

# Linear Algebra Review

Fei-Fei Li

# Vectors

- Vectors and matrices are just collections of ordered numbers that represent something: movements in space, scaling factors, pixel brightnesses, etc.
- A vector is a quantity that involves both magnitude and direction
- A column vector  $v \in \mathbf{R}^{n \times 1}$  where

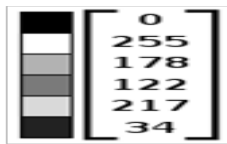
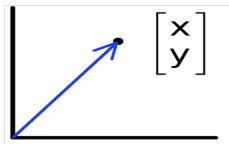
$$v = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad (1)$$

- A row vector  $v^T \in \mathbf{R}^{n \times 1}$  where

$$v^T = [x_1 \quad x_2 \quad \dots \quad x_n] \quad (2)$$

# Vectors

- Vectors can represent an offset in 2D or 3D space
- Points are just vectors from the origin



- Data (pixels, gradients at an image keypoint, etc) can also be treated as a vector
- Such vectors don't have a geometric interpretation, but calculations like "distance" can still have value

# Vectors

```
>> v = [ 9 ; 3 ; 4 ; 1 ; 0]
```

```
v =
```

```
9  
3  
4  
1  
0
```

```
>> vT = v'
```

```
vT =
```

```
9      3      4      1      0
```

```
>> v2 = [ 2 4 2.1 5 4]
```

```
v2 =
```

```
2.0000    4.0000    2.1000    5.0000    4.0000
```

```
>> v2T = v2'
```

```
v2T =
```

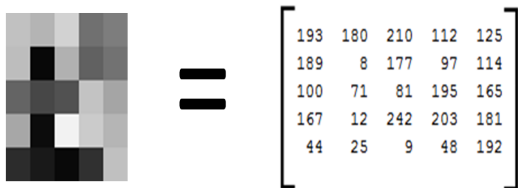
```
2.0000  
4.0000  
2.1000  
5.0000  
4.0000
```

- A matrix  $A \in \mathbf{R}^{m \times n}$  is an array of  $m \times n$  numbers arranged in  $m$  rows and  $n$  columns.

$$A = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & x_{m3} & \dots & x_{mn} \end{bmatrix}$$

- If  $m = n$ , we say that  $A$  is a square matrix
- MATLAB represents an image as a matrix of pixel brightnesses

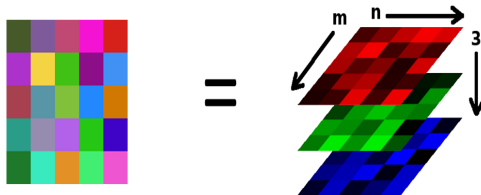
- Note that matrix coordinates are NOT Cartesian coordinates. The upper left corner is  $[y,x] = (1,1)$



- Grayscale images have one number per pixel, and are stored as an  $m \times n$  matrix.

# Images

- Color images have 3 numbers per pixel red, green, and blue brightnesses
- Stored as an  $m \times n \times 3$  matrix



# Matrix Operations

## Addition

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} a+1 & b+2 \\ c+3 & d+4 \end{bmatrix}$$

- Can only add a matrix with matching dimensions, or a scalar.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + 7 = \begin{bmatrix} a+7 & b+7 \\ c+7 & d+7 \end{bmatrix}$$

## Scaling

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times 3 = \begin{bmatrix} 3a & 3b \\ 3c & 3d \end{bmatrix}$$



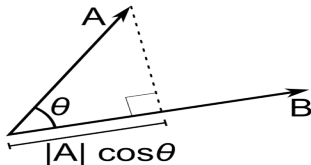
# Matrix Operations

## Inner product (dot product) of vectors

- Multiply corresponding entries of two vectors and add up the result
- $\mathbf{x}^T \mathbf{y}$  is also  $|\mathbf{x}| |\mathbf{y}| \cos(\theta)$  (the angle between  $\mathbf{x}$  and  $\mathbf{y}$ )

$$\mathbf{x}^T \mathbf{y} = \begin{bmatrix} x_1 & \dots & x_n \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \sum_{i=1}^n x_i y_i \quad (\text{scalar})$$

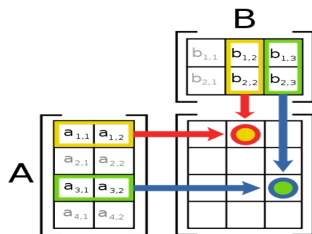
- If  $\mathbf{B}$  is a unit vector, then  $\mathbf{A} \cdot \mathbf{B}$  gives the length of  $\mathbf{A}$  which lies in the direction of  $\mathbf{B}$



# Matrix Operations

## Multiplication

- For  $A \cdot B$ ; each entry in the result is (that row of  $A$ ) dot product with (that column of  $B$ )



$$\begin{array}{ccc} A & \times & B \\ \downarrow & & \searrow \\ \begin{bmatrix} 0 & 2 \\ 4 & 6 \end{bmatrix} & & \begin{bmatrix} 1 & 3 \\ 5 & 7 \end{bmatrix} \\ & & \downarrow \\ \begin{bmatrix} 10 & 14 \\ 34 & 54 \end{bmatrix} \\ & & 0 \cdot 3 + 2 \cdot 7 = 14 \end{array}$$

# Matrix Operations

## Multiplication

```
>> A=floor(10*rand(3,4))
```

A =

0	6	3	4
0	1	4	8
4	4	5	9

```
>> B=floor(10*rand(4,5))
```

B =

6	8	6	2	5
2	3	1	5	1
0	9	7	5	9
1	1	4	7	0

```
>> R = A*B
```

R =

16	49	43	73	33
10	47	61	81	37
41	98	99	116	69

```
>> C=floor(10*rand(3,4))
```

C =

0	4	4	6
5	1	4	3
7	1	9	3

```
>> R2 = A.*C
```

R2 =

0	24	12	24
0	1	16	24
28	4	45	27

# Matrix Operations

## Transpose

- row  $r$  becomes column  $c$

$$\begin{bmatrix} 0 & 1 \\ 2 & 3 \\ 4 & 5 \end{bmatrix}^T = \begin{bmatrix} 0 & 2 & 4 \\ 1 & 3 & 5 \end{bmatrix}$$

- A useful identity

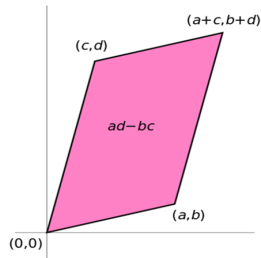
$$(ABC)^T = C^T B^T A^T$$

# Matrix Operations

## Determinant

- $\det(\mathbf{A})$  returns a scalar value for square matrix  $\mathbf{A}$ . For,

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \quad \det(\mathbf{A}) = ad - bc$$



- Properties:

$$\det(\mathbf{AB}) = \det(\mathbf{BA})$$

$$\det(\mathbf{A}^{-1}) = \frac{1}{\det(\mathbf{A})}$$

$$\det(\mathbf{A}^T) = \det(\mathbf{A})$$

$$\det(\mathbf{A}) = 0 \Leftrightarrow \mathbf{A} \text{ is singular}$$

# Special Matrices

## Identity matrix /

- Square matrix, 1s along diagonal, 0s elsewhere
- $I.A = A$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 3 & 0 & 0 \\ 0 & 7 & 0 \\ 0 & 0 & 2.5 \end{bmatrix}$$

## Diagonal matrix

- Square matrix with numbers along diagonal, 0s elsewhere
- $(A \text{ diagonal}).(Another matrix B)$  scales the rows of matrix B

# Special Matrices

## Symmetric matrix

$$\mathbf{A}^T = \mathbf{A} \quad \begin{bmatrix} 1 & 2 & 5 \\ 2 & 1 & 7 \\ 5 & 7 & 1 \end{bmatrix}$$

## Skew-symmetric matrix

$$\mathbf{A}^T = -\mathbf{A} \quad \begin{bmatrix} 1 & -2 & -5 \\ 2 & 1 & -7 \\ 5 & 7 & 1 \end{bmatrix}$$

# Transformations

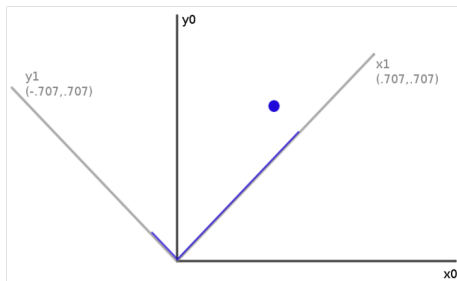
- Matrices can be used to transform vectors in useful ways, through multiplication:  $x' = Ax$
- Simplest is scaling:

$$\begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} s_x x \\ s_y y \end{bmatrix}$$



# Rotation

- How can you convert a vector represented in frame 0 to a new, rotated coordinate frame 1?
- Remember what a vector is: [component in direction of the frames x axis, component in direction of y axis]



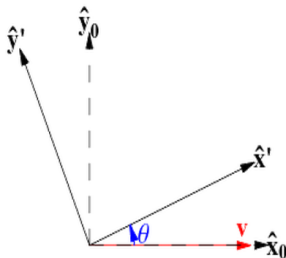
# Rotation

- So to rotate it we must produce this vector: [component in direction of new x axis, component in direction of new y axis]
- We can do this easily with dot products!
- New x coordinate is [original vector] dot [the new x axis]
- New y coordinate is [original vector] dot [the new y axis]

$$\begin{array}{ccc} R \times p = & & p \\ \text{rotated } p' & \nearrow & \begin{bmatrix} px \\ py \end{bmatrix} \\ & & \\ R & & \\ \begin{bmatrix} .707 & .707 \\ -.707 & .707 \end{bmatrix} & \searrow & \begin{bmatrix} px' \\ py' \end{bmatrix} \end{array}$$

# 2D Rotation Matrix Formula

- Counter-clockwise rotation by an angle  $\theta$



$$x' = x \cos \theta + y \sin \theta$$

$$y' = -x \sin \theta + y \cos \theta$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = R(\theta) \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\mathbf{P}' = \mathbf{R} \mathbf{P}$$

# Transformation Matrices

- Multiple transformation matrices can be used to transform a point:  
 $p' = R_2.R_1.S.p$
- The effect of this is to apply their transformations one after the other, from right to left.
- In the example above, the result is  $(R_2(R_1(S.p)))$
- The result is exactly the same if we multiply the matrices first, to form a single transformation matrix:  $p' = (R_2.R_1.S)p$

# Homogeneous System

- In general, a matrix multiplication lets us linearly combine components of a vector

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}$$

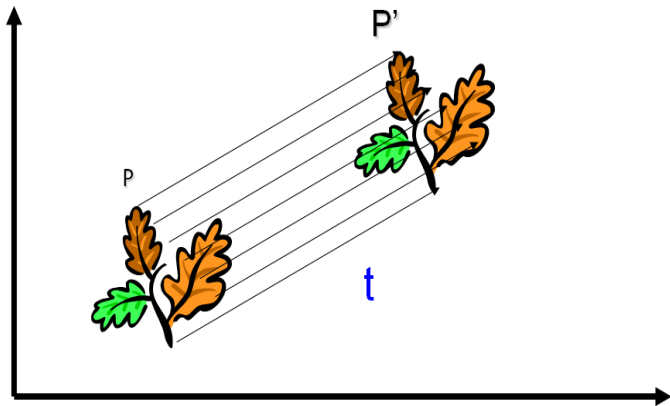
- This is sufficient for scale, rotate, skew transformations.
- But notice, we cant add a constant!
- The (somewhat hacky) solution? Stick a "1" at the end of every vector

# Homogeneous System

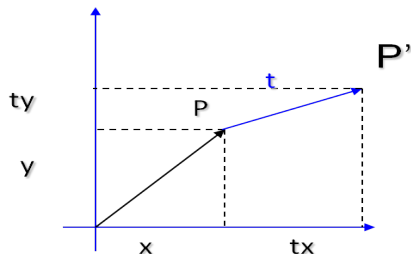
$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + c \\ dx + ey + f \\ 1 \end{bmatrix}$$

- Now we can rotate, scale, and skew like before, AND translate (note how the multiplication works out, above)
- This is called "homogeneous coordinates"
- In homogeneous coordinates, the multiplication works out so the rightmost column of the matrix is a vector that gets added

## 2D Translation



# 2D Translation with Homogeneous Coordinates



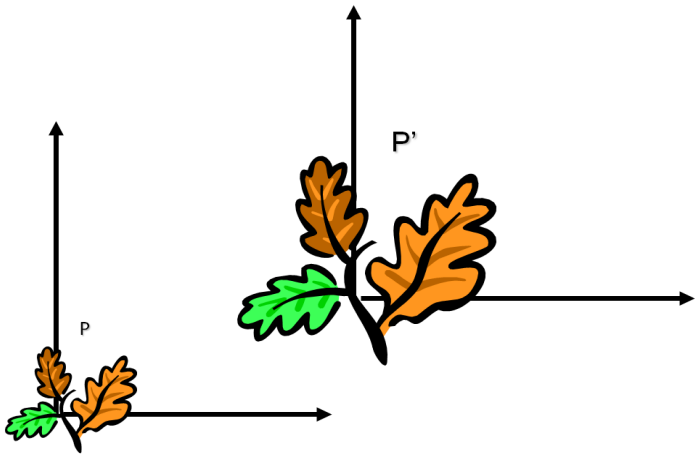
$$\mathbf{P} = (x, y) \rightarrow (x, y, 1)$$

$$\mathbf{t} = (t_x, t_y) \rightarrow (t_x, t_y, 1)$$

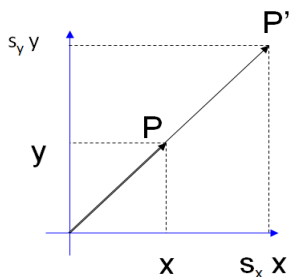
$$\begin{aligned} \mathbf{P}' &\rightarrow \begin{bmatrix} x+t_x \\ y+t_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{I} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \cdot \mathbf{P} = \mathbf{T} \cdot \mathbf{P} \end{aligned}$$



# Scaling



# Scaling



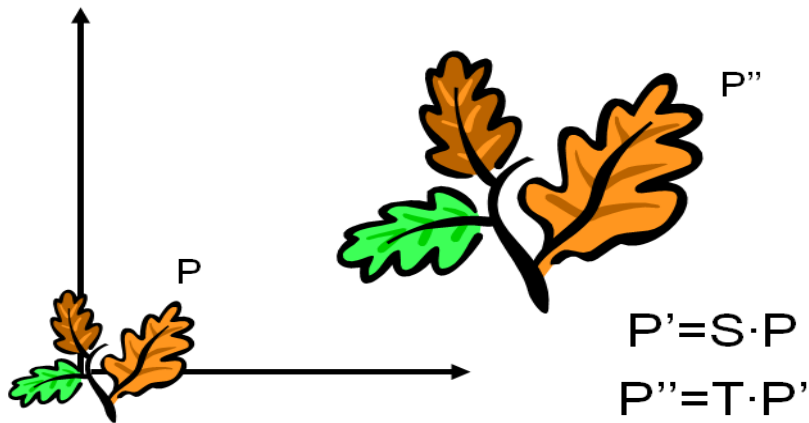
$$\mathbf{P} = (x, y) \rightarrow \mathbf{P}' = (s_x x, s_y y)$$

$$\mathbf{P} = (x, y) \rightarrow (x, y, 1)$$

$$\mathbf{P}' = (s_x x, s_y y) \rightarrow (s_x x, s_y y, 1)$$

$$\mathbf{P}' \rightarrow \begin{bmatrix} s_x x \\ s_y y \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{S}} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{S}' & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \cdot \mathbf{P} = \mathbf{S} \cdot \mathbf{P}$$

# Scaling & Translating



$$P'' = T \cdot P' = T \cdot (S \cdot P) = T \cdot S \cdot P = A \cdot P$$

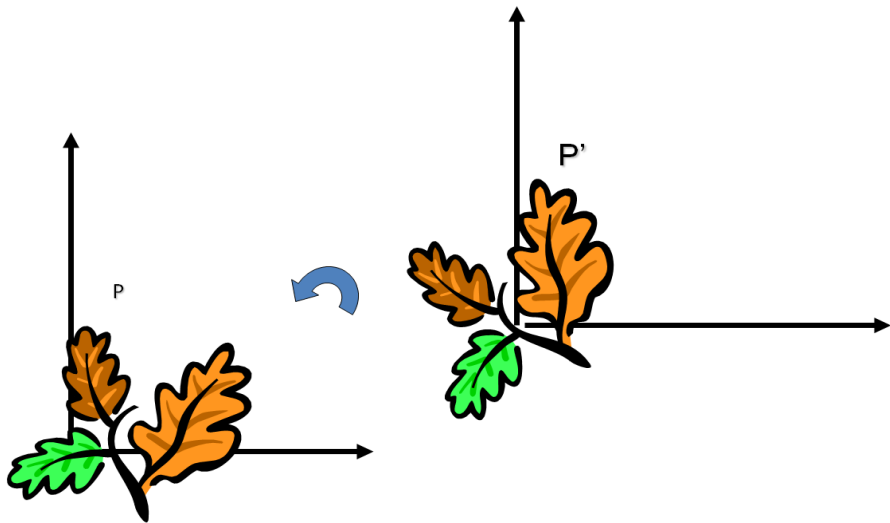
$$\begin{aligned}\mathbf{P}' &= \mathbf{T} \cdot \mathbf{S} \cdot \mathbf{P} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \\ &= \underbrace{\begin{bmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix}}_A \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x x + t_x \\ s_y y + t_y \\ 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} S & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}\end{aligned}$$

# Scaling & Translating

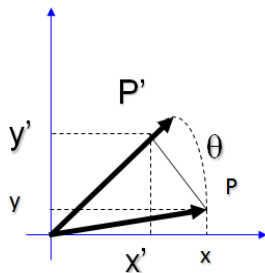
$$\mathbf{P}''' = \mathbf{T} \cdot \mathbf{S} \cdot \mathbf{P} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x x + t_x \\ s_y y + t_y \\ 1 \end{bmatrix}$$

$$\begin{aligned} \mathbf{P}''' = \mathbf{S} \cdot \mathbf{T} \cdot \mathbf{P} &= \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \\ &= \begin{bmatrix} s_x & 0 & s_x t_x \\ 0 & s_y & s_y t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x x + s_x t_x \\ s_y y + s_y t_y \\ 1 \end{bmatrix} \end{aligned}$$

# Rotation



Counter-clockwise rotation by an angle  $\theta$



$$x' = \cos \theta \, x - \sin \theta \, y$$

$$y' = \cos \theta \, y + \sin \theta \, x$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\mathbf{P}' = \mathbf{R} \, \mathbf{P}$$

# Rotation Matrix Properties

- Transpose of a rotation matrix produces a rotation in the opposite direction

$$RR^T = R^T R = I$$

$$\det(R) = 1$$

- Consider the rotation matrix from the previous slide



# Inverse of Matrix

- Given a matrix  $A$ , its inverse  $A^{-1}$  is a matrix such that  $AA^{-1} = A^{-1}A = I$

$$\begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}^{-1} = \begin{bmatrix} 1/2 & 0 \\ 0 & 1/3 \end{bmatrix}$$

- Inverse does not always exist. If  $A^{-1}$  exists,  $A$  is invertible or non-singular. Otherwise, its singular.
- Useful identities, for matrices that are invertible:

$$(\mathbf{A}^{-1})^{-1} = \mathbf{A}$$

$$(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$$

$$\mathbf{A}^{-T} \triangleq (\mathbf{A}^T)^{-1} = (\mathbf{A}^{-1})^T$$

- Pseudoinverse

- Say you have the matrix equation  $AX=B$ , where  $A$  and  $B$  are known, and you want to solve for  $X$
- You could use MATLAB to calculate the inverse and premultiply by it:  
 $A^{-1}AX = A^{-1}BX = A^{-1}B$
- MATLAB command would be `inv(A)*B`
- But calculating the inverse for large matrices often brings problems with computer floating-point resolution (because it involves working with very small and very large numbers together).
- Or, your matrix might not even have an inverse.
- Fortunately, there are workarounds to solve  $AX=B$  in these situations. And MATLAB can do them!

- Pseudoinverse

- Instead of taking an inverse, directly ask MATLAB to solve for  $X$  in  $AX=B$ , by typing  $A \setminus B$
- MATLAB will try several appropriate numerical methods (including the pseudoinverse if the inverse doesn't exist)
- MATLAB will return the value of  $X$  which solves the equation
- If there is no exact solution, it will return the closest one
- If there are many solutions, it will return the smallest one

- MATLAB example:

$$AX = B$$

$$A = \begin{bmatrix} 2 & 2 \\ 3 & 4 \end{bmatrix}, B = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

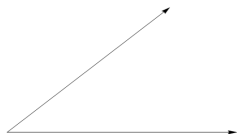
```
>> x = A\B  
x =  
    1.0000  
   -0.5000
```

# Rank of A Matrix

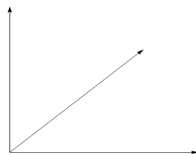
- Linear independence

- Suppose we have a set of vectors  $v_1, \dots, v_n$
- If we can express  $v_1$  as a linear combination of the other vectors  $v_2 \dots v_n$ , then  $v_1$  is linearly dependent on the other vectors.
- The direction  $v_1$  can be expressed as a combination of the directions  $v_2 \dots v_n$ . (E.g.  $v_1 = .7v_2 - .7v_4$ )
- If no vector is linearly dependent on the rest of the set, the set is linearly independent.
- Common case: a set of vectors  $v_1, \dots, v_n$  is always linearly independent if each vector is perpendicular to every other vector (and non-zero)

Linearly independent set



Not linearly independent



# Rank of A Matrix

- Column/row rank
  - $col - rank(A)$  = the max. number of linearly independent column vector of A
  - $row - rank(A)$  = the max. number of linearly independent row vector of A
- Column rank always equals row rank
- Matrix rank :  $rank(A) := col - rank(A) = row - rank(A)$
- For transformation matrices, the rank tells you the dimensions of the output
- E.g. if rank of A is 1, then the transformation  $p' = Ap$  maps points onto a line.

$$\begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x + y \\ 2x + 2y \end{bmatrix}$$

← All points get mapped to the line  $y=2x$

# Rank of A Matrix

- If an  $m \times m$  matrix is rank  $m$ , we say its "full rank"
  - Maps an  $m \times 1$  vector uniquely to another  $m \times 1$  vector
  - An inverse matrix can be found
- If  $rank < m$ , we say its "singular"
  - At least one dimension is getting collapsed. No way to look at the result and tell what the input was
  - Inverse does not exist
- Inverse also doesn't exist for non-square matrices

# Eigenvalues and Eigenvectors

- Suppose we have a square matrix  $A$ . We can solve for vector  $x$  and scalar  $\lambda$  such that  $Ax = \lambda x$
- In other words, find vectors where, if we transform them with  $A$ , the only effect is to scale them with no change in direction.
- These vectors are called eigenvectors (German for self vector of the matrix), and the scaling factors  $\lambda$  are called eigenvalues
- An  $m \times m$  matrix will have  $\leq m$  eigenvectors where  $\lambda$  is nonzero
- To find eigenvalues and eigenvectors rewrite the equation:

$$(A - \lambda I)x = 0$$

- $x = 0$  is always a solution but we have to find  $x \neq 0$ . This means  $A - \lambda I$  should not be invertible so we can have many solutions.

$$\det(A - \lambda I) = 0$$



# Eigenvalues and Eigenvectors

- For example;

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

$$\det(A - \lambda I)x = 0$$

$$\begin{vmatrix} 3 - \lambda & 1 \\ 1 & 3 - \lambda \end{vmatrix} = 0$$

then  $\lambda_1 = 4, x_1^T = [1 \ 1]$  and  $\lambda_2 = 2, x_2^T = [-1 \ 1]$

- Another example;

$$B = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

then  $\lambda_1 = 1, x_1^T = [1 \ 1]$  and  $\lambda_2 = -1, x_2^T = [-1 \ 1]$

- Relation between these two matrices ?

# Singular Value Decomposition (SVD)

- There are several computer algorithms that can factor a matrix, representing it as the product of some other matrices
- The most useful of these is the Singular Value Decomposition.
- Represents any matrix  $A$  as a product of three matrices:  $U\Sigma V^T$
- MATLAB command:  $[U,S,V]=\text{svd}(A)$

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

- where  $U$  and  $V$  are rotation matrices, and  $\Sigma$  is a scaling matrix.

$$\begin{matrix} U & & \Sigma & & V^T & & A \\ \begin{bmatrix} -.40 & .916 \\ .916 & .40 \end{bmatrix} & \times & \begin{bmatrix} 5.39 & 0 \\ 0 & 3.154 \end{bmatrix} & \times & \begin{bmatrix} -.05 & .999 \\ .999 & .05 \end{bmatrix} & = & \begin{bmatrix} 3 & -2 \\ 1 & 5 \end{bmatrix} \end{matrix}$$

# Singular Value Decomposition (SVD)

- Eigenvectors are for square matrices, but SVD is for all matrices
- To calculate  $U$ , take eigenvectors of  $AA^T$
- Square root of eigenvalues are the singular values (the entries of  $\Sigma$ ).
- To calculate  $V$ , take eigenvectors of  $A^T A$
- In general, if  $A$  is  $m \times n$ , then  $U$  will be  $m \times m$ ,  $\Sigma$  will be  $m \times n$ , and  $V^T$  will be  $n \times n$ .

# Singular Value Decomposition (SVD)

- $U$  and  $V$  are always rotation matrices.
  - Geometric rotation may not be an applicable concept, depending on the matrix. So we call them "unitary" matrices (each column is a unit vector)
- $\Sigma$  is a diagonal matrix
  - The number of nonzero entries = rank of  $A$
  - The algorithm always sorts the entries high to low

$$\begin{matrix} U & & \Sigma & & V^T & & A \\ \begin{bmatrix} -.39 & -.92 \\ -.92 & .39 \end{bmatrix} & \times & \begin{bmatrix} 9.51 & 0 & 0 \\ 0 & .77 & 0 \end{bmatrix} & \times & \begin{bmatrix} -.42 & -.57 & -.70 \\ .81 & .11 & -.58 \\ .41 & -.82 & .41 \end{bmatrix} & = & \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \end{matrix}$$

# Singular Value Decomposition (SVD)

- Look at how the multiplication works out, left to right
- Column 1 of  $U$  gets scaled by the first value from

$$\begin{matrix} U\Sigma \\ \begin{bmatrix} -3.67 & -.71 & 0 \\ -8.8 & .30 & 0 \end{bmatrix} \end{matrix} \times \begin{matrix} V^T \\ \begin{bmatrix} -.42 & -.57 & -.70 \\ .81 & .11 & -.58 \\ .41 & -.82 & .41 \end{bmatrix} \end{matrix} = \begin{matrix} A_{\text{partial}} \\ \begin{bmatrix} 1.6 & 2.1 & 2.6 \\ 3.8 & 5.0 & 6.2 \end{bmatrix} \end{matrix}$$

- The resulting vector gets scaled by row 1 of  $V^T$  to produce a contribution to the columns of  $A$

# Singular Value Decomposition (SVD)

$$\begin{aligned}
 & \begin{matrix} U\Sigma \\ \begin{bmatrix} -3.67 & -71 & 0 \\ -8.8 & .30 & 0 \end{bmatrix} \end{matrix} \times \begin{matrix} V^T \\ \begin{bmatrix} -.42 & -.57 & -.70 \\ .81 & .11 & -.58 \\ .41 & -.82 & .41 \end{bmatrix} \end{matrix} \begin{matrix} A_{\text{partial}} \\ \begin{bmatrix} 1.6 & 2.1 & 2.6 \\ 3.8 & 5.0 & 6.2 \end{bmatrix} \end{matrix} \\
 + & \begin{matrix} U\Sigma \\ \begin{bmatrix} -3.67 & -.71 & 0 \\ -8.8 & .30 & 0 \end{bmatrix} \end{matrix} \times \begin{matrix} V^T \\ \begin{bmatrix} -.42 & -.57 & -.70 \\ .81 & .11 & -.58 \\ .41 & -.82 & .41 \end{bmatrix} \end{matrix} \begin{matrix} A_{\text{partial}} \\ \begin{bmatrix} -.6 & -.1 & .4 \\ .2 & 0 & -.2 \end{bmatrix} \end{matrix} \\
 = & \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}
 \end{aligned}$$

- Each product of (column  $i$  of  $U$ ). (value  $i$  from  $\Sigma$ ). (row  $i$  of  $V^T$ ) produces a component of the final  $A$

# Singular Value Decomposition (SVD)

$$\begin{aligned}
 & \begin{matrix} U\Sigma \\ \begin{bmatrix} -3.67 & -.71 & 0 \\ -8.8 & .30 & 0 \end{bmatrix} \end{matrix} \times \begin{matrix} V^T \\ \begin{bmatrix} -.42 & -.57 & -.70 \\ .81 & .11 & -.58 \\ .41 & -.82 & .41 \end{bmatrix} \end{matrix} \quad \begin{matrix} A_{\text{partial}} \\ \begin{bmatrix} 1.6 & 2.1 & 2.6 \\ 3.8 & 5.0 & 6.2 \end{bmatrix} \end{matrix} \\
 + & \begin{matrix} U\Sigma \\ \begin{bmatrix} -3.67 & -.71 & 0 \\ -8.8 & .30 & 0 \end{bmatrix} \end{matrix} \times \begin{matrix} V^T \\ \begin{bmatrix} -.42 & -.57 & -.70 \\ .81 & .11 & -.58 \\ .41 & -.82 & .41 \end{bmatrix} \end{matrix} \quad \begin{matrix} A_{\text{partial}} \\ \begin{bmatrix} -.6 & -.1 & .4 \\ .2 & 0 & -.2 \end{bmatrix} \end{matrix} \\
 = & \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}
 \end{aligned}$$

- Were building A as a linear combination of the columns of U
- Using all columns of U, we'll rebuild the original matrix perfectly
- But, in real-world data, often we can just use the first few columns of U and we'll get something close (e.g. the first  $A_{\text{partial}}$ , above)

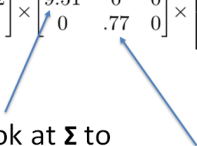
# Singular Value Decomposition (SVD)

$$\begin{aligned}
 & \begin{matrix} U\Sigma \\ \begin{bmatrix} -3.67 & -.71 & 0 \\ -8.8 & .30 & 0 \end{bmatrix} \end{matrix} \times \begin{matrix} V^T \\ \begin{bmatrix} -.42 & -.57 & -.70 \\ .81 & .11 & -.58 \\ .41 & -.82 & .41 \end{bmatrix} \end{matrix} \begin{matrix} A_{\text{partial}} \\ \begin{bmatrix} 1.6 & 2.1 & 2.6 \\ 3.8 & 5.0 & 6.2 \end{bmatrix} \end{matrix} \\
 + & \begin{matrix} U\Sigma \\ \begin{bmatrix} -3.67 & -.71 & 0 \\ -8.8 & .30 & 0 \end{bmatrix} \end{matrix} \times \begin{matrix} V^T \\ \begin{bmatrix} -.42 & -.57 & -.70 \\ .81 & .11 & -.58 \\ .41 & -.82 & .41 \end{bmatrix} \end{matrix} \begin{matrix} A_{\text{partial}} \\ \begin{bmatrix} -.6 & -.1 & .4 \\ .2 & 0 & -.2 \end{bmatrix} \end{matrix} \\
 = & \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}
 \end{aligned}$$

- We can call those first few columns of U the Principal Components of the data
- They show the major patterns that can be added to produce the columns of the original matrix
- The rows of  $V^T$  show how the principal components are mixed to produce the columns of the matrix



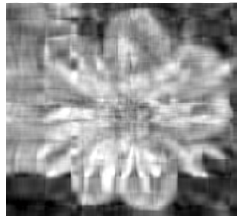
# SVD Applications

$$\begin{matrix} U \\ \begin{bmatrix} -.39 & -.92 \\ -.92 & .39 \end{bmatrix} \end{matrix} \times \begin{matrix} \Sigma \\ \begin{bmatrix} 9.51 & 0 & 0 \\ 0 & .77 & 0 \end{bmatrix} \end{matrix} \times \begin{matrix} V^T \\ \begin{bmatrix} -.42 & -.57 & -.70 \\ .81 & .11 & -.58 \\ .41 & -.82 & .41 \end{bmatrix} \end{matrix} = \begin{matrix} A \\ \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \end{matrix}$$


We can look at  $\Sigma$  to see that the first column has a large effect

while the second column has a much smaller effect in this example

# SVD Applications



- For this image, using only the first 10 of 300 principal components produces a recognizable reconstruction
- So, SVD can be used for image compression

# Principal Component Analysis

$$\begin{matrix} U\Sigma \\ \begin{bmatrix} -3.67 & -.71 & 0 \\ -8.8 & .30 & 0 \end{bmatrix} \end{matrix} \times \begin{matrix} V^T \\ \begin{bmatrix} -.42 & -.57 & -.70 \\ .81 & .11 & -.58 \\ .41 & -.82 & .41 \end{bmatrix} \end{matrix} = \begin{matrix} A_{\text{partial}} \\ \begin{bmatrix} 1.6 & 2.1 & 2.6 \\ 3.8 & 5.0 & 6.2 \end{bmatrix} \end{matrix}$$

- Remember, columns of  $U$  are the Principal Components of the data: the major patterns that can be added to produce the columns of the original matrix
- One use of this is to construct a matrix where each column is a separate data sample
- Run SVD on that matrix, and look at the first few columns of  $U$  to see patterns that are common among the columns
- This is called Principal Component Analysis (or PCA) of the data samples