L8 Updating the SVD

Ivan Slapničar

May 2, 2018

1 Updating the SVD

In many applications which are based on the SVD, arrival of new data requires SVD of the new matrix. Instead of computing from scratch, existing SVD can be updated.

1.1 Prerequisites

The reader should be familiar with concepts of singular values and singular vectors, related perturbation theory, and algorithms.

1.2 Competences

The reader should be able to recognise applications where SVD updating can be sucessfully applied and apply it.

1.3 Facts

For more details see M. Gu and S. C. Eisenstat, A Stable and Fast Algorithm for Updating the Singular Value Decomposition and M. Brand, Fast low-rank modifications of the thin singular value decomposition and the references therein.

1. Let $A \in \mathbb{R}^{m\times n}$ with $m \geq n$ and $\operatorname{rank}(A) = n$, and let $A = U\Sigma V^T$ be its SVD. Let $a \in \mathbb{R}^n$ be a vector, and let $\tilde A = \begin{bmatrix} A \\ a^T\end{bmatrix}$. Then

$$\begin{bmatrix} A \\ a^T\end{bmatrix} = \begin{bmatrix} U & \\ & 1\end{bmatrix} \begin{bmatrix} \Sigma \\ a^T V\end{bmatrix} V^T.$$

Let $\begin{bmatrix} \Sigma \\ a^T V\end{bmatrix} = \bar U \bar\Sigma \bar V^T$ be the SVD of the half-arrowhead matrix. *This SVD can be computed in $O(n^2)$ operations.* Then

$$\begin{bmatrix} A \\ a^T\end{bmatrix} = \begin{bmatrix} U & \\ & 1\end{bmatrix} \bar U \bar\Sigma \bar V^T V^T \equiv \tilde U \bar\Sigma \tilde V^T$$

is the SVD of $\tilde A$.

2. Direct computation of $\tilde{U}$ and $\tilde{V}$ requires $O(mn^2)$ and $O(n^3)$ operations. However, these multiplications can be performed using Fast Multipole Method. This is not (yet) implemented in Julia and is "not for the timid" (quote by Steven G. Johnson).

3. If $m < n$ and $\operatorname{rank}(A) = n$, then

$$\begin{bmatrix} A \\ a^T \end{bmatrix} = \begin{bmatrix} U & \\ & 1 \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ a^T V & \beta \end{bmatrix} \begin{bmatrix} V^T \\ v^T \end{bmatrix},$$

where $\beta = \sqrt{\|a\|_2^2 - \|V^T a\|_2^2}$ and $v = (I - VV^T)a$. Notice that $V^T v = 0$ by construction. Let $\begin{bmatrix} \Sigma & 0 \\ a^T V & \beta \end{bmatrix} = \bar{U}\bar{\Sigma}\bar{V}^T$ be the SVD of the half-arrowhead matrix. Then

$$\begin{bmatrix} A \\ a^T \end{bmatrix} = \begin{bmatrix} U & \\ & 1 \end{bmatrix} \bar{U}\bar{\Sigma}\bar{V}^T \begin{bmatrix} V^T \\ v^T \end{bmatrix} \equiv \tilde{U}\bar{\Sigma}\tilde{V}^T$$

is the SVD of $\tilde{A}$.

4. Adding a column a to A is equivalent to adding a row a^T to A^T.

5. If $\operatorname{rank}(A) < \min\{m, n\}$ or if we are using SVD approximation of rank r, and if we want to keep the rank of the approximation (this is the common case in practice), then the formulas in Fact 1 hold approximately. More precisely, the updated rank r approximation is **not** what we would get by computing the approximation of rank r of the updated matrix, but is sufficient in many applications.

1.3.1 Example - Adding row to a tall matrix

If $m >= n$, adding row does not increase the size of Σ.

```
In [1]: using Arrowhead
```

```
In [2]: function mySVDaddrow(svdA::Tuple,a::Vector)
            # Create the transposed half-arrowhead
            m,r,n=size(svdA[1],1),length(svdA[2]),size(svdA[3],1)
            T=typeof(a[1])
            b=svdA[3]'*a
            if m>=n || r<m
                M=HalfArrow(svdA[2],b)
            else
                β=sqrt(vecnorm(a)^2-vecnorm(b)^2)
                M=HalfArrow(svdA[2],[b;β])
            end
            tols=[1e2,1e2,1e2,1e2]
            U,σ,V=svd(M,tols)
            # Return the updated SVD
            if m>=n || r<m
                return [svdA[1] zeros(T,m); zeros(T,1,r) one(T)]*V, σ, svdA[3]*U
            else
                # Need one more row of svdA[3] - v is orthogonal projection
                v=a-svdA[3]*b
                v=v/norm(v)
                return [svdA[1] zeros(T,m); zeros(T,1,r) one(T)]*V, σ, [svdA[3] v]*U
            end
        end
```

```
Out[2]: mySVDaddrow (generic function with 1 method)
```

```
In [3]: s=srand(421)
        A=rand(10,6)
        a=rand(6)
```

```
Out[3]: 6-element Array{Float64,1}:
         0.336964
         0.916645
         0.832777
         0.844824
         0.886652
         0.344321
```

```
In [4]: svdA=svd(A)
```

```
Out[4]: ([-0.303731 -0.160825 ... -0.0983823 0.0874599; -0.308381 0.324931 ... -0.212191 0.13150
```

```
In [5]: typeof(svdA)
```

```
Out[5]: Tuple{Array{Float64,2},Array{Float64,1},Array{Float64,2}}
```

```
In [6]: U,σ,V=mySVDaddrow(svdA,a)
```

```
Out[6]: ([-0.276987 0.203469 ... 0.145536 0.065527; -0.285338 -0.271956 ... 0.262804 0.0867921;
```

```
In [7]: # Check the residual and orthogonality
        vecnorm([A;a']*V-U*diagm(σ)), vecnorm(U'*U-I), vecnorm(V'*V-I)
```

```
Out[7]: (8.226613472016954e-15, 1.9567217786511962e-15, 2.5903045614734177e-15)
```

1.3.2 Example - Adding row to a flat matrix

```
In [8]: # Now flat matrix
        A=rand(6,10)
        a=rand(10)
        svdA=svd(A)
```

```
Out[8]: ([-0.396523 -0.41672 ... -0.380992 -0.57937; -0.366007 -0.229076 ... 0.817008 -0.288577;
```

```
In [9]: U,σ,V=mySVDaddrow(svdA,a)
        norm([A;a']*V-U*diagm(σ)), norm(U'*U-I), norm(V'*V-I)
```

```
Out[9]: (3.2685916964195365e-15, 1.0076558439584477e-15, 1.5888194818579435e-15)
```

1.3.3 Example - Adding columns

This can be viewed as adding rows to the transposed matrix, an elegant one-liner in Julia.

```
In [10]: function mySVDaddcol(svdA::Tuple,a::Vector)
             reverse(mySVDaddrow(reverse(svdA),a))
         end
```

```
Out[10]: mySVDaddcol (generic function with 1 method)
```

```
In [11]: # Tall matrix
         A=rand(10,6)
         a=rand(10)
         svdA=svd(A)
         U,σ,V=mySVDaddcol(svdA,a)
         vecnorm([A a]*V-U*diagm(σ)), vecnorm(U'*U-I), vecnorm(V'*V-I)
```

```
Remedy 3
```

```
Out[11]: (2.018093616816287e-15, 2.885869687524597e-15, 1.2203017392912243e-15)
```

```
In [12]: # Flat matrix
         A=rand(6,10)
         a=rand(6)
         svdA=svd(A)
         U,σ,V=mySVDaddcol(svdA,a)
         vecnorm([A a]*V-U*diagm(σ)), vecnorm(U'*U-I), vecnorm(V'*V-I)
```

```
Remedy 3
```

```
Out[12]: (4.219249770465842e-15, 2.1190764959016045e-15, 3.4065348430906964e-15)
```

```
In [13]: # Square matrix
         A=rand(10,10)
         a=rand(10);
         svdA=svd(A);
```

```
In [14]: U,σ,V=mySVDaddrow(svdA,a)
         vecnorm([A;a']*V-U*diagm(σ)), vecnorm(U'*U-I), vecnorm(V'*V-I)

Out[14]: (7.001999503384417e-15, 3.4683414833379754e-15, 2.6833491333185156e-15)

In [15]: U,σ,V=mySVDaddcol(svdA,a)
         vecnorm([A a]*V-U*diagm(σ)), vecnorm(U'*U-I), vecnorm(V'*V-I)

Out[15]: (7.443251109620379e-15, 3.4729600362235705e-15, 2.4343704537550683e-15)
```

1.3.4 Example - Updating a low rank approximation

```
In [16]: # Adding row to a tall matrix
         A=rand(10,6)
         svdA=svd(A)
         a=rand(6)
         # Rank of the approximation
         r=4

Out[16]: 4

In [17]: svdAr=(svdA[1][:,1:r], svdA[2][1:r],svdA[3][:,1:r])
         U,σ,V=mySVDaddrow(svdAr,a)
         vecnorm([A;a']-U*diagm(σ)*V'), svdvals([A;a']), σ

Out[17]: (1.1166290977762616, [4.29878, 1.11594, 1.05239, 1.02696, 0.804526, 0.668058], [4.29596

In [18]: # Adding row to a flat matrix
         A=rand(6,10)
         svdA=svd(A)
         a=rand(10)
         # Rank of the approximation
         r=4

Out[18]: 4

In [19]: svdAr=(svdA[1][:,1:r], svdA[2][1:r],svdA[3][:,1:r])
         U,σ,V=mySVDaddrow(svdAr,a)
         vecnorm([A;a']-U*diagm(σ)*V'), svdvals([A;a']), σ

Out[19]: (1.1587774205933508, [4.71233, 1.2258, 1.04087, 0.894276, 0.776174, 0.633071, 0.277758]

In [ ]:
```