Partial Lanczos SVD methods for R

Bryan Lewis[1], adapted from the work of Jim Baglama[2] and Lothar Reichel[3]

[1]REvolution Computing
[2]University of Rhode Island
[3]Kent State University

UseR 2009

Outline

SVD and partial SVD

Partial Lanczos bidiagonalization

The irlba package

SVD

Let $A \in \mathbf{R}^{\ell \times n}$, $\ell \geq n$.

$$A = \sum_{j=1}^{n} \sigma_j u_j v_j^T,$$

$$v_j^T v_k = u_j^T u_k = \begin{cases} 1 & \text{if } j = k, \\ 0 & \text{o.w.}, \end{cases}$$

$u_j \in \mathbf{R}^{\ell}$, $v_j \in \mathbf{R}^{n}$, $j = 1, 2, \ldots, n$, and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$.

Partial SVD

Let $k < n$.

$$\tilde{A}_k \quad := \quad \sum_{j=1}^{k} \sigma_j u_j v_j^T$$

Partial Lanczos bi-diagonalization

Start with a given vector p_1. Compute m steps of the Lanczos process:

$$\begin{aligned} AP_m &= Q_m B_m \\ A^T Q_m &= P_m B_m^T + r_m e_m^T, \end{aligned}$$

$B_m \in \mathbf{R}^{m\times m},\ P_m \in \mathbf{R}^{n\times m},\ Q_m \in \mathbf{R}^{\ell\times m},$
$P_m^T P_m = Q_m^T Q_m = I_m,$
$r_m \in \mathbf{R}^n,\ P_m^T r_m = 0,$
$P_m = [p_1, p_2, \ldots, p_m].$

Approximating partial SVD with partial Lanczos bi-diagonalization

$$
\begin{aligned}
A^T A P_m &= A^T Q_m B_m \\
&= P_m B_m^T B_m + r_m e_m^T B_m,
\end{aligned}
$$

Approximating partial SVD with partial Lanczos bi-diagonalization

$$
\begin{aligned}
A^T A P_m &= A^T Q_m B_m \\
&= P_m B_m^T B_m + r_m e_m^T B_m,
\end{aligned}
$$

$$
\begin{aligned}
A A^T Q_m &= A P_m B_m^T + A r_m e_m^T, \\
&= Q_m B_m B_m^T + A r_m e_m^T.
\end{aligned}
$$

Approximating partial SVD with partial Lanczos bi-diagonalization

Let

$$B_m = \sum_{j=1}^{m} \sigma_j^B u_j^B \left(v_j^B \right)^T .$$

Approximating partial SVD with partial Lanczos bi-diagonalization

Let

$$B_m = \sum_{j=1}^{m} \sigma_j^B u_j^B \left(v_j^B\right)^T.$$

$$\left(\text{i.e., } B_m v_j^B = \sigma_j^B u_j^B, \text{ and } B_m^T u_j^b = \sigma_j^B v_j^B.\right)$$

Define:

Approximating partial SVD with partial Lanczos bi-diagonalization

Let

$$B_m = \sum_{j=1}^{m} \sigma_j^B u_j^B \left(v_j^B\right)^T.$$

$$\left(\text{i.e., } B_m v_j^B = \sigma_j^B u_j^B, \text{ and } B_m^T u_j^b = \sigma_j^B v_j^B.\right)$$

Define:

$$\begin{aligned} \tilde{\sigma}_j &:= \sigma_j^B, \\ \tilde{u}_j &:= Q_m u_j^B, \\ \tilde{v}_j &:= P_m v_j^B. \end{aligned}$$

Partial SVD approximation of A

$$
\begin{aligned}
A\tilde{v}_j &= AP_m v_j^B \\
&= Q_m B_m v_j^B \\
&= \sigma_j^B Q_m u_j^B \\
&= \tilde{\sigma}_j \tilde{u}_j,
\end{aligned}
$$

Partial SVD approximation of A

$$
\begin{aligned}
A\tilde{v}_j &= AP_m v_j^B \\
&= Q_m B_m v_j^B \\
&= \sigma_j^B Q_m u_j^B \\
&= \tilde{\sigma}_j \tilde{u}_j,
\end{aligned}
$$

$$
\begin{aligned}
A^T\tilde{u}_j &= A^T Q_m u_j^B \\
&= P_m B_m^T u_j^B + r_m e_m^T u_j^B \\
&= \sigma_j^B P_m v_j^B + r_m e_m^T u_j^B \\
&= \tilde{\sigma}_j \tilde{v}_j + \color{red}{r_m e_m^T u_j^B}.
\end{aligned}
$$

Augment and restart

1. Compute the Lanczos process up to step m.
2. Compute $k < m$ approximate singular vectors.
3. Orthogonalize against the approximate singular vectors to get a new starting vector.
4. Continue the Lanczos process with the new starting vector for m more steps.
5. Check for convergence tolerance and exit if met.
6. GOTO 1.

Sketch of the augmented process...

$$\bar{P}_{k+1} \quad := \quad [\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_k, p_{m+1}],$$

Sketch of the augmented process...

$$\begin{aligned} \bar{P}_{k+1} &:= [\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_k, p_{m+1}], \\ A\bar{P}_{k+1} &= [\tilde{\sigma}_1 \tilde{u}_1, \tilde{\sigma}_1 \tilde{u}_2, \ldots, \tilde{\sigma}_k \tilde{u}_k, Ap_{m+1}] \end{aligned}$$

Sketch of the augmented process...

$$\bar{P}_{k+1} := [\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_k, p_{m+1}],$$
$$A\bar{P}_{k+1} = [\tilde{\sigma}_1 \tilde{u}_1, \tilde{\sigma}_1 \tilde{u}_2, \ldots, \tilde{\sigma}_k \tilde{u}_k, Ap_{m+1}]$$

Orthogonalize Ap_{m+1} against $\{\tilde{u}_j\}_{j=1}^{k}$: $\quad Ap_{m+1} = \sum_{j=1}^{k} \rho_j \tilde{u}_j + r_k$.

Sketch of the augmented process...

$$\begin{aligned} \bar{P}_{k+1} &:= [\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_k, p_{m+1}], \\ A\bar{P}_{k+1} &= [\tilde{\sigma}_1 \tilde{u}_1, \tilde{\sigma}_1 \tilde{u}_2, \ldots, \tilde{\sigma}_k \tilde{u}_k, Ap_{m+1}] \end{aligned}$$

Orthogonalize Ap_{m+1} against $\{\tilde{u}_j\}_{j=1}^k$: $\quad Ap_{m+1} = \sum_{j=1}^k \rho_j \tilde{u}_j + r_k$.

$$\begin{aligned} \bar{Q}_{k+1} &:= [\tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_k, r_k/\|r_k\|], \\ \bar{B}_{k+1} &:= \begin{bmatrix} \tilde{\sigma}_1 & & & \rho_1 \\ & \tilde{\sigma}_2 & & \rho_2 \\ & & \ddots & \rho_k \\ & & & \|r_k\| \end{bmatrix}. \end{aligned}$$

Sketch of the augmented process...

$$\begin{aligned}\bar{P}_{k+1} &:= [\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_k, p_{m+1}],\\ A\bar{P}_{k+1} &= [\tilde{\sigma}_1 \tilde{u}_1, \tilde{\sigma}_1 \tilde{u}_2, \ldots, \tilde{\sigma}_k \tilde{u}_k, Ap_{m+1}]\end{aligned}$$

Orthogonalize Ap_{m+1} against $\{\tilde{u}_j\}_{j=1}^{k}$: $\quad Ap_{m+1} = \sum_{j=1}^{k} \rho_j \tilde{u}_j + r_k$.

$$\begin{aligned}\bar{Q}_{k+1} &:= [\tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_k, r_k/\|r_k\|],\\ \bar{B}_{k+1} &:= \begin{bmatrix} \tilde{\sigma}_1 & & & \rho_1 \\ & \tilde{\sigma}_2 & & \rho_2 \\ & & \ddots & \rho_k \\ & & & \|r_k\| \end{bmatrix}.\end{aligned}$$

$$A\bar{P}_{k+1} = \bar{Q}_{k+1}\bar{B}_{k+1}.$$

The irlba package

```
Usage:

   irlba (A,
          nu = 5,
          nv = 5,
          adjust = 3,
          aug = "ritz",
          sigma = "ls",
          maxit = 1000,
          reorth = 1,
          tol = 1e-06,
          V = NULL)
```

Small example

```
> A<-matrix (rnorm(5000*5000),5000,5000)
> require (irlba)

> system.time (L<-irlba (A,nu=5,nv=5))
   user  system elapsed
 41.640   0.470  36.985

> gc()
            used   (Mb) ...  max used   (Mb)
Ncells    143301    7.7 ...    350000   18.7
Vcells  25193378  192.3 ...  78709588  600.6
```

Small example (continued)

```
> system.time (S<-svd(A,nu=5,nv=5))
   user  system elapsed
616.035   4.396 187.371

> gc()
             used   (Mb) ...  max used   (Mb)
Ncells     143312    7.7 ...    144539    7.8
Vcells   25248388  192.7 ... 200285943 1528.1
```

Small example (continued)

```
> system.time (S<-svd(A,nu=5,nv=5))
   user  system elapsed
616.035   4.396 187.371

> gc()
            used  (Mb) ...  max used   (Mb)
Ncells    143312   7.7 ...    144539    7.8
Vcells  25248388 192.7 ... 200285943 1528.1


> sqrt (crossprod(S$d[1:5]-L$d)/crossprod(S$d[1:5]))
             [,1]
[1,] 1.56029e-12
```

Large examples (live demo)

The R implementation of IRLBA supports:

- Dense real/complex in-process matrices (normal R matrices)
- Sparse real in-process matrices (Matrix)
- Dense, real in- or out-of-process huge matrices with bigmemory + bigalgebra

References

1. http://www.rforge.net/irlba
2. http://www.math.uri.edu/~jbaglama
3. http://www.math.kent.edu/~reichel
4. http://www.math.kent.edu/~blewis
5. http://revolution-computing.com