Linear Algebra Tutorial I

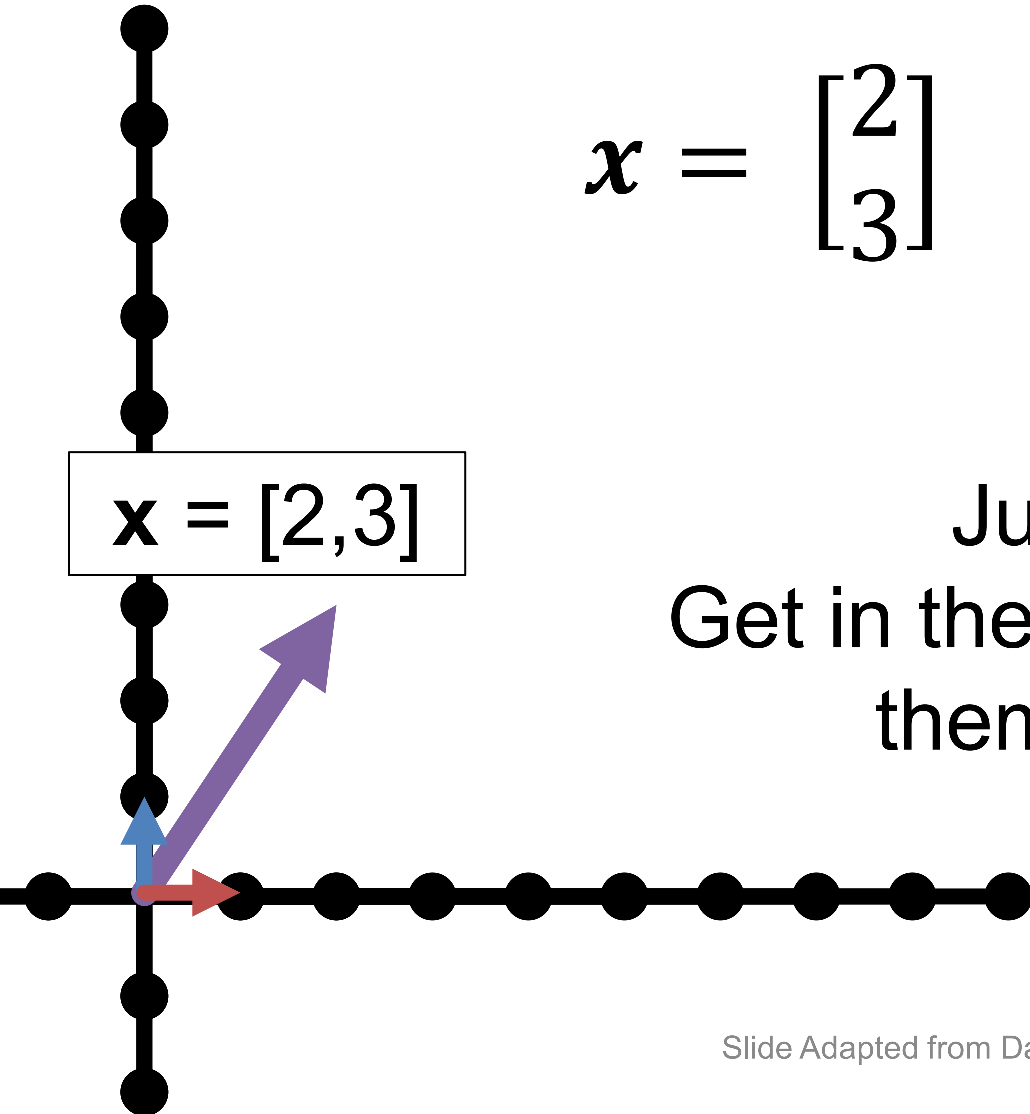
EEECS 442

Fall 2020, University of Michigan

Slide Adapted from David Fouhey

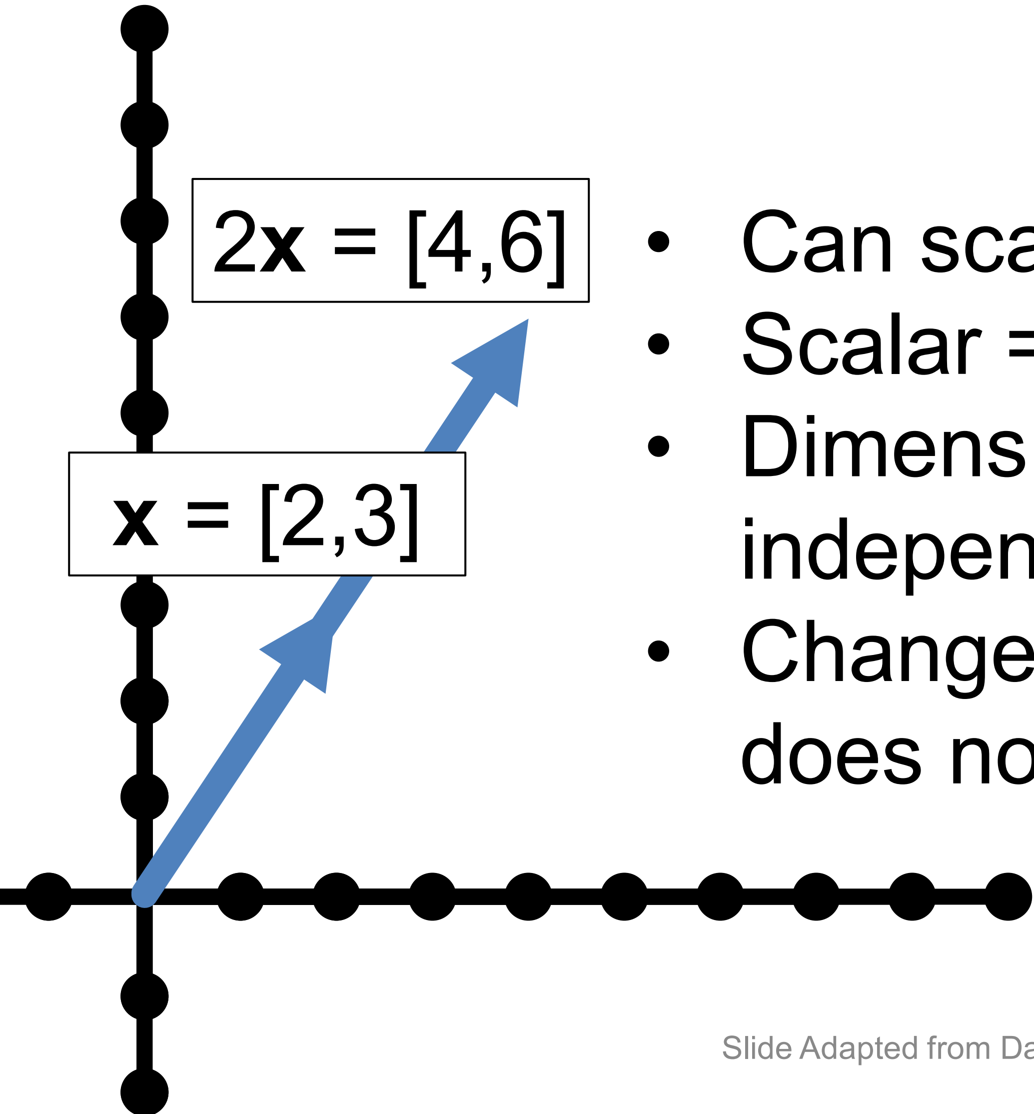
Vectors

$$\boldsymbol{x} = \begin{bmatrix} 2 \\ 3 \end{bmatrix} \qquad \boldsymbol{x}_1 = 2 \quad \boldsymbol{x}_2 = 3$$



Just an array!
Get in the habit of thinking of them as columns.

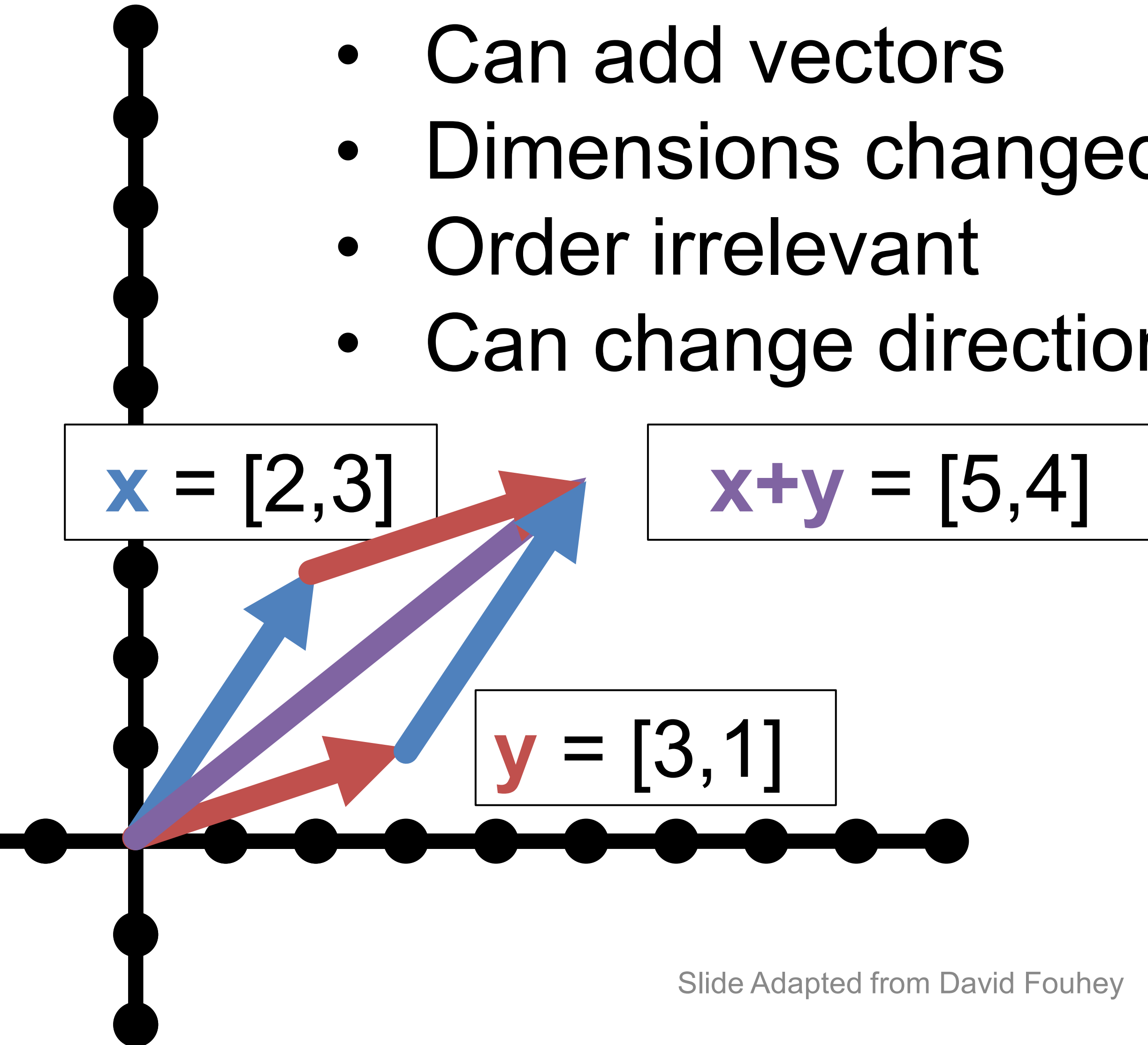
Scaling Vectors



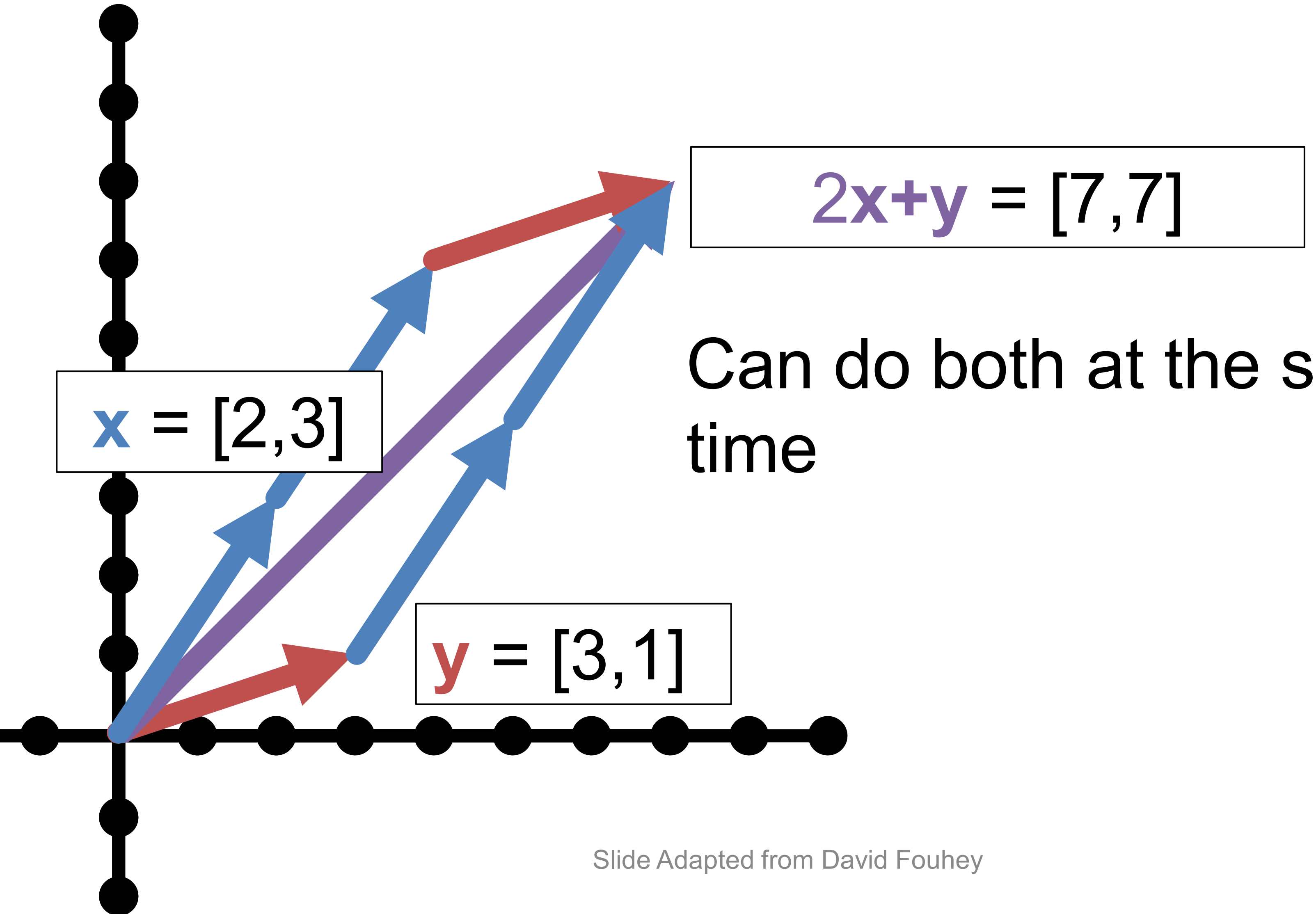
- Can scale vector by a *scalar*
- Scalar = single number
- Dimensions changed independently
- Changes *magnitude / length*, does not change *direction*.

Adding Vectors

- Can add vectors
- Dimensions changed independently
- Order irrelevant
- Can change direction and magnitude

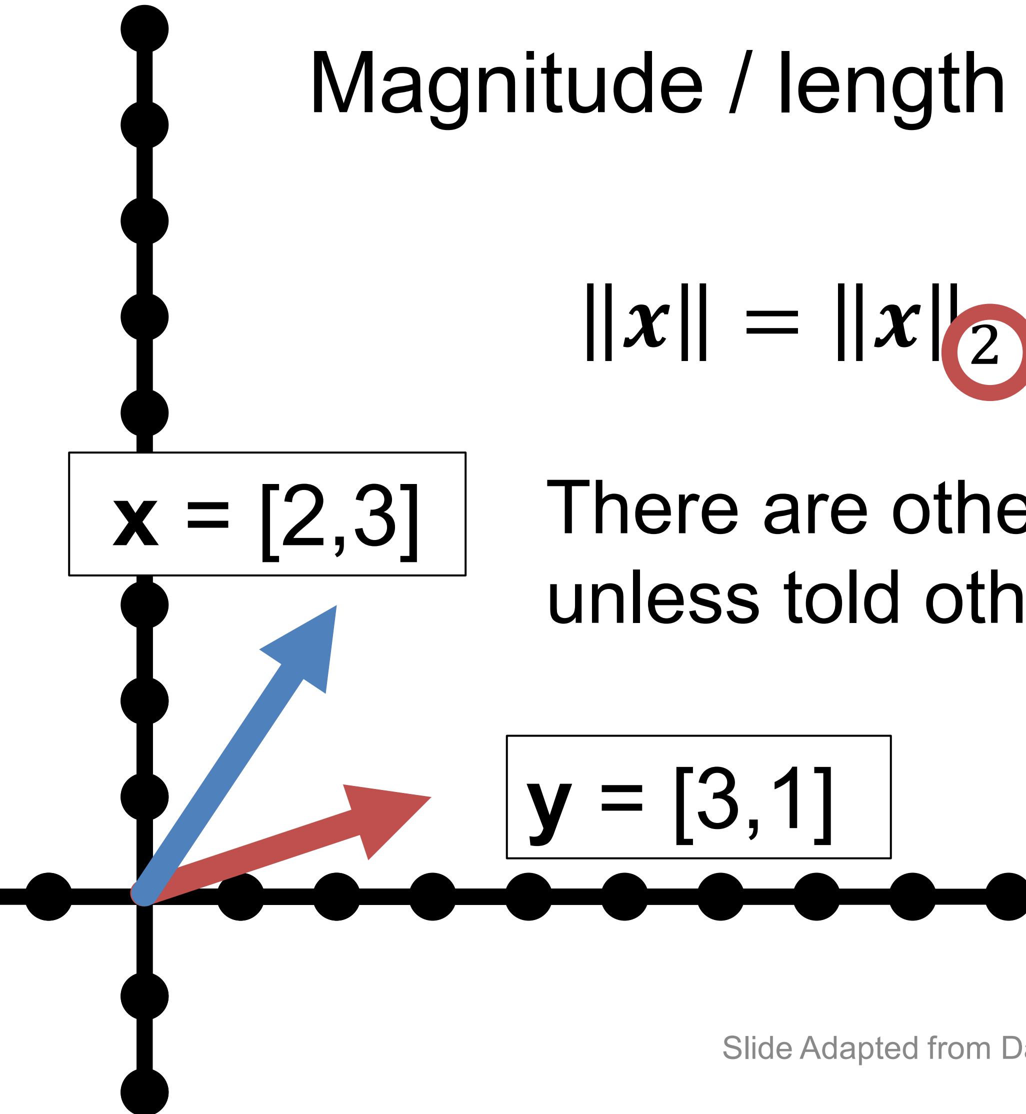


Scaling and Adding



Can do both at the same time

Measuring Length



Magnitude / length / (L2) norm of vector

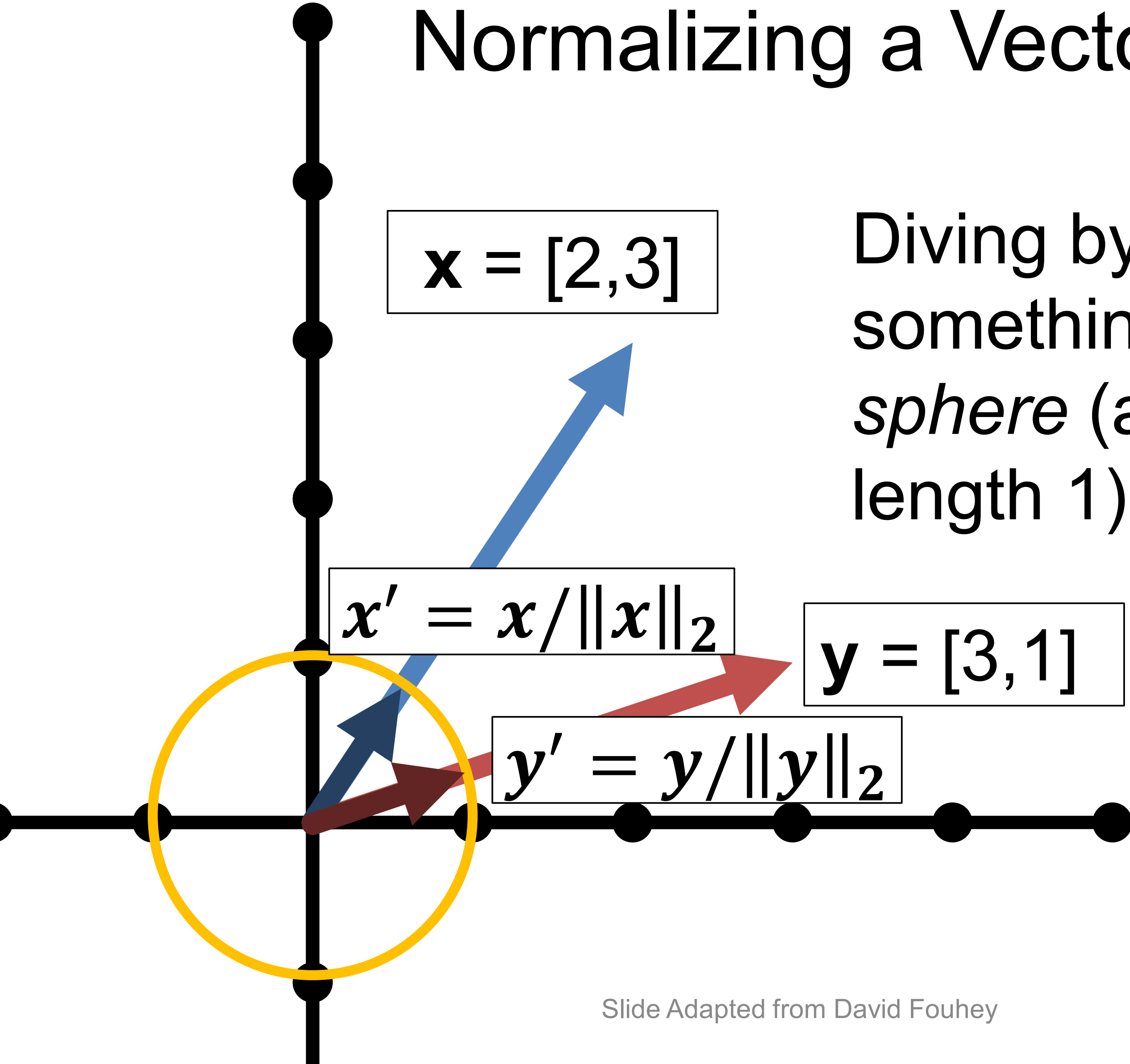
$$\|\boldsymbol{x}\| = \|\boldsymbol{x}\|_2 = \left(\sum_{i}^{n} x_i^2\right)^{1/2}$$

There are other norms; assume L2 unless told otherwise

$$\|\boldsymbol{x}\|_2 = \sqrt{13}$$
$$\|\boldsymbol{y}\|_2 = \sqrt{10}$$

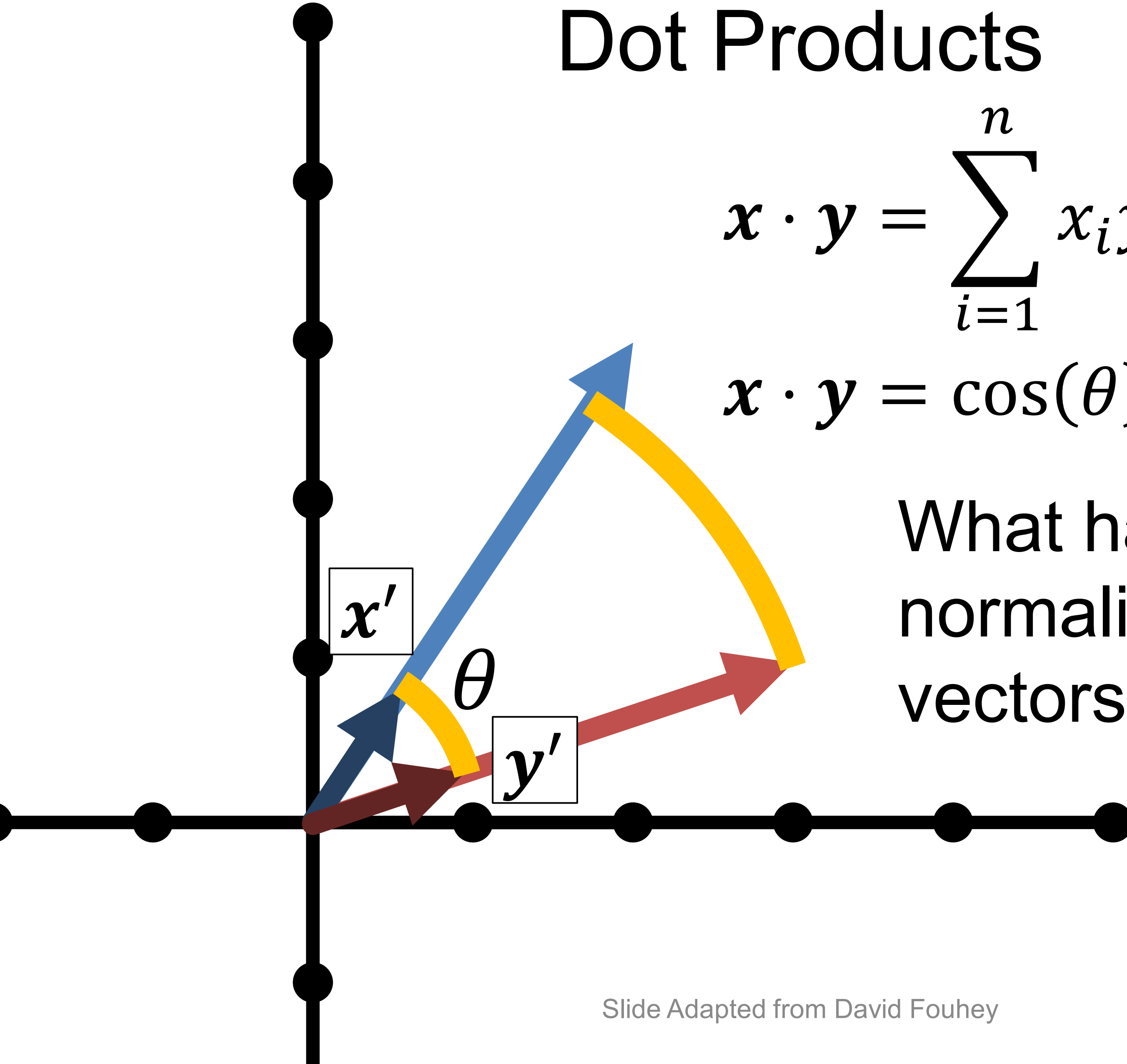
Why?

Normalizing a Vector



Diving by norm gives something on the *unit sphere* (all vectors with length 1)

Dot Products

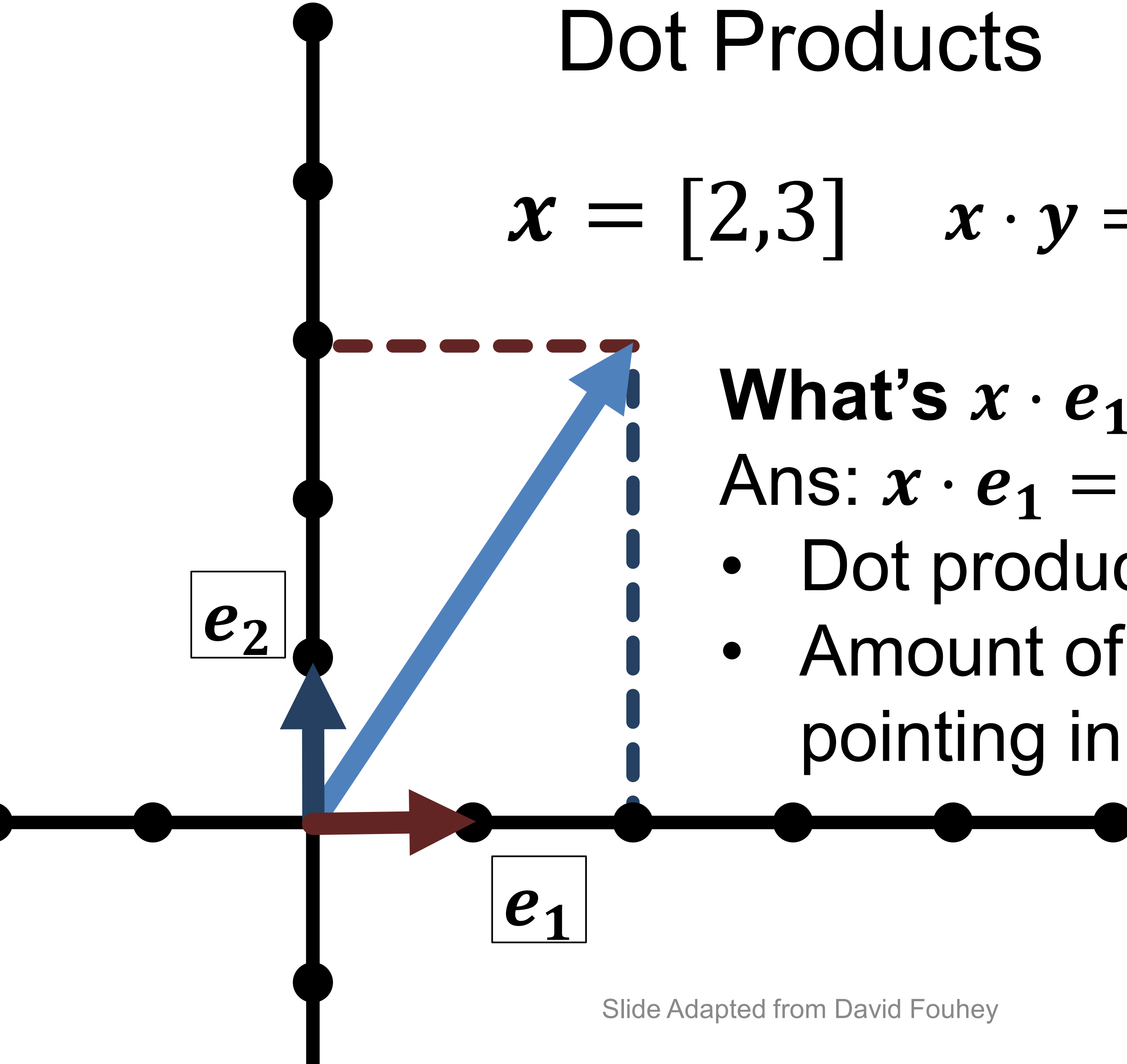


$$\boldsymbol{x} \cdot \boldsymbol{y} = \sum_{i=1}^{n} x_i y_i = \boldsymbol{x}^T \boldsymbol{y}$$

$$\boldsymbol{x} \cdot \boldsymbol{y} = \cos(\theta) \|\boldsymbol{x}\| \, \|\boldsymbol{y}\|$$

What happens with normalized / unit vectors?

Dot Products



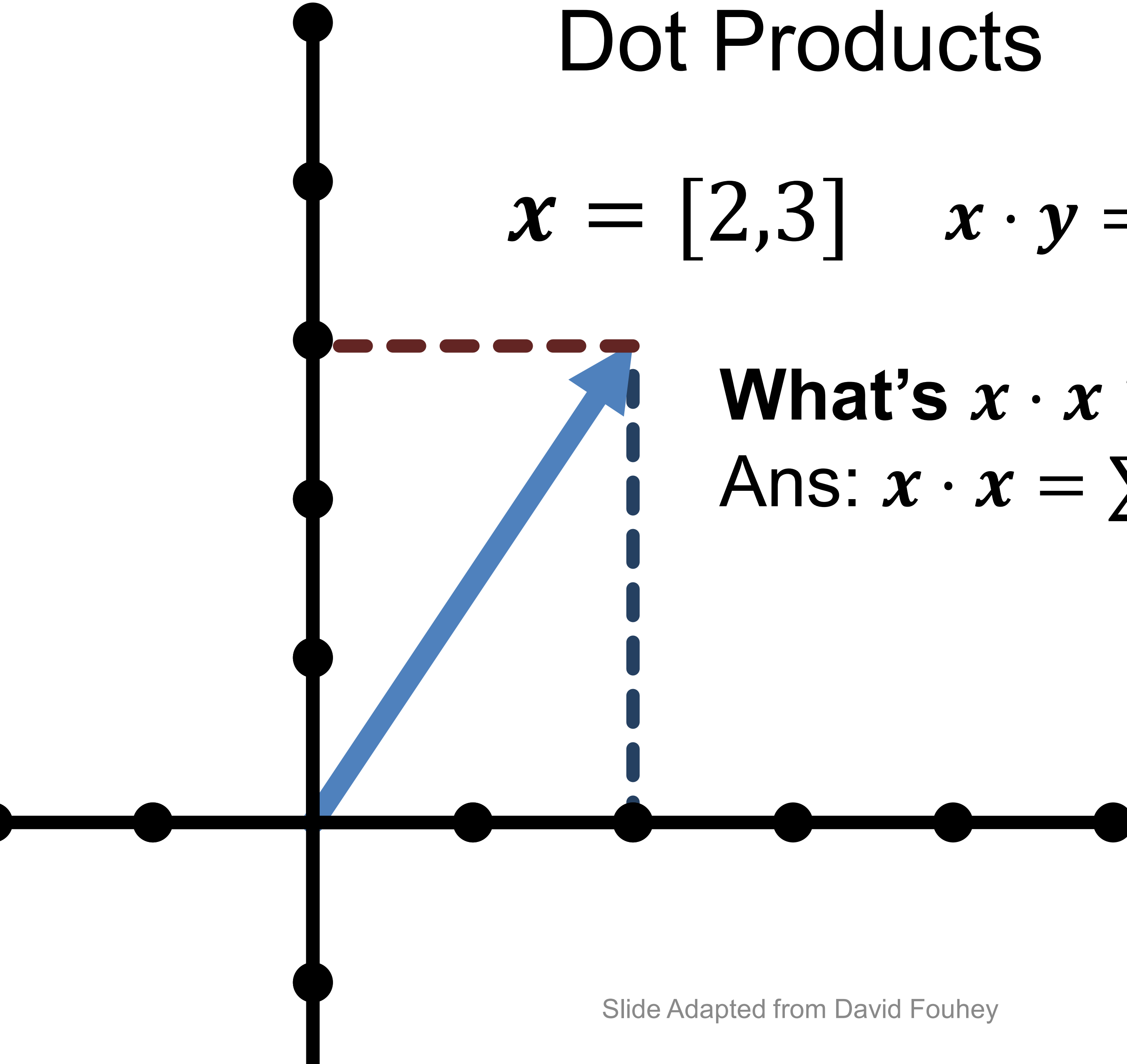
$$\boldsymbol{x} = [2,3] \qquad \boldsymbol{x} \cdot \boldsymbol{y} = \sum_i^n x_i y_i$$

What's $\boldsymbol{x} \cdot \boldsymbol{e_1}$, $\boldsymbol{x} \cdot \boldsymbol{e_2}$?

Ans: $\boldsymbol{x} \cdot \boldsymbol{e_1} = 2$; $\boldsymbol{x} \cdot \boldsymbol{e_2} = 3$

- Dot product is projection
- Amount of **x** that's also pointing in direction of **y**

Dot Products



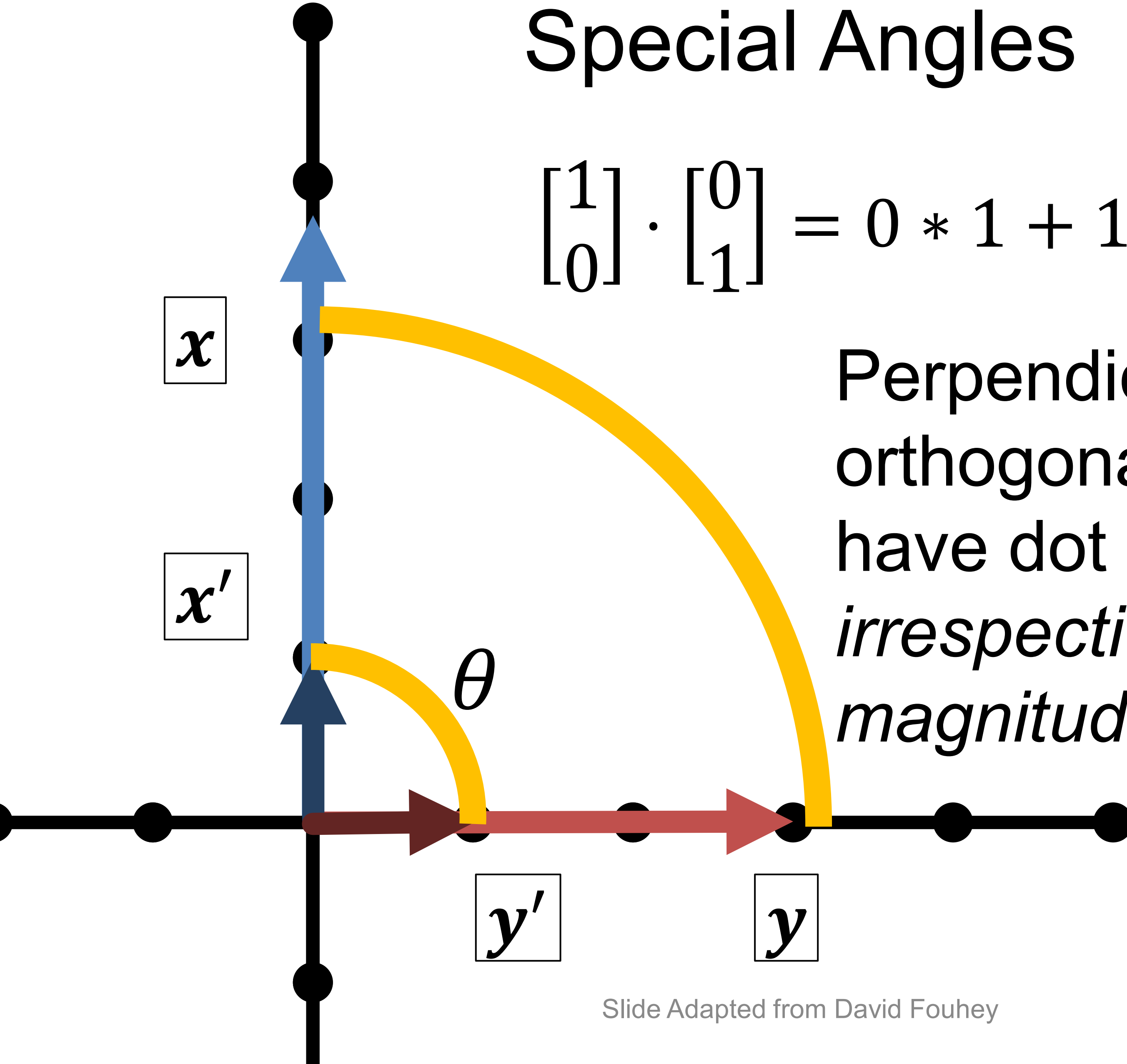
$$\boldsymbol{x} = [2,3] \qquad \boldsymbol{x} \cdot \boldsymbol{y} = \sum_{i}^{n} x_i y_i$$

What's $\boldsymbol{x} \cdot \boldsymbol{x}$?

Ans: $\boldsymbol{x} \cdot \boldsymbol{x} = \sum x_i x_i = \|\boldsymbol{x}\|_2^2$

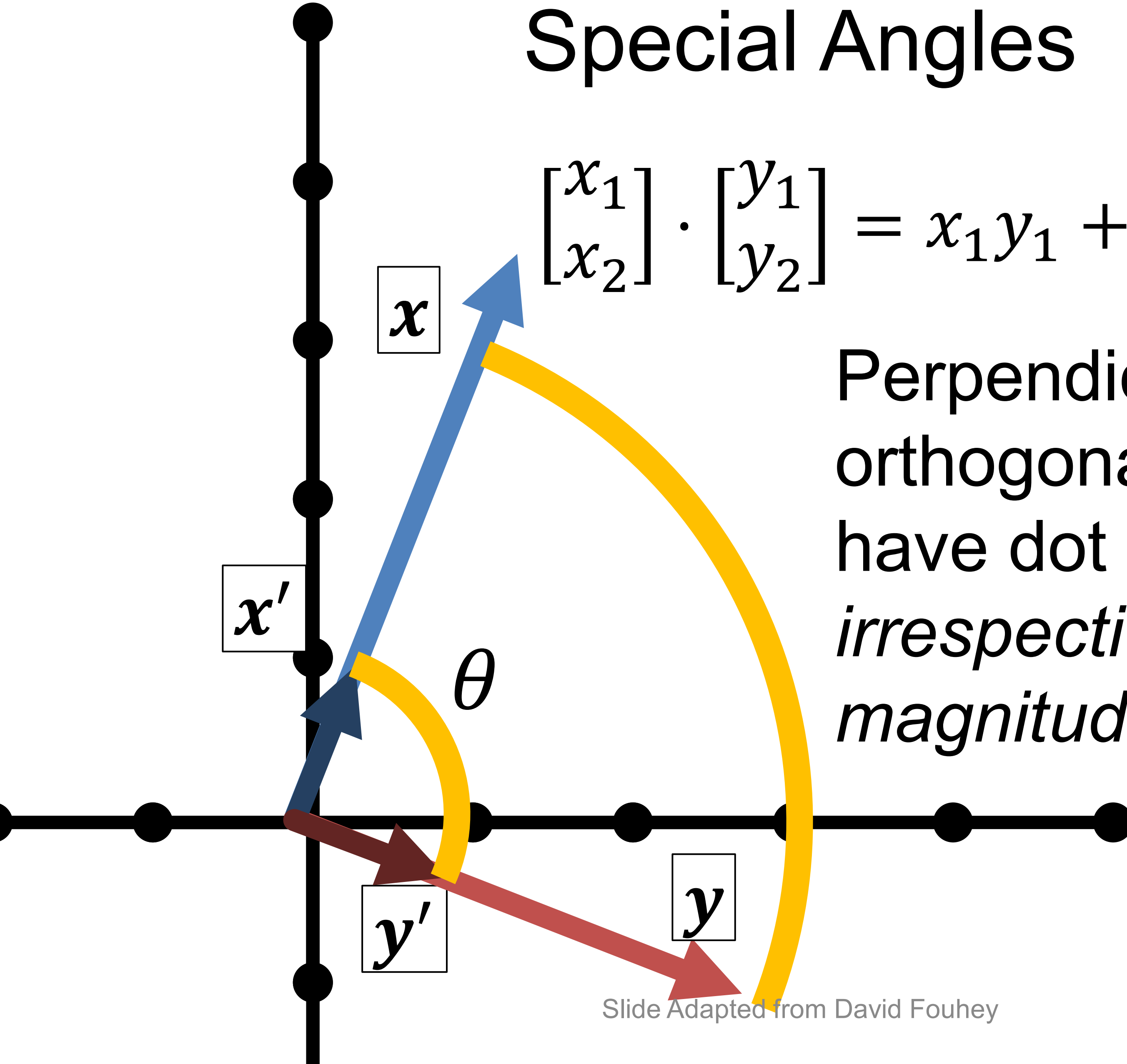
Special Angles

$$\begin{bmatrix}1\\0\end{bmatrix} \cdot \begin{bmatrix}0\\1\end{bmatrix} = 0 * 1 + 1 * 0 = 0$$



Perpendicular / orthogonal vectors have dot product 0 *irrespective of their magnitude*

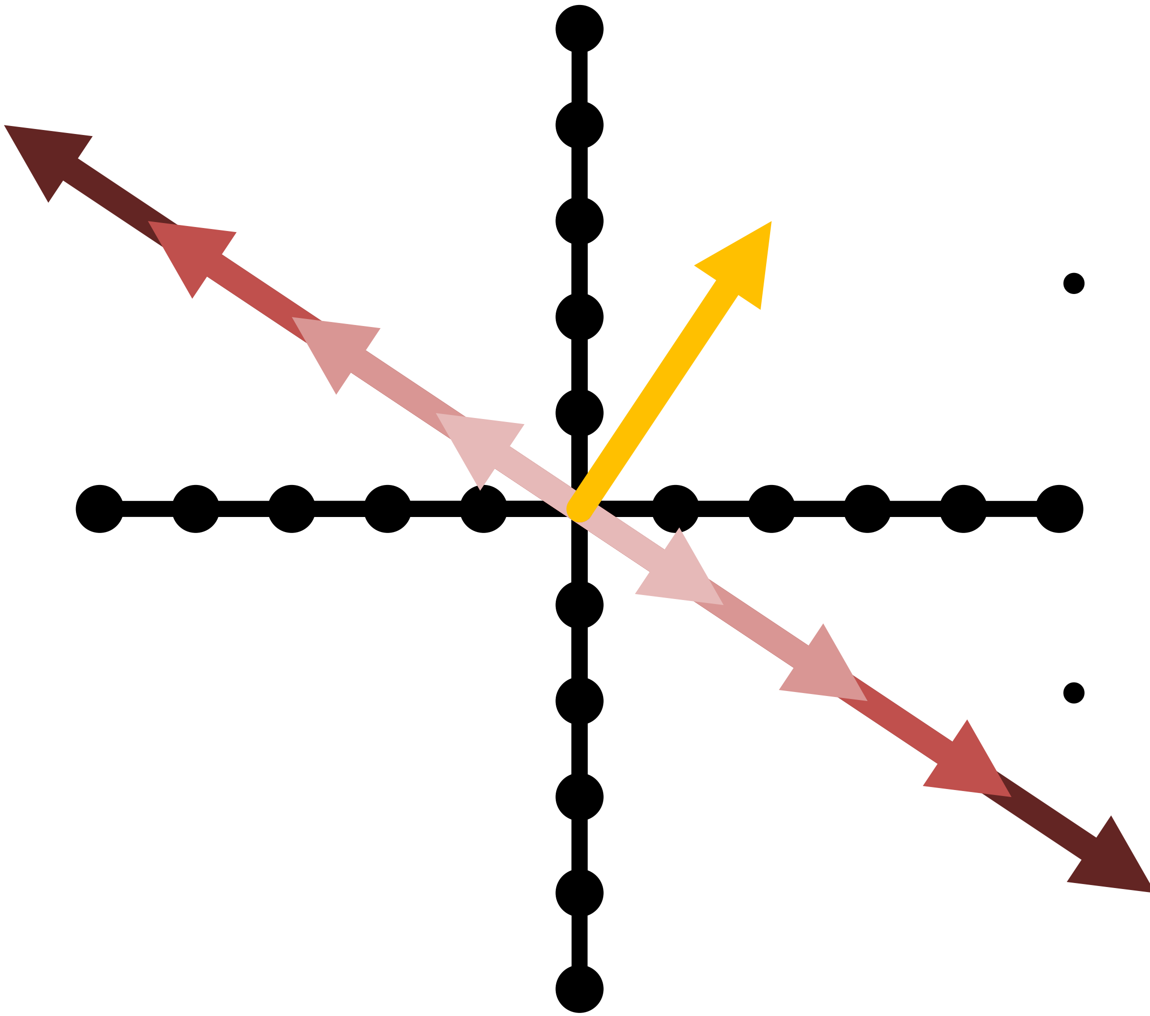
Special Angles



$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \cdot \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = x_1 y_1 + x_2 y_2 = 0$$

Perpendicular / orthogonal vectors have dot product 0 *irrespective of their magnitude*

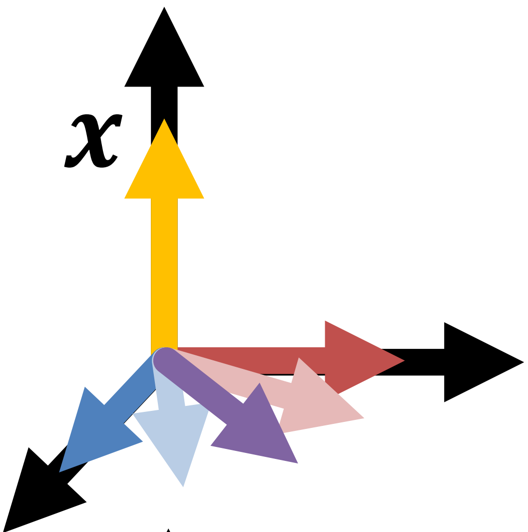
Orthogonal Vectors



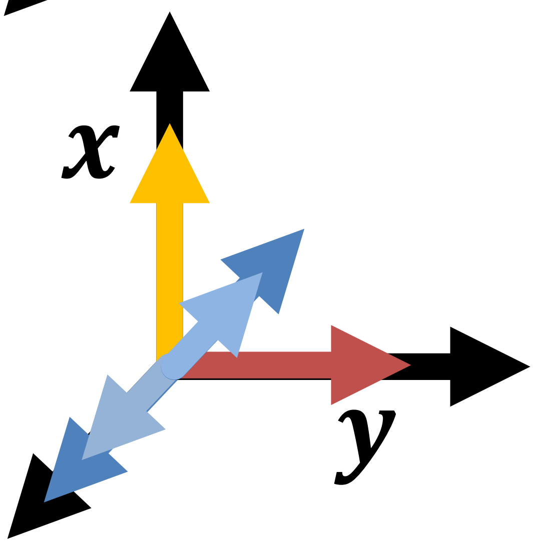
$\boldsymbol{x} = [2,3]$

- **Geometrically, what's the set of vectors that are orthogonal to x?**
- A line [3,-2]

Orthogonal Vectors

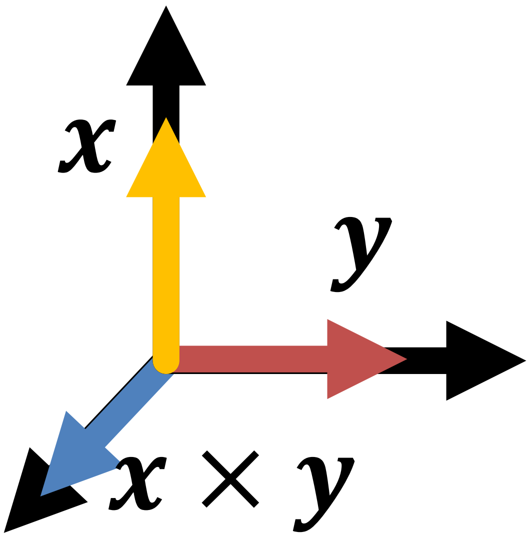


- **What's the set of vectors that are orthogonal to x = [5,0,0]?**
- A plane/2D space of vectors/any vector $[0, a, b]$



- **What's the set of vectors that are orthogonal to x <u>and</u> y = [0,5,0]?**
- A line/1D space of vectors/any vector $[0,0, b]$
- Ambiguity in *sign and magnitude*

Cross Product



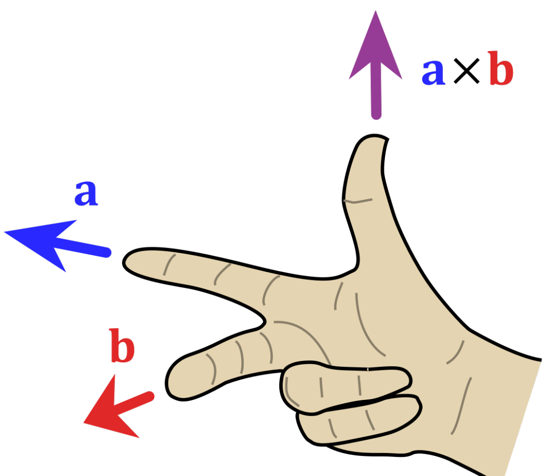


Image credit: Wikipedia.org

- Cross product $\boldsymbol{x} \times \boldsymbol{y}$ is: (1) orthogonal to x, y (2) has sign given by right hand rule and (3) has magnitude given by area of parallelogram of **x** and **y**
- **Important**: if x and y are the same direction or either is **0**, then $\boldsymbol{x} \times \boldsymbol{y} = \boldsymbol{0}$.
- Only in 3D!

Operations You Should Know

- Scale (vector, scalar → vector)
- Add (vector, vector → vector)
- Magnitude (vector → scalar)
- Dot product (vector, vector → scalar)
  - Dot products are projection / angles
- Cross product (vector, vector → vector)
  - Vectors facing same direction have cross product **0**
- You can **never** mix vectors of different sizes

Matrices

Horizontally concatenate n, m-dim column vectors and you get a mxn matrix A (here 2x3)

$$\boldsymbol{A} = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_n] = \begin{bmatrix} \boxed{\begin{matrix} v_{1_1} \\ v_{1_2} \end{matrix}} & \boxed{\begin{matrix} v_{2_1} \\ v_{2_2} \end{matrix}} & \boxed{\begin{matrix} v_{3_1} \\ v_{3_2} \end{matrix}} \end{bmatrix}$$

Matrices

Transpose: flip rows / columns $\begin{bmatrix} a \\ b \\ c \end{bmatrix}^T = \begin{bmatrix} a & b & c \end{bmatrix}$ $(3\text{x}1)^T = 1\text{x}3$

Vertically concatenate m, n-dim row vectors and you get a mxn matrix A (here 2x3)

$$A = \begin{bmatrix} \boldsymbol{u}_1^T \\ \vdots \\ \boldsymbol{u}_n^T \end{bmatrix} = \begin{bmatrix} \boxed{u_{1_1} \quad u_{1_2} \quad u_{1_3}} \\ \boxed{u_{2_1} \quad u_{2_2} \quad u_{2_3}} \end{bmatrix}$$

Matrix-Vector Product

$$\boldsymbol{y}_{2x1} = \boldsymbol{A}_{2x3}\boldsymbol{x}_{3x1}$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{v_1} & \boldsymbol{v_2} & \boldsymbol{v_3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\boldsymbol{y} = x_1\boldsymbol{v_1} + x_2\boldsymbol{v_2} + x_3\boldsymbol{v_3}$$

*Linear combination of columns of **A***

Matrix-Vector Product

$$\boldsymbol{y}_{2x1} = \boldsymbol{A}_{2x3}\boldsymbol{x}_{3x1}$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{array}{c} \boxed{3} \\ \begin{bmatrix} \boldsymbol{u}_1^T \\ \boldsymbol{u}_2^T \end{bmatrix} \end{array} \boldsymbol{x} \; \boxed{3}$$

$$y_1 = \boldsymbol{u}_1^T\boldsymbol{x} \qquad y_2 = \boldsymbol{u}_2^T\boldsymbol{x}$$

*Dot product between rows of **A** and **x***

Matrix Multiplication

Generally: $\boldsymbol{A}_{mn}$ *and* $\boldsymbol{B}_{np}$ *yield product* $(\boldsymbol{AB})_{mp}$

$$\boldsymbol{AB} = \begin{bmatrix} - & \boldsymbol{a}_1^T & - \\ & \vdots & \\ - & \boldsymbol{a}_m^T & - \end{bmatrix} \begin{bmatrix} | & & | \\ \boldsymbol{b}_1 & \cdots & \boldsymbol{b}_p \\ | & & | \end{bmatrix}$$

Yes – in **A**, I'm referring to the rows, and in **B**, I'm referring to the columns

Matrix Multiplication

Generally: $\boldsymbol{A}_{mn}$ *and* $\boldsymbol{B}_{np}$ *yield product* $(\boldsymbol{AB})_{mp}$

$$\begin{bmatrix} | & & | \\ \boldsymbol{b}_1 & \cdots & \boldsymbol{b}_p \\ | & & | \end{bmatrix}$$

$$\boldsymbol{AB} = \begin{bmatrix} - & \boldsymbol{a}_1^T & - \\ & \vdots & \\ - & \boldsymbol{a}_m^T & - \end{bmatrix} \begin{bmatrix} \boldsymbol{a}_1^T\boldsymbol{b}_1 & \cdots & \boldsymbol{a}_1^T\boldsymbol{b}_p \\ \vdots & \ddots & \vdots \\ \boldsymbol{a}_m^T\boldsymbol{b}_1 & \cdots & \boldsymbol{a}_m^T\boldsymbol{b}_p \end{bmatrix}$$

$$\boldsymbol{AB}_{ij} = \boldsymbol{a}_i^T\boldsymbol{b}_j$$

Slide Adapted from David Fouhey

Matrix Multiplication

- Dimensions must match
- Dimensions must match
- Dimensions must match
- (Yes, it's associative): ABx = (A)(Bx) = (AB)x
- (*No it's not commutative*): ABx ≠ (BA)x ≠ (BxA)

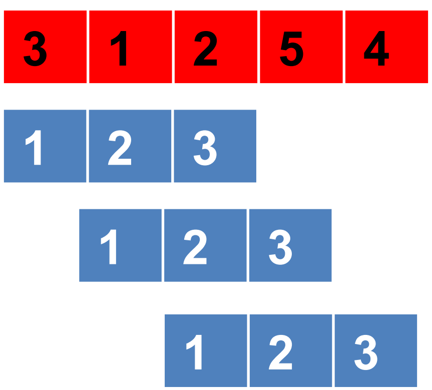
Cross-correlation

Consider 1D case for simplicity

- Correlation $c[m] = h * g = \sum_k h[m+k]g[k]$
- Convolution $f[m] = h \circ g = \sum_k h[m-k]g[k]$

Let h= [3,1,2,5,4], g = [1,2,3], then c = [11, 20, 24]:

$$c[0] = \sum_k h[0+k]g[k] = h[0]g[0] + h[1]g[1] + h[2]g[2] = \begin{bmatrix}3\\1\\2\end{bmatrix} \cdot \begin{bmatrix}1\\2\\3\end{bmatrix}$$



Each output element is from a dot product!

$$c[0] = \begin{bmatrix}3\\1\\2\end{bmatrix} \cdot \begin{bmatrix}1\\2\\3\end{bmatrix} = 3 + 2 + 6 = 11$$

$$c[1] = \begin{bmatrix}1\\2\\5\end{bmatrix} \cdot \begin{bmatrix}1\\2\\3\end{bmatrix} = 1 + 4 + 15 = 20$$

$$c[2] = \begin{bmatrix}2\\5\\4\end{bmatrix} \cdot \begin{bmatrix}1\\2\\3\end{bmatrix} = 2 + 10 + 12 = 24$$

Convolution

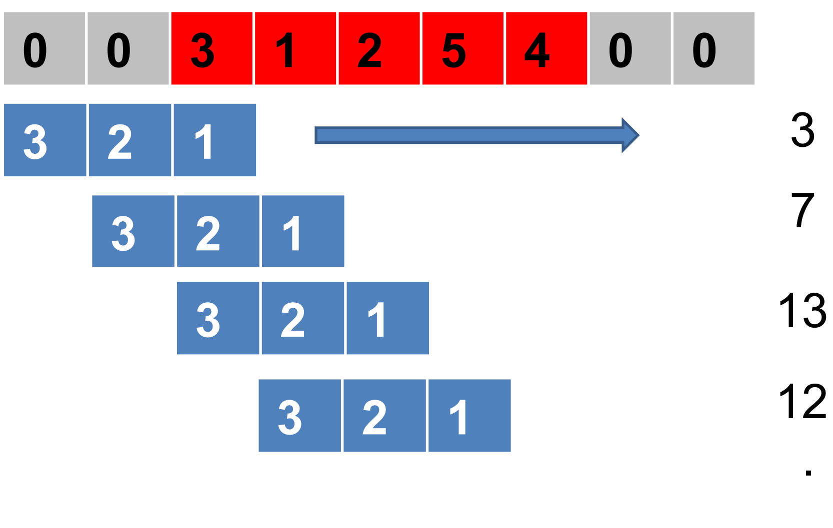
Consider 1D case for simplicity

- Correlation $c[m] = h * g = \sum_k h[m+k]g[k]$
- Convolution $f[m] = h \circ g = \sum_k h[m-k]g[k]$

Let h= [3,1,2,5,4], g = [1,2,3], then f = [3, 7, 13, 12, 20, 23, 12]:

$$f[0] = \sum_k h[0-k]g[k] = h[0]g[0] + h[-1]g[1] + h[-2]g[2] = \begin{bmatrix}0\\0\\3\end{bmatrix} \cdot \begin{bmatrix}3\\2\\1\end{bmatrix}$$

Each output element is from a dot product!



Operations They Don't Teach

You Probably Saw Matrix Addition

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} a+e & b+f \\ c+g & d+h \end{bmatrix}$$

What is this? FYI: e is a scalar

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

Broadcasting

If you want to be pedantic and proper, you expand e by multiplying a matrix of 1s (denoted **1**)

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + e = \begin{bmatrix} a & b \\ c & d \end{bmatrix} + \mathbf{1}_{2x2} e$$

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

Many smart matrix libraries do this automatically. This is the source of many bugs.

Broadcasting Example

Given: a nx2 matrix **P** and a 2D column vector **v**,
Want: nx2 difference matrix **D**

$$\boldsymbol{P} = \begin{bmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_n & y_n \end{bmatrix} \quad \boldsymbol{v} = \begin{bmatrix} a \\ b \end{bmatrix} \quad \boldsymbol{D} = \begin{bmatrix} x_1 - a & y_1 - b \\ \vdots & \vdots \\ x_n - a & y_n - b \end{bmatrix}$$

$$\boldsymbol{P} - \boldsymbol{v}^T = \begin{bmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_n & y_n \end{bmatrix} - \begin{matrix} [a & b] \\ \vdots & \\ [a & b] \end{matrix}$$

Blue stuff is assumed / broadcast

Broadcasting Rule

When operating on two arrays, NumPy compares their shapes element-wise. It starts with the trailing dimensions and works its way forward. Two dimensions are compatible when

1. they are equal, or
2. one of them is 1

```
A      (2d array):  5 x 4
B      (1d array):      1
Result (2d array):  5 x 4

A      (2d array):  5 x 4
B      (1d array):      4
Result (2d array):  5 x 4

A      (3d array):  15 x 3 x 5
B      (3d array):  15 x 1 x 5
Result (3d array):  15 x 3 x 5

A      (3d array):  15 x 3 x 5
B      (2d array):       3 x 5
Result (3d array):  15 x 3 x 5

A      (3d array):  15 x 3 x 5
B      (2d array):       3 x 1
Result (3d array):  15 x 3 x 5
```

Two Uses for Matrices

1. Storing things in a rectangular array (images, maps)
   - *Typical operations*: element-wise operations, convolution (which we'll cover next)
   - *Atypical operations*: almost anything you learned in a math linear algebra class
2. A linear operator that maps vectors to another space (**Ax**)
   - *Typical/Atypical:* reverse of above