large contributions from $\sigma_1, \ldots, \sigma_k$. Thus

$$A_{:,j} \approx \sum_{i=1}^{k} (\sigma_i u_i) v_i(j).$$

We want columns of A which correlate well with the top k left singular vectors. To measure this, we will introduce the notion of *leverage scores* of the matrix A.

2 Leverage Scores and Column Selection

We define the **normalized leverage scores** of matrix A as

$$\pi_j = \frac{1}{k}\sum_{i=1}^{k} v_i(j)^2 \qquad \text{for } j = 1, 2 \ldots n$$

Note that the normalized leverage scores constitutes a probability distribution, i.e. we have $\sum_j \pi_j = 1$. We also note that π_j is the j:th diagonal element of $V_k V_k^T$. We present the method `ColumnSelect` below. At a high level it is a two-stage procedure, the leverage scores are first calculated and columns are then selected randomly based on the probability distribution defined by the leverage scores.

`ColumnSelect` Algorithm

Starting with input matrix $A \in \mathbb{R}^{m \times n}$, rank parameter k, and error parameter ϵ.

1. Compute $V_1 \ldots V_k$ and leverage scores. Let C denote the set of columns we choose.
2. Add column j to C with probability $\min(1, c\pi_j)$, where c is a constant of order $\mathcal{O}\left(\frac{k}{\epsilon^2} \log k\right)$
3. Return C.

The method returns C' columns, where $\mathbb{E}[C'] \leq c$. It is possible to prove that we with high probability have

$$\|A - P_C A\|_f \leq \left(1 + \frac{\epsilon}{2}\right) \|A - A_k\|_F \tag{1}$$

Here P_C is the projection operator unto the space spanned by columns C.

3 CUR Algorithm

The column selection algorithm let us represent A by a subset of its columns. The main algorithm builds upon it to obtain such representation in terms of both columns and rows simultaneously. `AlgorithmCUR` performs the following steps,

CUR Matrix Decomposition Algorithm

Starting with input matrix $A \in \mathbb{R}^{m \times n}$, rank parameter k, and error parameter ϵ.

1. Run `ColumnSelect` on A with $c = O(k \log k/\epsilon^2)$ to choose columns of A and construct the matrix C.
2. Run `ColumnSelect` on A^T with $r = O(k \log k/\epsilon^2)$ to choose rows of A (columns of A^T) and construct the matrix R.
3. Compute $U = C^\dagger A R^\dagger$, where $X^\dagger$ denotes the Moore-Penrose pseudoinverse of a matrix X.

For the output C, U, R of this algorithm, we can prove that with probability at least 98%

$$\|A - CUR\|_F \leq (2 + \epsilon)\|A - A_k\|_F$$

By the way U is constructed,

$$\|A - CUR\|_F = \|A - CC^\dagger A R^\dagger R\|_F \tag{2}$$

Applying the triangle inequality,

$$\begin{aligned}
\|A - CUR\|_F &\leq \|A - CC^\dagger A\|_F + \|CC^\dagger A - CC^\dagger A R^\dagger R\|_F \\
&\leq \|A - CC^\dagger A\|_F + \|A - A R^\dagger R\|_F \\
&= \|A - P_C A\|_F + \|A - A P_R\|_F
\end{aligned}$$

The second inequality follows form the fact that $I - CC^\dagger$ is a projection operator, thus does not increase the Frobenius norm. Since C and U are selected by `ColumnSelect` on A and A^T respectively, we can use the error bound from equation (1) to get equation (2). Because `AlgorithmCUR` consists of two applications of `ColumnSelect` the failure probability is at most twice of the 1% of `ColumnSelect`, resulting in the 98% success rate.

References

[MD09] Michael W Mahoney and Petros Drineas. Cur matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences*, 106(3):697–702, 2009.