Notes on Rank-K Approximation

(and SVD for the uninitiated)

Robert A. van de Geijn
Department of Computer Science
The University of Texas
Austin, TX 78712
rvdg@cs.utexas.edu

April 19, 2011

1 Background

This notes assumes that the reader understands the following concepts:

- Linear combination of vectors.
- Linearly independent columns.
- Matrix-vector multiplication forms a linear combination of the columns of the matrix: Let $A \in \mathbb{R}^{m \times n}$ and $x \in \mathbb{R}^n$. Partition

$$A \rightarrow \left(\begin{array}{c|c|c|c} a_0 & a_1 & \cdots & a_{n-1} \end{array} \right) \quad \text{and} \quad x = \left(\begin{array}{c} \chi_0 \\ \hline \chi_1 \\ \hline \vdots \\ \hline \chi_{n-1} \end{array} \right)$$

then

$$Ax = \chi_0 a_0 + \chi_1 a_1 + \cdots + \chi_{n-1} a_{n-1}.$$

- $Ax = y$ only if y is in the column space of A ($y \in \mathcal{C}(A)$).
- If A has linearly independent columns and y is not in $\mathcal{C}(A)$ (and even if it is), the vector x that comes closest to solving $Ax \approx y$ is given by $x = (A^T A)^{-1} A^T y$. Here $(A^T A)^{-1} A^T$ is known as the *pseudo-inverse*. In this case the vector $z = Ax = A(A^T A)^{-1} A^T y$ is the *projection of y onto the column space of A, $\mathcal{C}(A)$*.

Figure 1: Original picture that will be approximated by a rank-k update.

2 Application

Let $Y \in \mathbb{R}^{m \times n}$ be a matrix that, for example, stores a picture. In this case, entry ψ_{ij} is, for example, a number that represents the gray level of pixel (i, j). The following instructions, executed in octave, generate the picture in Figure 1.

Approximation AX when $k = 20$ | Approximation $U_L \Sigma_{TL} V_L^T$ when $k = 20$

Approximation AX when $k = 30$ | Approximation $U_L \Sigma_{TL} V_L^T$ when $k = 30$

Figure 2: Multiple pictures generated by the rank-k approximations.

```
octave> lenna      % this loads the matrix Y with the picture in file lenna
octave> image( Y ) % this dispays the image
```

Now, pick out k columns of Y, and make them the columns of matrix A.

```
octave> k = 20;             % pick out 20 columns
octave> n = size( Y, 2 );  % n equals the number of columns in Y
octave> A = Y( :, 1:floor(n/k):n );
```

If $n = 400$, the above will set the first column of A to the first column of Y, the second column of A to column 21 of Y, etc. With a bit of luck, A has linearly independent columns. Let's assume that it does.

Now, columns in the picture vary slowly from one column to the next. So, it might be that some arbitrary column of Y, y_j, is actually a linear combination of the columns you chose. In other words, there is a x_j such that $Ax_j = y_j$. Well, that is probably a bit optimistic. So it is more likely that $Ax_j \approx y_j$. In that case the best choice for x_j is given by $x_j = (A^T A)^{-1} A^T y_j$, the linear least-squares solution. If x_j is chosen in that way, then $Ax_j = A(A^T A)^{-1} A^T y_j$ is the projection of y_j onto the column space of A, which means that it is the best linear combination of the columns of A.

What does this mean? If we partition X and Y by columns,

$$X = \left(\begin{array}{c|c|c|c} x_0 & x_1 & \cdots & x_{n-1} \end{array} \right) \quad \text{and} \quad Y = \left(\begin{array}{c|c|c|c} y_0 & y_1 & \cdots & y_{n-1} \end{array} \right)$$

then for each column of Y, y_j, we can approximate that column (vertical line in the picture) by $Ax_j \approx y_j$ where $x_j = (A^T A)^{-1} A^T y_j$.

Equivalently,

$$\left(\begin{array}{c|c|c|c} Ax_0 & Ax_1 & \cdots & Ax_{n-1} \end{array} \right) \approx \left(\begin{array}{c|c|c|c} y_0 & y_1 & \cdots & y_{n-1} \end{array} \right)$$

or

$$A \left(\begin{array}{c|c|c|c} x_0 & x_1 & \cdots & x_{n-1} \end{array} \right) \approx \left(\begin{array}{c|c|c|c} y_0 & y_1 & \cdots & y_{n-1} \end{array} \right),$$

where

$$\begin{array}{rcl} X = \left(\begin{array}{c|c|c|c} x_0 & x_1 & \cdots & x_{n-1} \end{array} \right) & = & \left(\begin{array}{c|c|c|c} (A^T A)^{-1} A^T y_0 & (A^T A)^{-1} A^T y_1 & \cdots & (A^T A)^{-1} A^T y_{n-1} \end{array} \right) \\ & = & (A^T A)^{-1} A^T \left(\begin{array}{c|c|c|c} y_0 & y_1 & \cdots & y_{n-1} \end{array} \right) \\ & = & (A^T A)^{-1} A^T Y. \end{array}$$

Notice: $A \in \mathbb{R}^{m \times k}$ and X is $\mathbb{R}^{k \times n}$:

- $A^T A \in \mathbb{R}^{k \times k}$.
- $(A^T A)^{-1} \in \mathbb{R}^{k \times k}$.
- $(A^T A)^{-1} A^T \in \mathbb{R}^{k \times n}$.

What we conclude is that $Y \approx AX$ where $X = (A^T A)^{-1} A^T Y$:

```
octave> X = ( A' * A ) \ ( A' * Y );   % X = inv( A' * A ) * A' * Y
octave> image( A * X )                 % this dispays the approximation of the image
```

3 Observations

Let $A \in \mathbb{R}^{m \times k}$ and $X \in \mathbb{R}^{k \times n}$. Then $B = AX$ has rank at most k.

- The rank of B equals the number of linearly independent columns.
- The rank of B also equals the dimension of its column space.
- The dimension of a space equals the number of vectors in its basis.
- The subset of columns of B that are linearly independent for a basis for the column space of B.
- Thus, there are at most k vectors in that basis and the rank of matrix B is at most k.

For this reason, AX, as computed in the previous section, is said to be a rank-k approximation of matrix Y. (In that section, we assumed A had linearly independent columns and those columns clearly become columns in AX. Hence, the rank is exactly k unless the rank of X is less than k.)

Typically, as one increases k, the approximation gets better. Try this!

4 The Singular Value Decomposition (SVD)

Definition 1. *Let $U \in \mathbb{R}^{m\times k}$. Then U is said to be orthonormal if each of its columns has length (2-norm) one, and its columns are mutually orthogonal.*

In other words,

- If $U = \left(u_0 \mid u_1 \mid \cdots \mid u_{k-1} \right)$ then $u_i^T u_j = \begin{cases} 0 \text{ if } i \neq j \\ 1 \text{ otherwise.} \end{cases}$
- Another, very concise, way of saying this is $U^T U = I$.

Theorem 2. *(Singular Value Decomposition Theorem) Given $Y \in \mathbb{R}^{m\times n}$, there exists orthogonal matrices $U \in \mathbb{R}^{m\times r}$, $V \in \mathbb{R}^{n\times r}$, and $\Sigma \in \mathbb{R}^{r\times r}$ such that $Y = U\Sigma V^T$, where U and V have orthonormal columns and*

$$\Sigma = \begin{pmatrix} \sigma_0 & & & \\ & \sigma_1 & & \\ & & \ddots & \\ & & & \sigma_{r-1} \end{pmatrix}$$

and $\sigma_0 \geq \sigma_1 \geq \cdots \geq \sigma_{r-1} > 0$.

This is probably the single most important result in linear algebra. We won't prove it.

Now, assume that $Y = U\Sigma V^T$ is the SVD of matrix Y. Partition, conformally,

$$U = \left(U_L \mid U_R \right), \quad V = \left(V_L \mid V_R \right), \quad \text{and} \quad \Sigma = \left(\begin{array}{c|c} \Sigma_{TL} & 0 \\ \hline 0 & \Sigma_{BR} \end{array} \right),$$

where U_L and V_L have k columns and Σ_{TL} is $k \times k$. so that

$$\begin{aligned} Y &= \left(U_L \mid U_R \right) \left(\begin{array}{c|c} \Sigma_{TL} & 0 \\ \hline 0 & \Sigma_{BR} \end{array} \right) \left(V_L \mid V_R \right)^T \\ &= \left(U_L \mid U_R \right) \left(\begin{array}{c|c} \Sigma_{TL} & 0 \\ \hline 0 & \Sigma_{BR} \end{array} \right) \left(\begin{array}{c} V_L^T \\ \hline V_R^T \end{array} \right) \\ &= \left(U_L \mid U_R \right) \left(\begin{array}{c} \Sigma_{TL} V_L^T \\ \hline \Sigma_{BR} V_R^T \end{array} \right) \\ &= U_L \Sigma_{TL} V_L^T + U_R \Sigma_{BR} V_R^T. \end{aligned}$$

Now, if $\sigma_0 \geq \cdots \geq \sigma_{k-1} \gg \sigma_k \geq \cdots \sigma_{r-1}$, then

$$Y = U_L \Sigma_{TL} V_L^T + U_R \Sigma_{BR} V_R^T \approx U_L \Sigma_{TL} V_L^T.$$

Notice: Previously, $Y \approx AX$, where $A \in \mathbb{R}^{m\times k}$ and $X \in \mathbb{R}^{k\times n}$. Now, $Y \approx U_L(\Sigma_{TL} V_L^T)$, where $U_L \in \mathbb{R}^{m\times k}$ and $(\Sigma_{TL} V_L^T) \in \mathbb{R}^{k\times n}$. While AX was <u>a</u> rank-k approximation to Y, $U_L \Sigma_{TL} V_L^T$ is the <u>best</u> rank-k approximation to Y.

Try

```
octave> lenna      % this loads the matrix Y with the picture in file lenna
octave> image( Y ) % this dispays the image
octave> k = 20;
octave> [ U, Sigma, V ] = svd( Y );
octave> UL = U( :, 1:k );             % first k columns
octave> VL = V( :, 1:k );             % first k columns
octave> SigmaTL = Sigma( 1:k, 1:k ); % TL submatrix of Sigma
octave> image( UL * SigmaTL * VL' );
```

and, hopefully, the result looks better than what we got before!