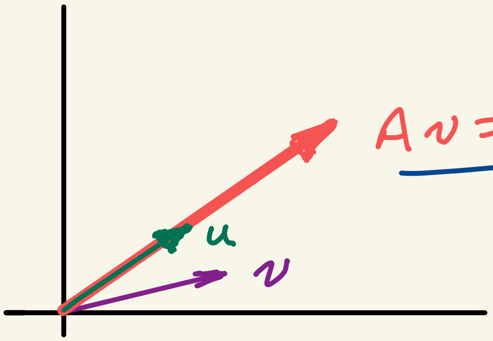
# Section 6.10.7
# Singular Value Decomposition (SVD)



$\underline{Av = \sigma u}$

$\|v\| = 1$

$\|u\| = 1$

$\|Av\| = \sigma$

$$\underset{\underline{m\times n}}{A}\begin{bmatrix} | & | & & | \\ v_1 & v_2 & \cdots & v_n \\ | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & \\ \sigma u_1 & \sigma u_2 & \cdots \\ | & | & \end{bmatrix}$$

$$\begin{bmatrix} | & | & \\ u_1 & u_2 & \cdots \\ | & | & \end{bmatrix}\begin{bmatrix} \sigma_1 & & 0 \\ 0 & \sigma_2 & \\ & & \ddots \ \sigma_m \end{bmatrix}$$

$AV = U\Sigma$

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

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

$U, V \perp$

$U^{-1} = U^T$

$V^{-1} = V^T$

$A^TA$ , $AA^T$

$$\boxed{S = Q\Lambda Q^T}$$

$$A^TA = (U\Sigma V^T)^T U\Sigma V^T$$
$$= V\Sigma^T \cancel{U^T U}\Sigma V^T$$
$$= V\Sigma^T\Sigma V^T = V\Sigma^2 V^T$$

$$\begin{bmatrix} \sigma_1^2 & & 0 \\ & \sigma_2^2 & \\ 0 & & \ddots \end{bmatrix}$$

V is eigenvector matrix of $A^TA$

$\Sigma$ is $\sqrt{\text{eigvalues of } A^TA}$

---

$$AA^T = (U\Sigma V^T)(U\Sigma V^T)^T$$
$$= U\Sigma \cancel{V^T V}\Sigma^T U^T$$
$$= U\underline{\Sigma\Sigma^T} U^T = U\Sigma^2 U^T$$

U is eigvectors of $AA^T$

Same eigenvalues as $A^TA$

$\Rightarrow$ same $\Sigma$

### 6.10.7 Singular Value Decomposition (SVD)

Here we will look at yet another factoring of a matrix. This factoring is perhaps the most important factoring for two reasons.

1. It (SVD) works for any matrix, even singular and non-square matrices.
2. SVD has application to artificial intelligence and data analytics. A statistical analysis algorithm known as *Principal Component Analysis (PCA)* relies on SVD.

Recall that in our introduction to *Application of Eigenvalues and Eigenvectors* that multiplication of a matrix vector stretches and rotates the vector. As before, we want to factor out the stretching and rotating component of the matrix. For a m-by-n matrix $\boldsymbol{A}$ and a n-by-1 vector $\boldsymbol{v}$, we have:

$$\boldsymbol{A}\,\boldsymbol{v} = \sigma\,\boldsymbol{u}$$

where $\sigma$ is a scalar that is the length of the resulting vector and $\boldsymbol{u}$ is a unit vector that shows the direction of the resulting vector.

Extending this to sets of vectors to accommodate the columns of $\boldsymbol{A}$, we have the following matrix relationships.

**Note:** The following matrix equation is written as for a square matrix (m = n). For non-square matrices, the $\boldsymbol{\Sigma}$ matrix needs to be padded with either rows or columns of zeros to accommodate the matrix multiplication requirements. We will see examples of this in the MATLAB examples. The number of non-zero $\sigma$s is r, the rank of $\boldsymbol{A}$, which is usually the lesser of m and n.

$$\begin{aligned}
\boldsymbol{A}\,\boldsymbol{V} &= \boldsymbol{A}\begin{bmatrix}\boldsymbol{v_1} & \boldsymbol{v_2} & \cdots & \boldsymbol{v_n}\end{bmatrix} \\
&= \begin{bmatrix}\sigma_1\,\boldsymbol{u_1} & \sigma_2\,\boldsymbol{u_2} & \cdots & \sigma_m\,\boldsymbol{u_m}\end{bmatrix} \\
&= \begin{bmatrix} | & | & & | \\ \boldsymbol{u_1} & \boldsymbol{u_2} & \cdots & \boldsymbol{u_m} \\ | & | & & | \end{bmatrix}
\begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_m \end{bmatrix} \\
&= \boldsymbol{U}\,\boldsymbol{\Sigma}
\end{aligned}$$

Then a factoring of $\boldsymbol{A}$ is just

$$\boldsymbol{A} = \boldsymbol{U}\,\boldsymbol{\Sigma}\,\boldsymbol{V^{-1}}.$$

The factoring finds two rotation matrices ($\boldsymbol{U}$ and $\boldsymbol{V^{-1}}$) and a diagonal stretching matrix ($\boldsymbol{\Sigma}$), which are called the *singular values*. But we still have not yet really stated anything significant or useful about $\boldsymbol{A}$ because we don't know what the sub-matrices are. We need to put some restrictions on the $\boldsymbol{V}$ matrix so that we have a strategy to compute sub-matrices that are really useful. You may have correctly guessed that eigenvectors and eigenvalues will come into play at this point.

We will require that the $\boldsymbol{V}$ matrix be *orthogonal*, which means that it is square with columns that are both unit vectors and orthogonal to each other. With the orthogonal requirement, $\boldsymbol{V}^{-1} = \boldsymbol{V}^T$ (see the spectral theorem in *Matrix Transpose Properties*).

The SVD factoring is:

$$\boxed{\boldsymbol{A} = \boldsymbol{U}\,\Sigma\,\boldsymbol{V}^T}$$

The size of each matrix is: $\boldsymbol{A}$: m-by-n, $\boldsymbol{A}\,\boldsymbol{V}$: m-by-n, $\boldsymbol{U}$: m-by-m, $\Sigma$: m-by-n, and $\boldsymbol{V}$: n-by-n.

Let us now consider matrices $\boldsymbol{A}^T\,\boldsymbol{A}$ and $\boldsymbol{A}\,\boldsymbol{A}^T$, which are always square, symmetric, semi-positive definite, have real eigenvalues, and have eigenvectors that are both real and orthogonal. The size of $\boldsymbol{A}^T\boldsymbol{A}$ is n-by-n, while the size of $\boldsymbol{A}\,\boldsymbol{A}^T$ is m-by-m. When we use the factoring to find $\boldsymbol{A}^T\boldsymbol{A}$ and $\boldsymbol{A}\,\boldsymbol{A}^T$, we can take advantage of a simplification and also discover what the sub-matrices should be.

$$\begin{aligned}
\boldsymbol{A^T A} &= \left(\boldsymbol{U}\,\Sigma\,\boldsymbol{V^T}\right)^{\boldsymbol{T}}\,\boldsymbol{U}\,\Sigma\,\boldsymbol{V^T} \\
&= \boldsymbol{V}\,\Sigma^{\boldsymbol{T}}\,\boldsymbol{U^T}\,\boldsymbol{U}\,\Sigma\,\boldsymbol{V^T} \\
&= \boldsymbol{V}\,\Sigma^{\boldsymbol{T}}\,\Sigma\,\boldsymbol{V^T} \\
&= \boldsymbol{V}\,\Sigma^{\boldsymbol{2}}\,\boldsymbol{V^T} \\
&= \boldsymbol{V}\begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_r^2 \end{bmatrix}\boldsymbol{V^T}
\end{aligned}$$

This factoring is the diagonalization of a symmetric matrix (see *Diagonalization and Powers of A*), thus it follows that the $\boldsymbol{V}$ matrix comes from the eigenvectors of $\boldsymbol{A}^T\boldsymbol{A}$. Likewise, the $\Sigma$ matrix is the square root of the diagonal eigenvalue matrix of $\boldsymbol{A}^T\boldsymbol{A}$.

Similarly, for the $\boldsymbol{U}$ matrix:

$$\begin{aligned}
\boldsymbol{A\,A^T} &= \boldsymbol{U}\,\Sigma\,\boldsymbol{V^T}\,\left(\boldsymbol{U}\,\Sigma\,\boldsymbol{V^T}\right)^{\boldsymbol{T}} \\
&= \boldsymbol{U}\,\Sigma^{\boldsymbol{T}}\,\boldsymbol{V^T}\,\boldsymbol{V}\,\Sigma\,\boldsymbol{U^T} \\
&= \boldsymbol{U}\,\Sigma^{\boldsymbol{T}}\,\Sigma\,\boldsymbol{U^T} \\
&= \boldsymbol{U}\,\Sigma^{\boldsymbol{2}}\,\boldsymbol{U^T} \\
&= \boldsymbol{U}\begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_r^2 \end{bmatrix}\boldsymbol{U^T}
\end{aligned}$$

So the $\boldsymbol{U}$ matrix is the eigenvector matrix of $\boldsymbol{A}\,\boldsymbol{A}^T$. The eigenvalues ($\Sigma^2$) in both cases are the same, so we just take the square root of the eigenvalues to get the $\Sigma$ matrix.

### SVD Example - Square, non-singular

```
>> A = [4 4; -3 3]
A =
     4     4
    -3     3
```

```
>> A'*A
ans =
    25      7
     7     25
>> A*A'
ans =
    32      0
     0     18
>> [V, S2] = eig(A'*A)
V =
  -0.7071    0.7071
   0.7071    0.7071
S2 =
    18   0
    0    32
>> Sigma = sqrt(S2)
Sigma =
      4.2426    0
      0         5.6569
>> [U, S2] = eig(A*A')
U =
    0      1
    1      0
S2 =
    18     0
    0    32
>> SVD = U * Sigma * V'
SVD =
    4.0000    4.0000
   -3.0000    3.0000
```

### SVD Example - Square, singular

As expected for a singular matrix, one of the eigenvalues is zero.

```
>> A = [2 3; 4 6]
A =
    2      3
    4      6
>> rank(A)
ans =
    1
>> A'*A
ans =
    20     30
    30    45
>> A*A'
ans =
    13    26
    26    52
>> [V,S2] = eig(A'*A)
V =
  -0.8321    0.5547
   0.5547    0.8321
S2 =
     0     0
     0    65
>> Sigma = sqrt(S2)
Sigma =
        0         0
        0    8.0623
>> [U,S2] = eig(A*A')
U =
   -0.8944    0.4472
    0.4472    0.8944
S2 =
     0     0
     0    65
>> SVD = U * Sigma * V'
SVD =
    2     3
    4     6
```

Non-zero $\sigma$s is rank(A)

### SVD Example - Rectangular

Here, I used the built-in `svd` MATLAB function. Notice that MATLAB sorted the results so that the singular values, $\sigma$s, are sorted in descending order. The eigenvectors in $\boldsymbol{U}$ and $\boldsymbol{V}$ are also sorted to match their corresponding singular values.

Sorting the results is useful for two applications. We need this for a type of data filtering achieved by *Dimensionality Reduction*, which is discussed later in this section. Secondly, it is used for *Principal Component Analysis (PCA)* (PCA), which is covered in the next section. With PCA, we want to find the primary contributing component towards some final result. The largest singular values identifies the most significant principal components. Knowing the principal components helps us to identify samples based on their most significant features (for example, face recognition).

```
>> A = [2 3; 4 10; 5 12]
A =
    2     3
    4    10
```

3x2

```
    5    12
>> rank(A)
ans =
    2
>> [U, Sigma, V] = svd(A)
U =
 -0.2053    0.9649   -0.1638
 -0.6243   -0.2580   -0.7373
 -0.7537   -0.0490    0.6554
Sigma =
      17.2482    0
        0        0.7077
        0        0
V =
 -0.3871    0.9220
 -0.9220   -0.3871
>> SVD = U * Sigma * V'
SVD =
    2.0000    3.0000
    4.0000   10.0000
    5.0000   12.0000
```

m>n

A - m×n

U - m×m

Σ - m×n

V - n×n

$\left.\begin{array}{l} U \\ \Sigma \end{array}\right\} m \times n$

$\left.\begin{array}{l} \Sigma \\ V \end{array}\right\} m \times n$

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \end{bmatrix}$$

m<n

6.10.7.4

### How Does SVD Change Vectors ?

```
function show_SVD(A)
% SHOW_SVD - A demononstration of how the U, S, and V matrices
%   from SVD rotate, stretch, and rotate vectors making a circle.
%   Try several 2-by-2 matrices to see how each behaves.

theta = linspace(0, 2*pi, 30);
x = [cos(theta); sin(theta)];

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

Vx = V'*x;
figure, plot(x(1,:), x(2,:), '*')
hold on
scatter(Vx(1,:), Vx(2,:))
for z = 1:30
    line([x(1,z), Vx(1,z)], [x(2,z), Vx(2,z)], 'Color', 'k')
end
title('Rotation by V^T Matrix')
hold off

svx = S*Vx;
```

```
figure, scatter(svx(1,:), svx(2,:))
daspect([1 1 1])
usvx = U*svx;
hold on
scatter(usvx(1,:), usvx(2,:))
title('Stretch by \Sigma and rotation by U')
hold off
```

6.10.7.5

### Computational Issues

- When $A^T A$ has repeating eigenvalues, then the corresponding $U$ and $V^T$ matrices may not match up as desired, so it may be desired to calculate $U$ as

  $AV = U\Sigma$

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

  $\begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}^{-1} = \begin{bmatrix} 1/\sigma_1 & 0 \\ 0 & 1/\sigma_2 \end{bmatrix}$

  In MATLAB, the matrix right-divide operator can be used.

```
U = (A*V)/Sigma;
```

- Some applications may have very large $A$ matrices (even hundreds of thousands of rows and columns), thus calculating $A\,A^T$ might take a very long time and risks having unacceptable round-off errors. Thus, the best software programs like MATLAB, will use other, more advanced algorithms.
- If the eigenvectors of $A^T\,A$ and $A\,A^T$ are computed independent of each other, there can be a problem with certain columns of $U$ or $V$ needing to be multiplied by -1 for the factoring to be correct. Thus, it is best to only do one eigenvector calculation. Below is a simple function that will correctly find the SVD factoring. As the comments indicate, the code shown here is not how MATLAB computes SVD. This function is just for educational purposes.

```
function [U, S, V] = mySVD(A)
% MYSVD - an implementation of Singular Value Decomposition (SVD)
%
% [U,S,V] = SVD(A) produces a diagonal matrix S, of the same
% dimension as A and with nonnegative diagonal elements in
% decreasing order, and unitary matrices U and V so that A = U*S*V'.
%
% MATLAB comes with a svd() function, which should normally be used.
% This is just an example showing how the SVD can be found from
% eigenvalues and eigenvectors. The algorithm that MATLAB's svd()
% function uses is more complex and computationally more efficient.
%
% Note that we sort the eigenvalues and eigenvalues, which become the
% singular values to make sure that they are in the right order.
%
% We compute the S and V matrix from the eigenvalues and eigenvectors
% of A'*A, but do not compute eigenvectors to find U. The eigenvectors
% from the eig() function can be multiplied by -1 and they are still
% eigenvectors. So if U and V were computed independently as
% eigenvectors, some columns of U might have the wrong sign in
% relation to the columns of V.
%
% A and S are mxn. A'*A and V are nxn. A*A' and U are mxm.
%
    [m, n] = size(A);
    [V2, S2] = eig(A'*A);
    [S2, order] = sort(diag(S2), 'descend');
    V = V2(:, order);

    % There are no negative eigenvalues of A'*A, but we still need
    % abs() in the next line just because a 0 can be -0, giving an
    % undesired complex result.

    S = diag(sqrt(abs(S2)));
    if m > n                         % pad S with zeros if needed
        z = zeros(m-n, n);
        S = [S; z];
    elseif n > m
        z = zeros(m, n-m);
        S = [S(1:m, 1:m) z];
    end
    U = (A*V)/S;
end
```

6.10.7.6

### Polar Decomposition

There is another factoring of a matrix that uses the sub-matrices of SVD. The *polar decomposition* splits the matrix up into a symmetric matrix and a rotation matrix. It is found by simply inserting an identity matrix in the form of $\boldsymbol{U^T U}$ into the SVD equation.

$$\begin{aligned} \boldsymbol{A} &= \boldsymbol{U}\,\boldsymbol{\Sigma}\,\left(\boldsymbol{U^T U}\right)\,\boldsymbol{V^T} \\ &= \left(\boldsymbol{U}\,\boldsymbol{\Sigma}\,\boldsymbol{U^T}\right)\left(\boldsymbol{U}\,\boldsymbol{V^T}\right) \\ &= \boldsymbol{S}\,\boldsymbol{Q} \end{aligned}$$

$$\boldsymbol{S} = \boldsymbol{U}\,\boldsymbol{\Sigma}\,\boldsymbol{U^T}$$

$$\boldsymbol{Q} = \boldsymbol{U}\,\boldsymbol{V^T}$$

6.10.7.7

### Dimensionality Reduction

Since the $\Sigma$ matrix is a diagonal matrix, the SVD equation can be written as a sum of matrix products. That is to say, the *outer product* view of matrix multiplication. See *Matrix Multiplication*