Section 7.4 Notes (The SVD)

The Singular Value Decomposition: Let A be any $m \times n$ matrix. Then there are orthogonal matrices U, V and a diagonal matrix Σ such that

$$A = U\Sigma V^T$$

Notes about this factorization:

- The ordering of the columns of U, V comes from the ordering of the singular values in Σ (largest to smallest).
- The columns of U are the eigenvectors of AA^T (and are o.n.)
- The columns of V are the eigenvectors of A^TA (and are o.n.)
- The non-zero eigenvalues (λ_i) of AA^T are the same as A^TA.
- The diagonal elements of Σ are the singular values, $\sigma_i = \sqrt{\lambda_i}$
- There are relationships between $\mathbf{v}_i$ and $\mathbf{u}_i$ (with normalization):

  $$A\mathbf{v}_i = \sigma_i \mathbf{u}_i \qquad A^T\mathbf{u}_i = \sigma_i \mathbf{v}_i$$

  The scaling factor comes from $\|A\mathbf{v}_i\| = \sigma_i = \|A^T\mathbf{u}_i\|$.
- If the matrix A is already a symmetric matrix, then $U = V$ and we get the decomposition from 7.1, which in that notation was PDP^T.

Numerical Example

(Exercise 11) Find the SVD of the matrix $A = \begin{bmatrix} -3 & 1 \\ 6 & -2 \\ 6 & -2 \end{bmatrix}$

SOLUTION: If we form A^TA and AA^T, then we just need to compute the eigenvalues. However, we might do this intelligently- It's really the eigenvalues we're after first, so use the smaller matrix. Below is an example.

The matrix $A^TA = \begin{bmatrix} 81 & -27 \\ -27 & 9 \end{bmatrix}$ has eigenvalues $\lambda = 0, 90$ (using a calculator).

For $\lambda = 0$, the reduced matrix is $\begin{bmatrix} 1 & -1/3 \\ 0 & 0 \end{bmatrix}$, so $\mathbf{v} = \frac{1}{\sqrt{10}} \begin{bmatrix} 1 \\ 3 \end{bmatrix}$

For $\lambda = 90$, the reduced matrix is $\begin{bmatrix} 1 & 3 \\ 0 & 0 \end{bmatrix}$, so $\mathbf{v} = \frac{1}{\sqrt{10}} \begin{bmatrix} -3 \\ 1 \end{bmatrix}$

So far, we know V and Σ, so now we need U. The first column can be found directly:

$$\mathbf{u}_1 = \frac{1}{\sigma_1} A\mathbf{v}_1 = \frac{1}{3}\begin{bmatrix} 1 \\ -2 \\ -2 \end{bmatrix}$$

We know there is only one non-zero eigenvalue, therefore, the rest of U must be in the null space of A (which is the eigenspace for $\lambda = 0$):

$$AA^T = \begin{bmatrix} 10 & -20 & -20 \\ -20 & 40 & 40 \\ -40 & 40 & 40 \end{bmatrix} \sim \begin{bmatrix} 1 & -2 & -2 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \Rightarrow \quad \left\{ \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix} \right\}$$

Now the full SVD is given by:

$$A = \begin{bmatrix} -3 & 1 \\ 6 & -2 \\ 6 & -2 \end{bmatrix} = \begin{bmatrix} 1/3 & 2/\sqrt{5} & 2/\sqrt{5} \\ -2/3 & 1/\sqrt{5} & 0 \\ -2/3 & 0 & 1/\sqrt{5} \end{bmatrix} \begin{bmatrix} 3\sqrt{10} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} -3/\sqrt{10} & 1/\sqrt{10} \\ 1/\sqrt{10} & 3/\sqrt{10} \end{bmatrix}^T$$

The SVD and the Four Subspaces

- If the columns of V are ordered by the magnitude of the singular values (largest to smallest), and the rank of A is r, then

  $$\text{Null}(A) = \text{span}\{\mathbf{v}_{r+1},\ \mathbf{v}_{r+2},\ \ldots\ \mathbf{v}_n\}$$

  is an orthonormal basis for the null space of A^TA (and thus also of A)- Note that the eigenspace E_0 for A^TA is the null space of A.

- The row space of A therefore has the basis:

  $$\text{Row}(A) = \text{span}\{\mathbf{v}_1,\ \mathbf{v}_2,\ \ldots\ \mathbf{v}_r\}$$

- The column space of A is therefore r dimensional:

  $$\text{Col}(A) = \text{span}\{\mathbf{u}_1,\ \mathbf{u}_2,\ \ldots\ \mathbf{u}_r\}$$

- The null space of A^T is given by:

  $$\text{Null}(A^T) = \text{span}\{\mathbf{u}_{r+1},\ \mathbf{u}_{r+2},\ \ldots\ \mathbf{u}_m\}$$

The Reduced SVD

Again, let A be $m \times n$ with rank r.

If we're not interested in getting the null spaces, we can get what's called the reduced SVD:

$$A = \hat{U}\hat{\Sigma}\hat{V}^T$$

where $\hat{U}$ is $m \times r$ with o.n. columns (ordered by the singular values), $\hat{\Sigma}$ is $r \times r$ with non-zero values along the diagonal, and where $\hat{V}$ is $n \times r$ with o.n. columns.

When there is no room for confusion, the hats are sometimes left off.

Going back to our previous example, notice that we could have found the reduced SVD faster, since we're ignoring the zero eigenspaces:

$$A = \begin{bmatrix} -3 & 1 \\ 6 & -2 \\ 6 & -2 \end{bmatrix} = \begin{bmatrix} 1/3 \\ -2/3 \\ -2/3 \end{bmatrix} 3\sqrt{10} \begin{bmatrix} -3/\sqrt{10} \\ 1/\sqrt{10} \end{bmatrix}^T = \begin{bmatrix} 1 \\ -2 \\ -2 \end{bmatrix} [-3, 1]$$

A Third Way

The SVD can be full or reduced, and we also have a decomposition similar to the spectral decomposition:

$$A = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \cdots + \sigma_r \mathbf{u}_r \mathbf{v}_r^T$$

Since our last example had rank 1, we saw this factorization there.

The SVD and Matlab

The basic command is:

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

If we only want the reduced SVD,

```
[U,S,V]=svd(A,'econ')
```

If we want to compute a partial sum:

$$\sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \cdots + \sigma_r \mathbf{u}_k \mathbf{v}_k^T$$

In Matlab this would be:

```
U(:,1:k)*S(1:k,1:k)*V(:,1:k)';
```

The SVD and Collections of Data

Suppose we have some data in very high dimensional space- For example, 300 vectors in $\mathbb{R}^{100000}$. Because there are few vectors, we don't need all 100000 vectors to form a basis- We may want to *approximate* the data using a smaller dimensional basis (just a few basis vectors).

As it turns out, the SVD gives the "best basis" by using the first k columns of U (there is a caveat- We usually subtract the mean from the data). For example, suppose A is 100000×300. The mean of the 300 vectors is also a vector in $\mathbb{R}^{100000}$, so we compute it and subtract it from each column of A. In Matlab, this would be:

```
A=randn(100000,300);  %Create random data to work with
mA=mean(A,2);  %mA is a column vector of means
Ahat=A-repmat(mA,1,300);
```

Now we take the SVD of A. The "reduced" representation of the data is found by projecting the data into the first k columns of U:

```
[U,S,V]=svd(Ahat,'econ');
Recon=U(:,1:k)*(U(:,1:k)'*Ahat);  %Projection
```

Of course, it's hard to visualize these large vectors, so after the next brief section, we'll try to work with some movie data.

The SVD and Solving $A\mathbf{x} = \mathbf{b}$

The SVD is how Matlab solves the system of equations for the least squares solution. Let's see how- Suppose we have the **reduced** SVD, and the equation below:

$$A\mathbf{x} = \mathbf{b} \quad \Rightarrow \quad U\Sigma V^T\mathbf{x} = \mathbf{b} \quad \Rightarrow \quad VV^T\mathbf{x} = V\Sigma^{-1}U^T\mathbf{b}$$

If we want a least squares solution, the left side of the equation is $\hat{x}$, and the **pseudoinverse** of the matrix A is:

$$A^\dagger = V\Sigma^{-1}U^T$$

Example in Matlab

```
% Make some test data:
x=10*rand(20,1);  x=sort(x);
y=3+4*x-2*x.^2+randn(size(x));

% Design matrix:
A=[ones(20,1) x x.^2];
[U,S,V]=svd(A,'econ');
```

```
% Least Square Solution using the PseudoInverse
c=V*diag(1./diag(S))*U'*y

% Plot results using new data (in t)
t=linspace(0,10);
yout=c(1)+c(2)*t+c(3)*t.^2;
plot(x,y,'*',t,yout,'k-')
```

SVD over a Collection of Photos

Suppose we have a collection of photographs, each of which are 640×480 pixels. Then, by concatenating the columns of the image, it can be represented as a vector in $\mathbb{R}^{640 \cdot 480}$, or $\mathbb{R}^{307200}$.

Given a collection of such images, we can translate them into a collection of vectors in this high dimensional space. Assuming that the faces occupy a very small region of this high dimensional space, we ought to be able to find a relatively small dimensional subspace W that captures most of the data. Such a subspace would a low dimensional approximation to the column space of photos.

We recall that the SVD gives us a way of constructing such a basis. For example, if we have 200 photos that are 640×480, we can translate that data into a single array that is 307200×200, and so the left singular vectors in U would form a basis for the column space.

Each of these vectors would themselves be vectors in $\mathbb{R}^{307200}$, so they could also be visualized as a 640×480 photograph. That what we'll do below, except that the "collection" is actually a short webcam video.

Download the data `author.mat` from the class website. The webcam consists of 109 frames, where each frame is 120×160, or 19200 pixels. Thus, loading the data will give you an array that is 19200×109.

Also, we should note that the data in matrix, saved as `Y1`, consists of integer values between 0 and 255 (representing 255 shades of gray from black (0) to white (255)). This is important to mention: we'll be translating these images into vectors of real numbers instead of integers, which may impact how the coloring takes place. Discussing this in detail now would take us too far away from the discussion, but is something to keep in mind.

Visualizing a vector as an array

When visualizing these high dimensional vectors, remember that any vector in $\mathbb{R}^{19200}$ could be visualized as a 120×160 array. In Matlab, this is performed by using the "reshape" command. We use `imagesc`, short for "image scaled", where Matlab will scale the values in the array to ready it for the color map. To see how the color mapping is performed, you can use the `colorbar` command.

In this case, if $\vec{u}$ is a vector in $\mathbb{R}^{19200}$, then we can reshape it into an array and visualize it as an array using a gray scale color scheme by issuing the following commands, issued on one line

```
imagesc(reshape(u,120,160)); colormap(gray); axis equal; colorbar
```

To watch the "movie", you can use the following code, which just goes through the data column by column and shows each as frame. I won't use the color bar, and the pause command is there so we can actually see the result (otherwise, the frames go by way too fast!).

```
for j=1:109
  A=reshape(Y1(:,j),120,160);
  imagesc(A); colormap(gray); axis equal; axis off;
pause(0.1)
end
```

A few sample frames and the movie's mean frame are shown in Figure 1. You might find it interesting that that the mean is the background. Once we've found the mean, then we can proceed with the Singular Value Decomposition. Now consider the following code "snippet":

```
X=Y1-repmat(Ymean,1,109);
[U,S,V]=svd(X,'econ');
```

In the script file, we'll compare a rank 2 reconstruction to rank 10 and rank 22- This is done by using two, 10, then 22 basis vectors for the subspace in which the data lies.

First, let's examine the actual basis vectors. From our code snippet, the basis vectors of interest are the columns of U (the left singular vectors of X). Since each basis vector is in $\mathbb{R}^{19200}$, the basis vector(s) can also be visualized as an array.

Figure 1: The figure above has three sample frames (Frames 23, 60 and 80) and the movie's mean.

Sample Code

```
%% Sample script for the "author" data

% Some cleaning commands:
close all; clear; clc;

% Download "author.mat" from the class website, so it can be loaded here:
load author
Y1=double(Y1);  %Changes the data from "integers" to "floating point"
                %The data was saved as integers to save on memory.

% We can visualize the "movie"
figure(1)
for j=1:109
   A=reshape(Y1(:,j),120,160);
   imagesc(A); colormap(gray); axis equal; axis off;
   pause(0.1)
end

%% Mean subtraction
Ymean=mean(Y1,2);
figure(2)
imagesc(reshape(Ymean,120,160)); colormap(gray); axis equal; axis off;
```

```
title('The Movie Mean');

%% The SVD
% We'll mean subtract, but keep the original data intact.  We'll call put
% the mean subtracted data in matrix X.
X=Y1-repmat(Ymean,1,109);
[U,S,V]=svd(X,'econ');

%% Projection to R^2
Xcoords=U(:,1:2)'*X;   %The matrix Xcoords is now 2 x 109
figure(3)
plot(Xcoords(1,:),Xcoords(2,:),'.');
title('The movie data presented in the best two dimensional basis');

%% The Reconstruction back into 19200 dim space
Xrecon=U(:,1:2)*Xcoords;  %Add back the mean if needed.

%% We can play back the reconstructed movie with the mean added back in:
figure(4)
for j=1:109
   A=reshape(Xrecon(:,j)+Ymean,120,160);
   imagesc(A); colormap(gray); axis equal; axis off;
   pause(0.1)
end
```

Homework

1. Exercises 1, 2 p. 481: Find the singular values of the matrices below:

$$\begin{bmatrix} 1 & 0 \\ 0 & -3 \end{bmatrix} \qquad \begin{bmatrix} -5 & 0 \\ 0 & 0 \end{bmatrix}$$

2. Exercises 7, 9 p. 481: Construct the SVD of each matrix below (by hand):

$$\begin{bmatrix} 2 & -1 \\ 2 & 2 \end{bmatrix} \qquad \begin{bmatrix} 7 & 1 \\ 0 & 0 \\ 5 & 5 \end{bmatrix}$$

3. Exercise 14, p. 481: Find a vector $\mathbf{x}$ at which $A\mathbf{x}$ has maximum length.

4. Suppose $A = U\Sigma V^T$ is the (full) SVD.

   (a) Suppose A is square and invertible. Find the SVD of the inverse.

(b) If A is square, show that $|\det(A)|$ is the product of the singular values.

(c) If A itself is symmetric, show that $U = V$ so that the SVD gives the eigenvalue-eigenvector factorization: PDP^T.

5. Let A, $\mathbf{b}$ be as defined below. Use the SVD to write $\mathbf{b} = \hat{\mathbf{b}} + \mathbf{z}$, where $\hat{\mathbf{b}} \in \text{Col}(A)$ and $\mathbf{z} \in \text{Null}(A)$.

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

Also write in Matlab how you would check to see if $\hat{\mathbf{b}}$ is actually in the column space of A (using the output of the SVD).

6. (Exercise 12, 6.5) Given the data below, and the model equation

$$\beta_0 + \beta_1 \ln(w) = p$$

form a linear systems of equations to find the constants β_0, β_1, and find them by computing the pseudoinverse (using an appropriate SVD).

w	48	61	81	113	131
p	91	98	103	110	112

7. Consider the matrix A below. We have two algorithms that will produce a basis for the column space- Gram-Schmidt and the SVD. Use Matlab to get both, then compare and contast the two bases. Hint: What vector comes first? Can we compare one, two or three dimensional subspaces?

$$A = \begin{bmatrix} 1 & 3 & 5 \\ 1 & 1 & 0 \\ 1 & 1 & 2 \\ 1 & 3 & 3 \end{bmatrix}$$

8. (Using the matrix A from the previous problem) Here is some data in $\mathbb{R}^4$ that we organize in an array (think

$$X = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & -1 & 1 & -1 \end{bmatrix}$$

Find the projection of this data into the space spanned by the first two columns of the matrix U from the SVD of A. It is OK to write only two significant digits if you're copying the output by hand.

9. We want to find a matrix P so that, given matrix A and vector $\mathbf{y}$, then $P\mathbf{y}$ is the projection of $\mathbf{y}$ into the column space of A (orthogonal projection).

   We know that we could do this with the SVD, but using the normal equations and assuming that A is full rank, show that the matrix is:

$$P = A(A^T A)^{-1} A^T$$

   (Hint: Start by writing $A\mathbf{x} = \mathbf{b}$ and get the normal equation.

10. Work through the movie example from the text- Be sure to run it in Matlab.