5 CUR Decomposition

Most matrix approximation techniques (including SVD) provide the basis vectors as linear combinations of data features, e.g. in a term-document matrix a basis vector could be $[(3/2)\text{ job} - (2/7)\text{ society} + \cdots + (1/\sqrt{10})\text{ salary}]$. Typically, these vectors are not understandable and particularly informative. In addition, analysts spend vast amount of time applying decomposition techniques to analyze the data and comprehend it better. Therefore, it is highly preferable and sometimes necessary to obtain an *interpretable* compact representation of data.

In this lecture we talk about matrix approximation techniques that are explicitly expressed in terms of a small number of columns and rows of the input matrix, and thereby not only are more amenable to interpretation, but also preserves additional structure of data such as sparsity or nonnegativity.

All CUR techniques decompose the input matrix $A \in \mathbb{R}^{n\times d}$ as a set of three matrices $C \in \mathbb{R}^{n\times c}, U \in \mathbb{R}^{c\times d}$ and $R \in \mathbb{R}^{r\times d}$ that when multiplied together, closely approximate A. The CUR matrix approximation is not unique and there are multiple algorithms for computing one.

5.1 LinearTimeCUR

First CUR algorithm named as *LinearTimeCUR* (LTCUR), is due to Drineas, Kannan and Mahoney[3]. This method, presented in algorithm 5.1.1, consists of three main steps: (1) first it constructs a matrix $C \in \mathbb{R}^{n\times c}$ by randomly sampling $c = O(k/\varepsilon^4)$ (or $c = O(1/\varepsilon^4)$ for spectral norm) columns of A in c independent identical trials; in each trial column $A_{:,j}$ is sampled with probability $p_j = \|A_{:,j}\|^2/\|A\|_F^2$ and if it is picked, it is rescaled by $1/\sqrt{cp_j}$, (2) then it forms a matrix $R \in \mathbb{R}^{r\times d}$ by independently sampling $r = O(k/\varepsilon^2)$ rows of A proportional to their squared norm; in each trial row $A_{i,:}$ is sampled with porbability $q_i = \|A_{i,:}\|^2/\|A\|_F^2$, and if it is selected it is rescaled by $1/\sqrt{rq_i}$ and (3) finally it constructs matrix $U \in \mathbb{R}^{c\times r}$ as the intersection of C and R and properly rescales such that $A = CUR$.

Algorithm 5.1.1 LinearTimeCUR

Input: $A \in \mathbb{R}^{n\times d},\ 1 \le c \le d,\ 1 \le r \le n,\ 1 \le k \le \min(n,d)$
Output: $C \in \mathbb{R}^{n\times c}, U \in \mathbb{R}^{c\times r}, R \in \mathbb{R}^{r\times d}$
for $t = 1$ to c **do**
  Pick $j \in \{1,\cdots,d\}$ with probability $p_j = \|A_{:,j}\|^2/\|A\|_F^2$
  Set $C_{:,t} = A_{:,j}/\sqrt{cp_j}$
Set $k = \min(k, \mathsf{rank}(C^TC))$
for $t = 1$ to r **do**
  Pick $i \in \{1,\cdots,n\}$ with probability $q_i = \|A_{i,:}\|^2/\|A\|_F^2$
  Set $R_{t,:} = A_{i,:}/\sqrt{rq_i}$
  Set $\Psi_{t,:} = C_{i,:}/\sqrt{rq_i}$
Let $U = ([C^TC]_k)^{-1}\Psi^T$
return C, U, R

Note this method is inspired by LinearTimeSVD algorithm of same authors[2], which is a column sampling method. They showed setting $c \ge 64k\mu_c^2/\varepsilon^4$ and $r \ge 4k/\delta_r^2\varepsilon^2$ where $\mu_c = 1 + \sqrt{8\log(1/\delta_c)}$ and $0 \le \varepsilon, \delta_r, \delta_c \le 1$, LTCUR achieves Frobenius error bound

$$\|A - CUR\|_F \le \|A - A_k\|_F + \varepsilon\|A\|_F$$

and setting $c \geq 64\mu_c^2/\varepsilon^4$ and $r \geq 4k/\delta_r^2\varepsilon^2$ it obtains spectral error bound as

$$\|A - CUR\|_2 \leq \|A - A_k\|_2 + \varepsilon\|A\|_F$$

with probability $1 - (\delta_r + \delta_c)$.

5.1.1 Error Analysis

Let $S_C \in \mathbb{R}^{n\times c}$ and $S_R \in \mathbb{R}^{r\times d}$ denote column sampling and row sampling matrices, i.e. they are zero-one matrices such that if in t-th trial, i-th row (similarly j-th column) of A is chosen then $(S_R)_{t,i} = 1$ (similarly $(S_C)_{j,t} = 1$), otherwise it is zero. Also consider rescaling diagonal matrices $D_C \in \mathbb{R}^{c\times c}$ and $D_R \in \mathbb{R}^{r\times r}$ such that $(D_R)_{t,t} = 1/\sqrt{rp_i}$ $\left(\text{similarly } (D_C)_{t,t} = 1/\sqrt{cq_j}\right)$ if in t-th trial, i-th row (similarly j-th column) of A is chosen. Using these notations, we can write R, C and U as

$$C = AS_CD_C \qquad \text{and} \qquad R = D_RS_RA \qquad \text{and} \qquad U = ([C^TC]_k)^{-1}C^T(D_RS_R)^T$$

Note that if we consider SVD decomposition of C as $C = H\Sigma Y^T$, due to LTSVD algorithm of [2], $H_kH_k^TA$ is already an approximation to A_k. In LTCUR, we show an approximation to $H_kH_k^TA$ is still a "good enough" approximation to A_k. Having this intuition and using triangle inequality we decompose the target error bound as

$$\|A - CUR\|_\zeta \leq \|A - H_kH_k^TA\|_\zeta + \|H_kH_k^TA - CUR\|_\zeta$$

Where $\zeta \in \{2, F\}$. Based on [2], Frobenius and spectral norm of first term are bounded as

$$\|A - H_kH_K^TA\|_F^2 \leq \|A - A_k\|_F^2 + 2\sqrt{k}\|AA^T - CC^T\|_F$$

$$\|A - H_kH_K^TA\|_2^2 \leq \|A - A_k\|_2^2 + 2\|AA^T - CC^T\|_2$$

For the second term, notice we can write CUR as

$$\begin{aligned} CUR &= [H\Sigma Y^T]\left[(Y_k\Sigma_k^2Y_k^T)^{-1}(Y\Sigma H^T)(D_RS_R)^T\right][D_RS_RA] \\ &= [H\Sigma Y^T]\left[Y_k\Sigma_k^{-1}H_k^T(D_RS_R)^T\right][D_RS_RA] \\ &= H_kH_k^T(D_RS_R)^TD_RS_RA \end{aligned}$$

Therefore

$$\begin{aligned} \|H_kH_k^TA - CUR\|_\zeta &= \|H_kH_k^TA - H_kH_k^T(D_RS_R)^TD_RS_RA\|_\zeta \\ &\leq \|H_kH_k^TA - H_kH_k^T(D_RS_R)^TD_RS_RA\|_F \\ &= \|H_k^TA - H_k^T(D_RS_R)^TD_RS_RA\|_F \end{aligned}$$

Notice that $H_k^T(D_RS_R)^T$ is a column subset of H_k^T sampled and rescaled by D_RS_R, and D_RS_RA is the corresponding row subset of A. Below we state a theorem used in *BasicMatrixMultiplication* algorithm [1] that helps us bound last term.

Theorem 5.1.1. *[1] Consider $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times p}$, $c \in \mathbb{Z}^+$ such that $1 \leq c \leq n$ and $\{p_i\}_{i=1}^n$ are such that $p_i \geq 0$ and $\sum_{i=1}^n p_i = 1$. Let C be a sample of c columns of A sampled with probabilities $\{p_i\}_{i=1}^n$, and let R contains corresponding c rows of B. If the probabilities are of form*

$$p_i = \frac{\|B_{i,:}\|^2}{\|B\|_F^2} \quad \textit{or} \quad p_i = \frac{\|A_{:,i}\|\|B_{i,:}\|}{\sum_{j=1}^n \|A_{:,j}\|\|B_{j,:}\|}$$

then

$$\boldsymbol{E}[\|AB - CR\|_F] \leq \frac{1}{\sqrt{c}}\|A\|_F\|B\|_F$$

Since matrix S_R is constructed using sampling probabilities $\|A_{i,:}\|^2/\|A\|_F^2$, we can say

$$\mathbf{E}\left[\|H_k^T A - H_k^T (D_R S_R)^T D_R S_R A\|_F\right] \leq \|H_k\|_F \|A\|_F/\sqrt{r} = \sqrt{k/r}\|A\|_F$$

Now using Markov's inequality with $\delta_r \in (0,1)$, we get

$$\mathbf{Pr}\left[\left\|H_k^T A - H_k^T (D_R S_R)^T D_R S_R A\right\|_F \geq \frac{1}{\delta_r}\sqrt{\frac{k}{r}}\|A\|_F\right] \leq \delta_r$$

Putting all results together we get Frobenius error bound as

$$\begin{aligned}
\|A - CUR\|_F &\leq \|A - A_k\|_F + (4k)^{1/4}\|AA^T - CC^T\|_F^{1/2} + \frac{1}{\delta_r}\sqrt{\frac{k}{r}}\|A\|_F \\
&\leq \|A - A_k\|_F + (4k/c)^{1/4}\frac{1}{\sqrt{\delta_c}}\|A\|_F + \frac{1}{\delta_r}\sqrt{\frac{k}{r}}\|A\|_F \\
&= \|A - A_k\|_F + \left((4k/c)^{1/4}\frac{1}{\sqrt{\delta_c}} + \frac{1}{\delta_r}\sqrt{\frac{k}{r}}\right)\|A\|_F \\
&= \|A - A_k\|_F + O(\varepsilon)\|A\|_F && \text{setting } c = k/\varepsilon^4 \text{ and } r = k/\varepsilon^2
\end{aligned}$$

Note that second transition comes from the fact that column sampling probabilities satisfy condition of theorem 5.1.1 (consider $B = A^T$), thus we can say $\mathbf{E}\left[\|AA^T - CC^T\|_F\right] \leq \frac{1}{\sqrt{c}}\|A\|_F^2$, and using Markov's inequality with $\delta_c \in (0,1)$ we get

$$\mathbf{Pr}\left[\|AA^T - CC^T\|_F \geq \frac{1}{\delta_c}\frac{1}{\sqrt{c}}\|A\|_F^2\right] \leq \delta_c$$

Now, we bound spectral error as following

$$\begin{aligned}
\|A - CUR\|_2 &\leq \|A - A_k\|_2 + \sqrt{2}\|AA^T - CC^T\|_2^{1/2} + \frac{1}{\delta_r}\sqrt{\frac{k}{r}}\|A\|_F \\
&\leq \|A - A_k\|_2 + \sqrt{2}\|AA^T - CC^T\|_F^{1/2} + \frac{1}{\delta_r}\sqrt{\frac{k}{r}}\|A\|_F \\
&\leq \|A - A_k\|_2 + \sqrt{\frac{2}{\delta_c}}(\frac{1}{c})^{1/4}\|A\|_F + \frac{1}{\delta_r}\sqrt{\frac{k}{r}}\|A\|_F \\
&= \|A - A_k\|_2 + (\sqrt{\frac{2}{\delta_c}}(\frac{1}{c})^{1/4} + \frac{1}{\delta_r}\sqrt{\frac{k}{r}})\ \|A\|_F \\
&= \|A - A_k\|_2 + O(\varepsilon)\|A\|_F && \text{setting } c = 1/\varepsilon^4 \text{ and } r = k/\varepsilon^2
\end{aligned}$$

Note both bounds hold with high probability $1 - \delta_r - \delta_c$.

5.1.2 Space and Run Time Analysis

In LTCUR, clearly sampling probabilities p_i and q_j can be computed in one pass, $O(nd)$ time (for reading data) and using $O(c + r)$ space. A second pass is needed to compute matrices C, R and Ψ; this will take $O(nd + dc + nr)$ time (since sampling is with replacement and each d column (n rows) of A can be selected for all c columns (r rows) of C (R)), also it needs $O(nc + dr + cr)$ additional space to store these three matrices. Having C in hand, one can compute C^TC and get rank k of that in $O(nc^2 + c^2k)$ time and $O(c^2)$ space. Therefore the total run time of algorithm 5.1.1 would be $O(nd + dc + nr + nc^2)$, and the space usage would be $O(nc + dr + cr)$.

5.2 LeverageScoreCUR

In this section, we describe LeverageScoreCUR (LSCUR) algorithm[4] which constructs matrices C and R based on "column leverage scores" and "row leverage scores" of input matrix A.

Recall that if $A = USV^T$ is SVD decomposition of $A \in \mathbb{R}^{n\times d}$, then one can write j-th column of A as linear combination of all left singular vectors (column of U) rescaled by corresponding singular values and entries of j-th row of right singular vector matrix (i.e. V); in other words

$$A_{:,j} = USV^T_{j,:} = \sum_{i=1}^{n} S_{i,i}U_{:,i}V_{j,i}$$

If we truncate this summation to first $k < n$ left singular vectors and values, we can approximate $A_{:,j}$ as

$$A_{:,j} \approx \sum_{i=1}^{k} S_{i,i}U_{:,i}V_{j,i}$$

We define the leverage score of j-th column as $score(A_{:,j}) = \sum_{i=1}^{k} V^2_{j,i}$. Note that sum of leverage scores for all column of A is exaclty k:

$$\sum_{j=1}^{d} score(A_{:,j}) = \sum_{j=1}^{d}\sum_{i=1}^{k} V^2_{j,i} = \sum_{i=1}^{k}\sum_{j=1}^{d} V^2_{j,i} = \sum_{i=1}^{k} \|V_{:,i}\|^2 = k$$

Therefore in order to define sampling probabilities we normalize leverage scores as $p_j = \frac{1}{k}\sum_{i=1}^{k} V^2_{j,i}$; with the normalization $\{p_j\}_{j=1}^{d}$ form probability distribution over columns of A as $p_j \geq 0$ and $\sum_{j=1}^{d} p_j = 1$. Using these sampling probabilities, authors of [4] proposed the *ColumnSelect* algorithm (depicted in algorithm 5.2.1) that samples $c = O(k \log k/\varepsilon^2)$ columns, forms a matrix $C \in \mathbb{R}^{n\times c}$ and achieve error bound $\|A - \pi_C(A)\|_F \leq (1 + \varepsilon/2)\|A - A_k\|_F$. The LSCUR which is described in algorithm 5.2.2 uses *ColumnSelect* algorithm on both A and A^T to achieve relative error bound $\|A - CUR\|_F \leq (2 + \varepsilon)\|A - A_k\|_F$.

Algorithm 5.2.1 ColumnSelect

Input: $A \in \mathbb{R}^{n\times d},\ 1 \leq c \leq d; 1 \leq k \leq \min(n, d)$
Output: $C \in \mathbb{R}^{n\times c}$
Compute svd of A as $A = U\Sigma V^T$ and define normalized leverage scores as $p_i = \frac{1}{k}\sum_{j=1}^{k} V^2_{i,j}$
for $t = 1$ to c **do**
  Pick $j \in \{1, \cdots, d\}$ with probability $\min(1, cp_j)$
  Set $C_{:,t} = A_{:,j}/\sqrt{c^2 p_j}$
return C

Algorithm 5.2.2 LeverageScoreCUR

Input: $A \in \mathbb{R}^{n\times d},\ 1 \leq c \leq d,\ 1 \leq r \leq n,\ 1 \leq k \leq \min(n, d)$
Output: $C \in \mathbb{R}^{n\times c}, U \in \mathbb{R}^{c\times r}, R \in \mathbb{R}^{r\times d}$
Run *ColumnSelect* on A with $c = O(k \log k/\varepsilon^2)$ to construct the matrix C.
Run *ColumnSelect* on A^T with $r = O(k \log k/\varepsilon^2)$ to construct the matrix R.
Set matrix U as $U = C^+AR^+$.
return C, U, R

	# PASSES	RUN TIME	SPACE USAGE	ERROR BOUND
LTCUR[3]	2	$O(n(d+k/\varepsilon^2+1/\varepsilon^8)+d/\varepsilon^4)$ $O(n(d+k/\varepsilon^2+k^2/\varepsilon^8)+dk/\varepsilon^4)$	$O(n/\varepsilon^4+dk/\varepsilon^2)$	$\|A-CUR\|_2 \leq OPT_2+\varepsilon\|A\|_F$ $\|A-CUR\|_F \leq OPT_F+\varepsilon\|A\|_F$
LSCUR[4]	2	$O(k\log k/\varepsilon^2 n^2 d + nd^2 k\log k/\varepsilon^2)$	$O(nd)$	$\|A-CUR\|_F \leq (2+\varepsilon)\,OPT_F$

Table 5.1: Comparing different CUR algorithms. We define $OPT_2 = \|A-A_k\|_2$ and $OPT_F = \|A-A_k\|_F$.

5.2.1 Error Analysis

We use triangle inequality and error bound of *ColumnSelect* algorithm to decompose $\|A-CUR\|_F$ as

$$\begin{aligned}
\|A-CC^+AR^+R\|_F &= \|A-CC^+AR^+R\|_F \\
&\leq \|A-CC^+A\|_F + \|CC^+A-CC^+AR^+R\|_F \\
&\leq \|A-CC^+A\|_F + \|A-AR^+R\|_F \\
&\leq \|A-\pi_C A\|_F + \|A-\pi_R A\|_F \\
&\leq (1+\varepsilon/2)\|A\|_F + (1+\varepsilon/2)\|A\|_F \\
&= (2+\varepsilon)\|A\|_F
\end{aligned}$$

5.2.2 Space and Run Time Analysis

Since LSCUR needs to take SVD of input matrix, it loads the whole matrix into memory, which takes $O(nd)$ time to read data, and $O(nd)$ space to store it. Taking SVD of it costs $O(nd^2)$ additional time. In another pass over data, LSCUR constructs matrices C and R, that takes $O(cd+rn)$ additional time and space. Finally computing matrix U requires $O(cn^2d+nd^2r)$ time and $O(cr)$ space. Overall LSCUR runs in $O(cn^2d+nd^2r)$ time and $O(nd)$ space.

Table 5.2.2 summarizes run time, space usage and error bounds of above mentioned CUR methods.

Bibliography

[1] Petros Drineas, Ravi Kannan, and Michael W Mahoney. Fast monte carlo algorithms for matrices i: Approximating matrix multiplication. *SIAM Journal on Computing*, 36(1):132–157, 2006.

[2] Petros Drineas, Ravi Kannan, and Michael W Mahoney. Fast monte carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix. *SIAM Journal on Computing*, 36(1):158–183, 2006.

[3] Petros Drineas, Ravi Kannan, and Michael W Mahoney. Fast monte carlo algorithms for matrices iii: Computing a compressed approximate matrix decomposition. *SIAM Journal on Computing*, 36(1):184–206, 2006.

[4] 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.