Consider that we have a matrix A whose columns are m observations and each of n rows are features. In this way we have non square matrix. Like we can factor numbers into product of primes, Singular vector decomposition method, or SVD in short, suggests that we can factor any matrix into components. Accordingly, any matrix can be factorized into .

Let’s look at matrices and . Both matrices are symmetric, square, at least positive semidefinite (have eigenvalues which are zeros or positive numbers), both have equal positive eigenvalues, and the rank r equal to the rank of original matrix A.

Moreover, let’s assume that the column vectors are the eigenvectors of , and column vectors are the eigenvectors of . Both matrices have same positive eigenvalues. The square roots of these eigenvalues are singular values.

If we now construct the matrices U and V using the corresponding column vectors comprising eigenvectors (after normalization), it is easy to show that  and , given that the rows of these matrices are orthonormal.

Let’s get back to the proposition of SVD that states (for any matrix A). Assume now that the columns of are ordered from largest to smallest (and so are the eigenvectors). To prove the proposition , we need to solve for three unknowns, namely: , , . Let’s write few facts:

We can now write and as follows:

Or after some algebraic manipulations we get:

It is easy to see that comprises eigenvectors and are eigenvalues of . In a similar fashion we can find U:

U and V are now vector of eigenvectors of and correspondingly, and are eigenvalues (note, the positive eigenvalues are equal for both matrices).

Now, let’s derive a useful equation. Since we have:

Therefore, we can obtain:

The above equation shows that we can factorize the entire matrix A into so called atoms. Moreover, we can exclude eigenvectors for which is too small. In this manner we can approximate the matrices with some loss of information.

A picture worth the thousand words. Let’s show the beauty of SVD using real example. We will use **Octave** to demonstrate how image compression can be done easily with SVD.

Here is the code we have used to compress the image:

[I, map] = imread ("nature.bmp");

J = rgb2gray(I);

function [Uc, Sc, Vc] = compress\_matrix(A, N)

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

Uc = U(:, 1:N);

Sc = S(1:N, 1:N);

Vc = V(:, 1:N);

end

[Uc, Sigmac, Vc] = compress\_matrix(J, 20);

Jc20 = uint8(Uc \* Sigmac \* Vc');

[Uc, Sigmac, Vc] = compress\_matrix(J, 50);

Jc50 = uint8(Uc \* Sigmac \* Vc');

[Uc, Sigmac, Vc] = compress\_matrix(J, 100);

Jc100 = uint8(Uc \* Sigmac \* Vc');

figure

subplot(2,2,1)

xlabel("original")

imshow(J)

subplot(2,2,2)

xlabel("Compressed (first 20 components)")

imshow(Jc20)

subplot(2,2,3)

xlabel("Compressed (first 50 components)")

imshow(Jc50)

subplot(2,2,4)

xlabel("Compressed (first 100 components)")

imshow(Jc100)

Here is how the result looks like for original image, compressed image using first 20, 50 and 100 components:

