ECS231

Low-rank approximation – revisited

(Introduction to Randomized Algorithms)

May 23, 2019

Outline

1. Review: low-rank approximation
2. Prototype randomized SVD algorithm
3. Accelerated randomized SVD algorithms
4. CUR decomposition

Review: optimak rank-k approximation

- The SVD of an $m \times n$ matrix A is defined by

$$A = U\Sigma V^T,$$

where U and V are $m \times m$ and $n \times n$ orthogonal matrices, respectively, $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots)$ and $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$.
- Computational cost $O(mn^2)$, assuming $m \geq n$.
- Rank-k truncated SVD of A:

$$A_k = U_{(:,1:k)} \cdot \Sigma_{(1:k,1:k)} \cdot V^T_{(:,1:k)}$$

Review: optimak rank-k approximation

- **Eckart-Young theorem.**

$$\min_{\text{rank}(B)\le k} \|A-B\|_2 = \|A-A_k\|_2 = \sigma_{k+1}$$

$$\min_{\text{rank}(B)\le k} \|A-B\|_F = \|A-A_k\|_F = \left(\sum_{j=k+1}^{n} \sigma_{k+1}^2\right)^{1/2}$$

- **Theorem A.**

$$\min_{\text{rank}(B)\le k} \|A-QB\|_F^2 = \|A-QB_k\|_F^2,$$

where Q is an $m\times p$ orthogonal matrix, and B_k is the rank-k truncated SVD of Q^TA, and $1\le k\le p$.

Remark: Given $m \times n$ matrix $A = (a_{ij})$, the Frobineous norm of A is defined by $\|A\|_F = \left(\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2\right)^{1/2} = (\text{trace}(A^TA))^{1/2}$.

Prototype randomized SVD algorithm

By Theorem A, we immediately have the following a prototype randomized SVD (low-rank approximation) algorithm:

- Input: $m \times n$ matrix A with $m \geq n$, integers $k > 0$ and $k < \ell < n$
- Steps:
  1. Draw a random $n \times \ell$ test matrix Ω.
  2. Compute $Y = A\Omega$ – "*sketching*".
  3. Compute an orthonormal basis Q of Y.
  4. Compute $\ell \times n$ matrix $B = Q^T A$.
  5. Compute B_k = the rank-truncated SVD of B.
  6. Compute $\widehat{A}_k = QB_k$.
- Output: $\widehat{A}_k$, a rank-k approximation of A.

Prototype randomized SVD algorithm

MATLAB demo code: `randsvd.m`

```
>> ...
>> Omega = randn(n,l);
>> C = A*Omega;
>> Q = orth(C);
>> [Ua,Sa,Va] = svd(Q'*A);
>> Ak = (Q*Ua(:,1:k))*Sa(1:k,1:k)*Va(:,1:k)';
>> ...
```

Prototype randomized SVD algorithm

- **Theorem.** With proper choice of an $m \times O(k/\epsilon)$ sketch Ω,

$$\min_{\mathsf{rank}(X) \leq k} \|A - QX|_F^2 \leq (1+\epsilon)\|A - A_k\|_2^2$$

  holds with high probability.
- *Reading: Halko et al, SIAM Rev., 53:217-288, 2011.*

Accelerated randomized SVD algorithm 1

The basic subspace iteration

- Input: $m \times n$ matrix A with $m \geq n$, $n \times \ell$ starting matrix Ω and positive integers k, ℓ, q and $n > \ell \geq k$.
- Steps:
  1. Compute $Y = (AA^T)^q A\Omega$.
  2. Compute an orthonormal basis Q of Y.
  3. Compute $\ell \times n$ matrix $B = Q^T A$.
  4. Compute B_k = the rank-truncated SVD of B.
  5. Compute $\widehat{A}_k = QB_k$.
- Output: $\widehat{A}_k$, a rank-k approximation of A.

Remark: When $k = \ell = 1$. This is the classical power method.

Accelerated randomized SVD algorithm 2

Remarks on the basic subspace iteration:

- The orthonormal basis Q of $Y = (AA^T)^q A\Omega$ should be stably computed by the following loop:

  compute $Y = A\Omega$
  compute $Y = QR$ (QR decompostion)
  for $j = 1, 2, \ldots, q$
  compute $Y = A^T Q$
  compute $Y = QR$ (QR decompostion)
  compute $Y = AQ$
  compute $Y = QR$ (QR deompostion)

- Convergence results:

  Under mild assumption of the starting matrix Ω,
  (a) the basic subspace iteration converges as $q \to \infty$.

  (b) $|\sigma_j - \sigma_j(Q^T B_k)| \leq O\left(\left(\frac{\sigma_{\ell+1}}{\sigma_k}\right)^{2q+1}\right)$

  Reading: M. Gu, Subspace iteration randomization and singular value problems, arXiv:1408.2208, 2014

Accelerated randomized SVD algorithm 3

- Input: $m \times n$ matrix A with $m \geq n$, positive integers k, ℓ, q and $n > \ell > k$.
- Steps:
  1. Draw a random $n \times \ell$ test matrix Ω.
  2. Compute $Y = (AA^T)^q A\Omega$ – "*sketching*".
  3. Compute an orthogonal columns basis Q of Y.
  4. Compute $\ell \times n$ matrix $B = Q^T A$.
  5. Compute B_k = the rank-truncated SVD of B.
  6. Compute $\widehat{A}_k = QB_k$.
- Output: $\widehat{A}_k$, a rank-k approximation of A.

Accelerated randomized SVD algorithm 4

MATLAB demo code: `randsvd2.m`

```
>> ...
>> Omega = randn(n,l);
>> C = A*Omega;
>> Q = orth(C);
>> for i = 1:q
>>    C = A'*Q;
>>    Q = orth(C);
>>    C = A*Q;
>>    Q = orth(C);
>> end
>> [Ua2,Sa2,Va2] = svd(Q'*A);
>> Ak2 = (Q*Ua2(:,1:k))*Sa2(1:k,1:k)*Va2(:,1:k)';
>> ...
```

The CUR decomposition

The CUR decomposition: find an optimal intersection U such that

$$A \approx CUR,$$

where C is the selected c columns of A, and R is the selected r rows of A.

The CUR decomposition

Theorem.

(a) $\|A - CC^{+}A\| \leq \|A - CX\|$ for any X

(b) $\|A - CC^{+}AR^{+}R\| \leq \|A - CXR\|$ for any X

(c) $U_{*} = \text{argmin}_{U} \|A - CUR\|_F^2 = C^{+}AR^{+}$

where $\|\cdot\|$ is a unitarily invariant norm.

Remark: Let $A = U\Sigma V^T$ is the SVD of an $m \times n$ matrix A with $m \geq n$. Then the pseudo-inverse (also called generalized inverse) A^{+} of A is given by $A^{+} = V\Sigma^{+}U^T$, where $\Sigma^{+} = \text{diag}(\sigma_1^{+}, \ldots)$ and $\sigma_j^{+} = 1/\sigma_j$ if $\sigma_j \neq 0$, otherwise $\sigma_j^{+} = 0$. If A is of full column rank, then $A^{+} = (A^T A)^{-1} A^T$. In MATLAB, `pinv(A)` is a built-in function of compute the pseudo-inverse of A.

The CUR decomposition

MATLAB demo code: `randcur.m`

```
>> ...
>> bound = n*log(n)/m;
>> sampled_rows = find(rand(m,1) < bound);
>> R = A(sampled_rows,:);
>> sampled_cols = find(rand(n,1) < bound);
>> C = A(:,sampled_cols);
>> U = pinv(C)*A*pinv(R);
>> ...
```

The CUR decomposition

- **Theorem.** With $c = O(k/\epsilon)$ columns and $r = O(k/\epsilon)$ rows selected by *adapative sampling* to for C and R,

$$\min_X \|A - CXR|_F^2 \leq (1+\epsilon)\|A - A_k\|_F^2$$

holds in expectation.

- *Reading: Boutsidis and Woodruff, STOC, pp.353-362, 2014*