Matrix factorizations and low rank approximation

Recall the two-stage approach to computing the approximate SVD of a given $m \times n$ matrix $\mathbf{A}$ of rank k: We seek $\mathbf{U}, \mathbf{V}$ orthonormal and $\mathbf{D}$ diagonal, $k \times k$ such that $\mathbf{A} \approx \mathbf{U}\mathbf{D}\mathbf{V}^*$.

Stage A: Find an $m \times k$ matrix $\mathbf{Q}$ with orthonormal columns such that $\mathbf{A} \approx \mathbf{Q}\mathbf{Q}^*\mathbf{A}$.

Stabe B: Compute the matrix-matrix product $\mathbf{B} = \mathbf{Q}^*\mathbf{A}$, the full SVD $\mathbf{B} = \hat{\mathbf{U}}\mathbf{D}\mathbf{V}^*$, and the matrix-matrix product $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

We then have that

$$\mathbf{A} \approx \mathbf{Q}\mathbf{Q}^*\mathbf{A} = \mathbf{Q}\mathbf{B} = \mathbf{Q}\hat{\mathbf{U}}\mathbf{D}\mathbf{V}^* = \mathbf{U}\mathbf{D}\mathbf{V}^*.$$

1. A RANDOMIZED ALGORITHM FOR "STAGE A"

For now, assume that A is exactly rank k. (This is a mathematical assumption. In finite precision arithmetic, a matrix is of course never exactly rank deficient, but mathematically, it is fine.)

The "range finder" approach proceeds as follows.

(1) Draw a Gaussian random vector $\mathbf{g}_1 \in \mathbb{R}^n$, so that each entry of $\mathbf{g}_1$ is drawn independently from a normalized Gaussian distribution. The idea is to sample *directions* uniformly (the length is more-or-less irrelevant), and drawing Gaussian random vectors is one way of achieving this result.

Compute the *sample vector* $\mathbf{y}_1 = \mathbf{A}\mathbf{g}_1 \in \mathrm{ran}(\mathbf{A})$.

(2) For $i = 2, ..., k$, draw a Gaussian random vector $\mathbf{g}_i \in \mathbb{R}^n$ and compute the sample vector $\mathbf{y}_i = \mathbf{A}\mathbf{g}_i \in \mathrm{ran}(\mathbf{A})$.

(3) Apply Gram-Schmidt to $\{\mathbf{y}_j\}_{j=1}^k$ to obtain the orthonormal basis $\{\mathbf{q}_j\}_{j=1}^k$ of $\mathrm{ran}(\mathbf{A})$. We can arrange the $\mathbf{q}_j$ into a matrix $\mathbf{Q} = [\mathbf{q}_1\ \mathbf{q}_2\ \cdots\ \mathbf{q}_k]$. Note that one can prove that $\{\mathbf{y}_j\}_{j=1}^k$ is linearly independent with probability 1, which shows that the Gram-Schmidt procedure will succeed.

There are some concerns with this idealized approach.

(1) Even though we are guaranteed that $\{\mathbf{y}_j\}_{j=1}^k$ is linearly independent with probability 1, in finite precision arithmetic the set of vectors $\{\mathbf{y}_j\}_{j=1}^k$ may be close to linearly dependent. Thus, the Gram-Schmidt procedure must be implemented carefully (e.g. Gram-Schmidt with reorthogonalization, Householder reflectors [used by LAPACK, MATLAB]).

(2) Typically A isn't exactly rank k. The singular values $\sigma_{k+1}, \sigma_{k+2}, ...$ will instead be small but non-zero. In the ideal case, $\mathrm{ran}(\mathbf{Q}) = \mathrm{span}\{\mathbf{u}_j\}_{j=1}^k$, where $\mathbf{u}_j$ is the jth left singular vector of $\mathbf{A}$; this provides the optimal (in the induced 2-norm) rank k approximation to A. The effect of small, but nonzero, singular values $\sigma_{k+1}, \sigma_{k+2}, ...$ is to "pollute" the sample vectors $\{\mathbf{y}_j\}_{j=1}^k$ by pushing them outside the span of the dominant k left singular vectors, $\{\mathbf{u}_j\}_{j=1}^k$.

Fortunately, there is a simple fix: *oversampling*. Pick a small integer p that specifies the amount of oversampling (e.g. $p = 5$ works well for a number of cases). Draw $k + p$ sample vectors instead of k, and perform the same procedure above to get $\{\mathbf{q}_j\}_{j=1}^{k+p}$. With high probability, the dominant k left singular vectors will be "almost contained" in $\mathrm{span}\{\mathbf{q}_j\}_{j=1}^{k+p}$.

Remark 1. The $k + p$ samples can be computed all at once. We can arrange the computation as a matrix-matrix product:

$$Y = \big[\mathbf{y}_1\ \mathbf{y}_2\ \cdots\ \mathbf{y}_{k+p}\big] = A\big[\mathbf{g}_1\ \mathbf{g}_2\ \cdots\ \mathbf{g}_{k+p}\big] = AG.$$

This formulation parallelizes well on a variety of platforms.

2. The Randomized SVD (RSVD)

In this section we combine the methods of the previous section with "Stage B", resulting in fast, randomized algorithm for computing an approximate SVD. We present first the algorithm and second a MATLAB implementation.

Algorithm 1 Randomized SVD (RSVD)

Input An $m \times n$ matrix $\mathbf{A}$, a rank k, and an oversampling parameter p.
Output Matrices $\mathbf{U}, \mathbf{D}, \mathbf{V}$ such that $\mathbf{A} \approx \mathbf{U}\mathbf{D}\mathbf{V}^*$, $\mathbf{U}, \mathbf{V}$ are orthonormal, and $\mathbf{D}$ is diagonal and $k \times k$.
procedure Stage A
    Draw a Gaussian random matrix $\mathbf{G}$ of size $n \times (k+p)$.
    Compute the *sampling matrix* $\mathbf{Y} = \mathbf{A}\mathbf{G}$.
    Orthonormalize the columns of $\mathbf{Y}$ to compute the ON $m \times (k+p)$ *basis matrix* $\mathbf{Q}$ s.t. $\mathrm{ran}(\mathbf{Q}) = \mathrm{ran}(\mathbf{Y})$.
end procedure
procedure Stage B
    Compute the matrix-matrix product $\mathbf{B} = \mathbf{Q}^*\mathbf{A}$.
    Compute the full SVD of $\mathbf{B} = \hat{\mathbf{U}}\mathbf{D}\mathbf{V}^*$.
    Compute the matrix-matrix product $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.
    If a rank-k approximation is desired, drop the p extra samples from $\mathbf{U}, \mathbf{D}, \mathbf{V}$.
end procedure

Note that at the end of Stage B, we (optionally) drop the p extra samples. If we do not drop the p extra samples, we'll get a *slightly* better approximation to $\mathbf{A}$. Note, however, that we typically will have $\sigma_{k+1}, \sigma_{k+2}, \ldots$ small, so that the approximation will not be significantly better than the rank-k approximation formed by dropping the extra p samples.

Listing 1. Randomized SVD

```
function [U,D,V] = rsvd(A, k, p)
[m,n] = size(A);

% Stage A
G = randn(n, k+p);
Y = A*G;
[Q,~,~] = qr(Y,0); % Defaults to CPQR
                   % 0 is a flag to produce an "economy size" decomp.

% Stage B
B = Q'*A;
[Uhat,D,V] = svd(B, 'econ');
U = Q*Uhat;
U = U(:,1:k); D = D(1:k,1:k); V = V(:,1:k); % optionally drop extra p samples

end
```

REFERENCES