

# Introduction to SVD and Applications

Eric Kostelich and Dave Kuhl  
MSRI Climate Change Summer School

July 18, 2008

## Introduction

The goal of this exercise is to familiarize you with the basics of the singular value decomposition (SVD). The SVD is known by many names, such as *principal component analysis*. It has many uses. In numerical analysis, the SVD provides a measure of the effective rank of a given matrix. In statistics and time series analysis, the SVD a particularly useful tool for finding least-squares solutions and approximations.

Don't worry if you don't finish all the exercises today. However, I strongly encourage you to complete them over the weekend. Monday's lectures will be easier to understand once you have some geometric intuition about the SVD.

Let  $\mathbf{A}$  be an  $m \times n$  real matrix;  $m$  and  $n$  may be any positive integers. The SVD of  $\mathbf{A}$  is the factorization

$$\mathbf{A}_{m \times n} = \mathbf{U}_{m \times n} \mathbf{S}_{n \times n} \mathbf{V}_{n \times n}^T, \quad (1)$$

where  $\mathbf{S} = \text{diag}(s_1, \dots, s_n)$ . The  $s_i$ 's are the *singular values* of  $\mathbf{A}$ . By convention, they are ordered so that  $s_1 \geq s_2 \geq \dots \geq s_n \geq 0$ . The columns of  $\mathbf{U}$  and  $\mathbf{V}$  may be chosen so that they form an orthonormal basis of the column space and row space, respectively, of  $\mathbf{A}$ . If  $\mathbf{A}$  has full rank, then its singular values are all positive, and when they are ordered as indicated, then the SVD is unique up to the signs of the columns of  $\mathbf{U}$  and  $\mathbf{V}$ .

**Note.** There are other essentially equivalent formulations of Eq. (1). For example, when  $m \geq n$  and  $\text{rank } \mathbf{A} = n$ , it is possible to extend  $\mathbf{U}$  to be an  $m \times m$  matrix whose columns form an orthonormal basis for  $\mathbf{R}^m$  (and whose first  $n$  columns span

the column space of  $\mathbf{A}$ ). In this case,  $\mathbf{S}$  may be taken to be an  $m \times n$  matrix such that  $S_{ii} = s_i$ ,  $i = 1, \dots, n$ , and whose other entries are zero.

All of this can be extended to a general  $m \times n$  complex matrix  $\mathbf{A}$ . You can find more details in most any textbook on advanced linear algebra and on a variety of Web sites. We'll stick to real matrices for today.

The decomposition in Eq. (1) implies that

$$\begin{aligned}\mathbf{A}^T \mathbf{A} &= (\mathbf{U} \mathbf{S} \mathbf{V}^T)^T (\mathbf{U} \mathbf{S} \mathbf{V}^T) \\ &= \mathbf{V} \mathbf{S} \mathbf{U}^T \mathbf{U} \mathbf{S} \mathbf{V}^T \\ &= \mathbf{V} \mathbf{S}^2 \mathbf{V}^T \quad \text{since } \mathbf{U} \text{ is orthonormal and } \mathbf{S} \text{ is diagonal.}\end{aligned}$$

Also, because  $\mathbf{V}^T = \mathbf{V}^{-1}$ , we have  $(\mathbf{A}^T \mathbf{A}) \mathbf{V} = \mathbf{V} \mathbf{S}^2$ , which shows that the columns of  $\mathbf{V}$  are eigenvectors of  $\mathbf{A}^T \mathbf{A}$ . The singular values of  $\mathbf{A}$  are the square roots of the corresponding eigenvalues.

**Practice Problem 1.** Show that the columns of  $\mathbf{U}$  are eigenvectors of  $\mathbf{A} \mathbf{A}^T$ .

**Practice Problem 2.** Consider the linear least-squares problem, where we fit a  $p$ -parameter model of the form

$$y_i = b_1 + b_2 x + \dots + b_p x^{p-1} \quad (2)$$

to a collection of data points  $\{(x_i, y_i)\}_{i=1}^n$ . Equation (2) can be written in matrix-vector form as

$$\mathbf{y} = \mathbf{X} \boldsymbol{\beta} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_1 & \cdots & x_1^{p-1} \\ 1 & x_2 & \cdots & x_2^{p-1} \\ & & \vdots & \\ 1 & x_n & \cdots & x_n^{p-1} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_p \end{pmatrix}.$$

The least-squares solution is given by the solution to the *normal equation*

$$(\mathbf{X}^T \mathbf{X}) \boldsymbol{\beta} = \mathbf{X}^T \mathbf{y}. \quad (3)$$

Assuming that  $n \geq p$  and  $\text{rank } \mathbf{X} = p$ , show that

$$\boldsymbol{\beta} = \mathbf{V} \mathbf{S}^{-1} \mathbf{U}^T \mathbf{y}$$

where  $\mathbf{X} = \mathbf{U} \mathbf{S} \mathbf{V}^T$  is the SVD of  $\mathbf{X}$ . Hence the SVD is an alternative to the *QR* decomposition for solving linear least-squares problems.

## Matrix times circle equals ellipse

**Practice Problem 3.** Generate a circle of points with  $1^\circ$  separation as follows (mind the semicolons!):

```
circ = zeros(2, 360);  a 2 x 360 zero array
deg = [ 1 : 1 : 360 ];
circ(1, :) = cos(deg * pi / 180);  pi is a built-in constant
circ(2, :) = sin(deg * pi / 180);
```

Now define  $\mathbf{A}$  to be your favorite nonsingular  $2 \times 2$  matrix (choose  $\mathbf{A} \neq \mathbf{I}$  to keep things interesting). The syntax

```
A = [ a b ; c d ]
```

defines  $\mathbf{A} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ . Compute the SVD of  $\mathbf{A}$ :

```
[ U S V ] = svd(A)
```

Now plot  $\mathbf{A} \times$  circle as follows:

```
hold on  concatenate all subsequent plots on the same graph
axis equal
plot(circ(1, :), circ(2, :), 'o')  no semicolon
Acirc = A * circ;  semicolon!
plot(Acirc(1, :), Acirc(2, :), 'o')  no semicolon
```

The major and minor axes of the ellipse are given by the eigenvectors of  $\mathbf{A}\mathbf{A}^T$ , that is, the columns of  $\mathbf{U}$ . Add them to your plot as follows:

```
Uline = [ U(:,1)*S(1,1) [0 0]' U(:,2)*S(2,2) ];  note the prime
plot(Uline(1, :), Uline(2, :), '-r')
hold off
```

## Rank- $r$ approximations

Often it is convenient to approximate a given full-rank matrix  $\mathbf{A}$  by a matrix  $\hat{\mathbf{A}}$  of less than full rank. The approximation  $\hat{\mathbf{A}}$  is obtained by Eq. (1) where the smallest  $n - r$  singular values in  $\mathbf{S}$  are replaced with 0. Equivalently, we may write

$$\hat{\mathbf{A}} = \hat{\mathbf{U}}_{m \times r} \hat{\mathbf{S}}_{r \times r} \hat{\mathbf{V}}_{r \times n}^T \quad (4)$$

that is, the product of the first  $r$  columns of  $\mathbf{U}$ , the upper  $r \times r$  block of  $\mathbf{S}$ , and the first  $r$  columns of  $\mathbf{V}$ . (In other words, for every  $\mathbf{x}$ , we project  $\mathbf{Ax}$  onto the subspace spanned by the first  $r$  columns of  $\mathbf{U}$ .)

The notion of a rank- $r$  approximation is a key idea in the data assimilation algorithm that we'll discuss next week. One goal today is to illustrate that  $r$  need not be very large to get a good approximation. The example that we'll consider gives an image compression algorithm.

**Step 1.** Download the file `dog.jpg` from the Web page to the your working MATLAB directory. Then load it into Matlab with the command

```
A = imread('./dog.jpg');  semicolon!
```

The semicolon is necessary so that MATLAB does not print out many screenfuls of data. The result is a  $310 \times 338$  matrix of grayscale values corresponding to a black and white picture of a dog. (The matrix has 104,780 entries.)

**Step 2.** We need to do some initial processing. Type

```
B = double(A(:, :, 1)) + 1;  semicolon!
```

which converts  $A$  into the double-precision format that is needed for the singular value decomposition. Now say

```
B = B / 256;  semicolon!  
[ U S V ] = svd(B);  semicolon!
```

The gray scale goes from 0 to 256 in a black-and-white JPEG image. We divide  $\mathbf{B}$  by 256 to obtain values between 0 and 1, which is required for the MATLAB imaging routines that we'll use later.

**Practice Problem 4.** What are the dimensions of  $\mathbf{U}$ ,  $\mathbf{S}$ , and  $\mathbf{V}$ ?

Here  $\mathbf{S}$  has more columns than rows; in fact, columns 311 to 338 are all zeros. (When  $m < n$ , we pad  $\mathbf{S}$  on the right with zero columns to turn  $\mathbf{S}$  into an  $m \times n$  matrix rather than an  $n \times n$  matrix.)

**Practice Problem 5.** Compute the best rank-1 approximation to  $\mathbf{B}$  and store it in the matrix `rank1`. (See Eq. (4). The matrix slice consisting of the first  $r$  columns of  $\mathbf{U}$  is denoted `U(:, 1:r)` in MATLAB. The upper  $r \times r$  portion of  $\mathbf{S}$  is `S(1:r, 1:r)`, and so on.)

**Step 3.** Let's visualize `rank1`. To do that, first create

```
C = zeros(size(A));  semicolon!
```

This creates an array of zeroes,  $\mathbf{C}$ , of the same dimensions as the original image matrix  $\mathbf{A}$ . Now  $\mathbf{A}$  is actually  $310 \times 338 \times 3$ . To create an image from the matrix  $\mathbf{C}$ , MATLAB uses  $\mathbf{C}(i, j, 1 : 3)$  as the RGB values that color the  $ij$ th pixel. We have a black-and-white photo, so we'll set all the RGB values to be the same, namely  $1/2$ :

```
C(:, :, :) = 0.5;    semicolon!
```

**Step 4.** Copy the rank-1 image into  $\mathbf{C}$  as follows:

```
C(:, :, 1) = rank1;    semicolons on all of these  
C(:, :, 2) = rank1;  
C(:, :, 3) = rank1;
```

**Step 5.** We're almost done, except for one hitch. MATLAB does all its scaling using values from 0 to 1 (and maps them to values between 0 and 256 for the graphics hardware). Lower-rank approximations to the actual image can have values that are slightly less than 0 and greater than 1. So we'll truncate them to fit, as follows:

```
C(:, :, :) = min(1, C(:, :, :));    semicolon!  
C(:, :, :) = max(0, C(:, :, :));    semicolon!
```

**Step 6.** View the resulting image:

```
image(C)    no semicolon
```

**Practice Problem 6.** Create and view a rank-10 approximation to the original picture. (Use Steps 4–6 but with rank10 instead of rank1. If you mess up—for example, you get an all-black picture—then start over from Step 3.)

**Practice Problem 7.** Repeat with approximations of rank 20, 30, and 40 (and any others that you'd like to experiment with). What is the smallest rank that, in your opinion, gives an acceptable approximation to the original picture?

**Practice Problem 8.** What rank- $r$  approximation exactly reproduces the original picture?