Numerical Linear Algebra

EECS 442 – David Fouhey

Winter 2023, University of Michigan

http://web.eecs.umich.edu/~fouhey/teaching/EECS442_W23/

Today – Math

Two goals for the class:

- Math with computers ≠ Math
- Practical math you need to know but may not have been taught

~~LSA MATHEMATICS~~
~~UNIVERSITY OF MICHIGAN~~

Adding Numbers

- Suppose $b > 0$
- **Is $a+b > a$?**
- **Is $a+b = a$?**

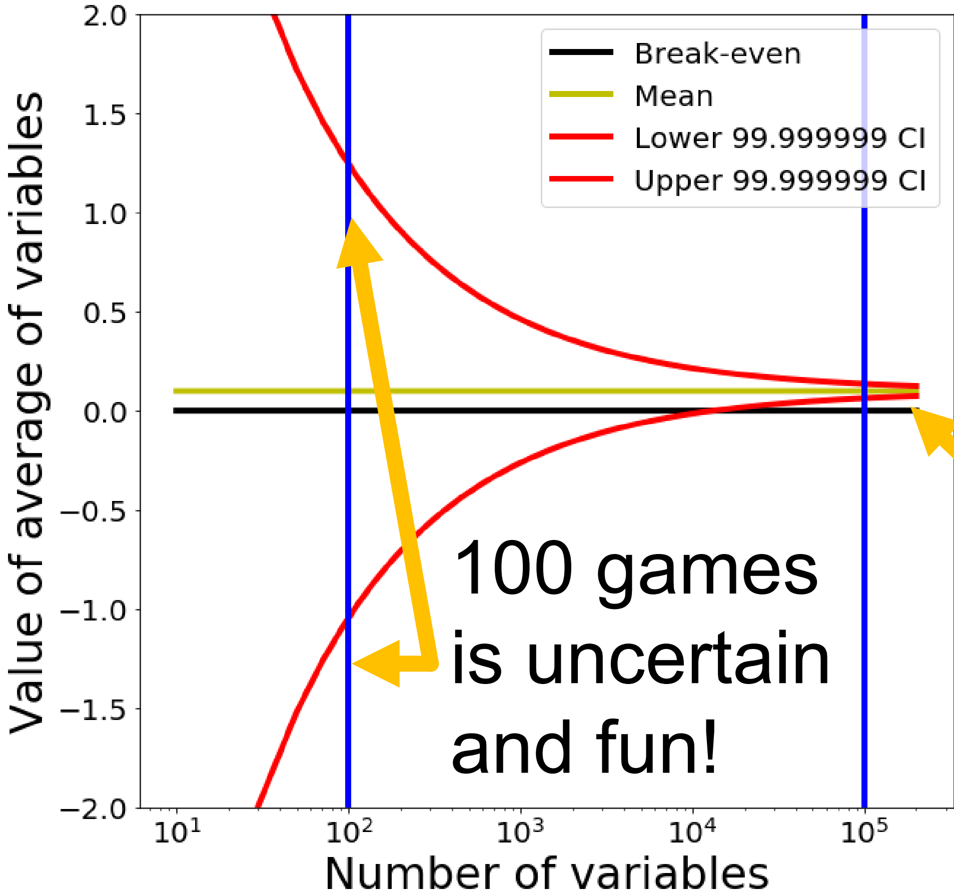
Adding Numbers

- **1 + 1 = ?**
- Suppose x_i is normally distributed with mean μ and standard deviation σ for $1 \leq i \leq N$
- **How is the average, or $\boldsymbol{\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i}$, distributed (qualitatively), in terms of variance?**
- *The Free Drinks in Vegas Theorem**: $\hat{\mu}$ has mean μ and standard deviation $\frac{\sigma}{\sqrt{N}}$.

**Not the real name.* More un-fun name: law of large numbers.

Free Drinks in Vegas

Each game/variable has mean $0.10, std $2



100K games is guaranteed profit: 99.999999% lowest value is $0.064.
$0.01 for drinks
$0.054 for profits

Let’s Make It Big

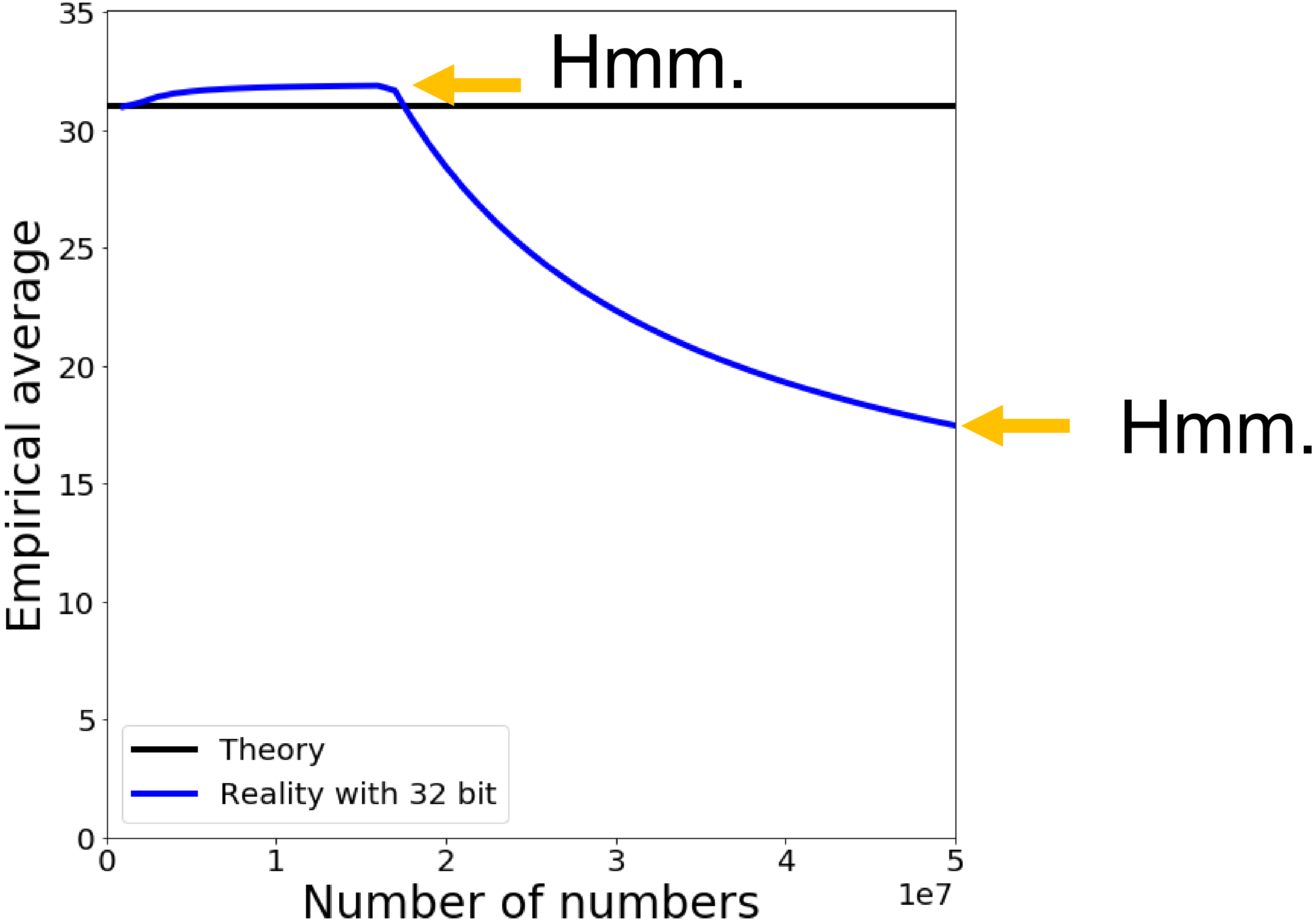
- Suppose I average 50M normally distributed numbers (mean: 31, standard deviation: 1)
- For instance: have predicted and actual depth for 200 480x640 images and want to know the average error (|predicted – actual|)

```
numerator = 0
for x in xs:
  numerator += x
return numerator / len(xs)
```

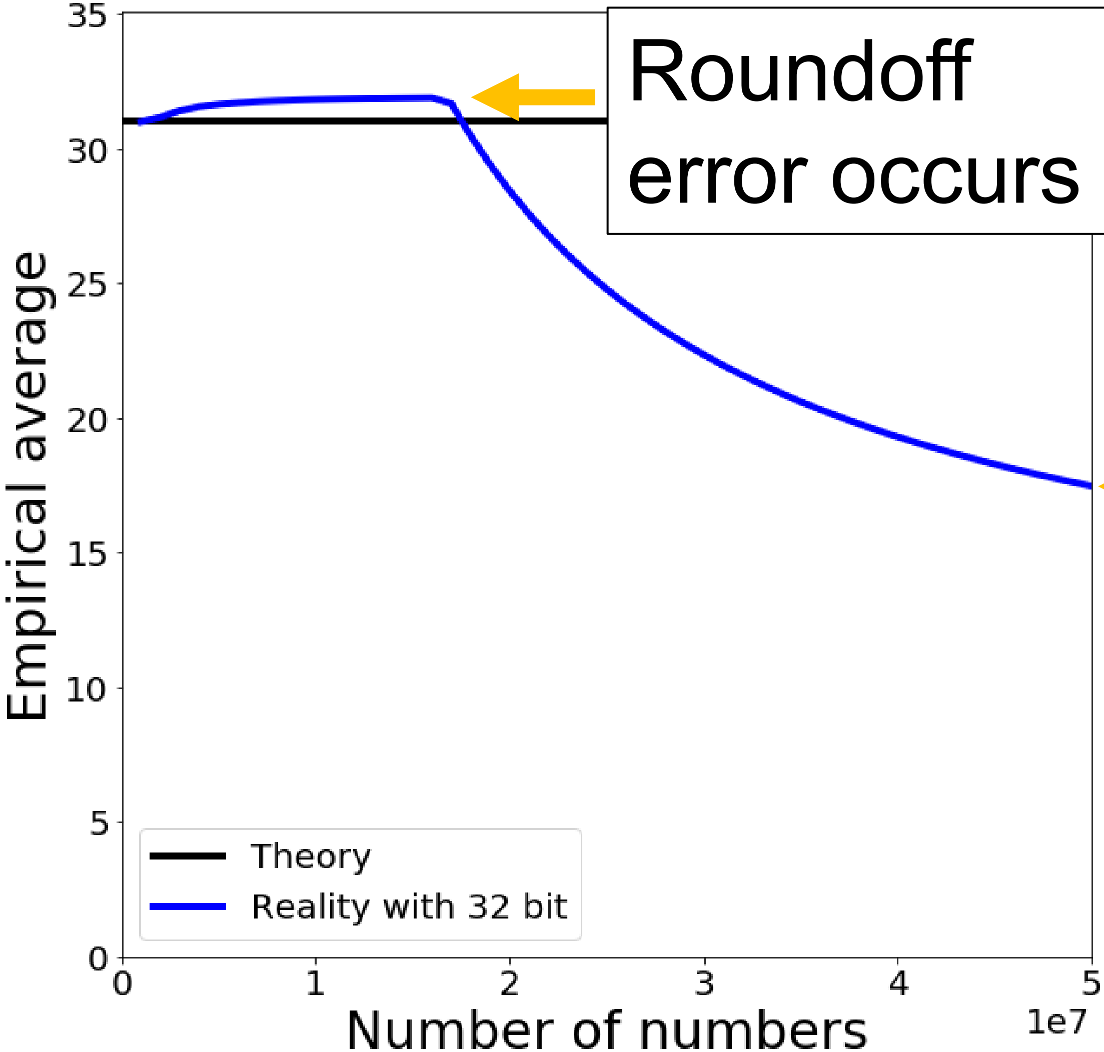
Let’s Make It Big

- **What should happen qualitatively?**
- Theory says that the average is distributed with mean 31 and standard deviation $\frac{1}{\sqrt{50M}} \approx (10^{-5})$
- **What will happen?**
- Reality: 17.47

Trying it Out



Trying it Out



Roundoff error occurs

a+b=a -> numerator is stuck, denominator isn’t

Take-homes

- Computer numbers aren't math numbers
- Overflow, accidental zeros, roundoff error, and basic equalities are almost certainly incorrect for some values
- Floating points and numpy try to protect you.
- Generally safe to use a double and use built-in-functions in numpy (not necessarily others!)
- Spooky behavior = look for numerical issues

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 if 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, many bugs.

Broadcasting Example

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

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

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

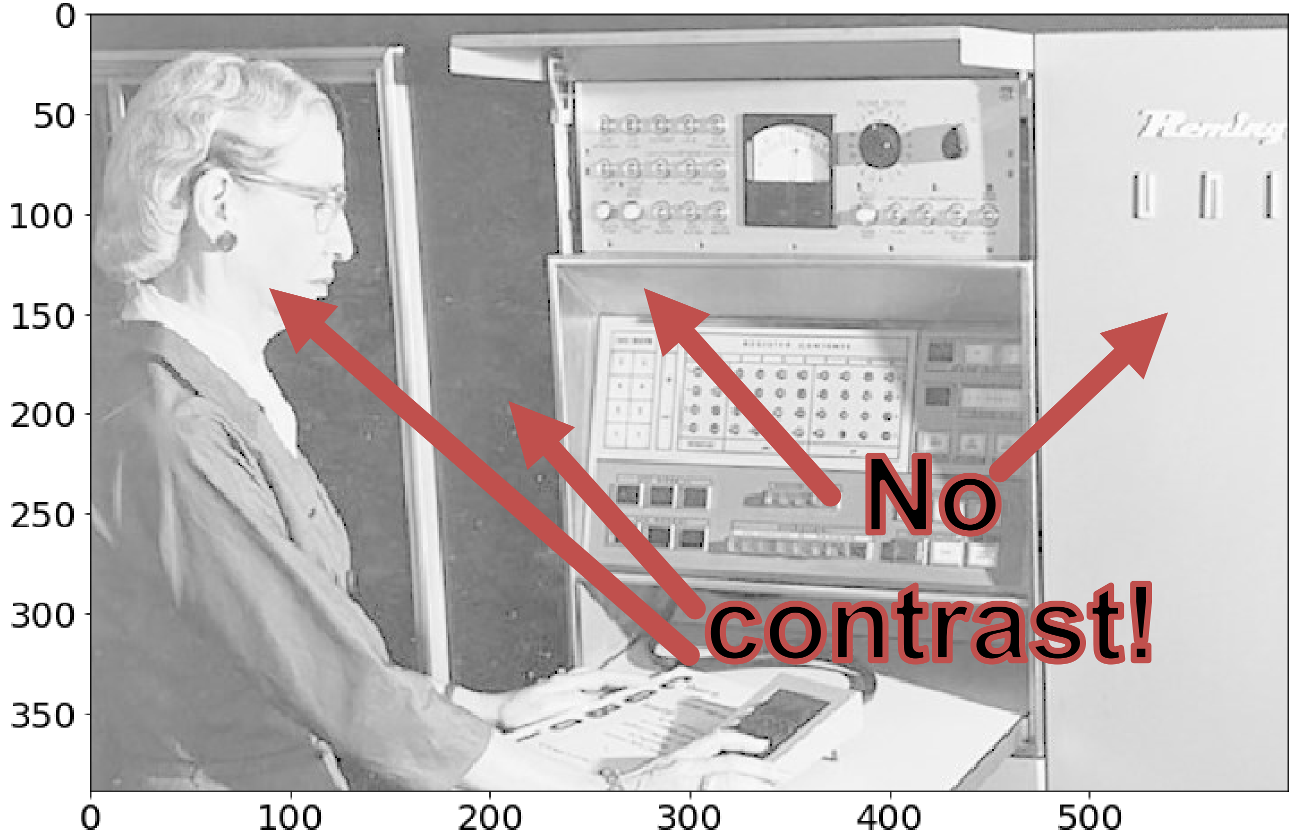
Blue stuff is assumed / broadcast

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

Images as Matrices

Suppose someone hands you this matrix.
What’s wrong with it?

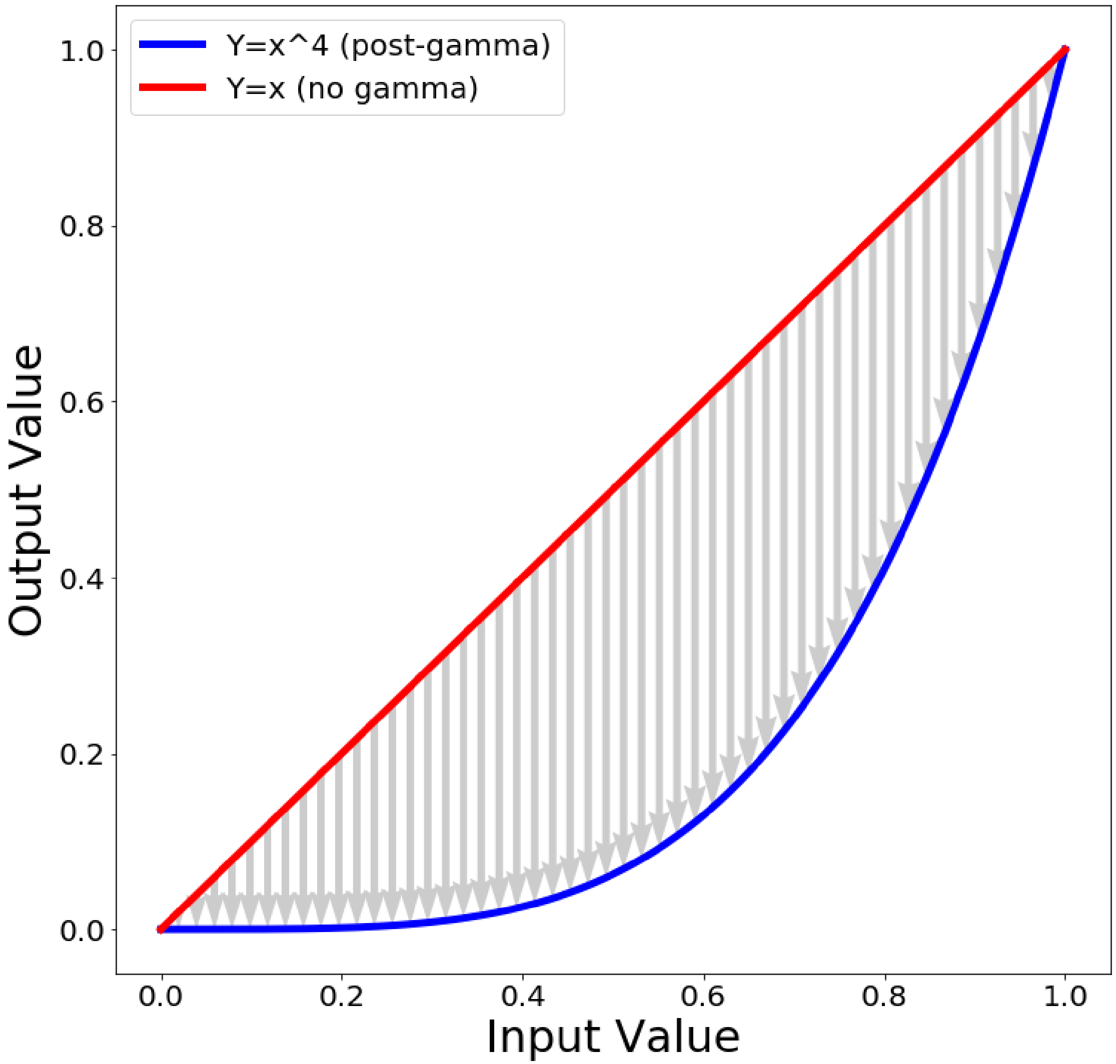


Contrast – Gamma curve

Typical way to change the contrast is to apply a nonlinear correction

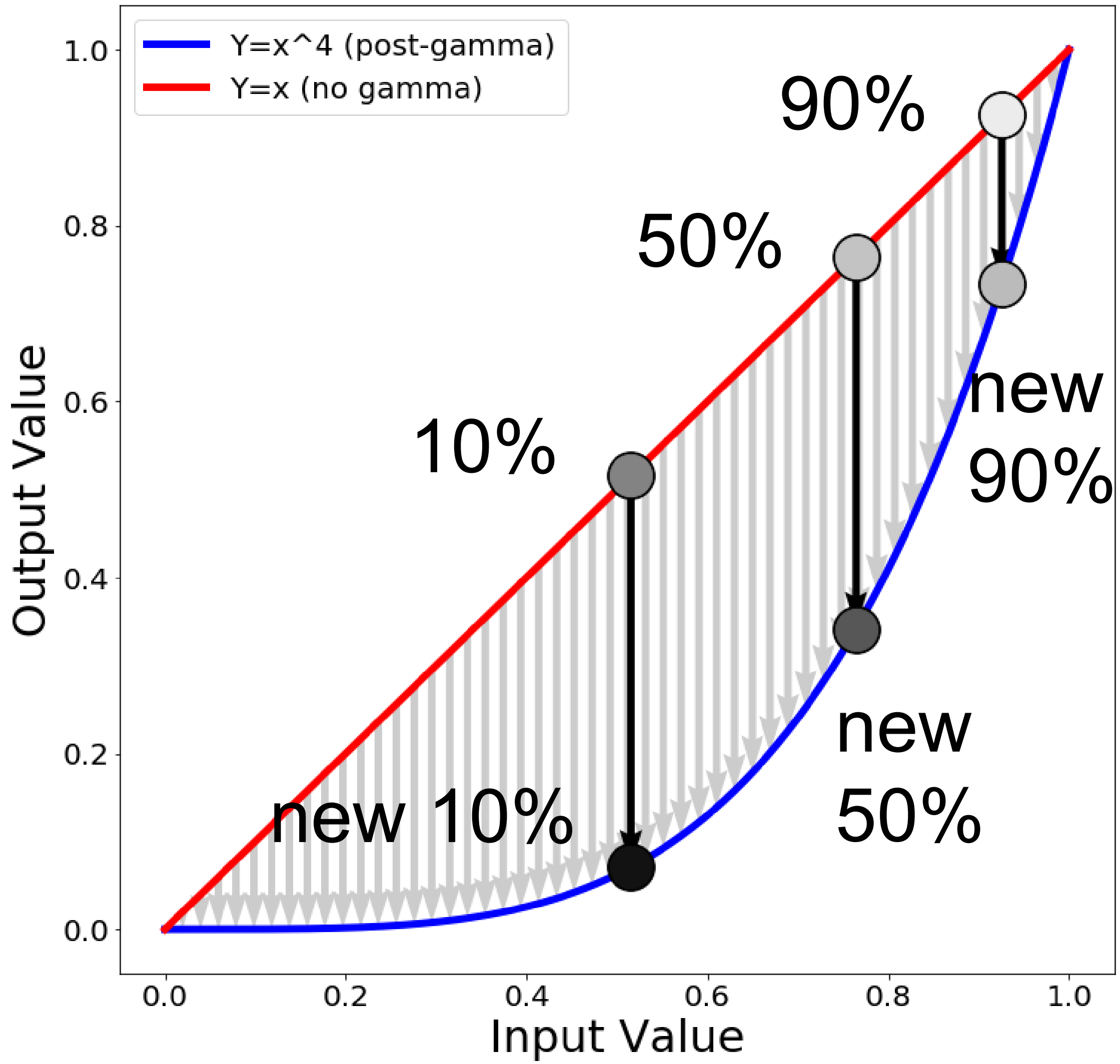
$$\text{pixelvalue}^{\gamma}$$

The quantity γ controls how much contrast gets added



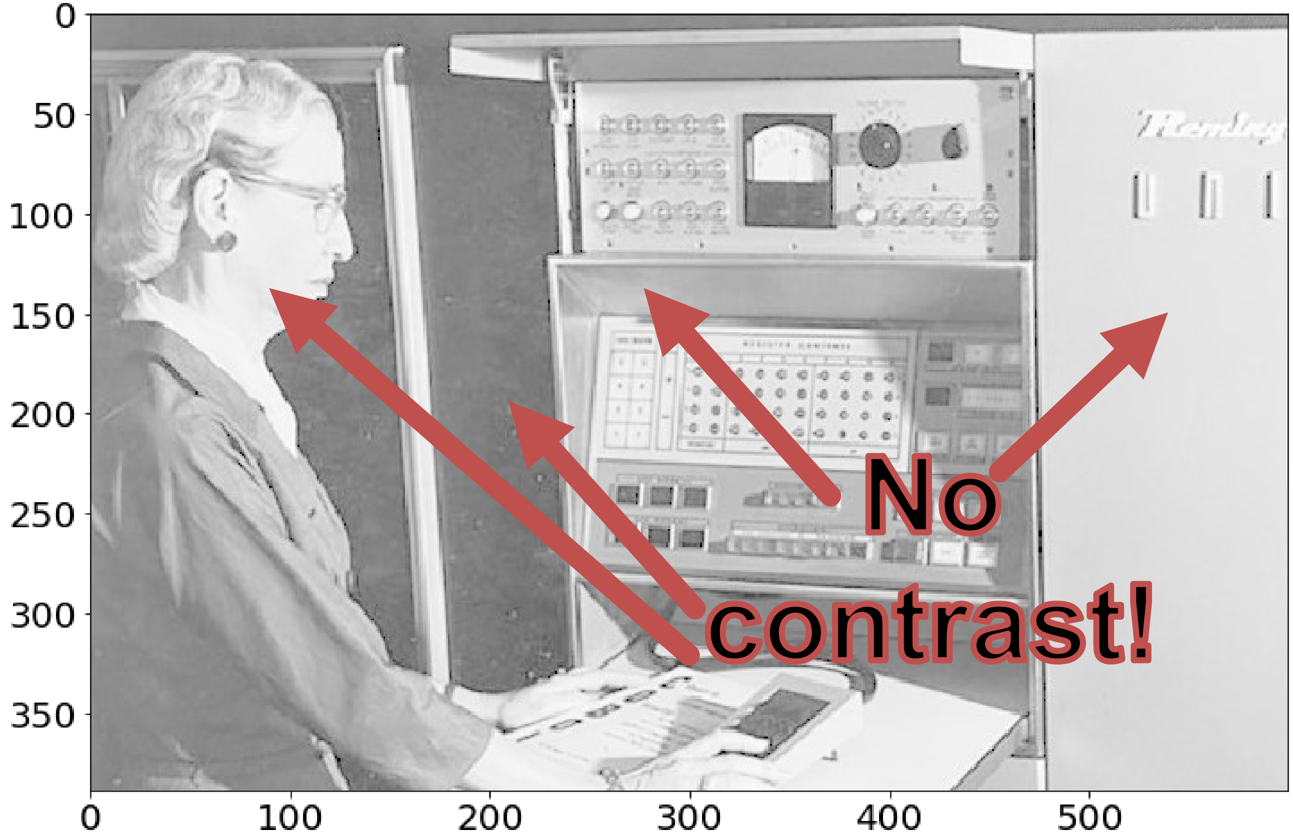
Contrast – Gamma curve

Now the darkest regions (10th pctile) are **much** darker than the moderately dark regions (50th pctile).



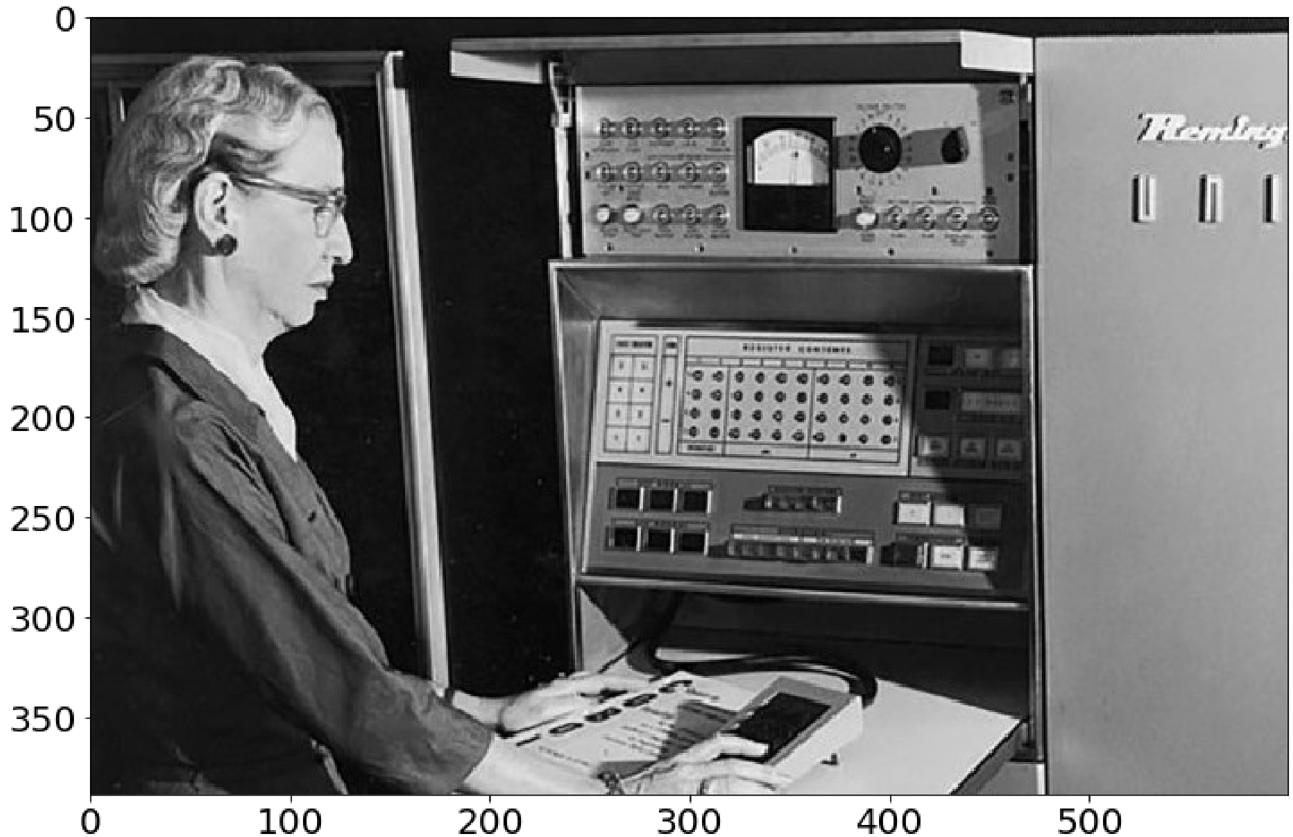
Images as Matrices

Suppose someone hands you this matrix.
The contrast is wrong!



Results

Phew! Much Better.



Implementation

Python+Numpy (right way):

```
imNew = im**4
```

Python+Numpy (slow way – **why?**):

```
imNew = np.zeros(im.shape)
for y in range(im.shape[0]):
  for x in range(im.shape[1]):
    imNew[y,x] = im[y,x]**expFactor
```

Element-wise Operations

Element-wise power – beware notation

$$(\boldsymbol{A}^p)_{ij} = A_{ij}^p$$

"Hadamard Product" / Element-wise multiplication

$$(\boldsymbol{A} \odot \boldsymbol{B})_{ij} = \boldsymbol{A}_{ij} * \boldsymbol{B}_{ij}$$

Element-wise division

$$(\boldsymbol{A}/\boldsymbol{B})_{ij} = \frac{A_{ij}}{B_{ij}}$$

Story time: I swear this is relevant

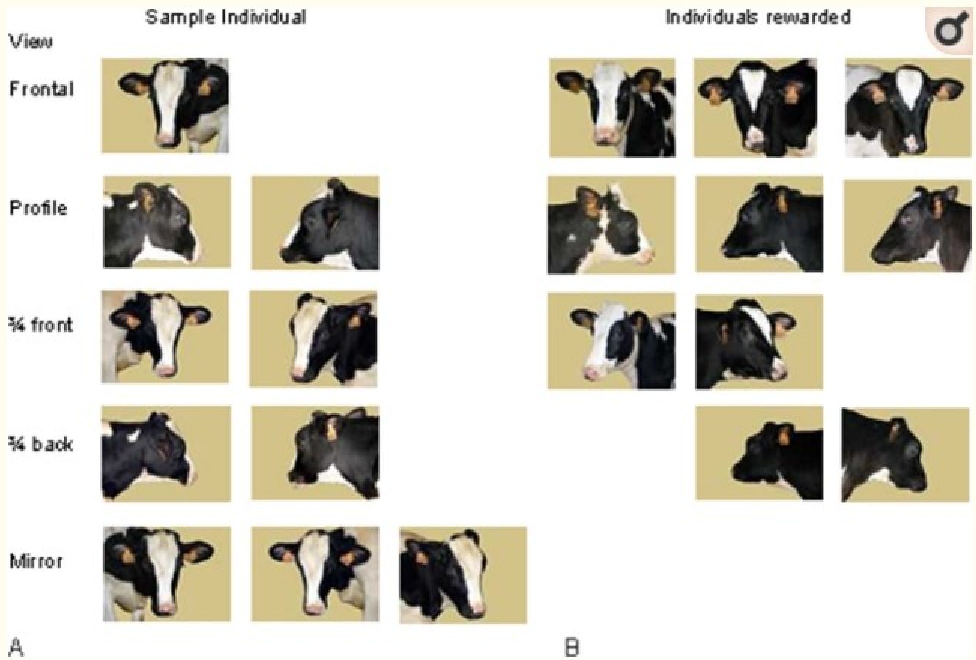


Figure 1

Example of stimuli used in the experiment of the recognition of a familiar *Prim'Holstein* individual.

Ten views represented the sample individuals (A) and ten views represented three other individuals (B). In training, a frontal view of a face (the first line of the figure) of the sample individual (A) had to be discriminated from a frontal view of an individual in the group (B). In generalization test, for each trial, an image of the sample individual (A) and an image of a cow from the group (B) were randomly selected and presented simultaneously. For the second experiment, individuals were unfamiliar *Prim'Holstein* cows, for the third experiment unfamiliar *Normande* cows and for the last experiment unfamiliar *Charolaise* cows.

Image credit: Coulon et al. Individual Recognition in Domestic Cattle (Bos taurus): Evidence from 2D-Images of Heads from Different Breeds. PLoS One. 2009; 4(2): e4441.

Sums Across Axes

Suppose have Nx2 matrix **A**

$$\boldsymbol{A} = \begin{bmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_n & y_n \end{bmatrix}$$

ND col. vec.

$$\Sigma(\boldsymbol{A}, 1) = \begin{bmatrix} x_1 + y_1 \\ \vdots \\ x_n + y_n \end{bmatrix}$$

2D row vec

$$\Sigma(\boldsymbol{A}, 0) = \left[\sum_{i=1}^{n} x_i \quad , \sum_{i=1}^{n} y_i \right]$$

Note – libraries distinguish between N-D column vector and Nx1 matrix.

Vectorizing Example

- Suppose I represent each image as a 128-dimensional vector
- I want to compute all the pairwise distances between {$\mathbf{x}_1$, …, $\mathbf{x}_N$} and {$\mathbf{y}_1$, …, $\mathbf{y}_M$} so I can find, the nearest $\mathbf{y}_j$ for every $\mathbf{x}_i$
- Identity: $\|\boldsymbol{x}-\boldsymbol{y}\|^2 = \|\boldsymbol{x}\|^2 + \|\boldsymbol{y}\|^2 - 2\boldsymbol{x}^T\boldsymbol{y}$
- Or: $\|\boldsymbol{x}-\boldsymbol{y}\| = \left(\|\boldsymbol{x}\|^2 + \|\boldsymbol{y}\|^2 - 2\boldsymbol{x}^T\boldsymbol{y}\right)^{1/2}$

Vectorizing Example

$$\boldsymbol{X} = \begin{bmatrix} - & \boldsymbol{x}_1 & - \\ & \vdots & \\ - & \boldsymbol{x}_N & - \end{bmatrix} \quad \boldsymbol{Y} = \begin{bmatrix} - & \boldsymbol{y}_1 & - \\ & \vdots & \\ - & \boldsymbol{y}_M & - \end{bmatrix} \quad \boldsymbol{Y}^{\boldsymbol{T}} = \begin{bmatrix} | & & | \\ \boldsymbol{y}_1 & \cdots & \boldsymbol{y}_M \\ | & & | \end{bmatrix}$$

Compute a Nx1 vector of norms (can also do Mx1)

$$\boldsymbol{\Sigma}\left(\boldsymbol{X}^2, \boldsymbol{1}\right) = \begin{bmatrix} \|\boldsymbol{x}_{\boldsymbol{1}}\|^2 \\ \vdots \\ \|\boldsymbol{x}_{\boldsymbol{N}}\|^2 \end{bmatrix}$$

Compute a NxM matrix of dot products

$$\left(\boldsymbol{X}\boldsymbol{Y}^{\boldsymbol{T}}\right)_{ij} = \boldsymbol{x}_{\boldsymbol{i}}^{\boldsymbol{T}} \boldsymbol{y}_{\boldsymbol{j}}$$

Vectorizing Example

$$\mathbf{D} = \left(\Sigma(\boldsymbol{X}^2, 1) + \Sigma(\boldsymbol{Y}^2, 1)^{\boldsymbol{T}} - 2\boldsymbol{X}\boldsymbol{Y}^{\boldsymbol{T}}\right)^{1/2}$$

$$\begin{bmatrix} \|\boldsymbol{x}_1\|^2 \\ \vdots \\ \|\boldsymbol{x}_N\|^2 \end{bmatrix} + \begin{bmatrix} \|\boldsymbol{y}_1\|^2 & \cdots & \|\boldsymbol{y}_M\|^2 \end{bmatrix}$$

$$\begin{bmatrix} \|\boldsymbol{x}_1\|^2 + \|\boldsymbol{y}_1\|^2 & \cdots & \|\boldsymbol{x}_1\|^2 + \|\boldsymbol{y}_M\|^2 \\ \vdots & \ddots & \vdots \\ \|\boldsymbol{x}_N\|^2 + \|\boldsymbol{y}_1\|^2 & \cdots & \|\boldsymbol{x}_N\|^2 + \|\boldsymbol{y}_M\|^2 \end{bmatrix}$$

Why?

$$\left(\Sigma(\boldsymbol{X}^2, 1) + \Sigma(\boldsymbol{Y}^2, 1)^T\right)_{ij} = \|\boldsymbol{x}_i\|^2 + \|\boldsymbol{y}_j\|^2$$

Vectorizing Example

$$\mathbf{D} = \left(\Sigma(\boldsymbol{X}^{\mathbf{2}}, 1) + \Sigma(\boldsymbol{Y}^{\mathbf{2}}, 1)^{\boldsymbol{T}} - 2\boldsymbol{X}\boldsymbol{Y}^{\boldsymbol{T}}\right)^{1/2}$$

$$\mathbf{D}_{ij} = \|\boldsymbol{x}_{\boldsymbol{i}}\|^2 + \left\|\boldsymbol{y}_{\boldsymbol{j}}\right\|^2 - 2\boldsymbol{x}_{\boldsymbol{i}}^{\boldsymbol{T}}\boldsymbol{y}_{\boldsymbol{j}}$$

Numpy code:

```
XNorm = np.sum(X**2,axis=1,keepdims=True)
YNorm = np.sum(Y**2,axis=1,keepdims=True)
D = (XNorm+YNorm.T-2*np.dot(X,Y.T))**0.5
```

*May have to make sure this is at least 0 (sometimes roundoff issues happen)

Does it Make a Difference?

Computing pairwise distances between 300 and 400 128-dimensional vectors

1. for x in X, for y in Y, using native python: 9s
2. for x in X, for y in Y, using numpy to compute distance: 0.8s
3. vectorized: 0.0045s (~2000x faster than 1, 175x faster than 2)

Expressing things in primitives that are optimized is usually faster

Rank

- Rank of a nxn matrix **A** – number of linearly independent columns (**or rows**) of A / the dimension of the span of the columns
- Matrices with *full rank* (n x n, rank n) behave nicely: can be inverted, span the full output space, are one-to-one.

Symmetric Matrices

- Symmetric: $\boldsymbol{A}^{\boldsymbol{T}} = \boldsymbol{A}$ or $\boldsymbol{A}_{ij} = \boldsymbol{A}_{ji}$
- Have **lots** of special properties

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Any matrix of the form $\boldsymbol{A} = \boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}$ is symmetric.

Quick check:

$$\boldsymbol{A}^{\boldsymbol{T}} = \left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\right)^{\boldsymbol{T}}$$
$$\boldsymbol{A}^{\boldsymbol{T}} = \boldsymbol{X}^{\boldsymbol{T}}\left(\boldsymbol{X}^{\boldsymbol{T}}\right)^{\boldsymbol{T}}$$
$$\boldsymbol{A}^{\boldsymbol{T}} = \boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}$$

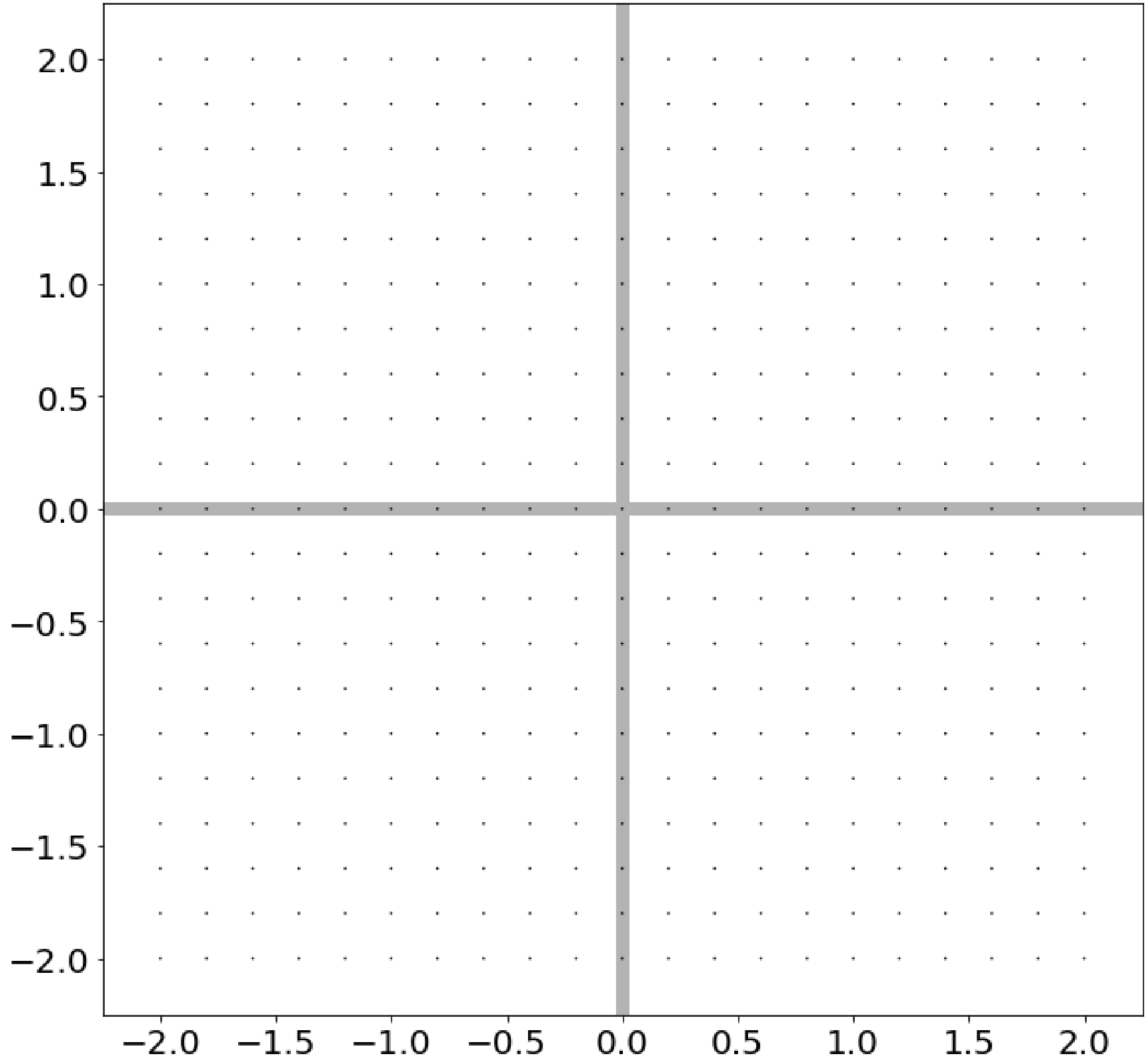
Special Matrices – Rotations

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

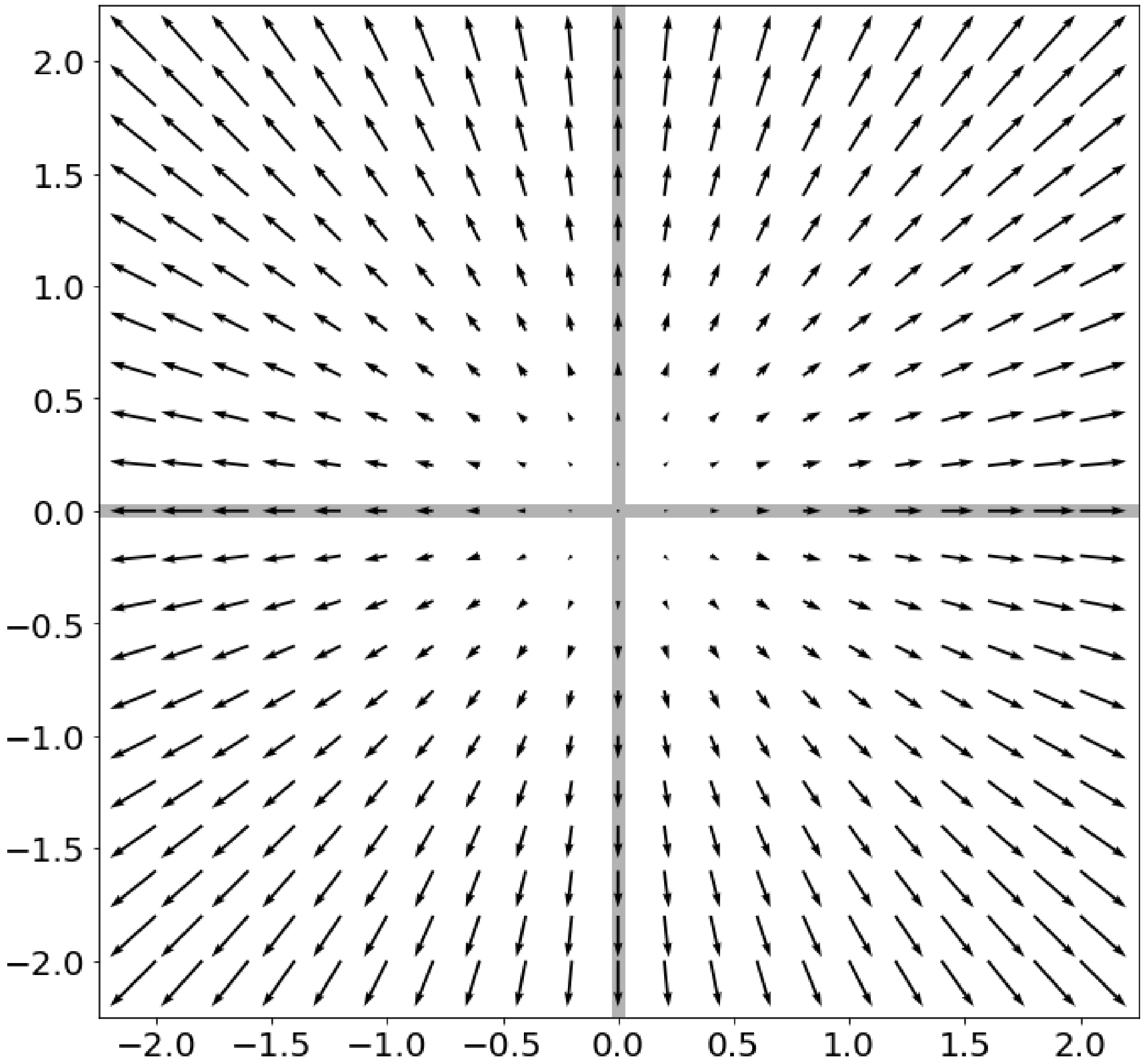
- Rotation matrices $\boldsymbol{R}$ rotate vectors and ***do not change vector L2 norms*** ($\|\boldsymbol{Rx}\|_2 = \|\boldsymbol{x}\|_2$)
- Every row/column is unit norm
- Every row is linearly independent
- Transpose is inverse $\boldsymbol{RR^T} = \boldsymbol{R^T R} = \boldsymbol{I}$
- Determinant is 1 (otherwise it’s also a coordinate flip/reflection), eigenvalues are 1

Eigensystems

- An eigenvector $\boldsymbol{v_i}$ and eigenvalue λ_i of a matrix $\boldsymbol{A}$ satisfy $\boldsymbol{A}\boldsymbol{v_i} = \lambda_i \boldsymbol{v_i}$ ($\boldsymbol{A}\boldsymbol{v_i}$ is scaled by λ_i)
- Vectors and values are always paired and typically you assume $\|\boldsymbol{v_i}\|^2 = 1$
- Biggest eigenvalue of A gives bounds on how much $f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$ stretches a vector **x**.
- Hints of what people really mean:
  - "Largest eigenvector" = vector w/ largest value
  - Spectral just means there's eigenvectors



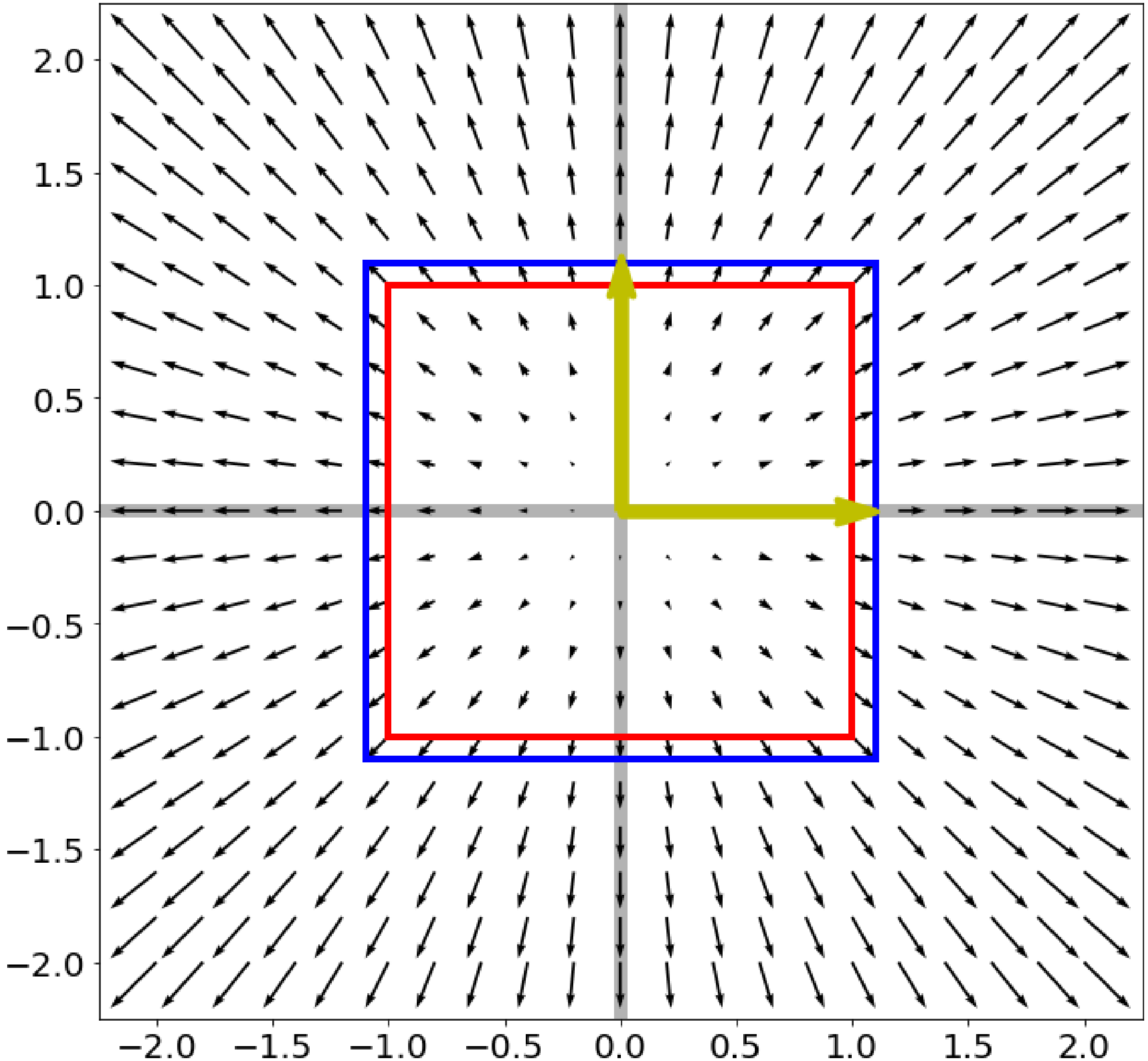
Suppose I have points in a grid



Now I apply f(**x**) = **Ax** to these points

Pointy-end: **Ax** . Non-Pointy-End: **x**

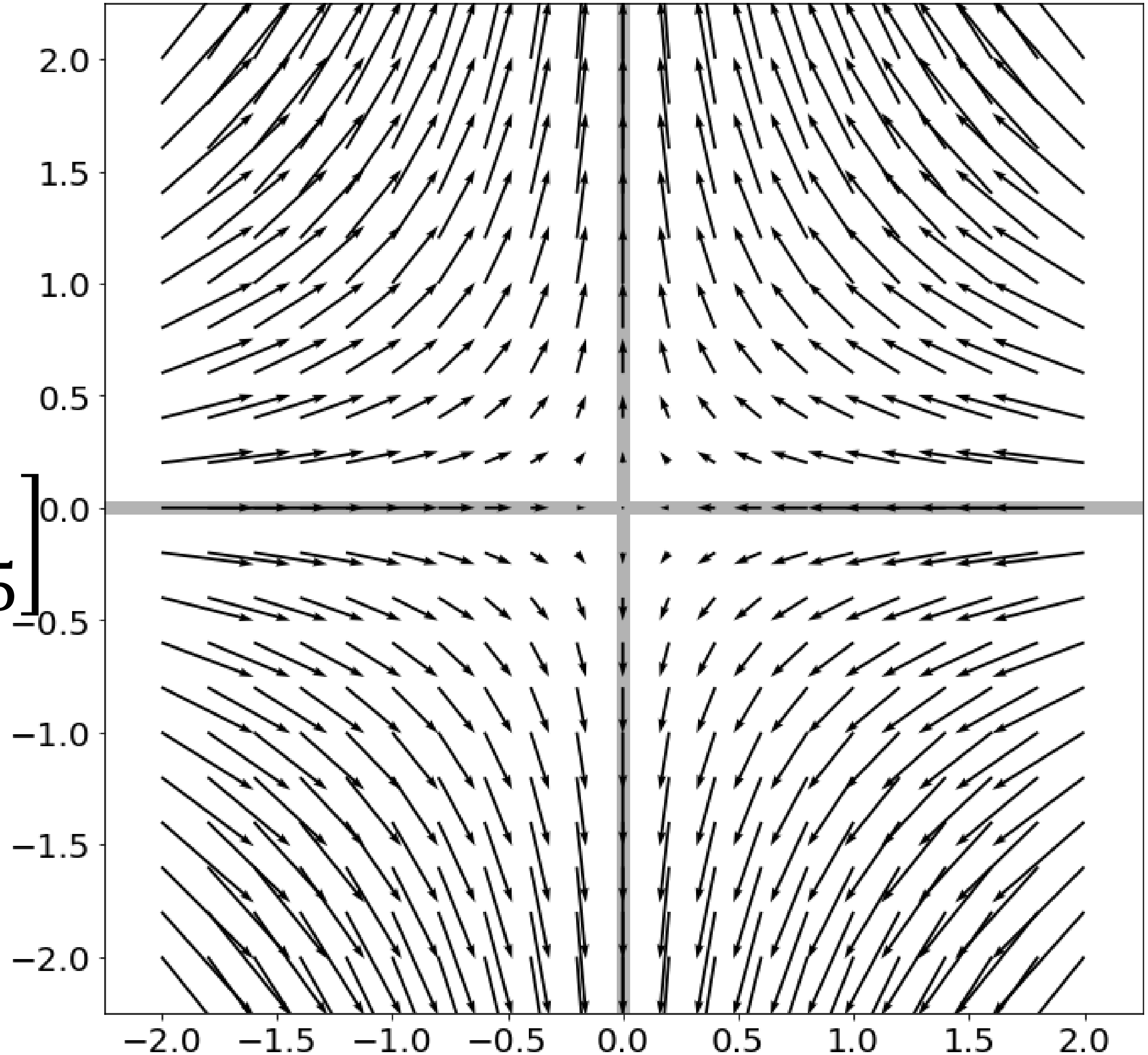
$$A = \begin{bmatrix} 1.1 & 0 \\ 0 & 1.1 \end{bmatrix}$$



Red box – unit square, Blue box – after f(**x**) = **Ax**.

What are the yellow lines and why?

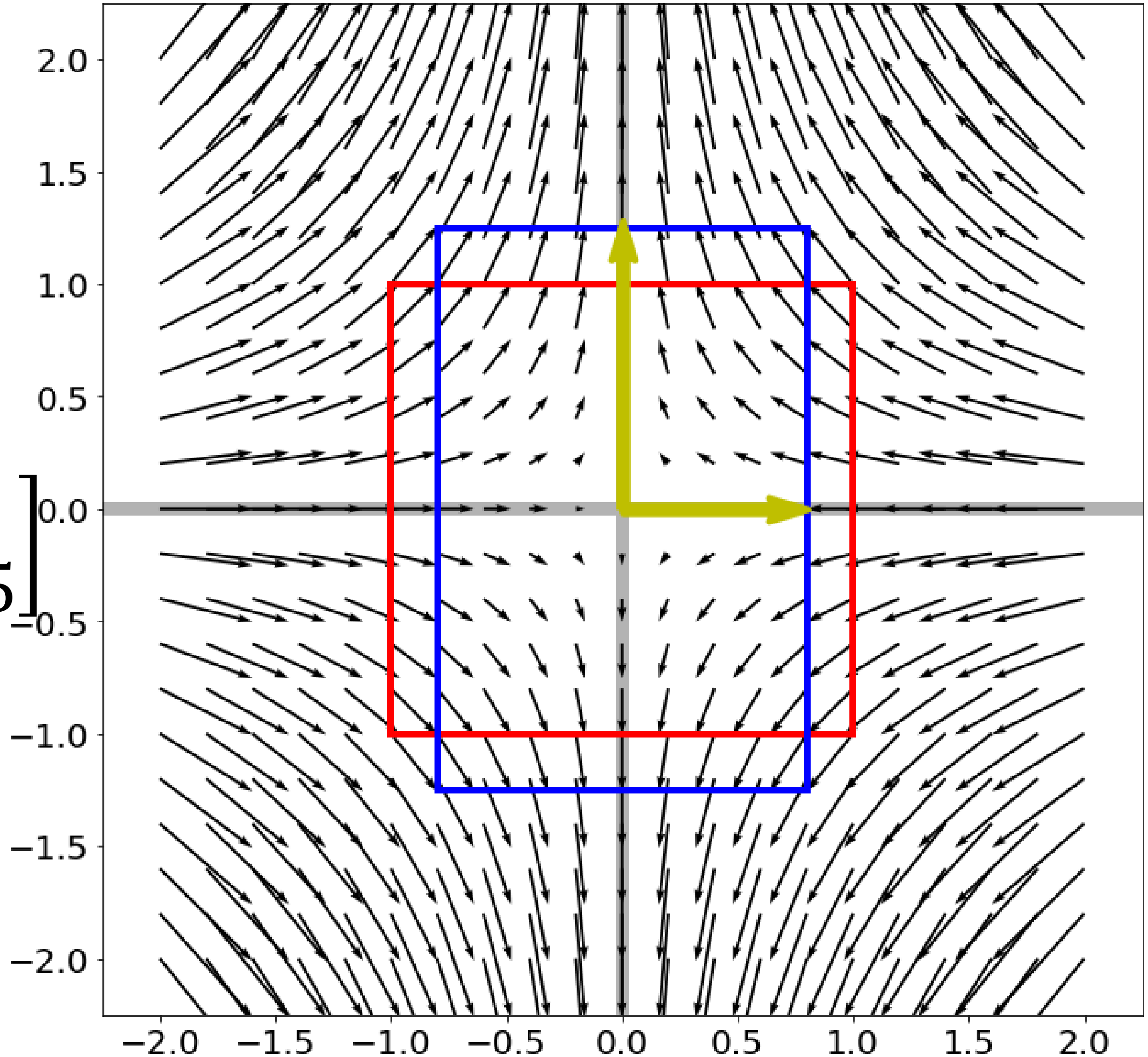
$$\boldsymbol{A} = \begin{bmatrix} 0.8 & 0 \\ 0 & 1.25 \end{bmatrix}$$



Now I apply f(**x**) = **Ax** to these points

Pointy-end: **Ax** . Non-Pointy-End: **x**

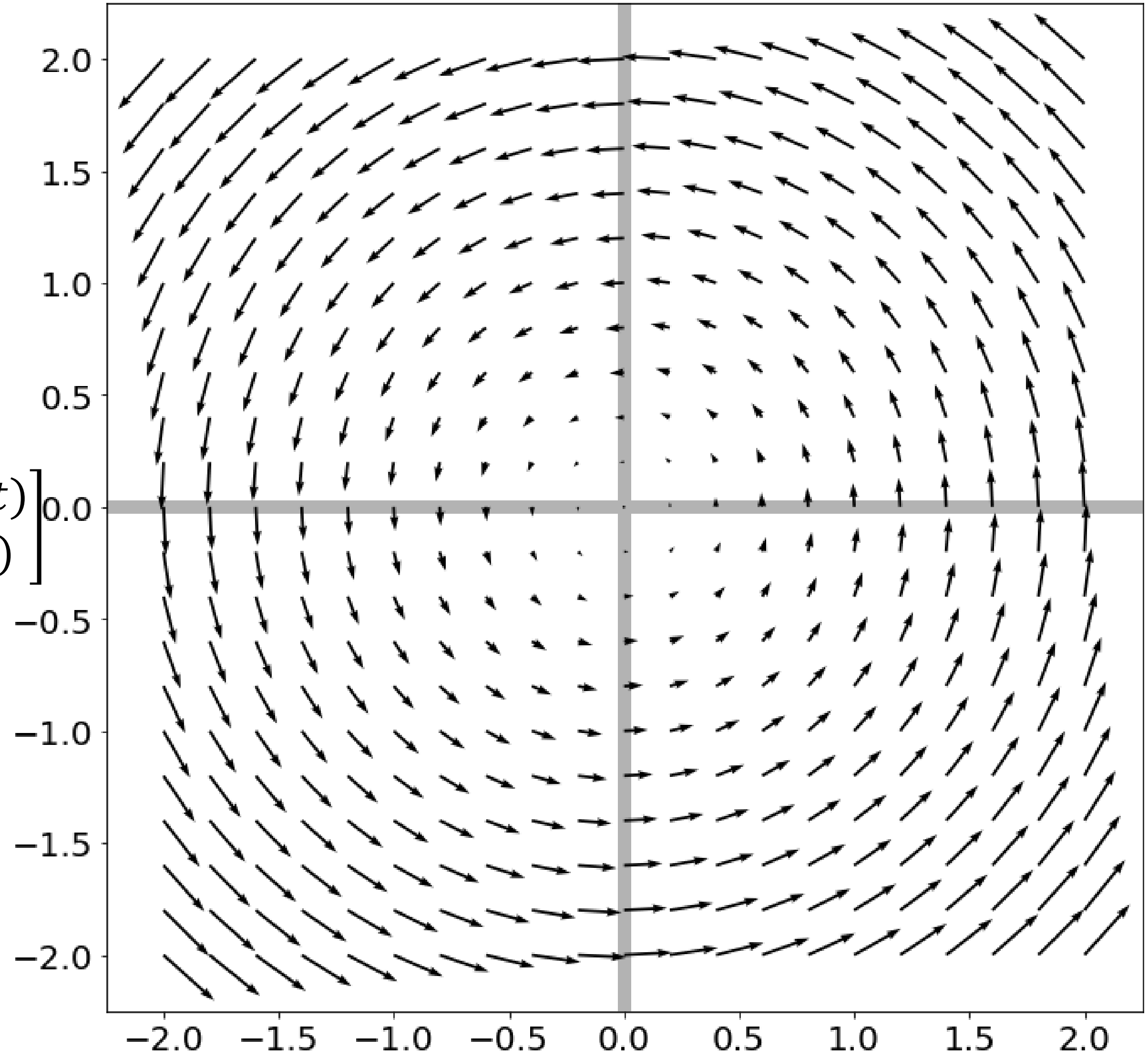
$$A = \begin{bmatrix} 0.8 & 0 \\ 0 & 1.25 \end{bmatrix}$$



Red box – unit square, Blue box – after f(**x**) = **Ax**.

What are the yellow lines and why?

$$A = \begin{bmatrix} \cos(t) & -\sin(t) \\ \sin(t) & \cos(t) \end{bmatrix}$$



Red box – unit square, Blue box – after f($\mathbf{x}$) = $\mathbf{Ax}$.

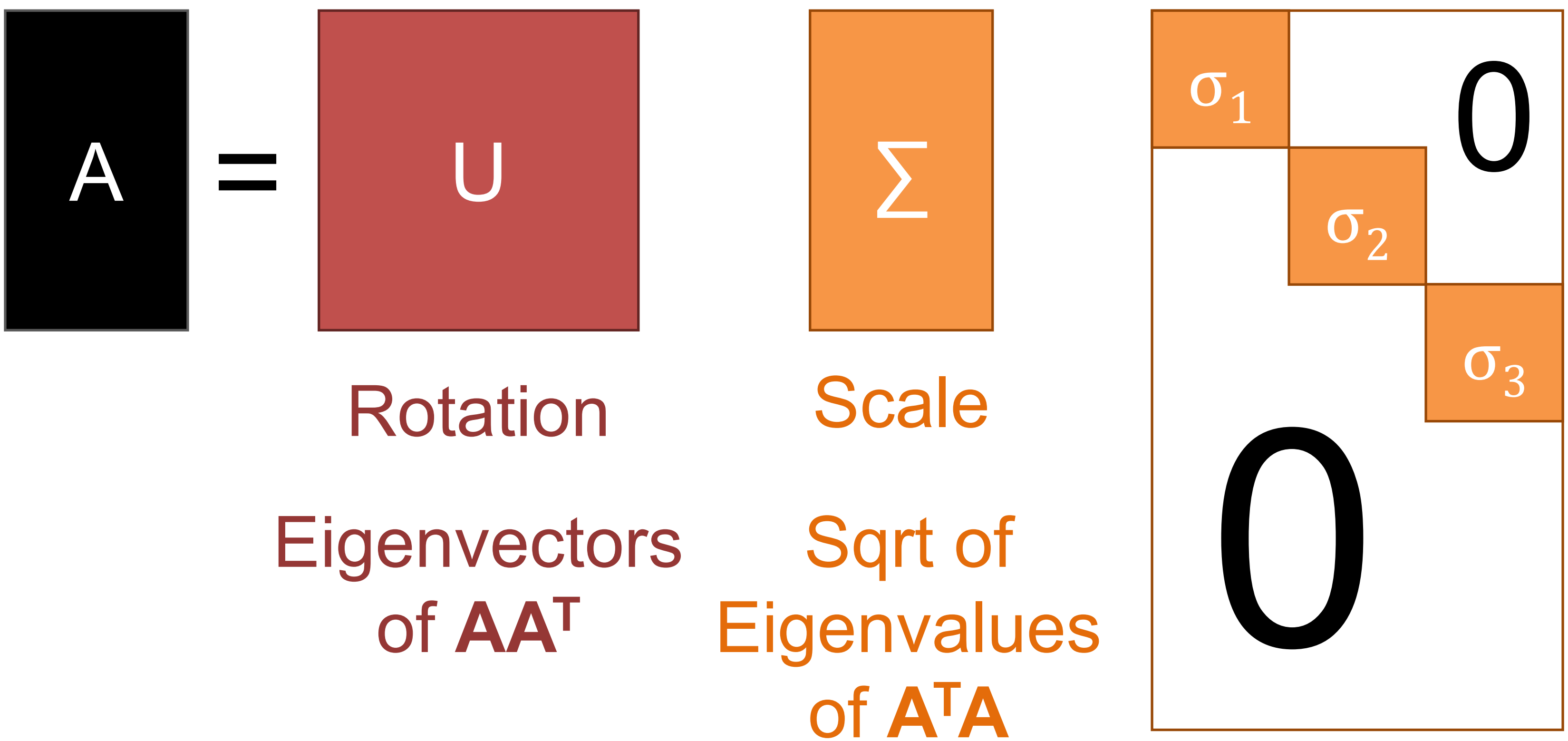
Can we draw any yellow lines?

Eigenvectors of Symmetric Matrices

- Always n mutually orthogonal eigenvectors with n (not necessarily) distinct eigenvalues
- For symmetric $\boldsymbol{A}$, the eigenvector with the largest eigenvalue maximizes $\frac{\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}}$ (smallest/min)
- So for unit vectors (where $\boldsymbol{x}^T\boldsymbol{x} = 1$), that eigenvector maximizes $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$
- A surprisingly large number of optimization problems rely on (max/min)imizing this

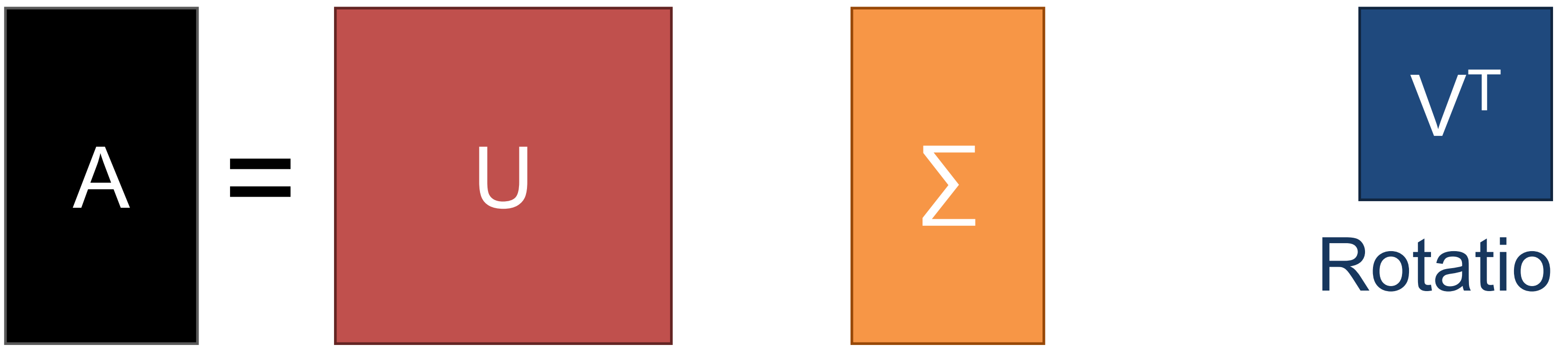
The Singular Value Decomposition

Can **always** write a mxn matrix **A** as: $A = U\Sigma V^T$



The Singular Value Decomposition

Can **always** write a mxn matrix **A** as: $\boldsymbol{A} = \boldsymbol{U\Sigma V^T}$



Rotation

Rotation

Scale

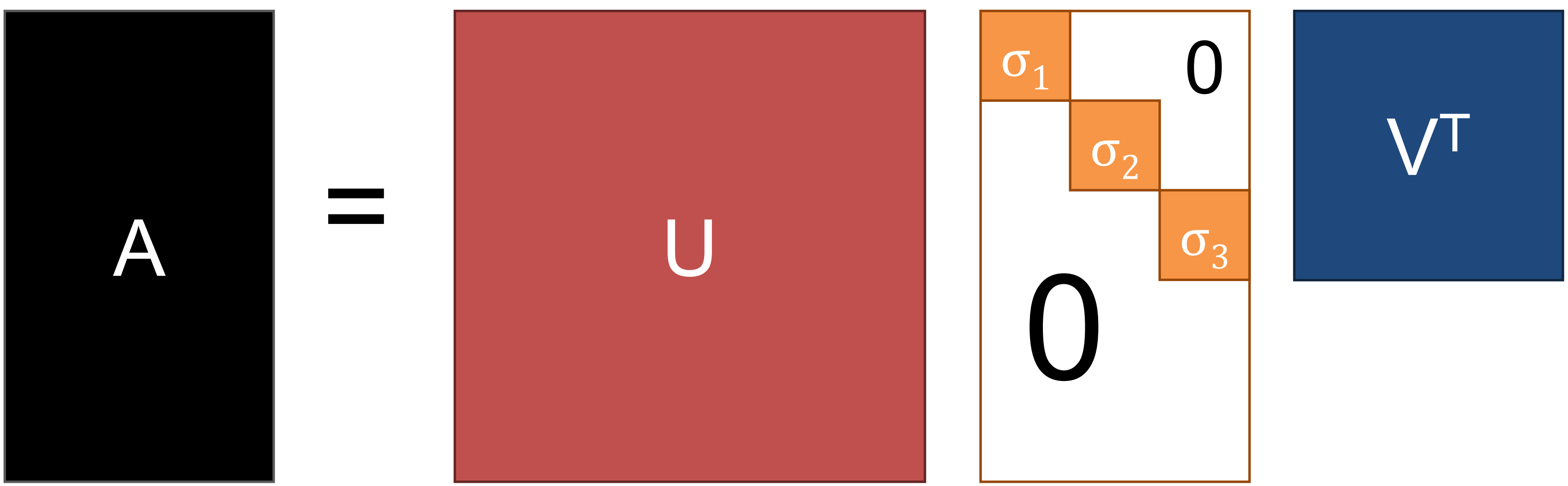
Eigenvectors of $\mathbf{AA^T}$

Sqrt of Eigenvalues of $\mathbf{A^TA}$

Eigenvectors of $\mathbf{A^TA}$

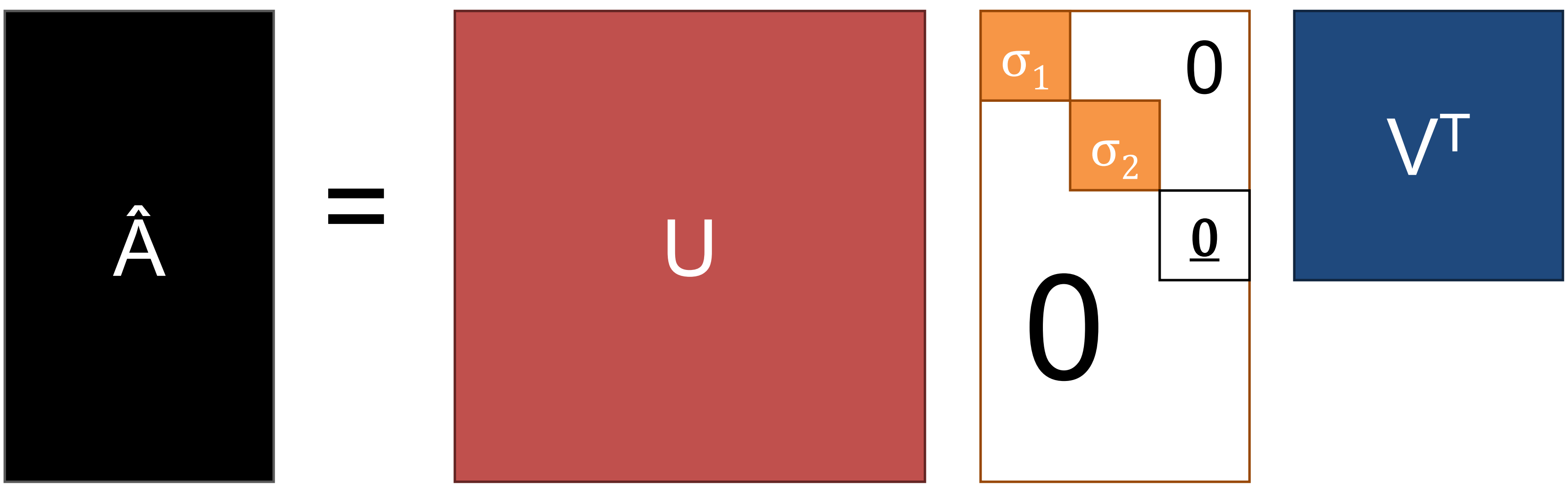
Singular Value Decomposition

- *Every* matrix is a rotation, scaling, and rotation
- Number of non-zero singular values = rank / number of linearly independent vectors
- "Closest" matrix to **A** with a lower rank



Singular Value Decomposition

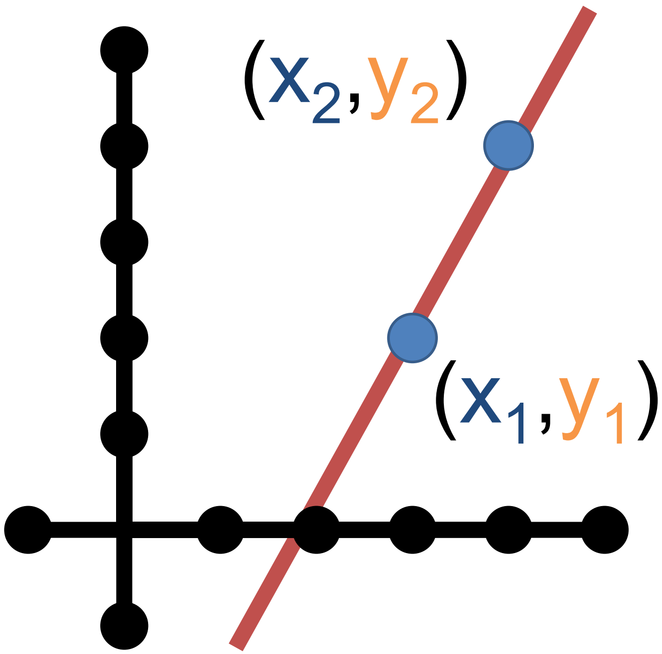
- *Every* matrix is a rotation, scaling, and rotation
- Number of non-zero singular values = rank / number of linearly independent vectors
- "Closest" matrix to **A** with a lower rank



Singular Value Decomposition

- *Every* matrix is a rotation, scaling, and rotation
- Number of non-zero singular values = rank / number of linearly independent vectors
- "Closest" matrix to **A** with a lower rank
- Secretly behind basically many things you do with matrices

Least-Squares



Start with two points (x_i,y_i)

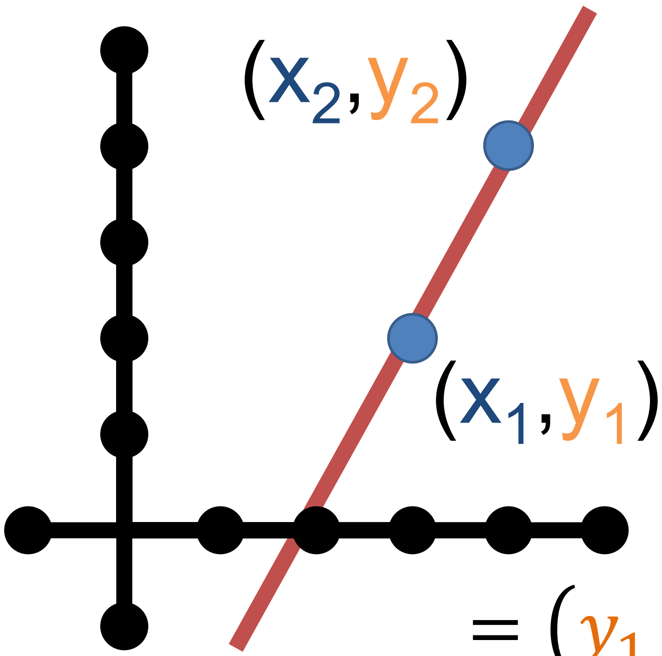
$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{v}$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix}$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} mx_1 + b \\ mx_2 + b \end{bmatrix}$$

We know how to solve this – invert **A** and find **v** (i.e., (m,b) that fits points)

Least-Squares



Start with two points (x_i, y_i)

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{v}$$

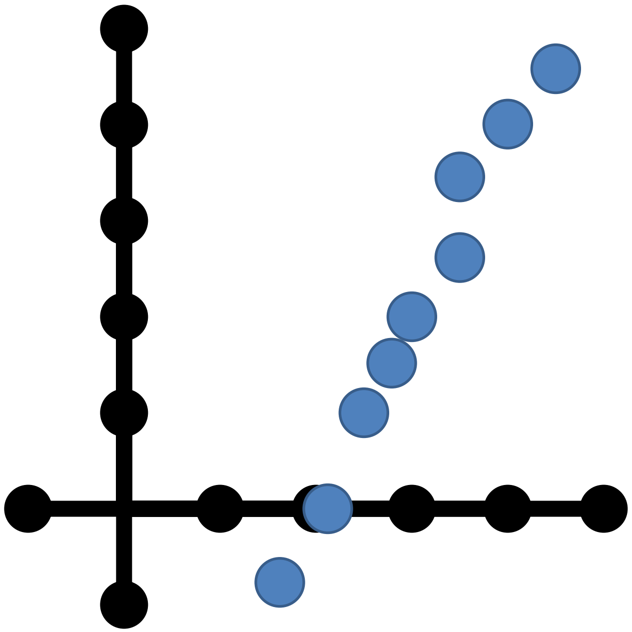
$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix}$$

$$\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{v}\|^2 = \left\| \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} - \begin{bmatrix} mx_1 + b \\ mx_2 + b \end{bmatrix} \right\|^2$$

$$= \left(y_1 - (\underline{mx_1 + b})\right)^2 + \left(y_2 - (\underline{mx_2 + b})\right)^2$$

The sum of squared differences between **the actual value of y** and **what the model says y should be.**

Least-Squares



Suppose there are n > 2 points

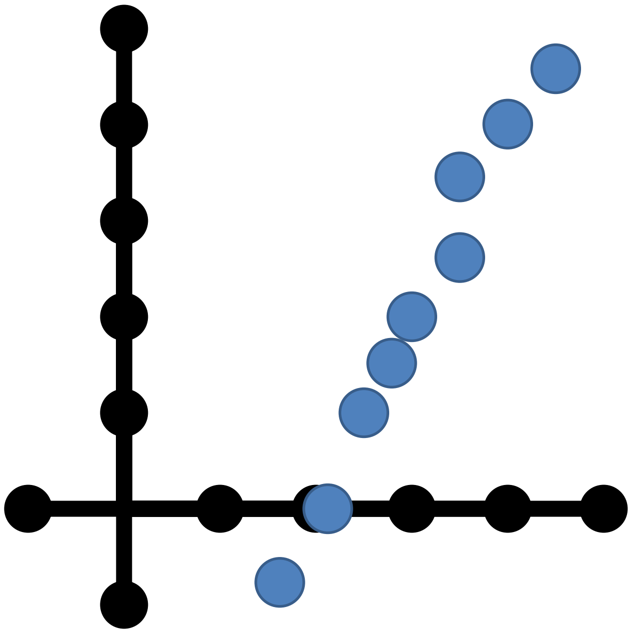
$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{v}$$

$$\begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_N & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix}$$

Compute $||\boldsymbol{y} - \boldsymbol{A}\boldsymbol{v}||^2$ again

$$||\boldsymbol{y} - \boldsymbol{A}\boldsymbol{v}||^2 = \sum_{i=1}^{n} (y_i - (mx_i + b))^2$$

Least-Squares



Suppose there are n > 2 points

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{v}$$

$$\begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_N & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix}$$

Want to minimize $||\boldsymbol{y} - \boldsymbol{A}\boldsymbol{v}||^2$

We can control the entries of **v**, but columns of **A** can't possibly be put together in any way to produce **y**

Solving Least-Squares

Given **y**, **A**, and **v** with **y** = **Av** overdetermined
(**A** tall / more equations than unknowns)
We want to minimize $||\boldsymbol{y} - \boldsymbol{A}\boldsymbol{v}||^2$, or find:

$$\boxed{\arg\min_{\boldsymbol{v}}} ||\boldsymbol{y} - \boldsymbol{A}\boldsymbol{v}||^2$$

(The value of v that makes the expression smallest)

Solution satisfies $\left(\boldsymbol{A}^T\boldsymbol{A}\right)\boldsymbol{v}^* = \boldsymbol{A}^T\boldsymbol{y}$

<u>or</u>

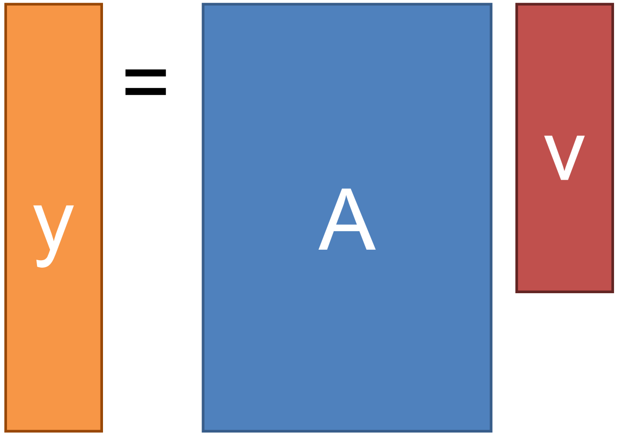
$$\boldsymbol{v}^* = \left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-1}\boldsymbol{A}^T\boldsymbol{y}$$

(Don't actually compute the inverse!)

When is Least-Squares Possible?

Given **y**, **A**, and **v**. Want **y** = **Av**

Want n outputs, have n knobs to fiddle with, every knob is useful if A is full rank.

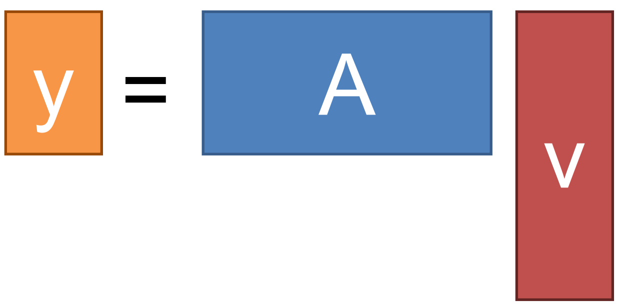


A: rows (outputs) > columns (knobs). Thus can't get precise output you want (not enough knobs). So settle for "closest" knob setting.

When is Least-Squares Possible?

Given **y**, **A**, and **v**. Want **y** = **Av**

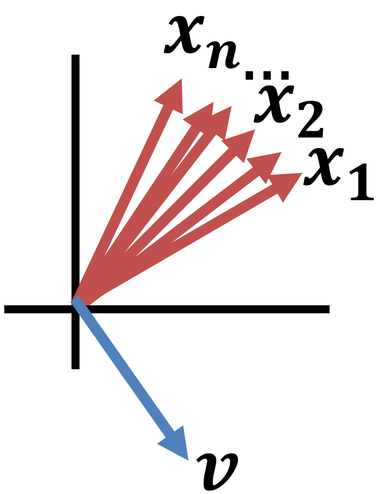
Want n outputs, have n knobs to fiddle with, every knob is useful if A is full rank.



A: columns (knobs) > rows (outputs). Thus, any output can be expressed in infinite ways.

Homogeneous Least-Squares

Given a set of unit vectors (aka directions) $x_1, \ldots, x_n$ and I want vector v that is as orthogonal to all the x_i as possible (for some definition of orthogonal)



Stack x_i into **A**, compute **Av**

$$Av = \begin{bmatrix} - & x_1^T & - \\ & \vdots & \\ - & x_n^T & - \end{bmatrix} v = \begin{bmatrix} x_1^T v \\ \vdots \\ x_n^T v \end{bmatrix}$$

0 if orthog

Compute $\|Av\|^2 = \boxed{\sum_i^n \left(x_i^T v\right)^2}$

Sum of how orthog. v is to each x

Homogeneous Least-Squares

- A lot of times, given a matrix **A** we want to find the **v** that minimizes $\|\boldsymbol{Av}\|^2$.
- I.e., want $\mathbf{v}^* = \arg\min_{\boldsymbol{v}} \|\boldsymbol{Av}\|_2^2$
- What's a trivial solution?
- Set **v = 0** → **Av = 0**
- Exclude this by forcing v to have unit norm

Homogeneous Least-Squares

Let's look at $\|\boldsymbol{A}\boldsymbol{v}\|_2^2$

$\|\boldsymbol{A}\boldsymbol{v}\|_2^2 =$ Rewrite as dot product

$\|\boldsymbol{A}\boldsymbol{v}\|_2^2 = (\mathbf{Av})^{\mathrm{T}}(\mathbf{Av})$ Distribute transpose

$\|\boldsymbol{A}\boldsymbol{v}\|_2^2 = \boldsymbol{v}^{\boldsymbol{T}}\boldsymbol{A}^{\boldsymbol{T}}\mathbf{Av} = \mathbf{v}^{\mathbf{T}}\left(\mathbf{A}^{\mathbf{T}}\mathbf{A}\right)\mathbf{v}$

We want the vector minimizing this quadratic form
Where have we seen this?

Homogeneous Least-Squares

Ubiquitious tool in vision:

$$\arg \min_{\|\boldsymbol{v}\|^2=1} \|\boldsymbol{A}\boldsymbol{v}\|^2$$

$\rightarrow$ **(1)** "Smallest"* eigenvector of $\boldsymbol{A}^T\boldsymbol{A}$

(2) "Smallest" right singular vector of $\boldsymbol{A}$

For min → max, switch smallest → largest

*Note: $\boldsymbol{A}^T\boldsymbol{A}$ is positive semi-definite so it has all non-negative eigenvalues

Derivatives

Remember derivatives?

Derivative: rate at which a function f(x) changes at a point as well as the direction that increases the function

Given quadratic function f(x)

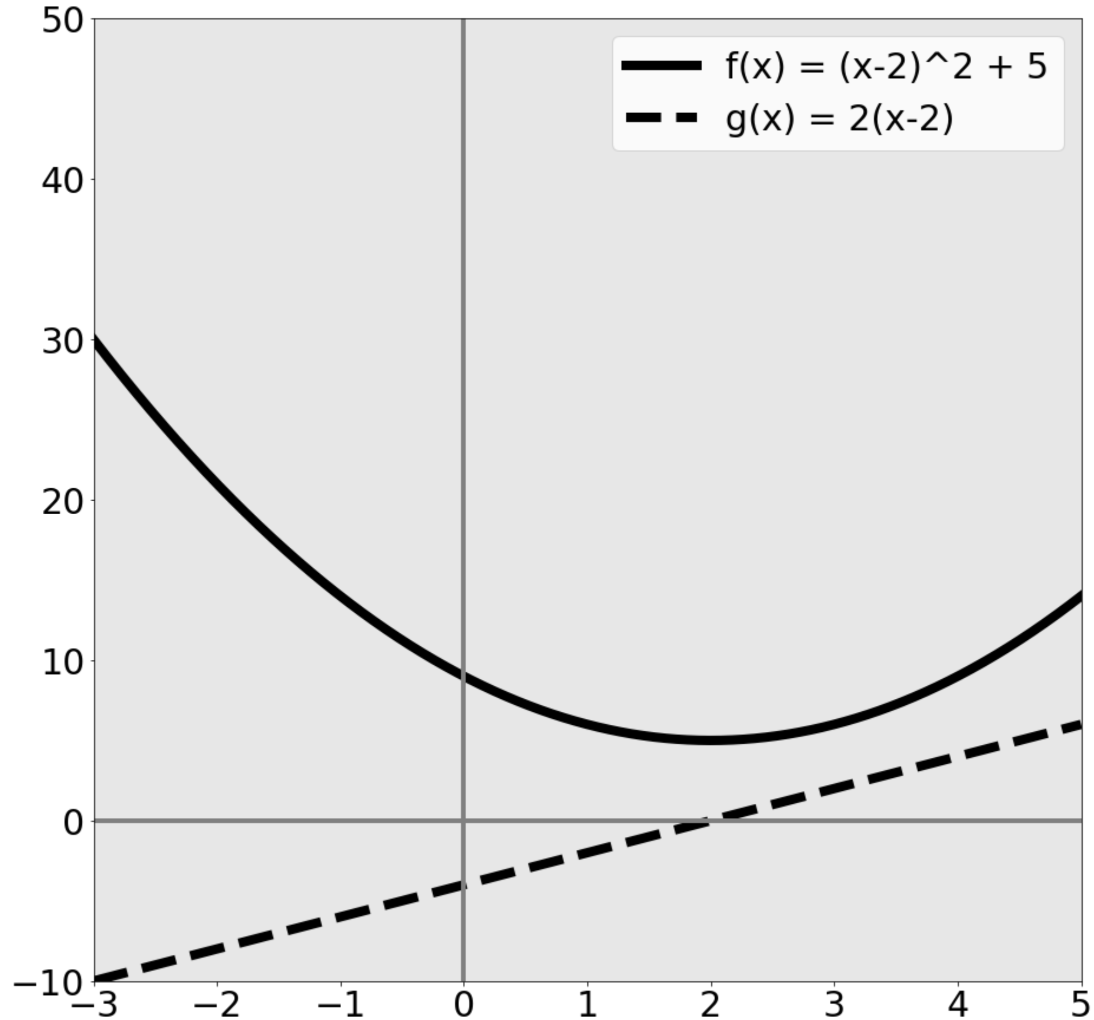
$$f(x,y) = (x-2)^2 + 5$$

$f(x)$ is function

$$g(x) = f'(x)$$

aka

$$g(x) = \frac{d}{dx} f(x)$$



Given quadratic function f(x)

$$f(x, y) = (x-2)^2 + 5$$

What's special about x=2?

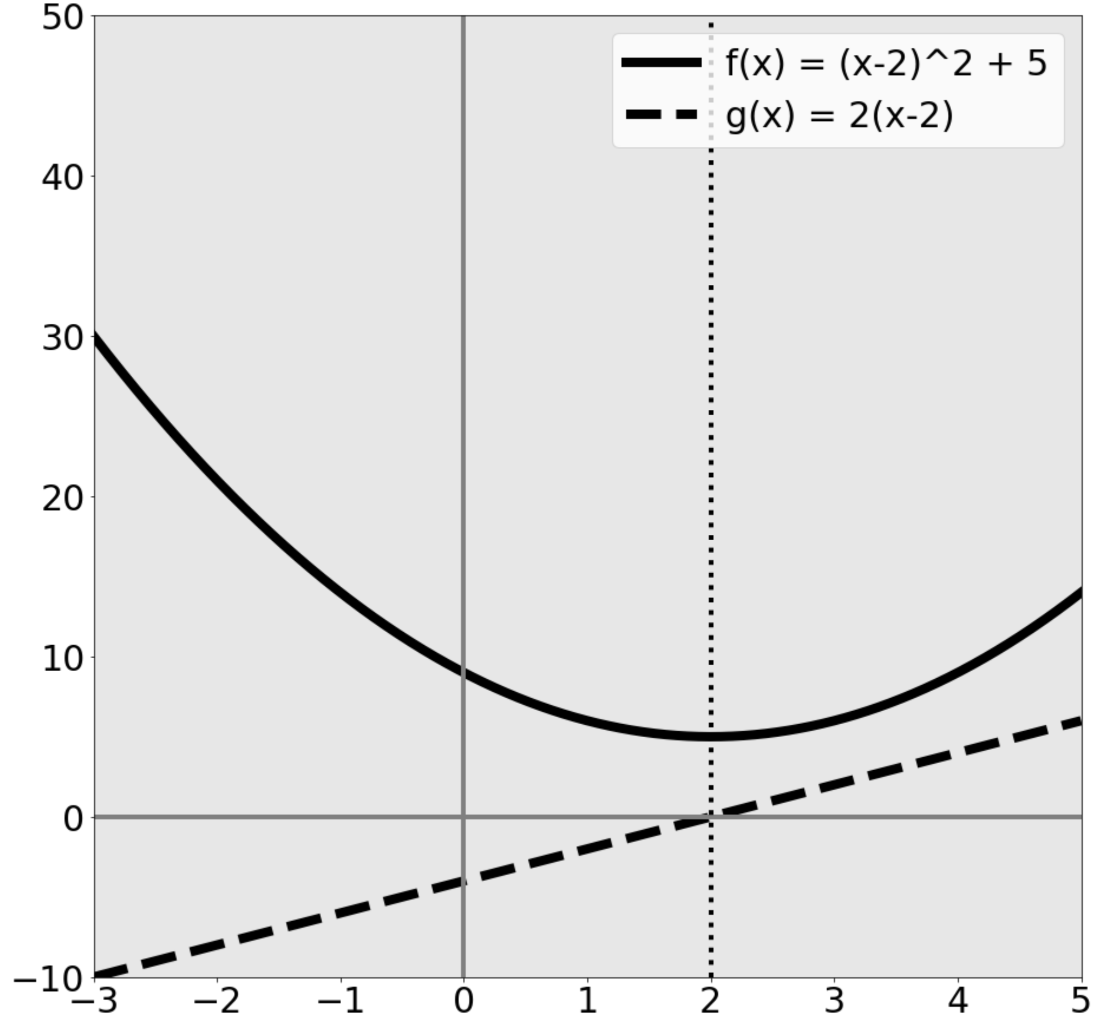
$f(x)$ minim. at 2

$g(x) = 0$ at 2

a = minimum of f →

$g(a) = 0$

Reverse is ***<u>not true</u>***



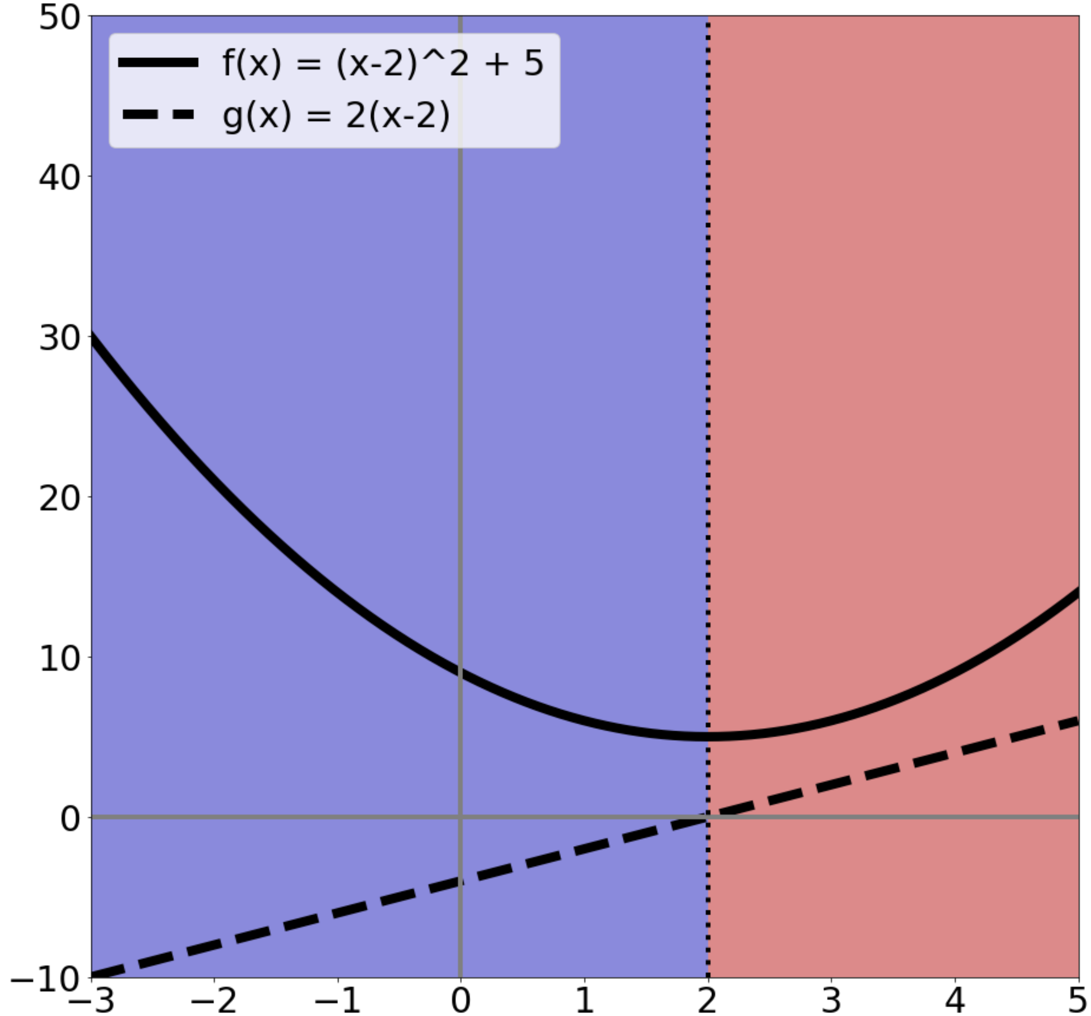
Rates of change

$$f(x, y) = (x-2)^2 + 5$$

Suppose I want to increase f(x) by changing x:

Blue area: move left
Red area: move right

Derivative tells you direction of ascent and rate



What Calculus Should I Know

- Really need intuition
- Need chain rule
- Rest you should look up / use a computer algebra system / use a cookbook
- Partial derivatives (and that’s it from multivariable calculus)

Partial Derivatives

- Pretend other variables are constant, take a derivative. That's it.
- Make our function a function of two variables

$$f(x) = (x-2)^2 + 5$$

$$\frac{\partial}{\partial x} f(x) = 2(x-2) * 1 = 2(x-2)$$

$$f_2(x,y) = (x-2)^2 + 5 + \boxed{(y+1)^2}$$

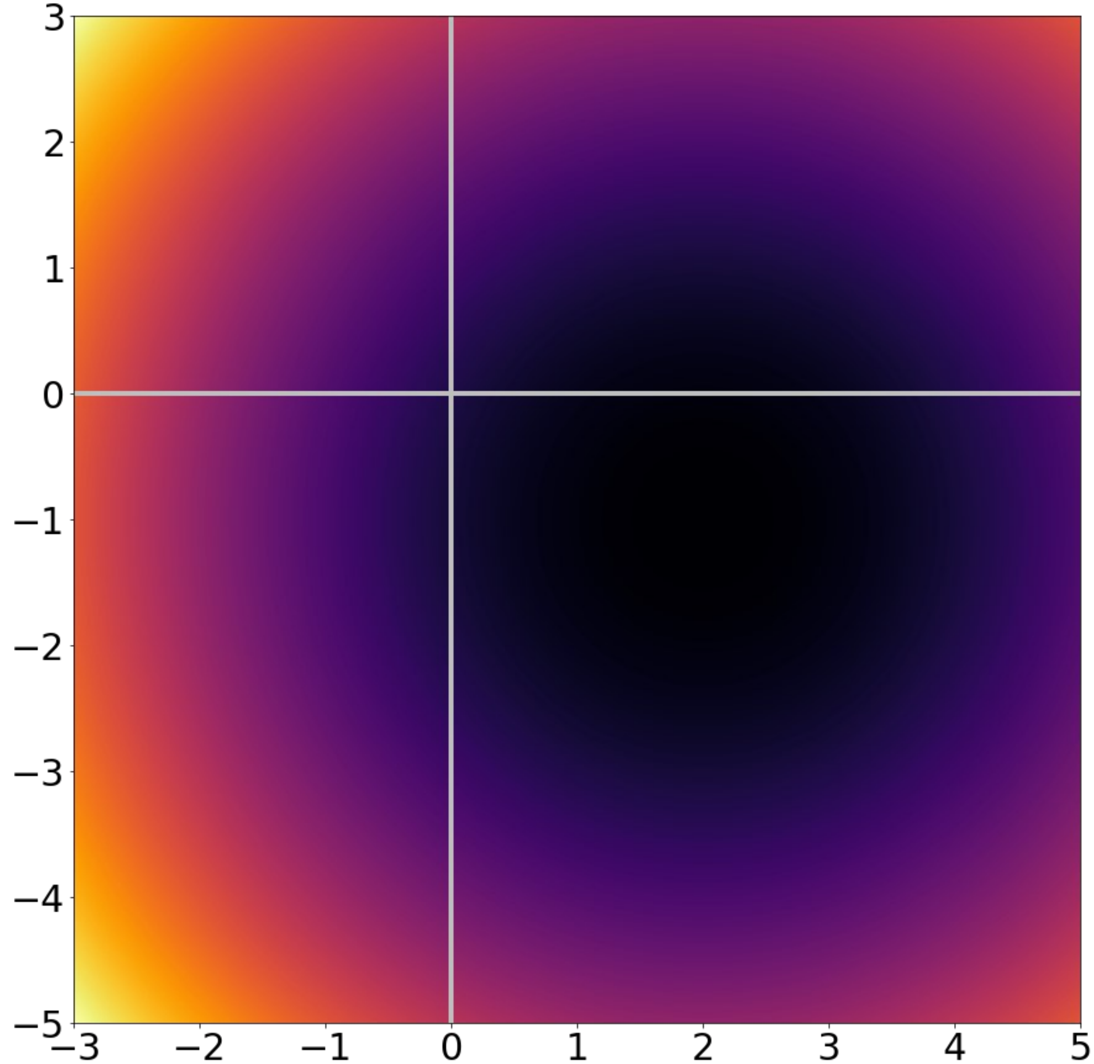
Pretend it's constant → derivative = 0

$$\frac{\partial}{\partial x} f_2(x) = 2(x-2)$$

Zooming Out

$$f_2(x, y) = (x-2)^2 + 5 + (y+1)^2$$

Dark = f(x,y) low
Bright = f(x,y) high



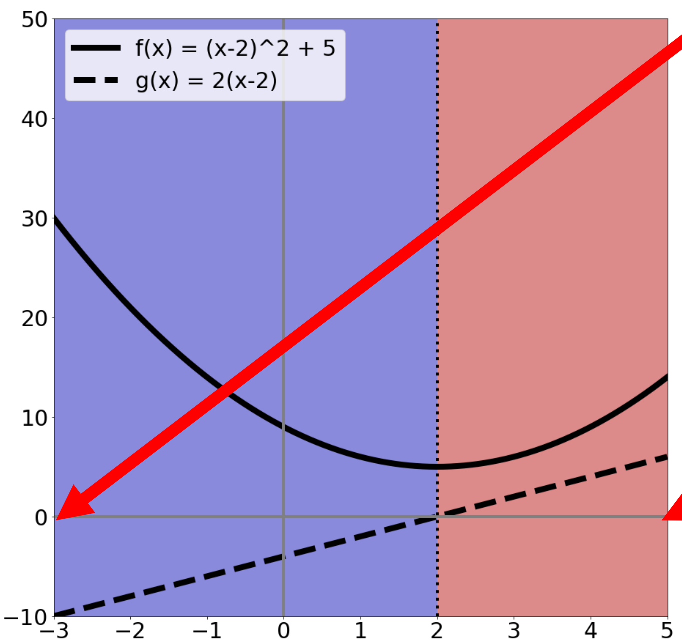
Taking a slice of

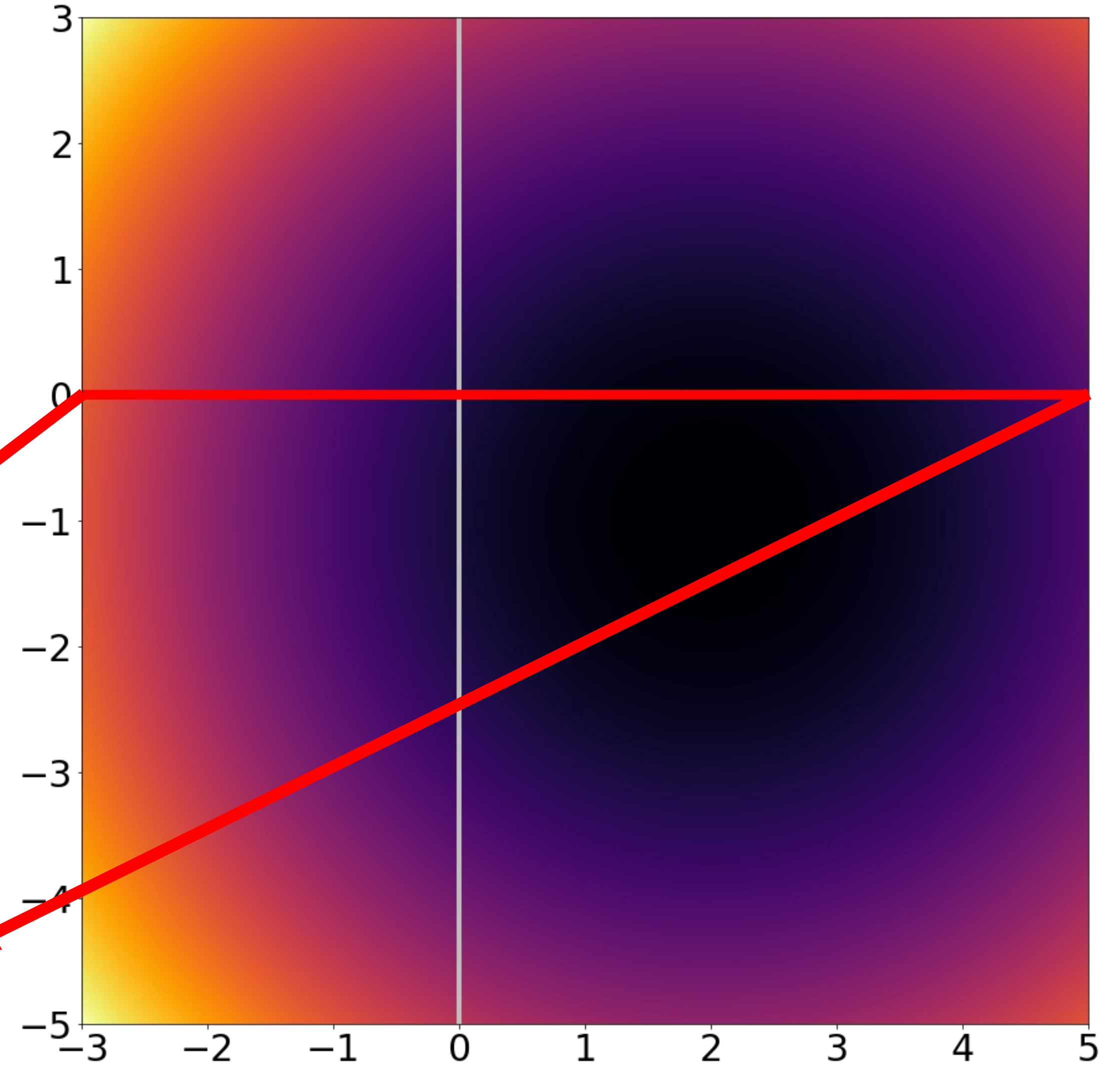
$$f_2(x, y) = (x-2)^2 + 5 + (y+1)^2$$

Slice of y=0 is the function from before:

$$f(x) = (x-2)^2 + 5$$

$$f'(x) = 2(x-2)$$

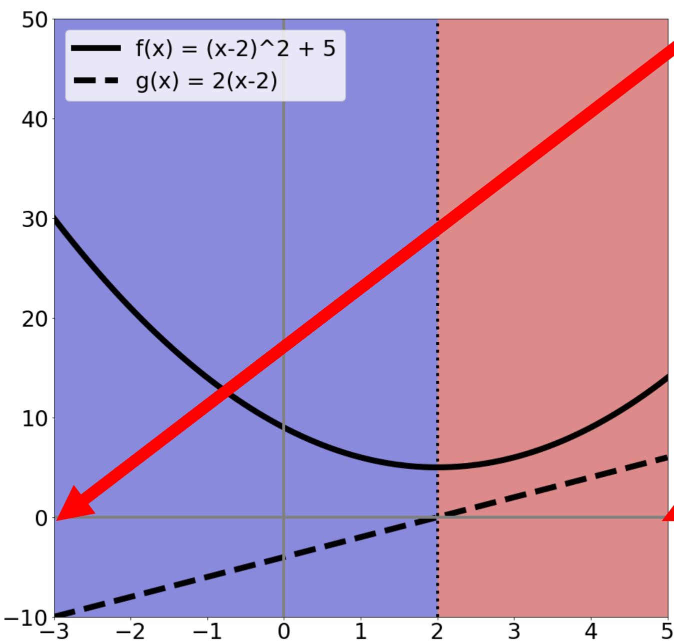


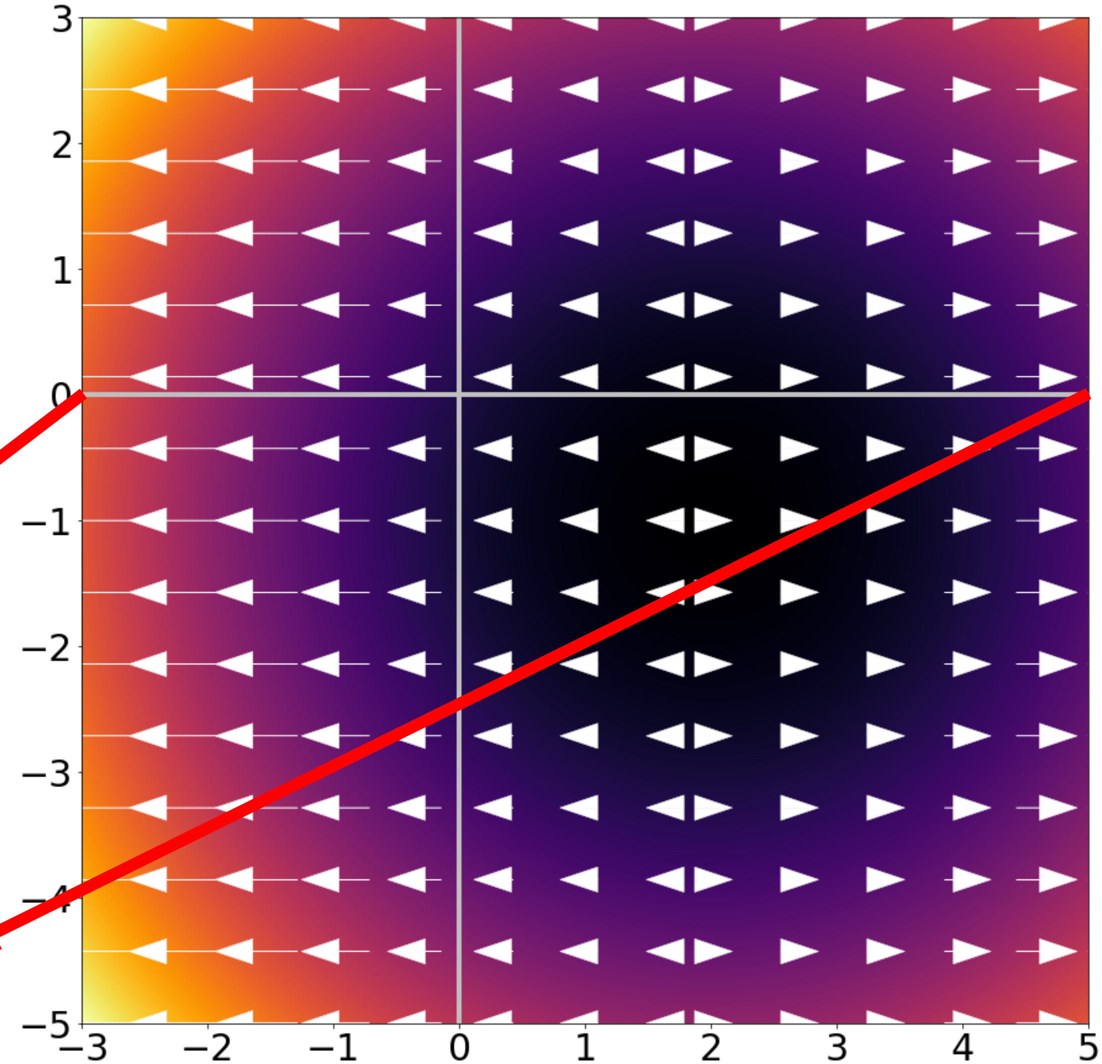


Taking a slice of

$$f_2(x, y) = (x - 2)^2 + 5 + (y + 1)^2$$

$\frac{\partial}{\partial x} f_2(x, y)$ is rate of change & direction in x dimension





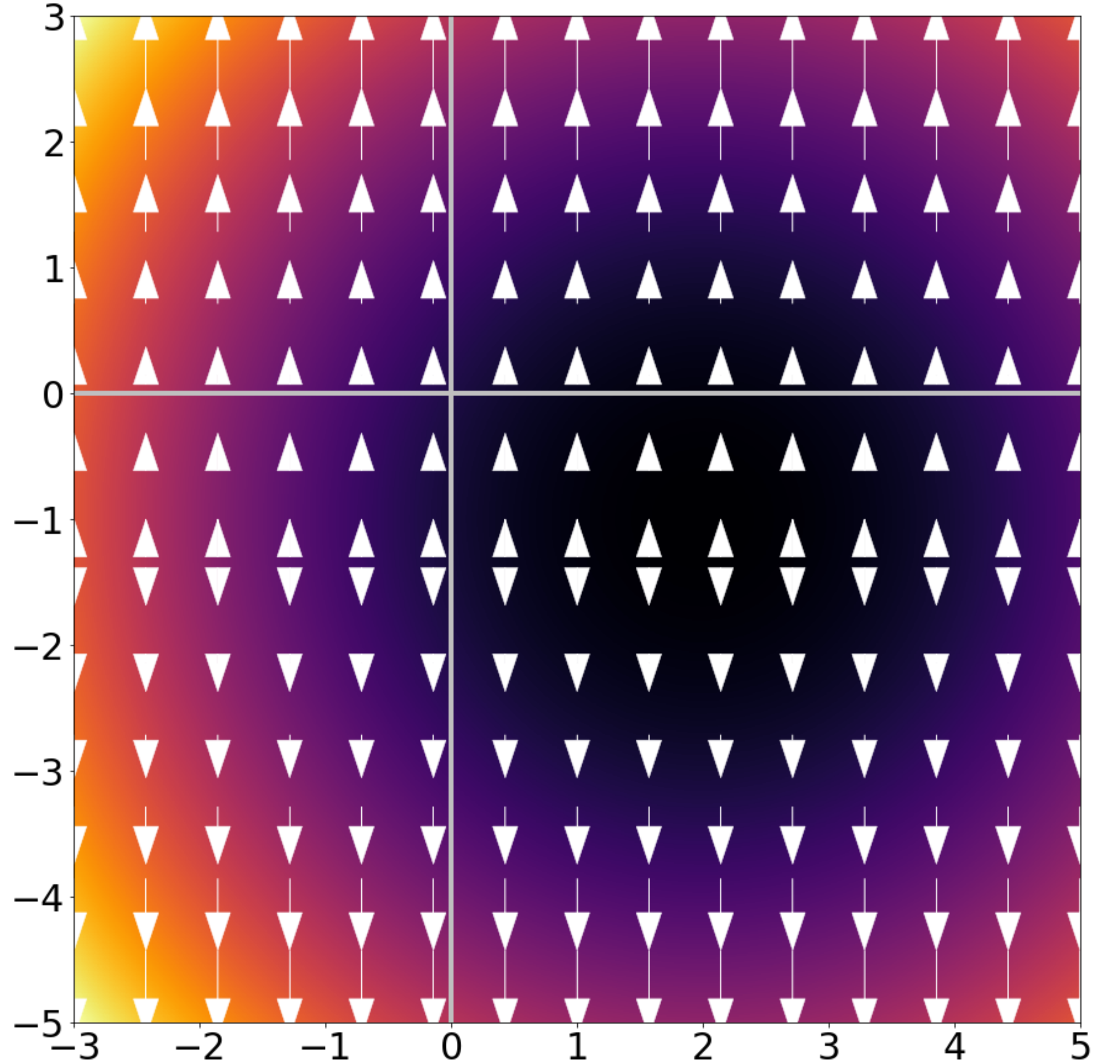
Zooming Out

$$f_2(x, y) = (x-2)^2 + 5 + (y+1)^2$$

$\frac{\partial}{\partial y} f_2(x, y)$ is

$$2(y+1)$$

and is the rate of change & direction in y dimension



Zooming Out

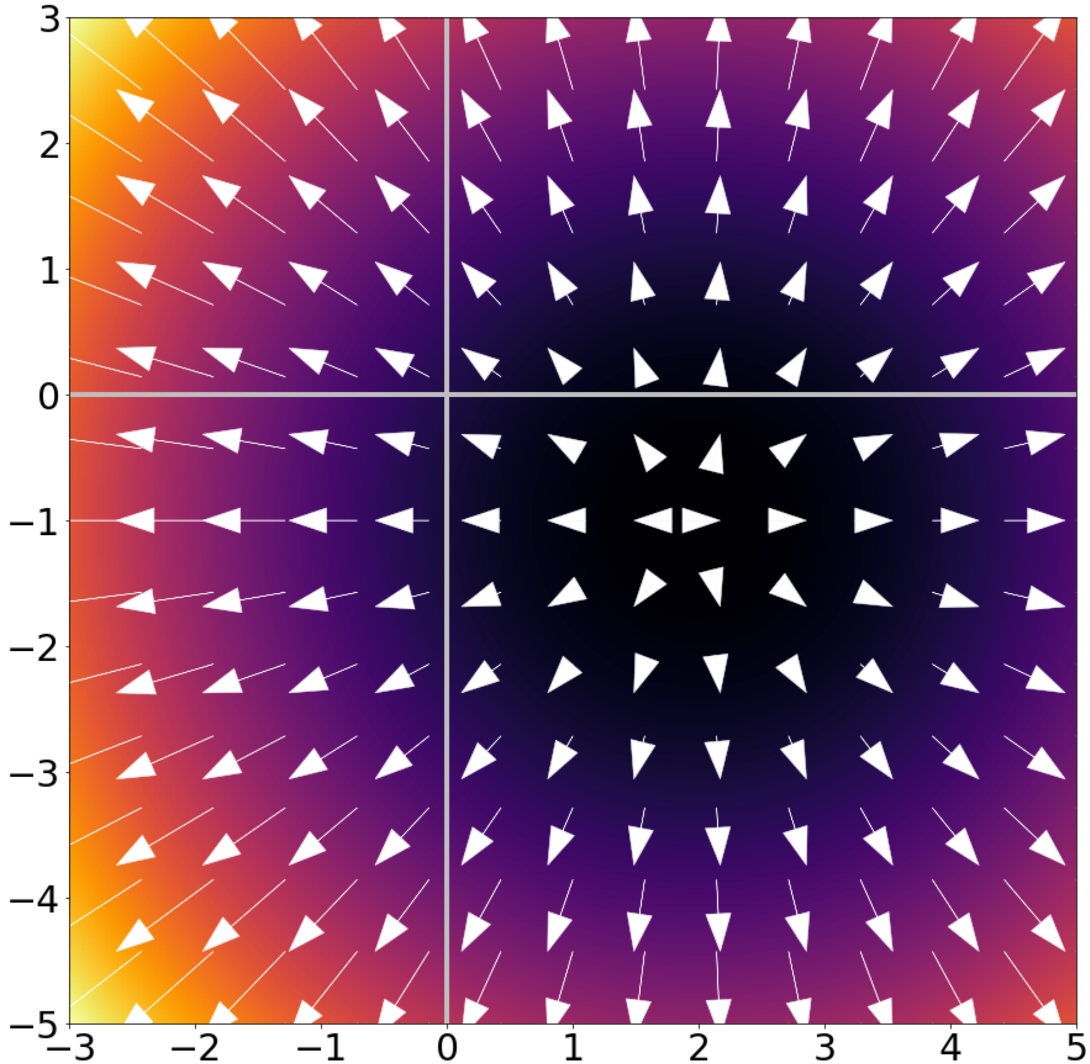
$$f_2(x, y) = (x-2)^2 + 5 + (y+1)^2$$

Gradient/Jacobian: Making a vector of

$$\nabla_f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

gives rate and direction of change.

Arrows point OUT of minimum / basin.



What Should I Know?

- Gradients are simply partial derivatives per-dimension: if $\boldsymbol{x}$ in $f(\boldsymbol{x})$ has n dimensions, $\nabla_f(x)$ has n dimensions
- Gradients point in direction of ascent and tell the rate of ascent
- If a is minimum of $f(\boldsymbol{x}) \rightarrow \nabla_{\mathrm{f}}(\mathrm{a}) = \mathbf{0}$
- Reverse is not true, especially in high-dimensional spaces

For the Curious

- I used to teach floating point stuff. Here’s a condensed explanation
- The tl;dr is that floating points are not real numbers.

What's a Number?

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
1	0	1	1	1	0	0	1	185

$128 + 32 + 16 + 8 + 1 =$ 185

Adding Two Numbers

2^8		2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
		1	0	1	1	1	0	0	1	185
	+	0	1	1	0	1	0	0	1	105
1		0	0	1	0	0	0	1	0	34

Carry Flag

Result

"Integers" on a computer are integers ***modulo*** $\underline{2^k}$

Some Gotchas

32 + (3 / 4) x 40 = 32

32 + (3 x 40) / 4 = 62

Why?

<u>Underflow</u>	<u>No Underflow</u>
32 + 3 / 4 x 40 =	32 + 3 x 40 / 4 =
32 + 0 x 40 =	32 + 120 / 4 =
32 + 0 =	32 + 30 =
32	62

Ok – you have to multiply before dividing

Some Gotchas

Should be:
9x4=36

math 32 + $\boxed{(9 \times 40) / 10}$ = 68

uint8 32 + (9 x 40) / 10 = 42

<u>Overflow</u>

32 + 9 x 40 / 10 =
32 + $\boxed{104}$ / 10 =
32 + 10 =
42

Why 104?

$$\boxed{\begin{array}{l} 9 \times 40 = 360 \\ 360 \% 256 = 104 \end{array}}$$

What’s a Number?

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
1	0	1	1	1	0	0	1	185

How can we do fractions?

2^5	2^4	2^3	2^2	2^1	2^0	•	2^{-1}	2^{-2}	
1	0	1	1	1	0	•	0	1	45.25

45 0.25

Fixed-Point Arithmetic

$2^5 \quad 2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0 \bullet 2^{-1} \quad 2^{-2}$

1 0 1 1 1 0 0 1 45.25

What's the largest number we can represent?

63.75 – **Why?**

How precisely can we measure at 63?

0.25

How precisely can we measure at 0?

0.25

Fine for many purposes but for science, seems silly

Floating Point

Sign (S)	Exponent (E)	Fraction (F)
1	0 1 1 1	0 0 1
1	**7**	**1**
-1	$2^{7-7} = 2^0 = 1$	**1+1/8 = 1.125**

$$\left(-1^{S}\right)\left(2^{E+bias}\right)\left(1+\frac{F}{2^3}\right)$$

Bias: allows exponent to be negative; Note: fraction = significant = mantissa; exponents of all ones or all zeros are special numbers

Floating Point

Sign	Exponent
1	0 1 1 1
-1	7-7=0 *(-bias)*

Fraction

0/8	0 0 0	$-2^0 \times 1.00 = -1$
1/8	0 0 1	$-2^0 \times 1.125 = -1.125$
2/8	0 1 0	$-2^0 \times 1.25 = -1.25$
	...	
6/8	1 1 0	$-2^0 \times 1.75 = -1.75$
7/8	1 1 1	$-2^0 \times 1.875 = -1.875$

Floating Point

Sign	Exponent
1	1 0 0 1
-1	9-7=2 *(-bias)*

Fraction

	Fraction	
0/8	0 0 0	$-2^2 \times 1.00 = -4$
1/8	0 0 1	$-2^2 \times 1.125 = -4.5$
2/8	0 1 0	$-2^2 \times 1.25 = -5$
	…	
6/8	1 1 0	$-2^2 \times 1.75 = -7$
7/8	1 1 1	$-2^2 \times 1.875 = -7.5$

Floating Point

Sign	Exponent	Fraction	
1	0 1 1 1	0 0 0	$-2^0 \times 1.00 = -1$
		0 0 1	$-2^0 \times 1.125 = -1.125$
1	1 0 0 1	0 0 0	$-2^2 \times 1.00 = -4$
		0 0 1	$-2^2 \times 1.125 = -4.5$

Gap between numbers is *relative*, not absolute

Revisiting Adding Numbers

	Sign	Exponent	Fraction	
	1	0 1 1 0	0 0 0	-2^{-1} x 1.00 = -0.5
+	1	1 0 0 1	0 0 0	-2^{2} x 1.00 = -4
	1	1 0 0 1	0 0 1	-2^{2} x 1.125 = -4.5

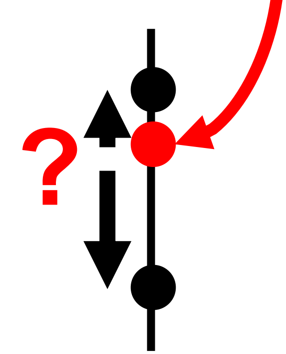
Actual implementation is complex

Revisiting Adding Numbers

	Sign	Exponent	Fraction	
	1	0 1 0 0	0 0 0	$-2^{-3} \times 1.00 = -0.125$
+	1	1 0 0 1	0 0 0	$-2^{2} \times 1.00 = -4$

$-2^{2} \times 1.03125 = -4.125$

1	1 0 0 1	0 0 0	$-2^{2} \times 1.00 = -4$
1	1 0 0 1	0 0 1	$-2^{2} \times 1.125 = -4.5$

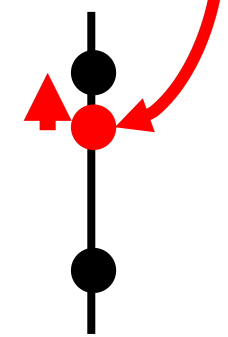


Revisiting Adding Numbers

	Sign	Exponent	Fraction	
	1	0 1 0 0	0 0 0	$-2^{-3} \times 1.00 = -0.125$
+	1	1 0 0 1	0 0 0	$-2^{2} \times 1.00 = -4$

$-2^{2} \times 1.03125 = -4.125$

1	1 0 0 1	0 0 0	$-2^{2} \times 1.00 = -4$



For a and b, these can happen

a + b = a   **a+b-a ≠ b**

Revisiting Adding Numbers

IEEE 754 Single Precision / Single

8 bits

$2^{127} \approx 10^{38}$

23 bits

≈ 7 decimal digits

IEEE 754 Double Precision / Double

11 bits

$2^{1023} \approx 10^{308}$

52 bits

≈ 15 decimal digits

Revisiting Adding Numbers

IEEE 754 Half Precision

5 bits

$2^{16} \approx 10^5$

10 bits

$\approx$ 3 decimal digits

BFloat16 From Google

8 bits

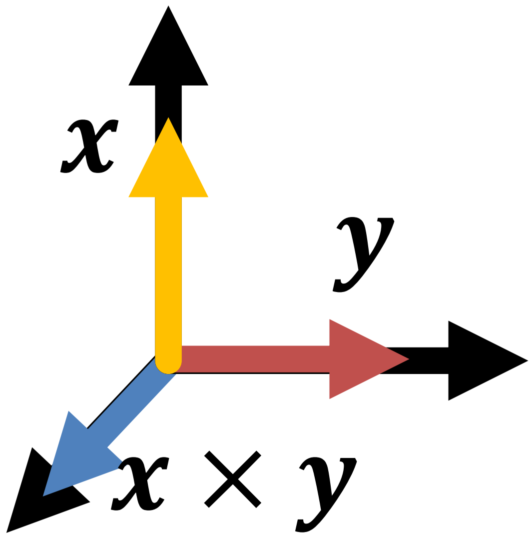
$2^{127} \approx 10^{38}$

7 bits

$\approx$ 2 decimal digits

Past Stuff

Cross Product



- Set $\{z\colon z \cdot x = 0, z \cdot y = 0\}$ has an ambiguity in sign and magnitude
- Cross product $x \times y$ 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 $x \times y = \mathbf{0}$.
- Only in 3D!

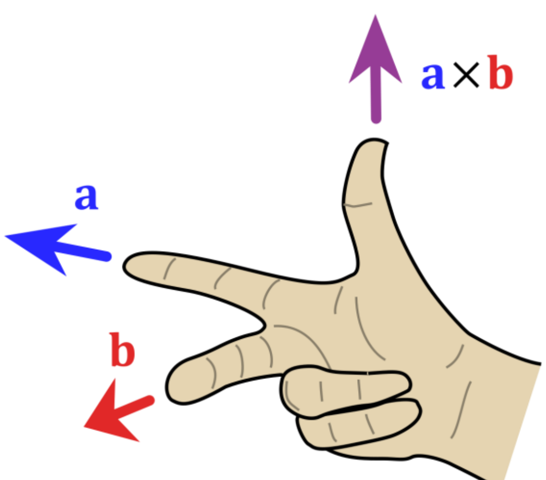
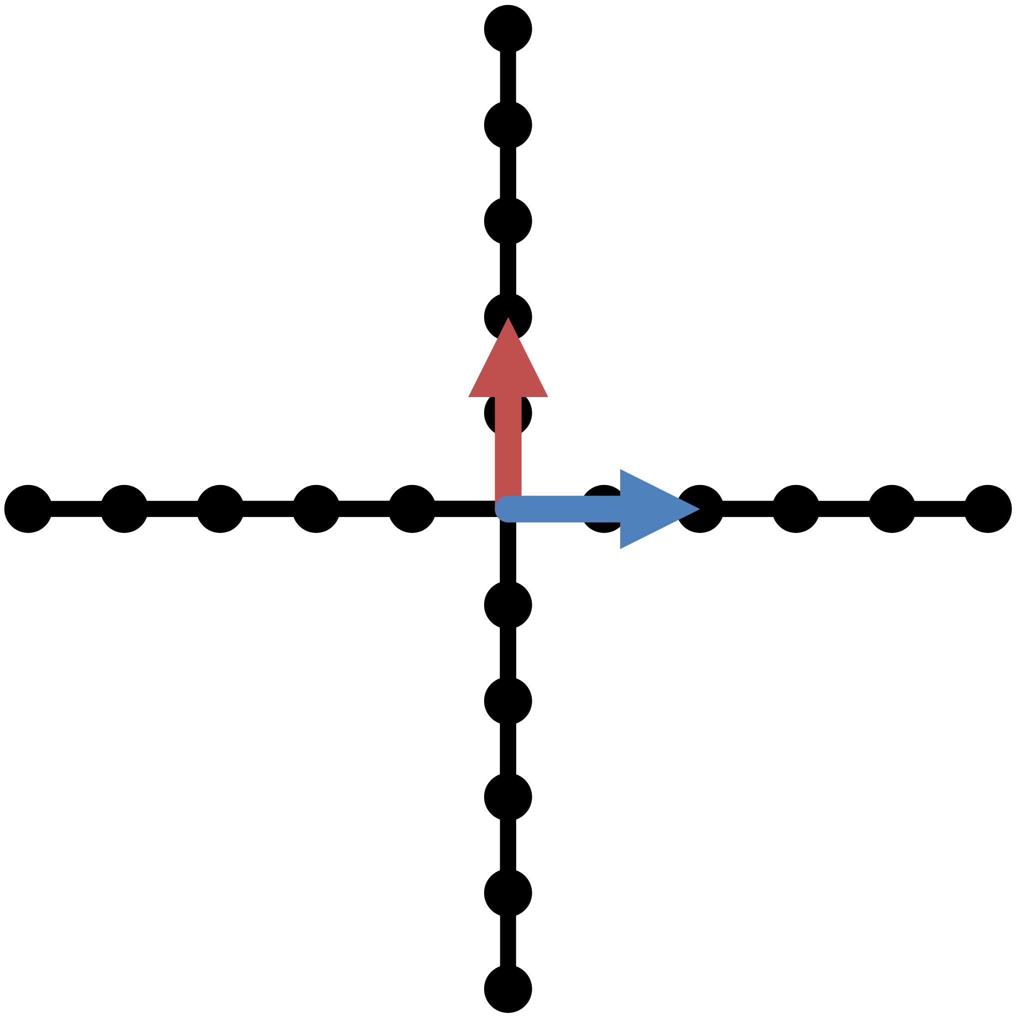


Image credit: Wikipedia.org

Span

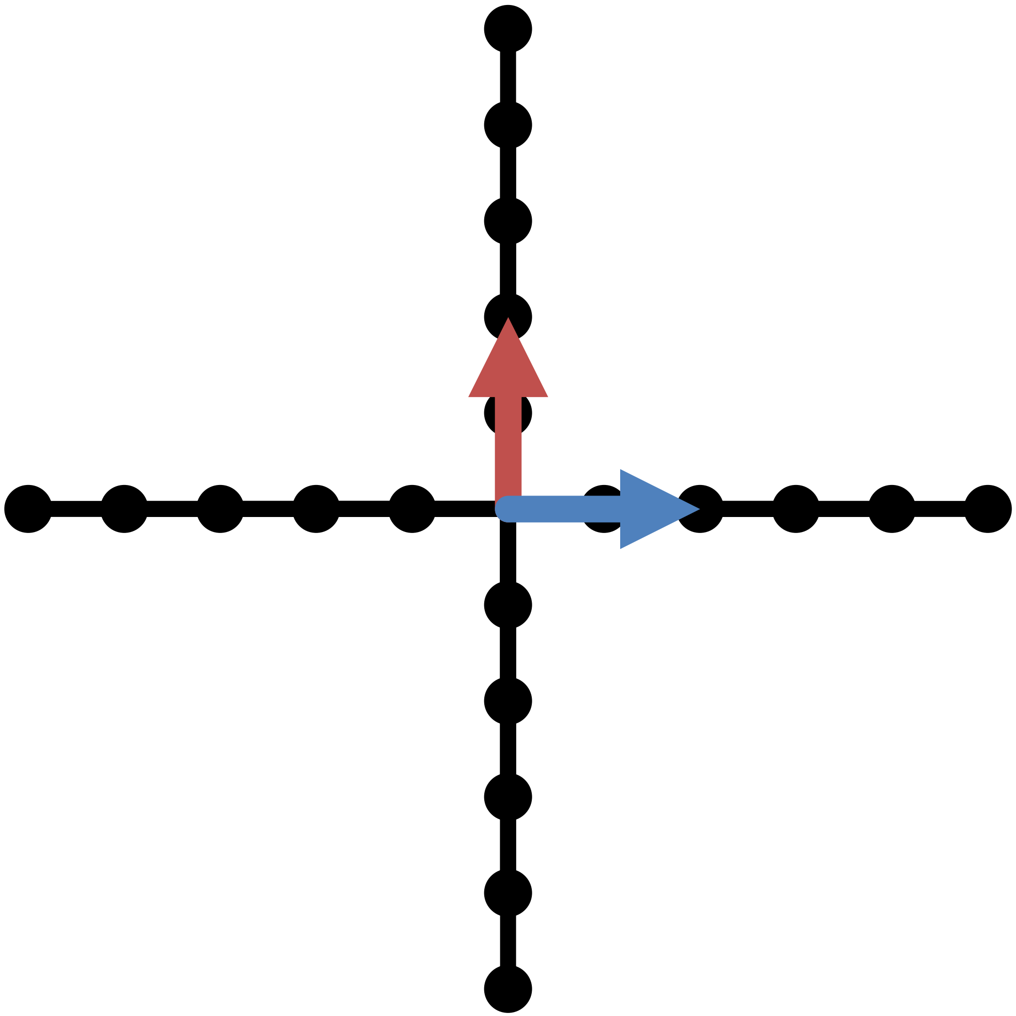


Span: all linear combinations of a set of vectors

Span({  }) =
Span({[0,2]}) = ?
All vertical lines through origin =
$\{\lambda[0,1]: \lambda \in R\}$

Is blue in {red}'s span?

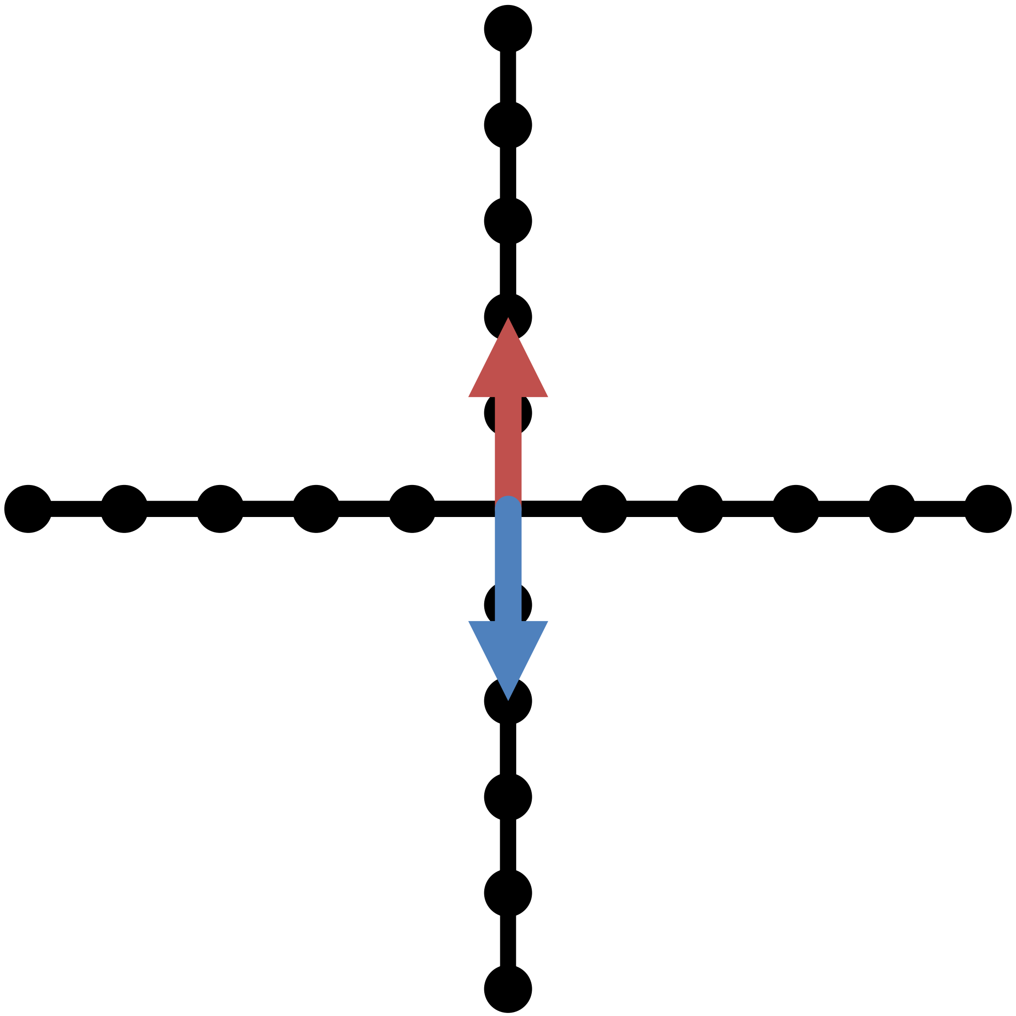
Span



Span: all linear combinations of a set of vectors

Span({↑ , →}) = **?**

Span



Span: all linear combinations of a set of vectors

Span({$\uparrow$, $\downarrow$ }) = **?**

Linear Independence

Recall: $$\boldsymbol{Ax} = (x_1 + \alpha x_2)\boldsymbol{c_1} + x_3\boldsymbol{c_2}$$

$$\boldsymbol{y} = \boldsymbol{A}\begin{bmatrix} x_1 + \beta \\ x_2 - \beta/\alpha \\ x_3 \end{bmatrix} = \left(x_1 + \cancel{\beta} + \alpha x_2 - \cancel{\alpha\frac{\beta}{\alpha}}\right)c_1 + x_3 c_2$$

- Can write **y** an infinite number of ways by adding β to **x$_1$** and subtracting $\frac{\beta}{\alpha}$ from **x$_2$**
- Or, given a vector **y** there's not a unique vector **x** s.t. **y** =**Ax**
- Not all **y** have a corresponding **x** s.t. **y=Ax**

Linear Independence

$$\boldsymbol{Ax} = (x_1 + \alpha x_2)\boldsymbol{c_1} + x_3\boldsymbol{c_2}$$

$$\boldsymbol{y} = \boldsymbol{A}\begin{bmatrix} \beta \\ -\beta/\alpha \\ 0 \end{bmatrix} = \left(\beta - \alpha\frac{\beta}{\alpha}\right)\boldsymbol{c_1} + 0\boldsymbol{c_2}$$

- What else can we cancel out?
- An infinite number of non-zero vectors **x** can map to a zero-vector **y**
- Called the right null-space of A.

Linear Independence

A set of vectors is linearly independent if you can't write one as a linear combination of the others.

Suppose: $\boldsymbol{a} = \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix}$ $\boldsymbol{b} = \begin{bmatrix} 0 \\ 6 \\ 0 \end{bmatrix}$ $\boldsymbol{c} = \begin{bmatrix} 5 \\ 0 \\ 0 \end{bmatrix}$

$$\boldsymbol{x} = \begin{bmatrix} 0 \\ 0 \\ 4 \end{bmatrix} = 2\boldsymbol{a} \qquad \boldsymbol{y} = \begin{bmatrix} 0 \\ -2 \\ 1 \end{bmatrix} = \frac{1}{2}\boldsymbol{a} - \frac{1}{3}\boldsymbol{b}$$

- **Is the set {a,b,c} linearly independent?**
- **Is the set {a,b,x} linearly independent?**
- **Max # of independent 3D vectors?**

Matrix-Vector Product

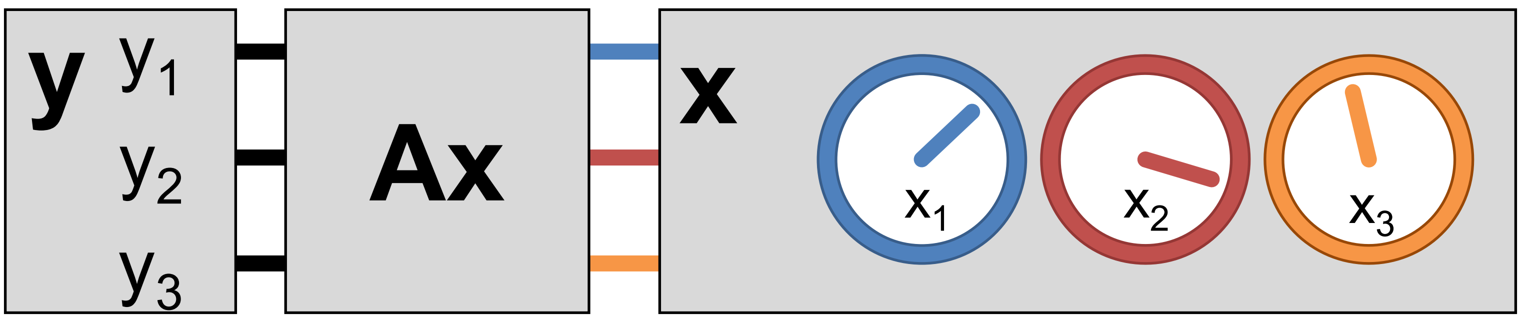
$$\boldsymbol{Ax} = \begin{bmatrix} | & & | \\ \boldsymbol{c_1} & \cdots & \boldsymbol{c_n} \\ | & & | \end{bmatrix} \boldsymbol{x}$$

Right-multiplying **A** by **x** mixes columns of **A** according to entries of **x**

- The output space of f(**x**) = **Ax** is constrained to be the *span* of the columns of **A**.
- Can’t output things you can’t construct out of your columns

An Intuition

$$\boldsymbol{y} = \boldsymbol{Ax} = \begin{bmatrix} | & | & | \\ \boldsymbol{c}_1 & \boldsymbol{c}_2 & \boldsymbol{c}_n \\ | & | & | \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$



x – knobs on machine (e.g., fuel, brakes)

y – state of the world (e.g., where you are)

A – machine (e.g., your car)

Linear Independence

Suppose the columns of 3x3 matrix **A** are *not* linearly independent (c_1, αc_1, c_2 for instance)

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} = \begin{bmatrix} | & | & | \\ \boldsymbol{c_1} & \alpha\boldsymbol{c_1} & \boldsymbol{c_2} \\ | & | & | \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

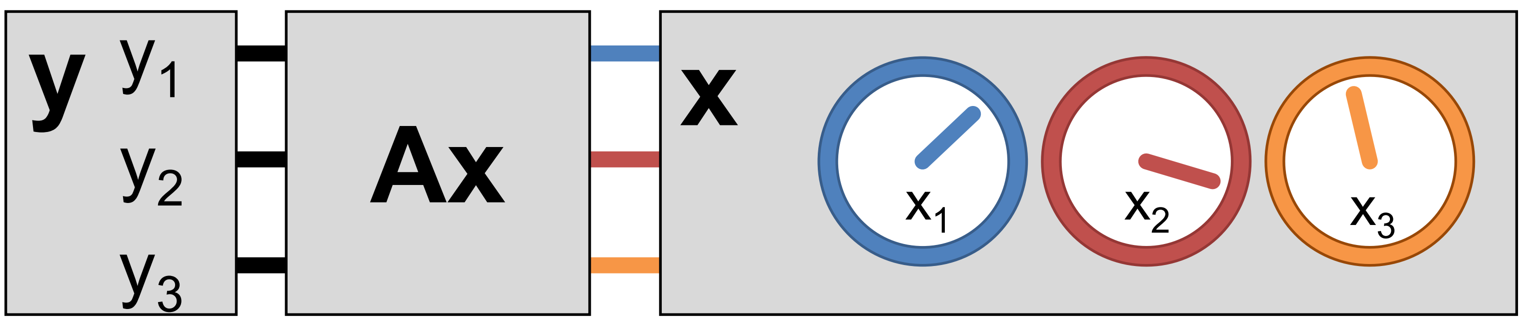
$$\boldsymbol{y} = x_1\boldsymbol{c_1} + \alpha x_2\boldsymbol{c_1} + x_3\boldsymbol{c_2}$$

$$\boldsymbol{y} = (x_1 + \alpha x_2)\boldsymbol{c_1} + x_3\boldsymbol{c_2}$$

Linear Independence Intuition

Knobs of **x** are redundant. Even if **y** has 3 outputs, you can only control it in two directions

$$\boldsymbol{y} = (x_1 + \alpha x_2)\boldsymbol{c_1} + x_3\boldsymbol{c_2}$$



Inverses

- Given $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$, **y** is a linear combination of columns of **A** proportional to **x**. If A is full-rank, we should be able to invert this mapping.
- Given some **y** (output) and **A**, what **x** (inputs) produced it?
- $\mathbf{x} = \mathbf{A}^{-1}\mathbf{y}$
- Note: if you don't need to compute it, **don't compute it.** Solving for **x** is much faster and stable than obtaining $\mathbf{A}^{-1}$.