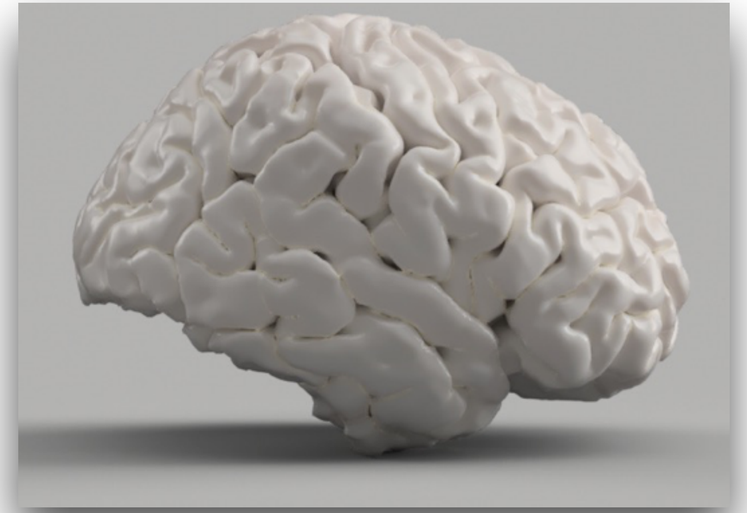
COGS109: Lecture 18



Modeling and Data analysis overview and closing thoughts

July 27, 2023

Modeling and Data Analysis

Summer Session 1, 2023

C. Alex Simpkins Jr., Ph.D.

RDPRobotics LLC | Dept. of CogSci, UCSD

Review

Week #	***Topics***
Week 1	Introduction and definitions, datahub, python, version control/git/github
Week 2	(Python/Jupyter II) Data manipulation and processing
Week 3	Extracting basic information from data and visualizing that info
Week 4	Modeling the data and evaluating models, data fits
Week 5	Presenting and communicating results

What is '***Data***?'

- "Facts and statistics collected together for reference or analysis" (Webster's Dictionary)
- Can be recorded from any form, situation, or field of study
- "Anything that exists, exists in some quantity, anything that exists in some quantity can be measured…" (Thorndike)

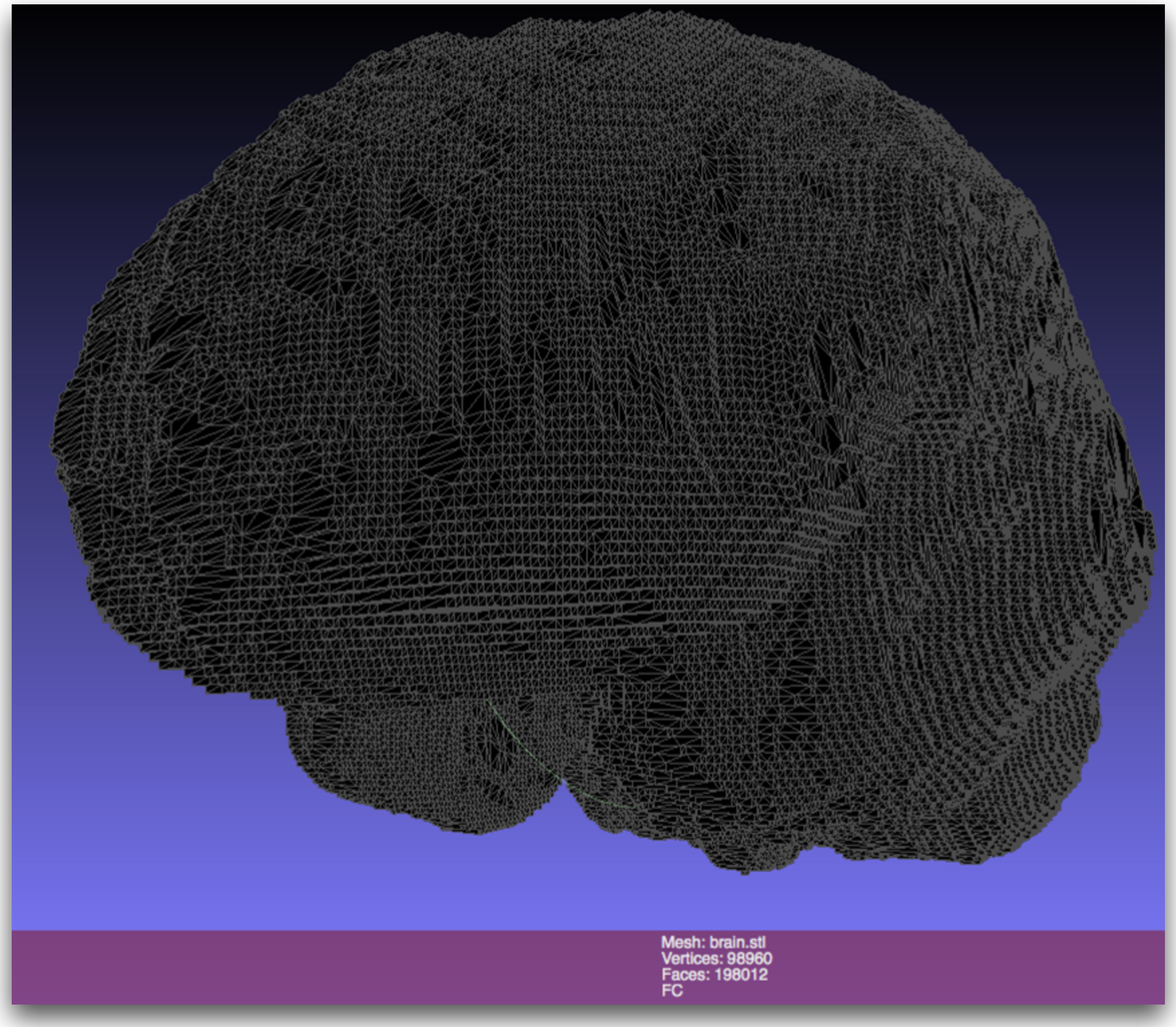
What is '***Data***?'

- "Facts and statistics collected together for reference or analysis" (Webster's Dictionary)
- Can be recorded from any form, situation, or field of study
- "Anything that exists, exists in some quantity, anything that exists in some quantity can be measured…" (Thorndike)

How we'll approach learning about *and doing* Modeling and Data Analysis in COGS 109

Types of data files (low level format)

- But how do we encode files in 1’s and 0’s?
- Files can typically be classified into two different formats
  - ASCII (“Text”)
  - Binary
- STL example
  - Brain.STL (ASCII - 52MB)
  - Brain.STL (BINARY - 9MB)



Decimal - Binary - Octal - Hex – ASCII Conversion Chart

Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII
0	00000000	000	00	NUL	32	00100000	040	20	SP	64	01000000	100	40	@	96	01100000	140	60	`
1	00000001	001	01	SOH	33	00100001	041	21	!	65	01000001	101	41	A	97	01100001	141	61	a
2	00000010	002	02	STX	34	00100010	042	22	“	66	01000010	102	42	B	98	01100010	142	62	b
3	00000011	003	03	ETX	35	00100011	043	23	#	67	01000011	103	43	C	99	01100011	143	63	c
4	00000100	004	04	EOT	36	00100100	044	24	$	68	01000100	104	44	D	100	01100100	144	64	d
5	00000101	005	05	ENQ	37	00100101	045	25	%	69	01000101	105	45	E	101	01100101	145	65	e
6	00000110	006	06	ACK	38	00100110	046	26	&	70	01000110	106	46	F	102	01100110	146	66	f
7	00000111	007	07	BEL	39	00100111	047	27	‘	71	01000111	107	47	G	103	01100111	147	67	g
8	00001000	010	08	BS	40	00101000	050	28	(	72	01001000	110	48	H	104	01101000	150	68	h
9	00001001	011	09	HT	41	00101001	051	29	)	73	01001001	111	49	I	105	01101001	151	69	i
10	00001010	012	0A	LF	42	00101010	052	2A	*	74	01001010	112	4A	J	106	01101010	152	6A	j
11	00001011	013	0B	VT	43	00101011	053	2B	+	75	01001011	113	4B	K	107	01101011	153	6B	k
12	00001100	014	0C	FF	44	00101100	054	2C	,	76	01001100	114	4C	L	108	01101100	154	6C	l
13	00001101	015	0D	CR	45	00101101	055	2D	-	77	01001101	115	4D	M	109	01101101	155	6D	m
14	00001110	016	0E	SO	46	00101110	056	2E	.	78	01001110	116	4E	N	110	01101110	156	6E	n
15	00001111	017	0F	SI	47	00101111	057	2F	/	79	01001111	117	4F	O	111	01101111	157	6F	o
16	00010000	020	10	DLE	48	00110000	060	30	0	80	01010000	120	50	P	112	01110000	160	70	p
17	00010001	021	11	DC1	49	00110001	061	31	1	81	01010001	121	51	Q	113	01110001	161	71	q
18	00010010	022	12	DC2	50	00110010	062	32	2	82	01010010	122	52	R	114	01110010	162	72	r
19	00010011	023	13	DC3	51	00110011	063	33	3	83	01010011	123	53	S	115	01110011	163	73	s
20	00010100	024	14	DC4	52	00110100	064	34	4	84	01010100	124	54	T	116	01110100	164	74	t
21	00010101	025	15	NAK	53	00110101	065	35	5	85	01010101	125	55	U	117	01110101	165	75	u
22	00010110	026	16	SYN	54	00110110	066	36	6	86	01010110	126	56	V	118	01110110	166	76	v
23	00010111	027	17	ETB	55	00110111	067	37	7	87	01010111	127	57	W	119	01110111	167	77	w
24	00011000	030	18	CAN	56	00111000	070	38	8	88	01011000	130	58	X	120	01111000	170	78	x
25	00011001	031	19	EM	57	00111001	071	39	9	89	01011001	131	59	Y	121	01111001	171	79	y
26	00011010	032	1A	SUB	58	00111010	072	3A	:	90	01011010	132	5A	Z	122	01111010	172	7A	z
27	00011011	033	1B	ESC	59	00111011	073	3B	;	91	01011011	133	5B	[	123	01111011	173	7B	{
28	00011100	034	1C	FS	60	00111100	074	3C	<	92	01011100	134	5C	\	124	01111100	174	7C	|
29	00011101	035	1D	GS	61	00111101	075	3D	=	93	01011101	135	5D	]	125	01111101	175	7D	}
30	00011110	036	1E	RS	62	00111110	076	3E	>	94	01011110	136	5E	^	126	01111110	176	7E	~
31	00011111	037	1F	US	63	00111111	077	3F	?	95	01011111	137	5F	_	127	01111111	177	7F	DEL

source: https://bournetocode.com/projects/GCSE_Computing_Fundamentals/pages/3-3-5-ascii.html

ASCII Files

- American Standard Code for Information Interchange
- Any word processor, straight text
  - M-files are ASCII text files, so any word processor can create them,
    - As long as you are saving as ASCII text
    - Word has its own format, but can create ASCII text files
- Python, Matlab, C, JAVA, etc can load and save specialized data files and standard text files (look for the extension on the end of the file name, ie “demo.m,” “data.dat,” “data.txt”)
- Often you will be dealing with data, either in survey format, or in files which come from data acquisition systems (stored in text or binary files)

Binary files and .mat files

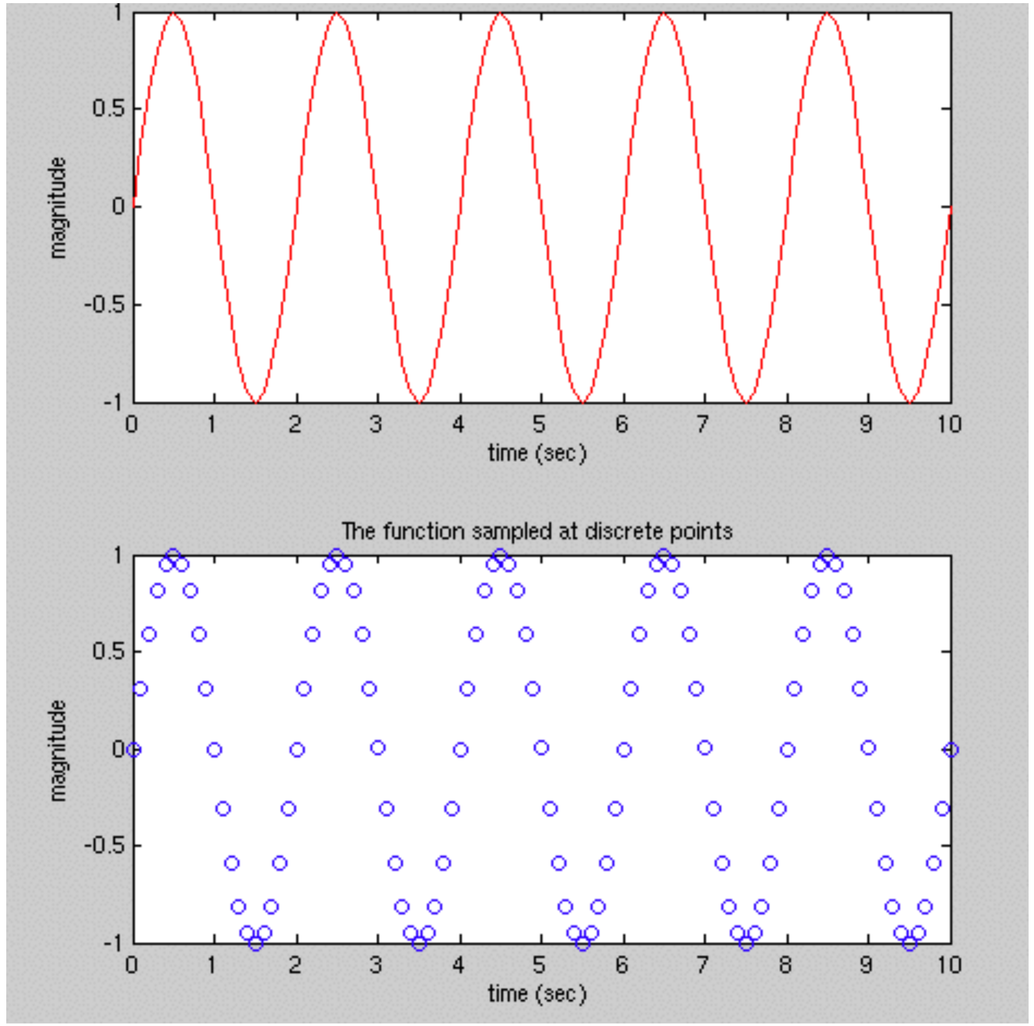
- A more efficient way to store files is binary format
  - Smaller
  - But…Less platform independent - ie need to know exactly what the format is to read the file
  - Matlab stores a binary format with the extension .mat
  - Can’t load these files into just any text editor like you can with ASCII

Sampling, discretization, filtering

- Review of continuous vs. discrete quantities
- Analog vs. Digital
- Discretization, sampling, aliasing
- Filter theory, frequency response, filter types
- Linearity

Continuous vs. Discrete quantities

- Information storage
  - **Continuous** signals have information at every point in time
  - **Discrete** signals have info only at specified intervals (fixed or variable)



Analog vs. Digital quantities

- Information storage
  - **Analog** contains infinite information
  - **Digital** contains limited information, depending on the number of bits of information the digital value can store
    - 0 or 1 in each bit means each bit multiplies the possible combinations of numbers by 2
    - 2^4 = 0-15 (a 4-bit number, 16 different values)
    - 2^8 = 0-255 (an 8-bit number, 256 different values)
    - 2^16 = 0-65535 (a 16-bit number, 65536 different values)

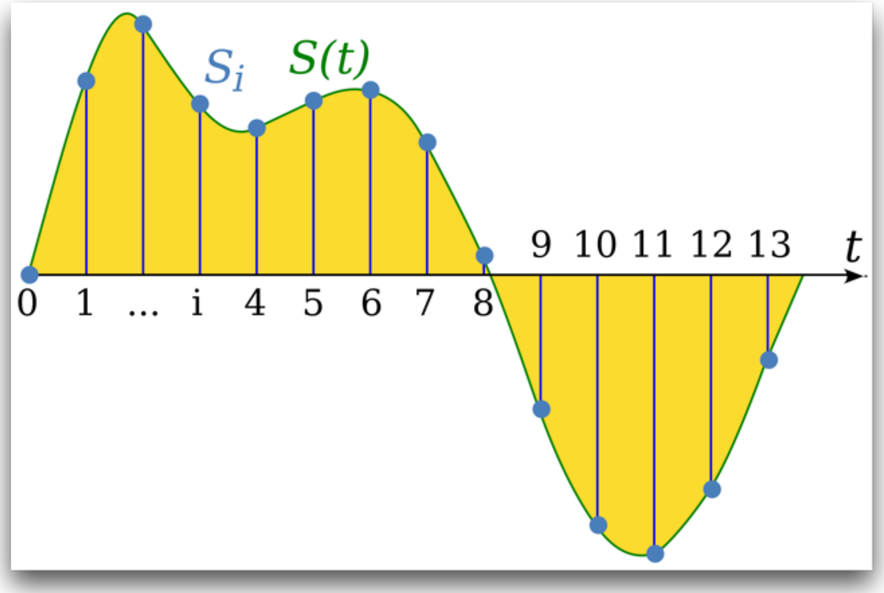
Discretization

- Measuring a continuous (analog) signal means capturing information at specified (fixed or variable) intervals
  - **Sampling frequency -** the frequency at which data is recorded from a signal (Typically in Hz, ie 5kHz)
- When capturing data, or when manipulating data which has been discretized, there are several issues to consider
  - Aliasing (not the TV show:)
  - Sampling rates
  - Post-processing – filtering data to remove unwanted information while retaining desired information

Sampling

- **Sample -** We record data at specific points in time
- **Period -** The time between samples, ***T*** **[sec]**
- **Sample frequency -** The frequency of sampling, *f* [Hz]

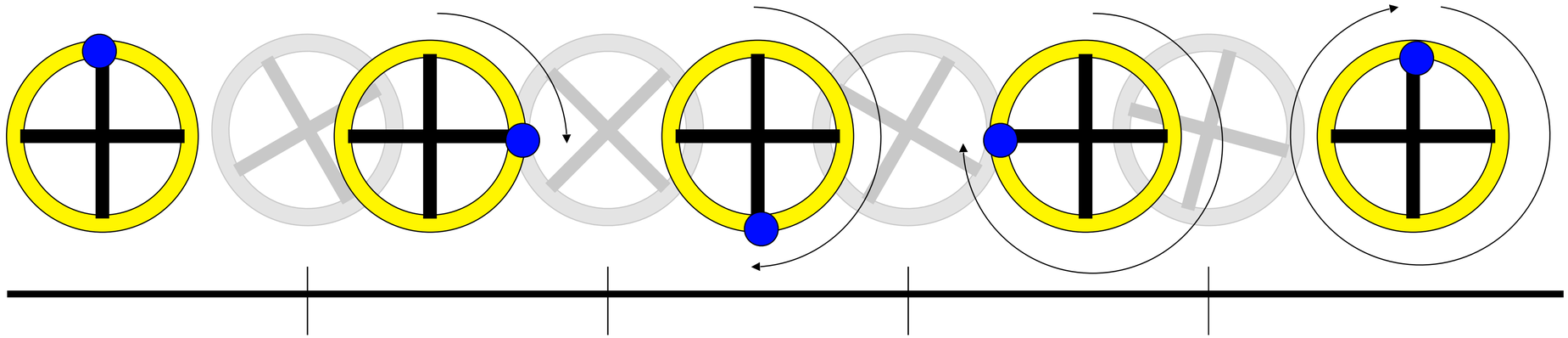
$$f = \frac{1}{\Delta T}$$



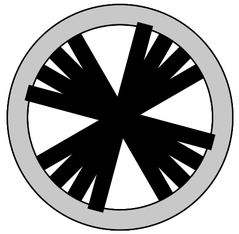
https://en.wikipedia.org/wiki/Sampling_%28signal_processing%29

Example: Sampling and Aliasing

- The wheel spokes example…<Live demo>

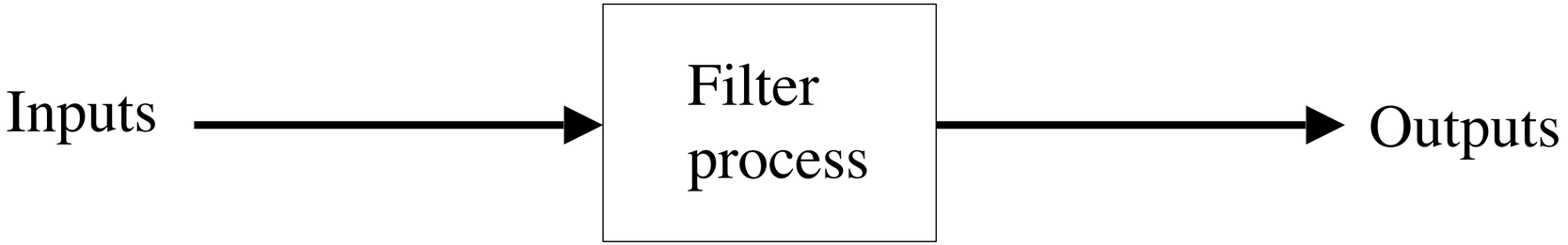


- We're sampling at too slow a rate to accurately see the spokes rotate, and at a particular rotational velocity of the wheel, we see an 'aliased' reverse rotation!



Thus we filter our data…

- **Filter -** an operation or process which alters input data according to some mathematical relationship or heuristic rule to produce output data which is more desirable

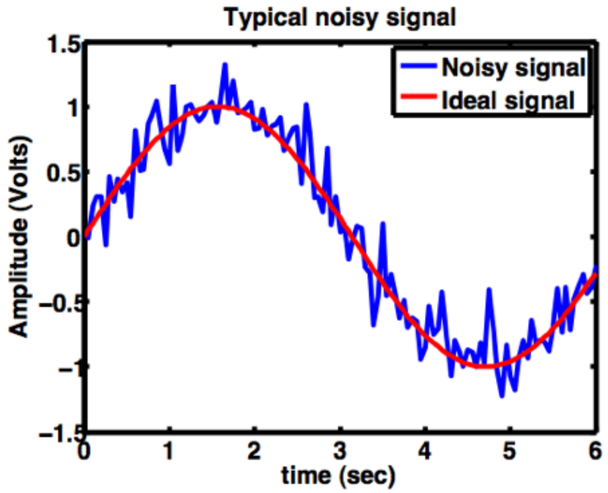


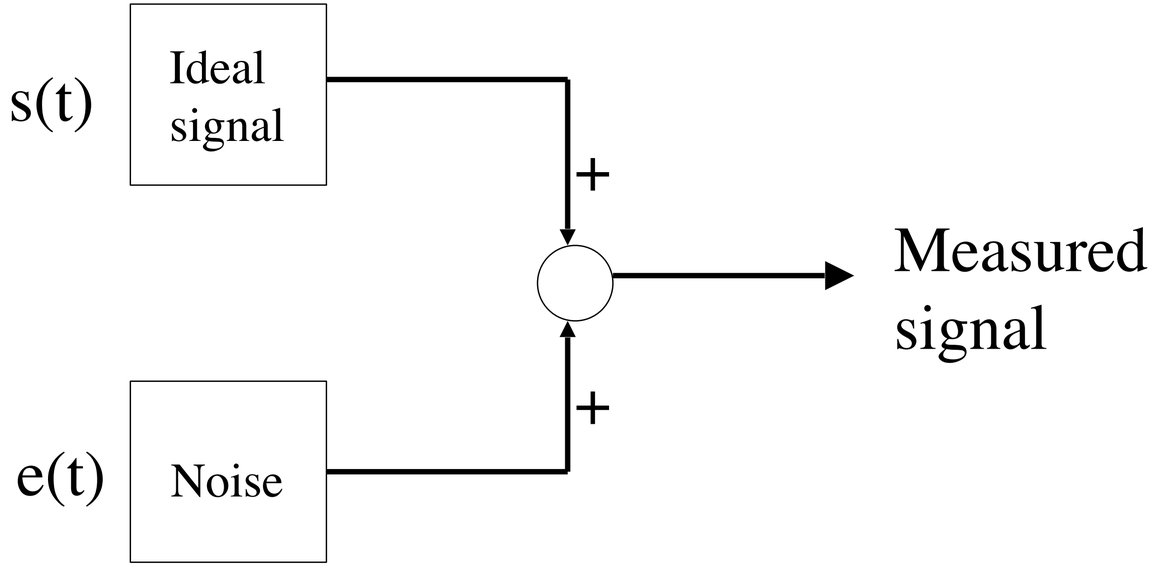
Common filter types in signal processing

- **Low-pass filter** - (ideal) attenuates high frequency data, while allowing low frequency data to pass unchanged
- **High-pass filter** - (ideal) attenuates low frequency data, while allowing high frequency data to pass unchanged
- **Band-pass filter** - (ideal) attenuates all frequencies except a particular frequency band (or bands)
- **Band-stop filter** - (ideal) attenuates one or a selection of frequency ranges of data, allowing all the rest to pass unchanged
- Actual filters are not exactly ideal…which we will discuss

Signals and noise…

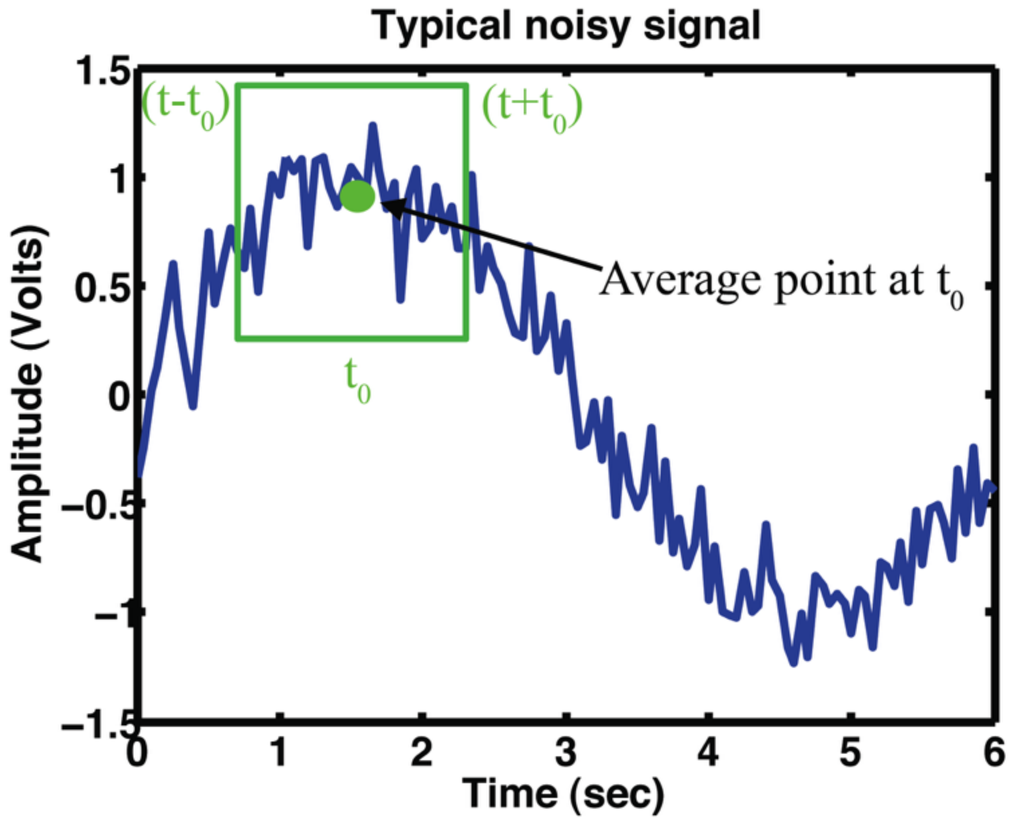
- By making assumptions about the properties of the unwanted ‘noise’ *e(t),* we can reconstruct an appropriate *estimate* of the original signal *s(t)*
  - **Noise - any unwanted portion of a signal, lumped together. It may come from multiple sources but tends toward some statistically predictable properties**





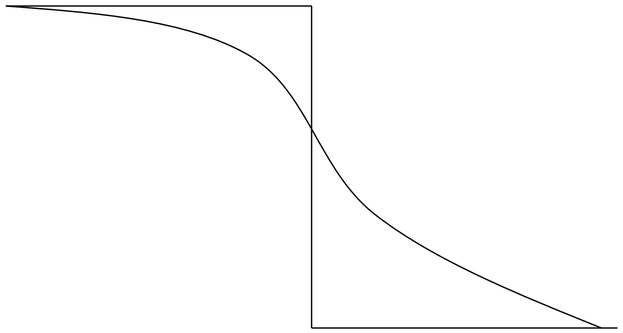
Low-pass filtering

- So the effect is this



Disadvantages…

- Need to have all data in memory already, so it isn't an 'online' filter
- Causality
  - **If we care about an exact event timing, this is a poor filter to use:**



Signal anticipates changes!

Solution

- Recursive filter
  - Solves causality issue
  - Easy to implement as we saw last time

Tidy data == rectangular data

A

	A	B	C	D	E
1	id	sex	glucose	insulin	triglyc
2	101	Male	134.1	0.60	273.4
3	102	Female	120.0	1.18	243.6
4	103	Male	124.8	1.23	297.6
5	104	Male	83.1	1.16	142.4
6	105	Male	105.2	0.73	215.7

Broman KW, Woo KH. (2017) Data organization in spreadsheets. *PeerJ Preprints* 5:e3183v1 https://doi.org/10.7287/peerj.preprints.3183v1

Data wrangling vs. data cleaning

- Data wrangling focuses on transforming the data from a ‘raw’ format into a format suitable for computational use
- Data cleaning focuses on, as discussed, fixing/removing incorrect, corrupted, incorrectly formatted, duplicate, incomplete, data within a dataset

Data wrangling vs. data cleaning

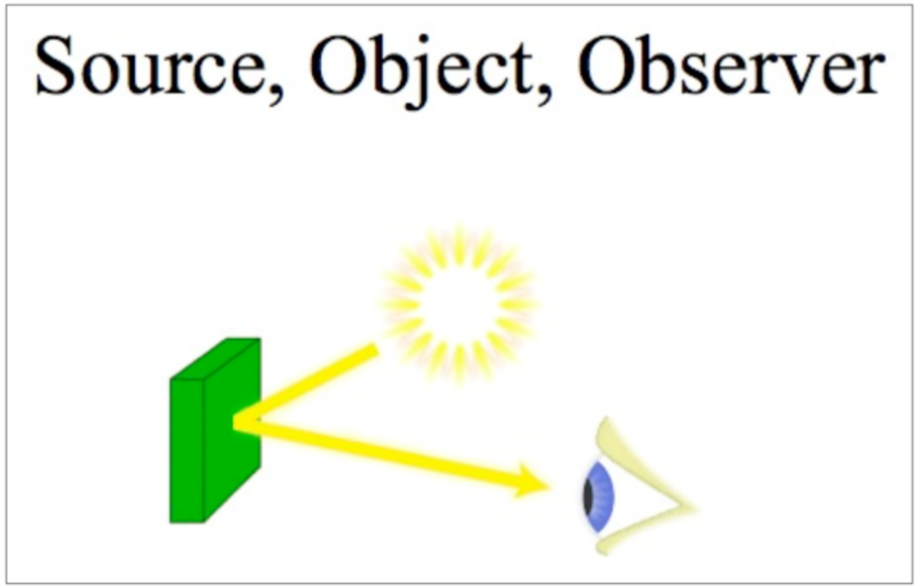
- Data wrangling focuses on transforming the data from a ‘raw’ format into a format suitable for computational use
- Data cleaning focuses on, as discussed, fixing/removing incorrect, corrupted, incorrectly formatted, duplicate, incomplete, data within a dataset

Visualization

- Human brain has trouble making sense of large amounts of data produced by computational modeling and experimentation
- As more computational methods are applied, more and more information is being created
- Scientific visualization is one way of making important information explicit and simple to process
- http://svs.gsfc.nasa.gov/

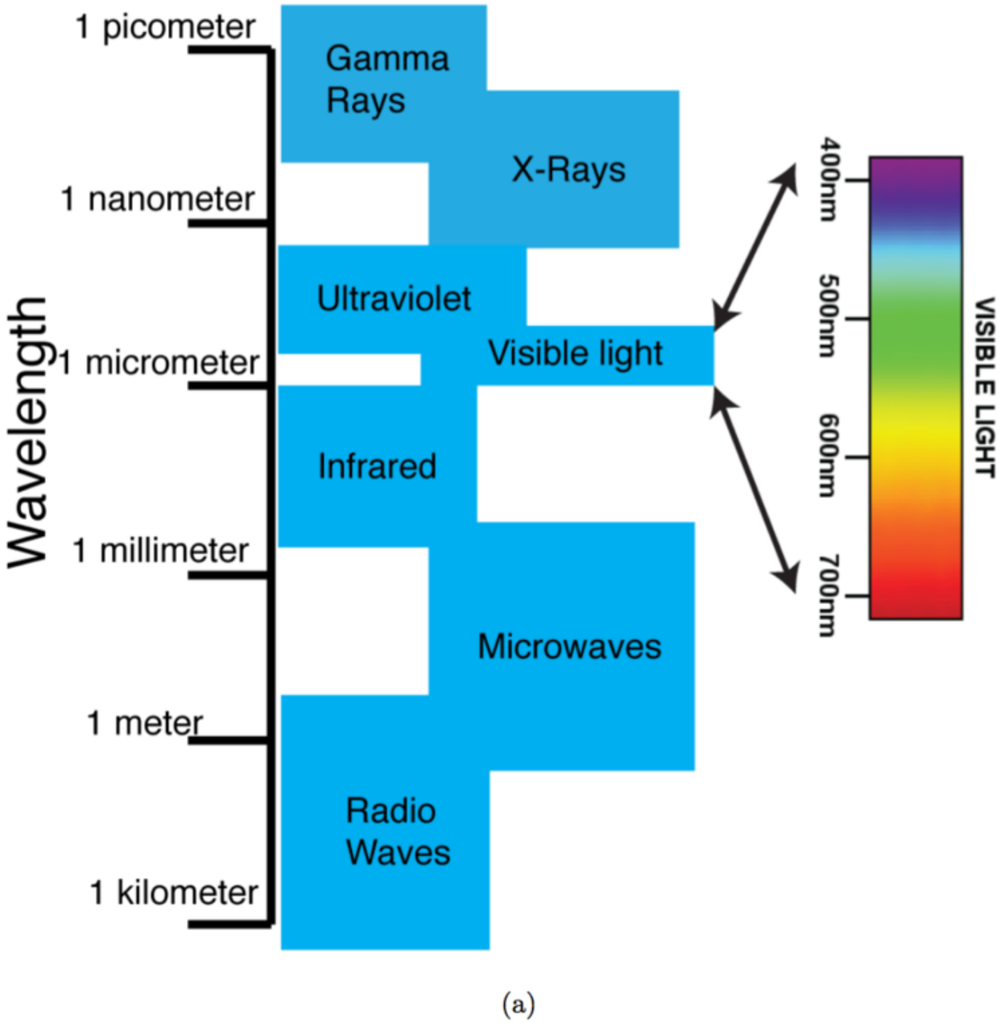
What is color?

- Reflected light = color of object
- Color is the set of wavelengths of light reflected from an object
- A light source can be a light bulb, the sun, etc or another object



Electromagnetic Spectrum

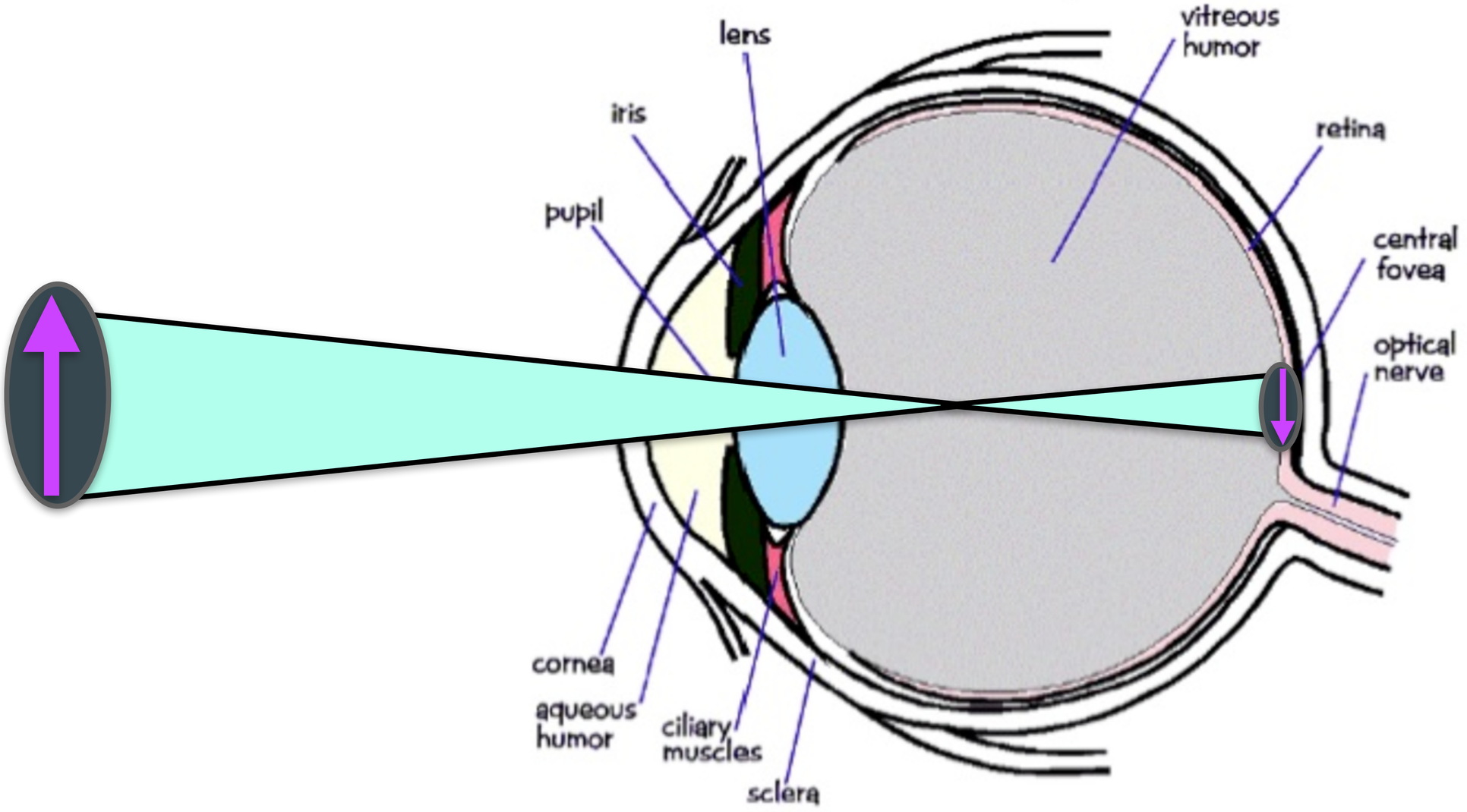
- Visual light is a tiny part
- How can we visualize these quantities in a perceptually useful way?



(a)

Figure 5.2: Visible light (i.e. the portion of the electromagnetic spectrum that human beings can perceive using their eyes) is a very small subset of the entire electromagnetic spectrum, as the reader can see here.

The Eye


lens
vitreous humor
iris
retina
pupil
central fovea
optical nerve
cornea
aqueous humor
ciliary muscles
sclera

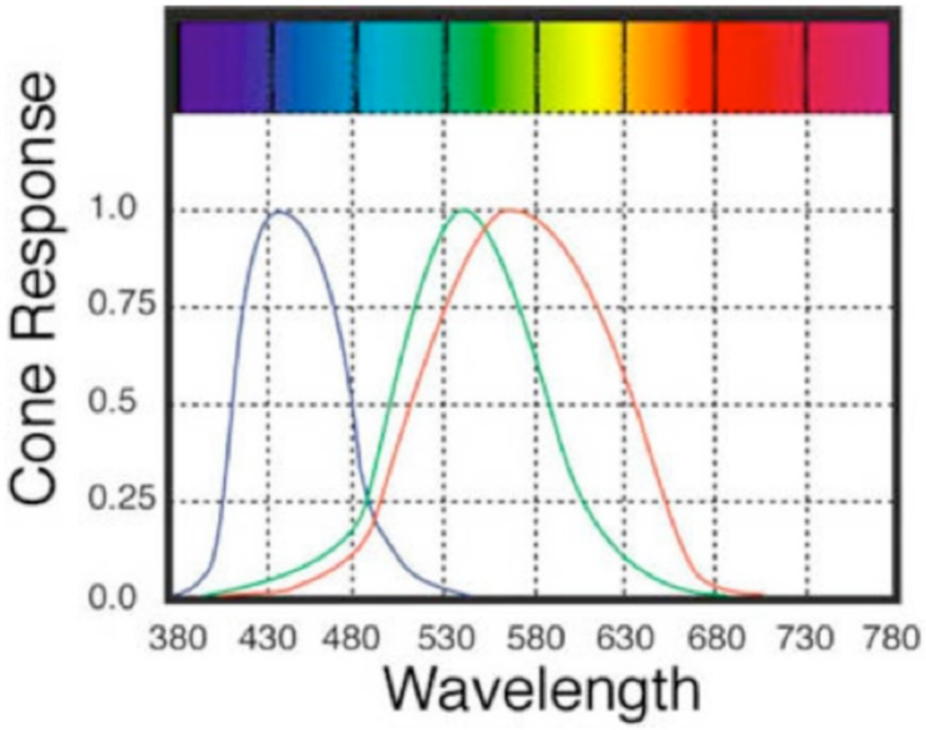
Human perception of color

- **Color constancy** - our visual perception is constantly adjusting to compensate for changing surroundings
- Human color perception is ***context dependent***
  - Ever try to perceive the difference between two colors of clothing in low light?
  - Movie example - Abyss Yellow/green light source, “Cut the blue wire with the white stripe, NOT the black wire with the yellow strip”
  - Side note- how to fix this as the designer of the device?
    - Use one wire with dashes instead of a stripe - “Cut the wire with the dashes.” Person cutting: “Easy. It’s done!”

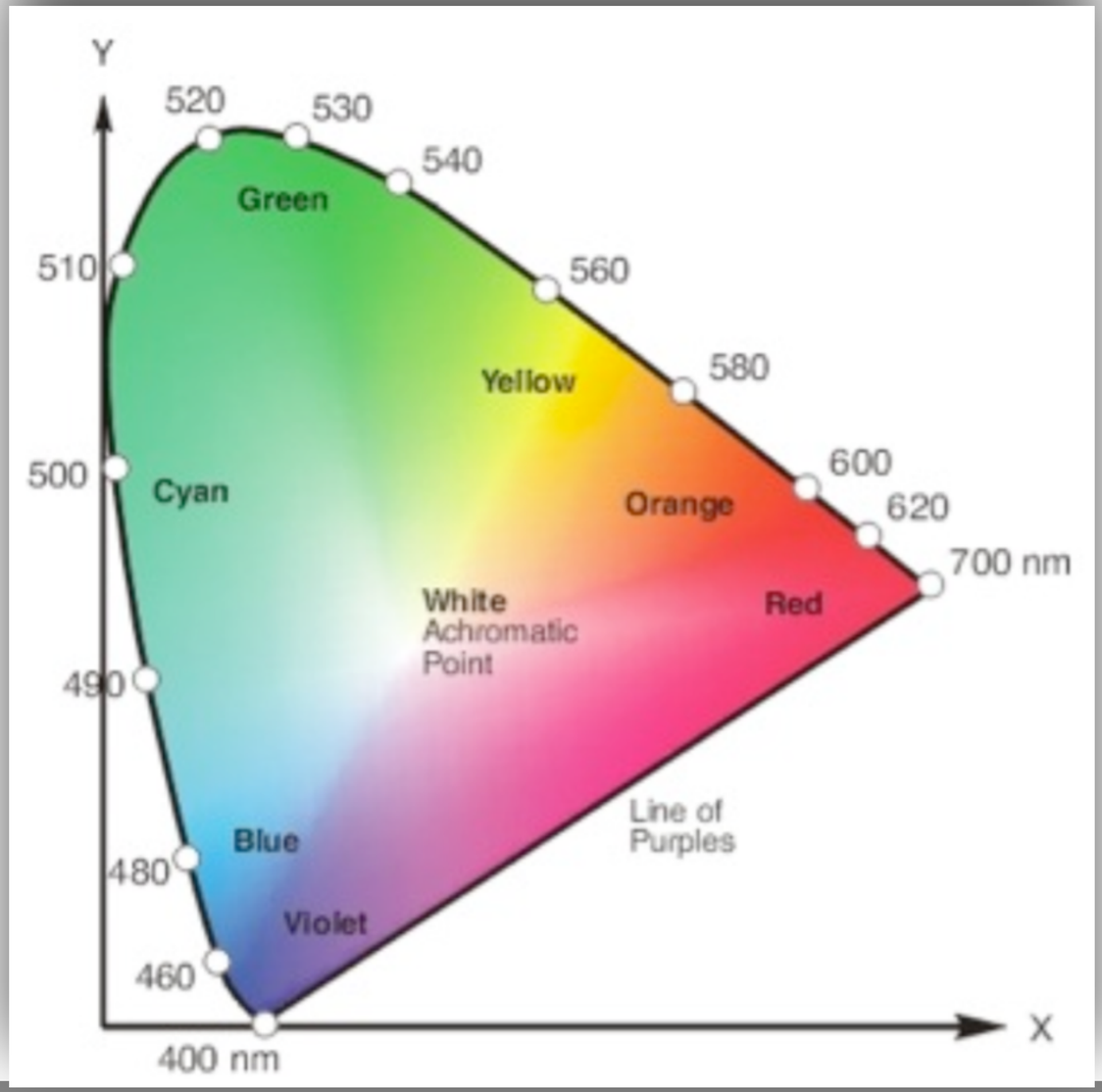
Rods and Cones - Color vs. Intensity

Rods - sensitive to intensity (black and white sensitivity in low light conditions)

Cones - three types, S, M and L corresponding to short, medium and long wavelength light sensitivities



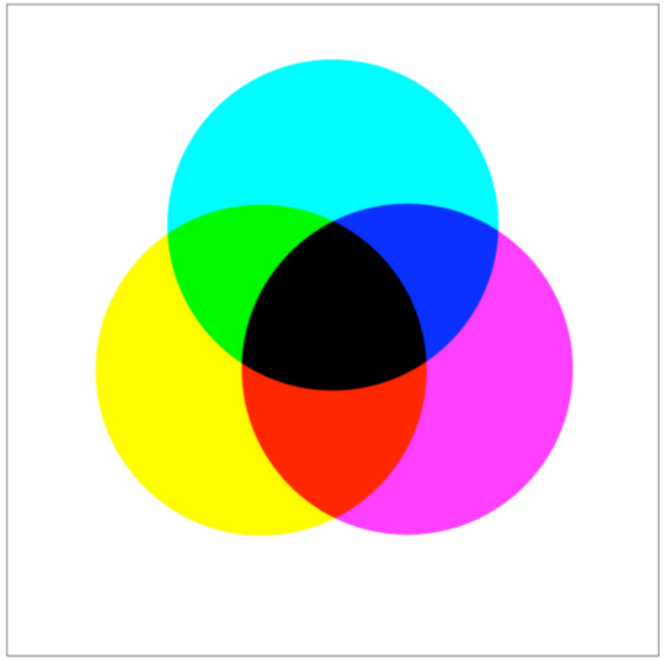
CIE Color Chromaticity Chart



Additive vs. Subtractive Color

Additive (RGB)

Subtractive (CMY)



$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1-R \\ 1-G \\ 1-B \end{bmatrix}$$

Additive vs. Subtractive Color

- RGB
  - **red-green-blue**
  - **Additive scheme**
- CMY
  - **Cyan-magenta-yellow**
  - **Subtractive scheme**
  - **Black (CMYK) is typically added to inkjet printers**
    - Difficult to make exact black by mixing CMY, requires precision
    - Typically one uses black the most so it makes sense to have a separate ink cartridge for black
- HSV
  - **Hue-saturation-value**
  - **Many feel this is a more natural way to describe color for humans**

Example: Bad color matching

- Eeeghh!
- The red and blue are on opposite ends of the visual color spectrum, so we have trouble focusing on both colors simultaneously
- I could have made this worse by adding all equations, but last time too many people passed out!
- AVOID REDS ON BLUES OR BLUES ON REDS

Example: Good color matching

- Ahhh…
- This is much more comfortable for the eyes.
- Choose colors which are based on luminance differences
- generally avoid two fully saturated colors as foreground and background
- Increase contrast by reducing the perceived intensity of either the foreground or background

Luminance Equation

$$Y = 0.30 * Red + 0.59 * Green + 0.11 * Blue$$

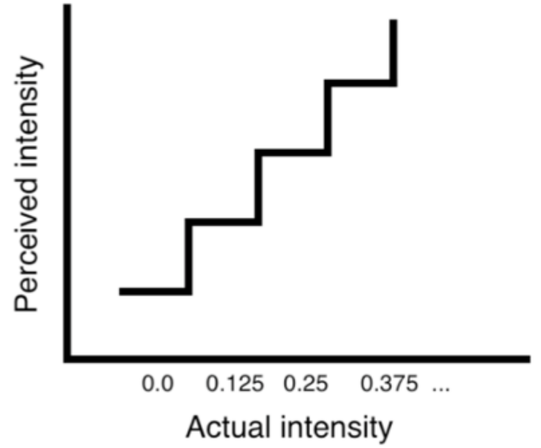
- Perceived intensity due to a color
  - Different contributions of red/green/blue components
  - Empirically determined

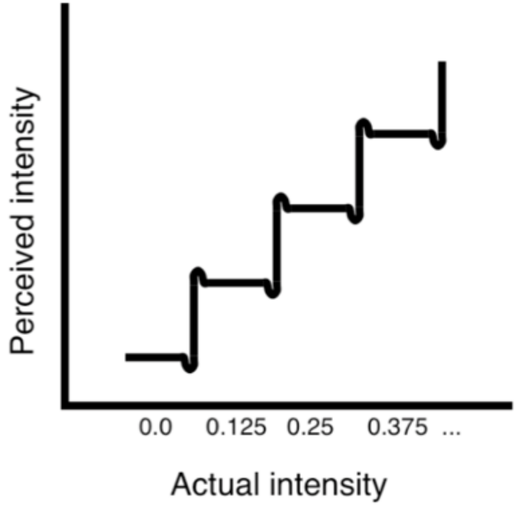
Contrast tables

	Black	White	Red	Green	Blue	Cyan	Magenta	Orange	Yellow
Black	0.00	1.00	0.30	0.59	0.11	0.70	0.41	0.60	0.89
White	1.00	0.00	0.70	0.41	0.89	0.30	0.59	0.41	0.11
Red	0.3	0.7	0.00	0.29	0.19	0.40	0.11	0.30	0.59
Green	0.59	0.41	0.29	0.00	0.48	0.11	0.18	0.01	0.30
Blue	0.11	0.89	0.19	0.48	0.00	0.59	0.30	0.49	0.78
Cyan	0.70	0.30	0.40	0.11	0.59	0.00	0.29	0.11	0.19
Magenta	0.41	0.59	0.11	0.18	0.30	0.29	0.00	0.19	0.48
Orange	0.60	0.41	0.30	0.01	0.49	0.11	0.19	0.00	0.30
Yellow	0.89	0.11	0.59	0.30	0.78	0.19	0.48	0.30	0.00

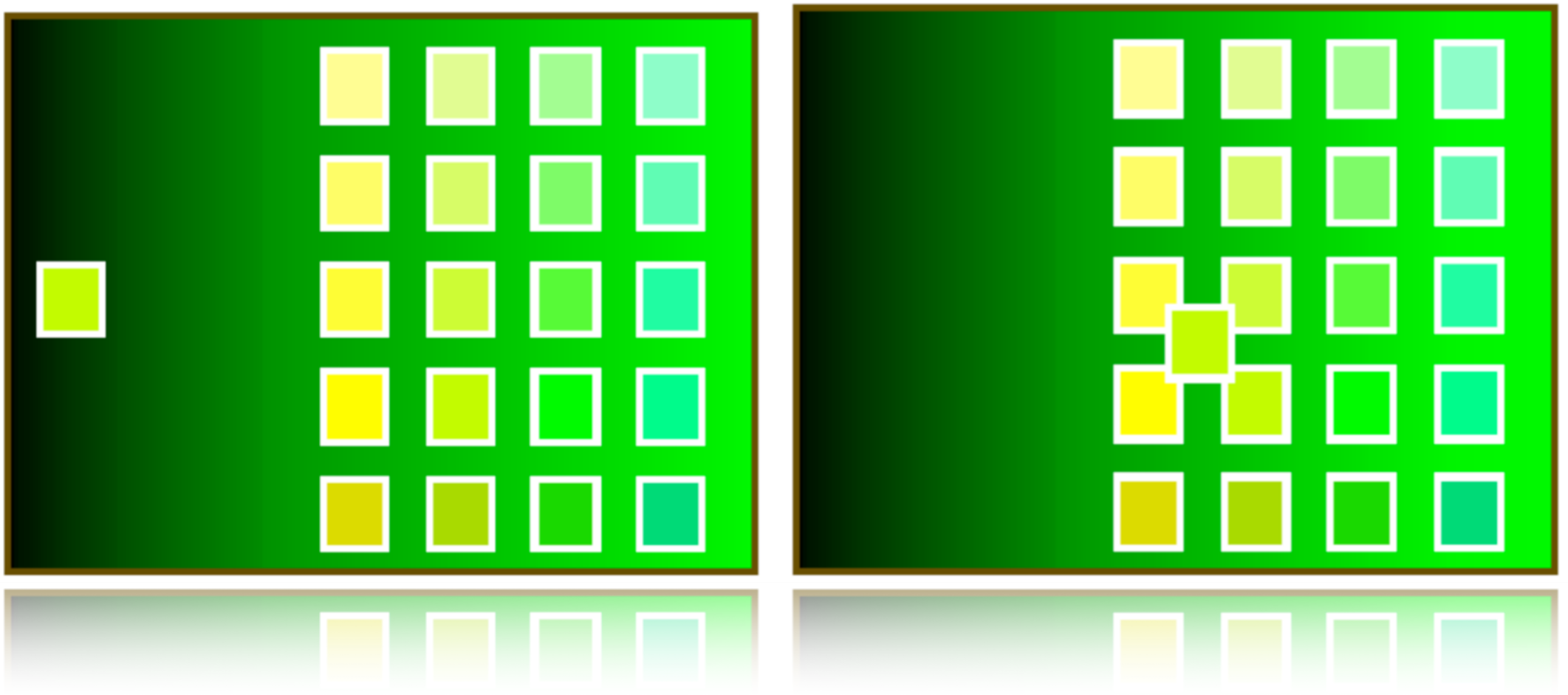
Table 5.1: A color contrast table can be formed by subtracting the luminance equation values for two different colors, then taking the absolute value.

Beware of Mach Banding



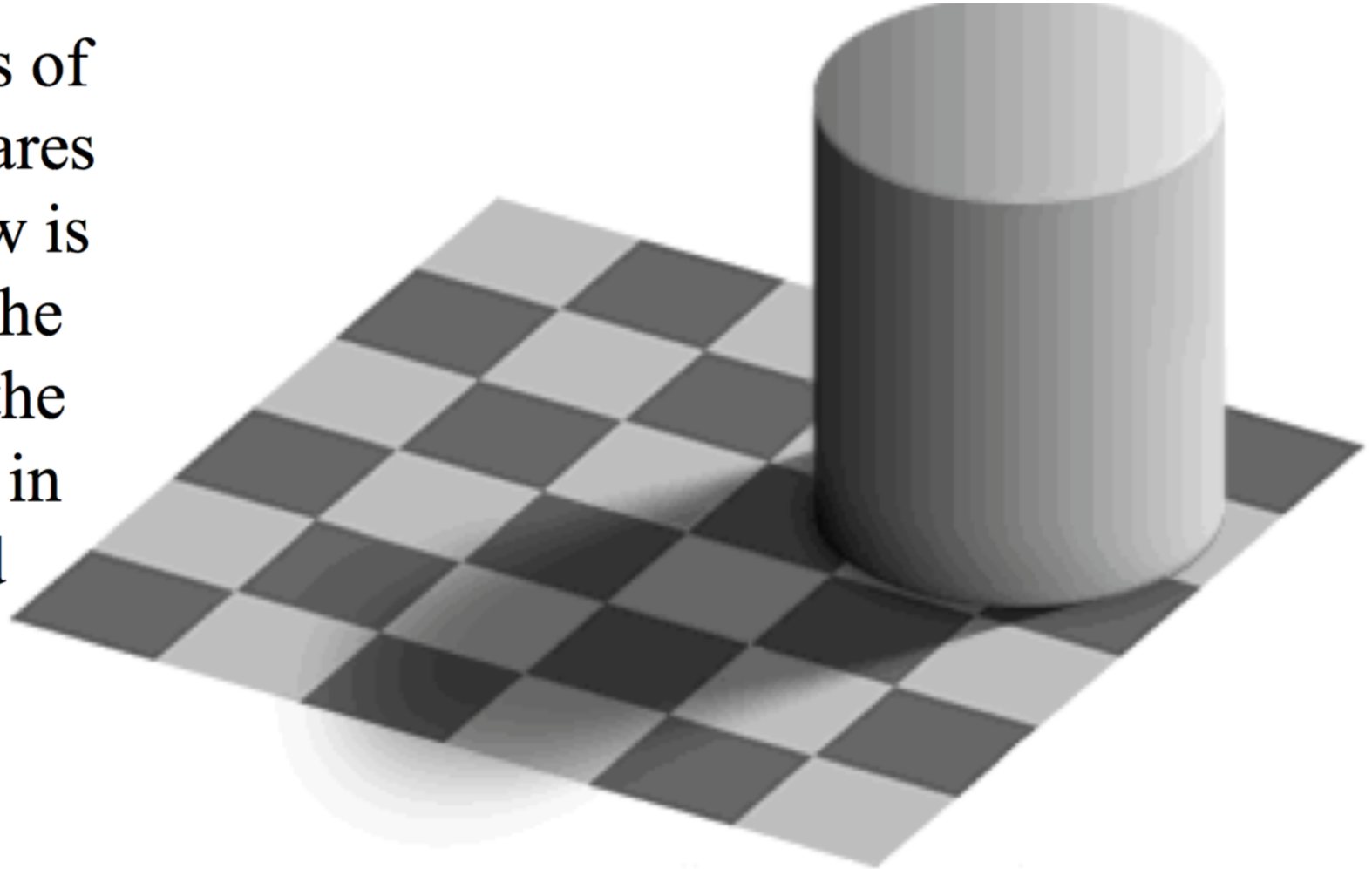


Recall that perceived color intensity is also context dependent



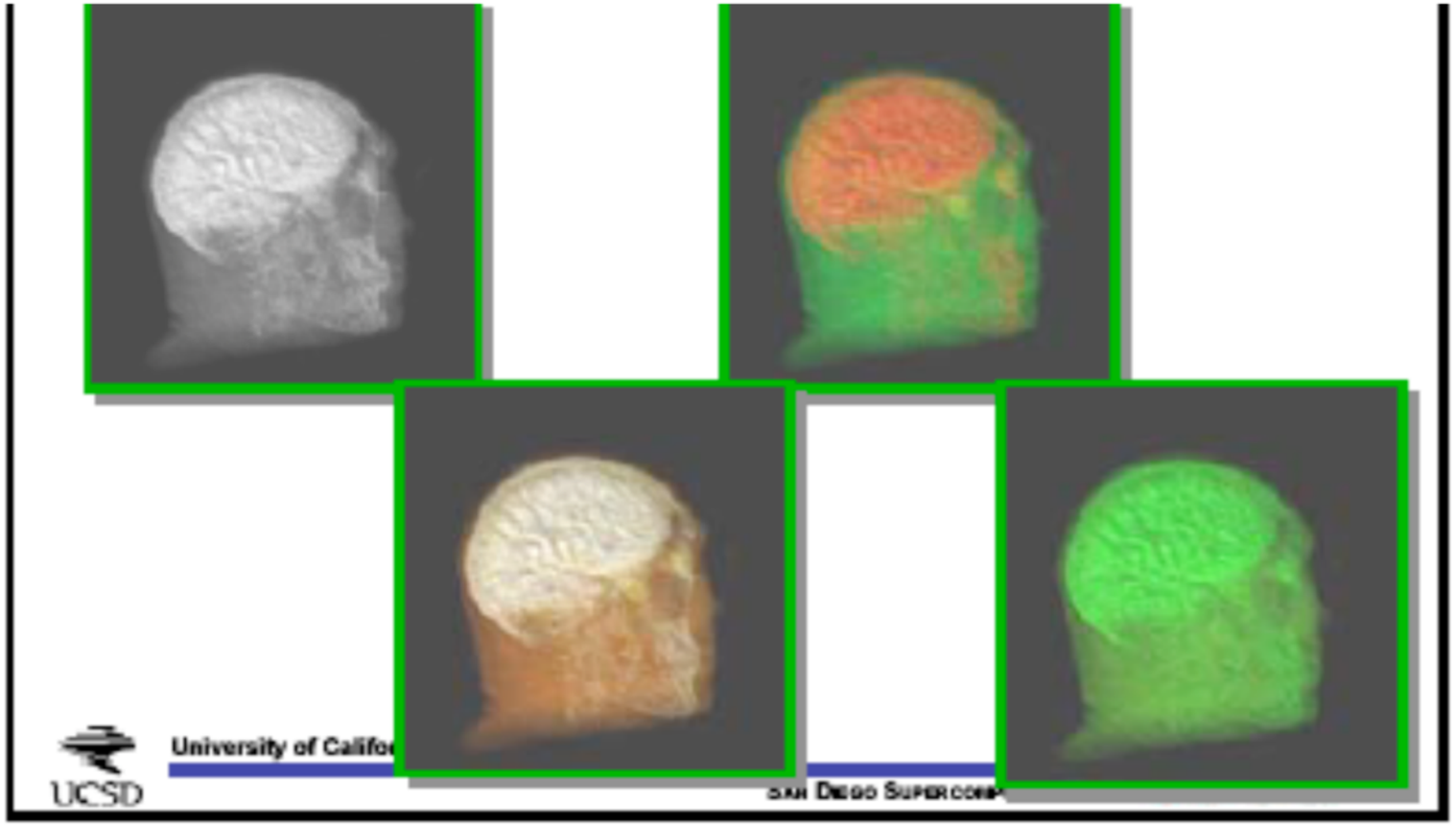
Perceived lightness is context dependent as well

- The lightness of the light squares in the shadow is the same as the lightness of the dark squares in the unshaded region



False color representation and color maps

- Map values from any range to a map of colors
  - i.e. a matrix of 0-1 range-> white-black



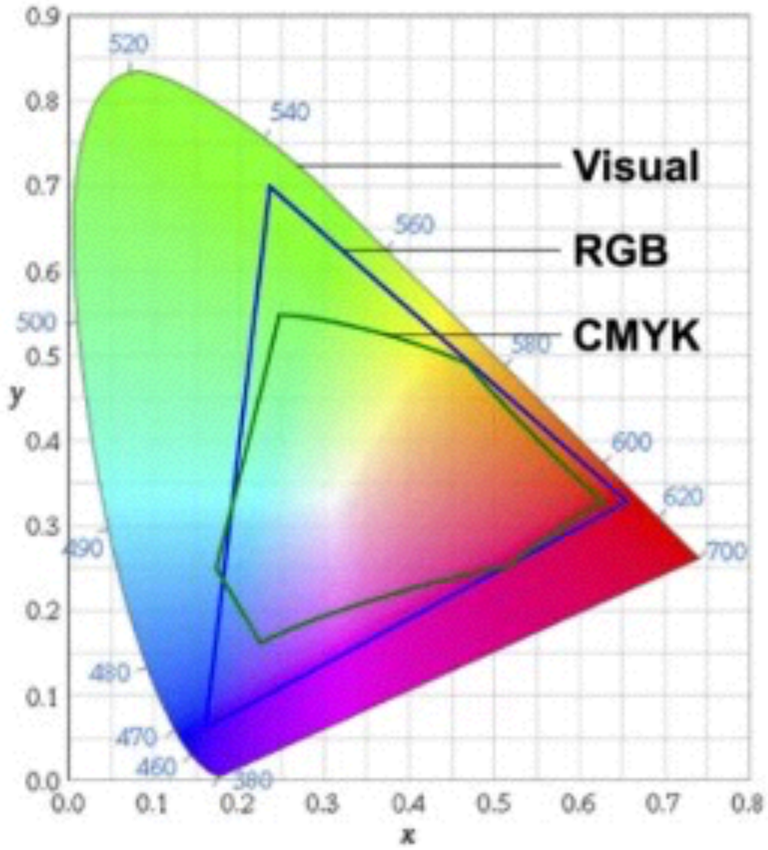
More color maps

- Rainbow color scale - magenta is not directly in the EM spectrum

- Heated object color scale - intensity increases left->right

Color **Gamut comparison**

- The range of colors a device can display
- This can be a triangle or more complex shapes
- Typically a subset of human perception
  - Stay away from what cannot be printed when creating for papers



Output

- If you are creating visualizations for multiple contexts (video, computer monitors, printed papers, etc) be aware of device limitations
- Use **redundant encoding of information** if you don’t know what the output is or who will be looking at it
  - **Different fonts**
  - **Symbols**
  - **Fill pattern**
  - **Outline pattern**
  - **Outline thickness**

Output

- If you are creating visualizations for multiple contexts (video, computer monitors, printed papers, etc) be aware of device limitations
- Use **redundant encoding of information** if you don’t know what the output is or who will be looking at it
  - **Different fonts**
  - **Symbols**
  - **Fill pattern**
  - **Outline pattern**
  - **Outline thickness**

Math and symbol review

- http://casimpkinsjr.radiantdolphinpress.com/pages/cogs138_sp23/handouts/greek_letters_review.pdf
- http://casimpkinsjr.radiantdolphinpress.com/pages/cogs138_sp23/handouts/math_review.pdf
- Handouts page on website:
  - http://casimpkinsjr.radiantdolphinpress.com/pages/cogs138_sp23/handouts.html

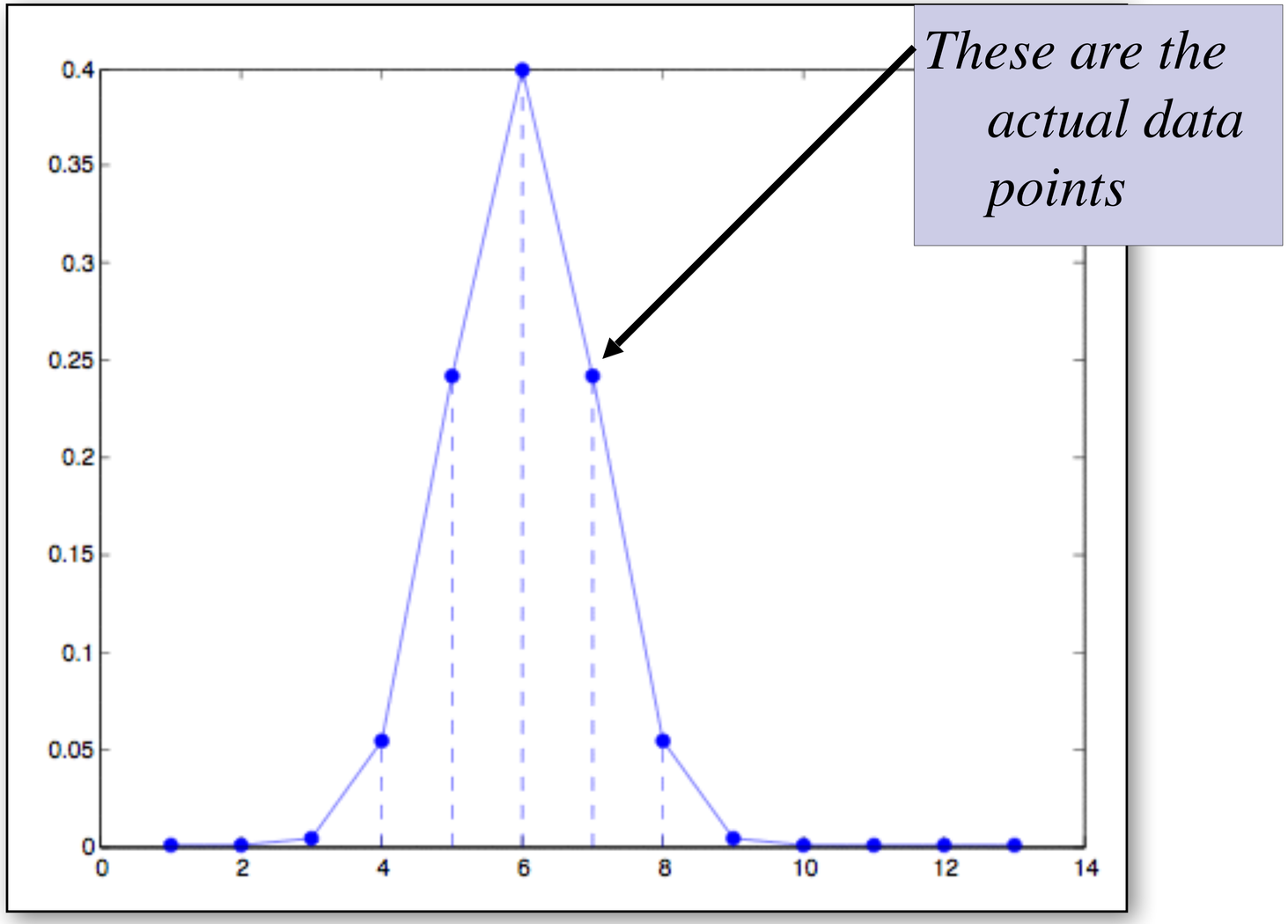
Python docs on statistics

- Individual stats:
  - https://docs.python.org/3/library/statistics.html
- Comparisons:
  - https://github.com/drsimpkins-teaching/cogs138/blob/main/Tutorials-master/12-StatisticalComparisons.ipynb

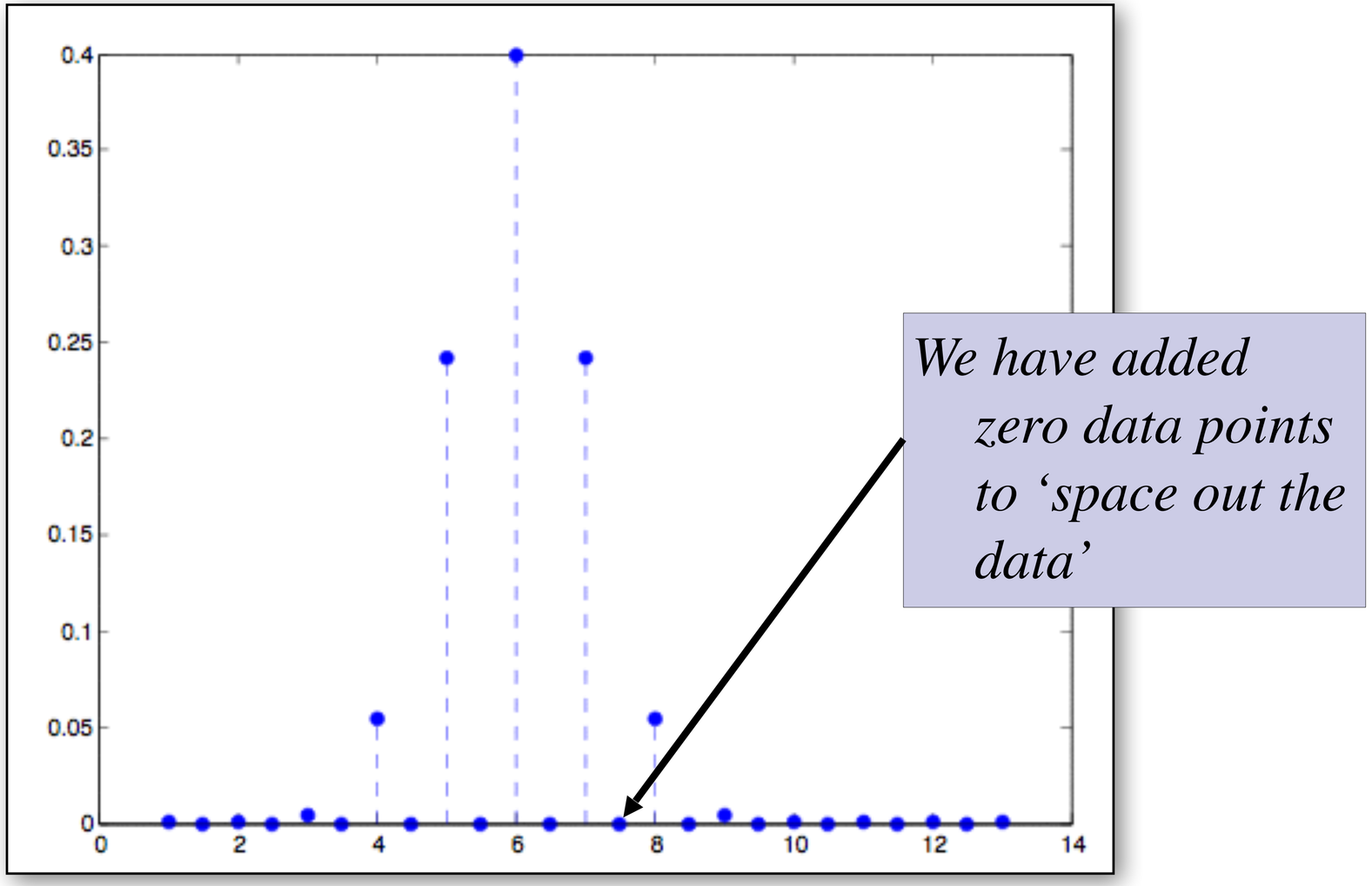
Data analysis I : Central Tendency

- Considers the general sense of the data
- What does the data look like?

What does super-sampling look like?



Example: Up-sampling

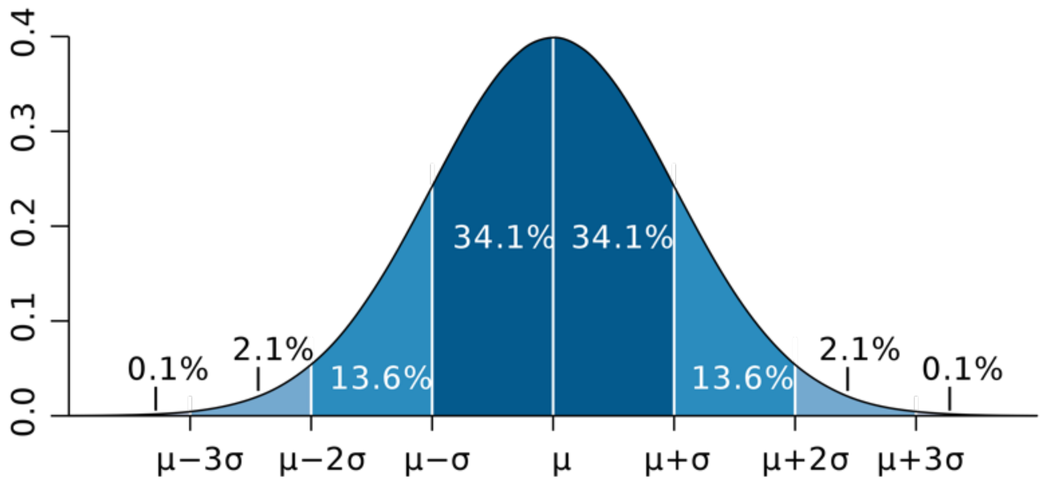


Quantitatively: Another way to ‘look’ at data

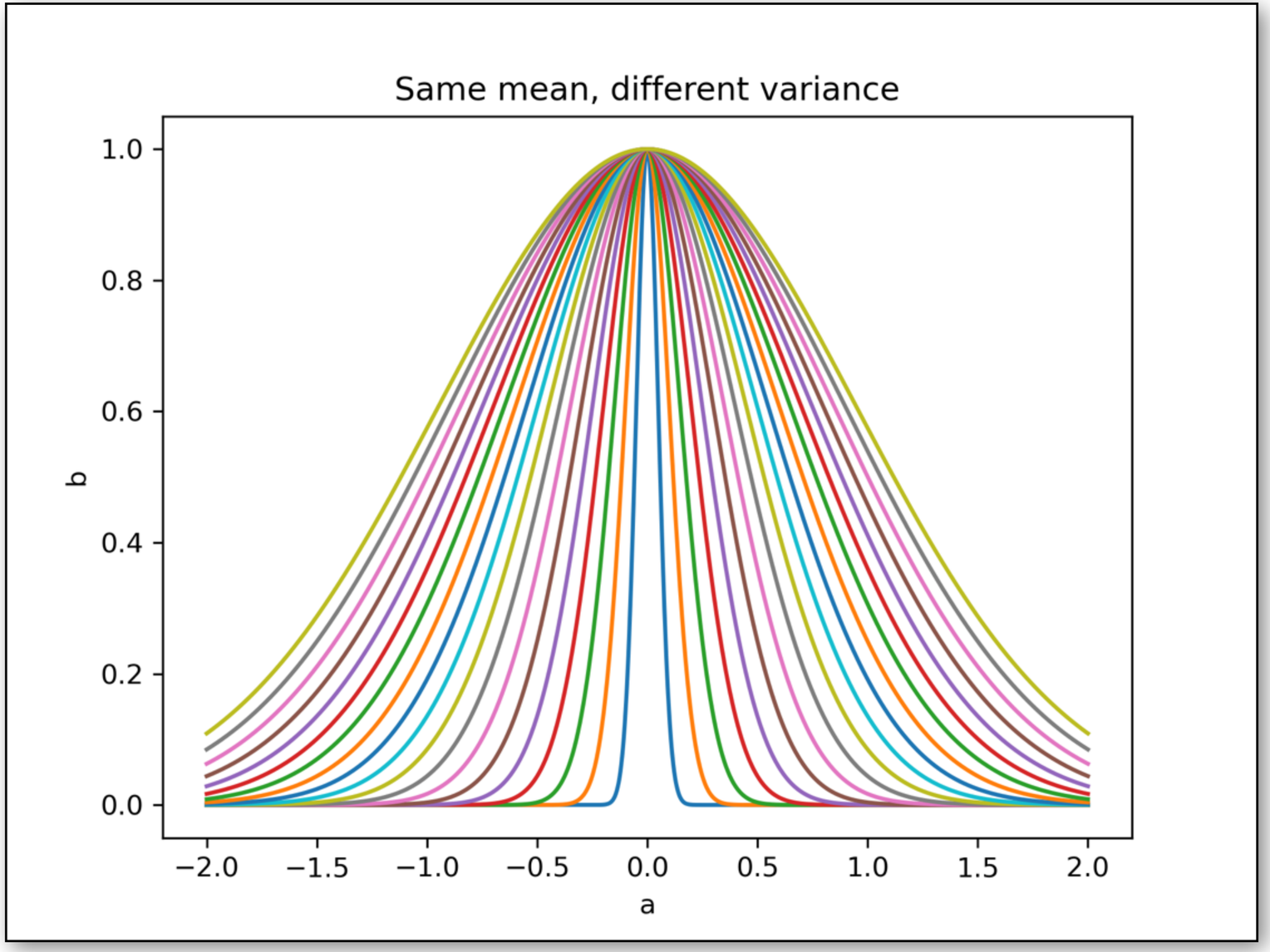
- How do we look at data quantitatively and extract meaningful information?
- Computing basic statistics is a start
  - **Mode**
  - **Mean**
  - **Median**
  - **Standard Deviation**
  - **Variance**
  - **Covariance**
  - **Correlation**
- https://www.w3schools.com/python/module_statistics.asp

Central Limit Theorem and Law of Large Numbers

- If X is taken independently from the same distribution, then X_i is said to be a random sample from that distribution
- X_i are said to be independent identically distributed (i.i.d.)
- **Law of large numbers** (LLN)- sample mean approaches population mean as n approaches infinity
- **Central limit theorem** (CLT) - the distribution of the sample mean approaches a normal distribution for n approaching infinity



The mean isn't everything! These all have the same mean



The describe method

- DataFrame.describe()
- Computes basic statistics as well and presents a summary
- There are more thorough stats summaries but this is simple and fast
- Provides:

- count - The number of not-empty values.
- mean - The average (mean) value.
- std - The standard deviation.
- min - the minimum value.
- 25% - The 25% percentile*.
- 50% - The 50% percentile*.
- 75% - The 75% percentile*.
- max - the maximum value.

*Percentile meaning: how many of the values are less than the given percentile.

Underlying dynamics need to be exposed

- Tacoma narrows bridge disaster
  - 1st order vs. higher order
  - https://en.wikipedia.org/wiki/Tacoma_Narrows_Bridge_(1940)

https://en.wikipedia.org/wiki/Tacoma_Narrows_Bridge_(1940)#/media/File:Opening_day_of_the_Tacoma_Narrows_Bridge,_Tacoma,_Washington.jpg

How do we then learn about unknown dynamics?

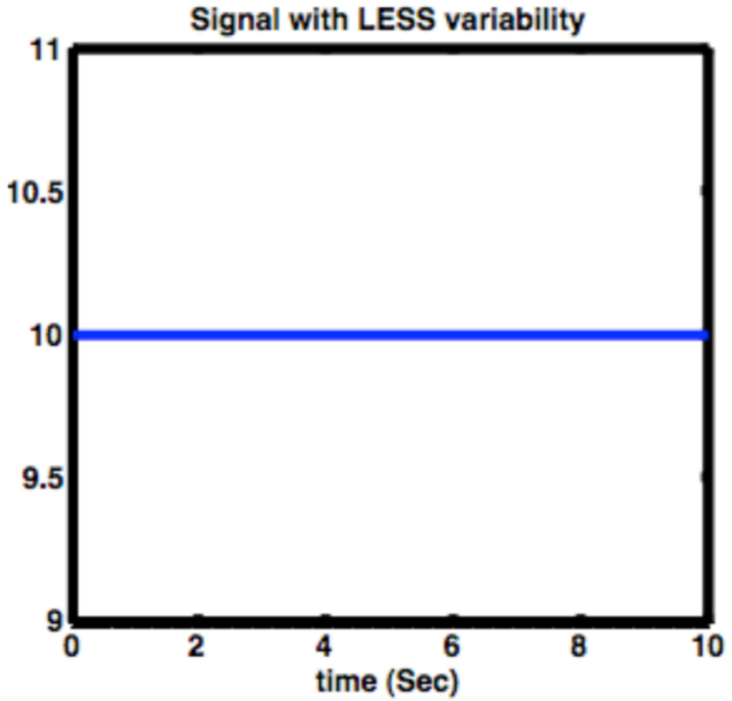
- Learn by **experience**, **experimentation**, **hypothesis generation**, **data science!**

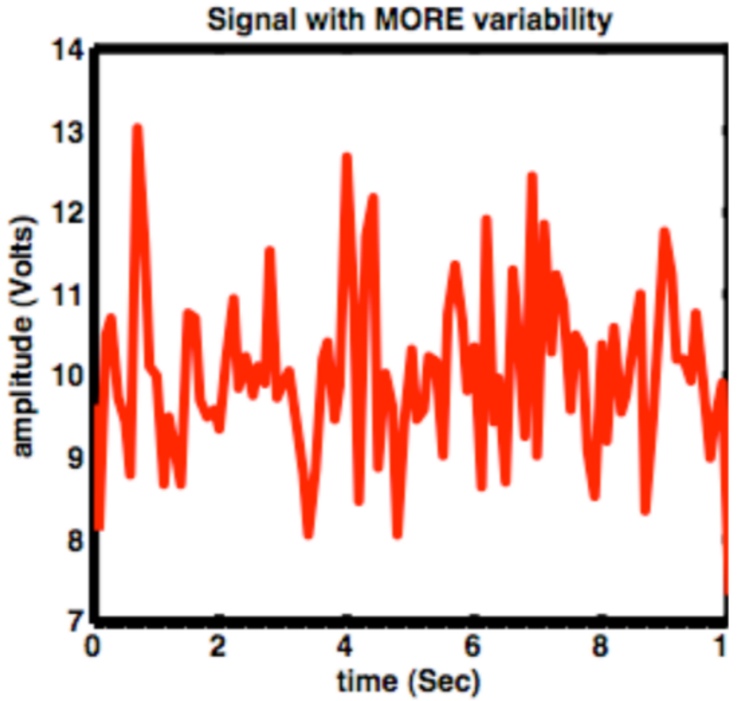
  - “The Tacoma Narrows bridge failure has given us invaluable information ... It has shown [that] every new structure [that] projects into new fields of magnitude involves new problems for the solution of which neither theory nor practical experience furnish an adequate guide. It is then that we must rely largely on judgment and if, as a result, errors, or failures occur, we must accept them as a price for human progress.” [Othmar Ammann]

  - Following the incident, engineers took extra caution to incorporate aerodynamics into their designs, and wind tunnel testing of designs was eventually made mandatory.

Why we need a measure of variability

Same means, different variability of the signal





We need a measure of Variability, here are a few…

- Range
  - From math review, difference between max and min values of the data

$$Range(x) = Max(x) - Min(x)$$

- Variance
  - Mean of squared deviations from the mean
  - In square units of the sample variable
- Standard deviation
  - Square root of variance
  - In units of the sample variable - sometimes easier to interpret

Variance (part II)

- Steps to compute the variance
  - **Compute the deviations from the mean for all the data**

$$d_i = \left(x_i - \bar{x}\right)$$

  - **Compute the square of each of the deviations**

$$sd_i = \left(d_i\right)^2$$

  - **Sum up all these squared deviations**

$$ssqd = \sum_{i=1}^{N} \left(sd_i\right)$$

  - **Divide the mean squared deviations by N, the number of observations**

$$Var = \frac{ssqd}{N}$$

Standard Deviation

- Typical ‘deviation’ from the mean
- Ie how far on average scores depart on either side from the mean
- Easy to compute after the variance - just take the square root of the variance

$$SD = \sqrt{Var} = \sqrt{\frac{\sum (x_i - \bar{x})^2}{N}}$$

$$\bar{x} = \frac{\sum x_i}{N}$$

Z scores

- A Z score is simply a measure of how many standard deviations away from the mean a score is
- Units are standard deviations

$$Z_i = \frac{X_i - \mu}{SD}$$

Correlation coefficient motivation

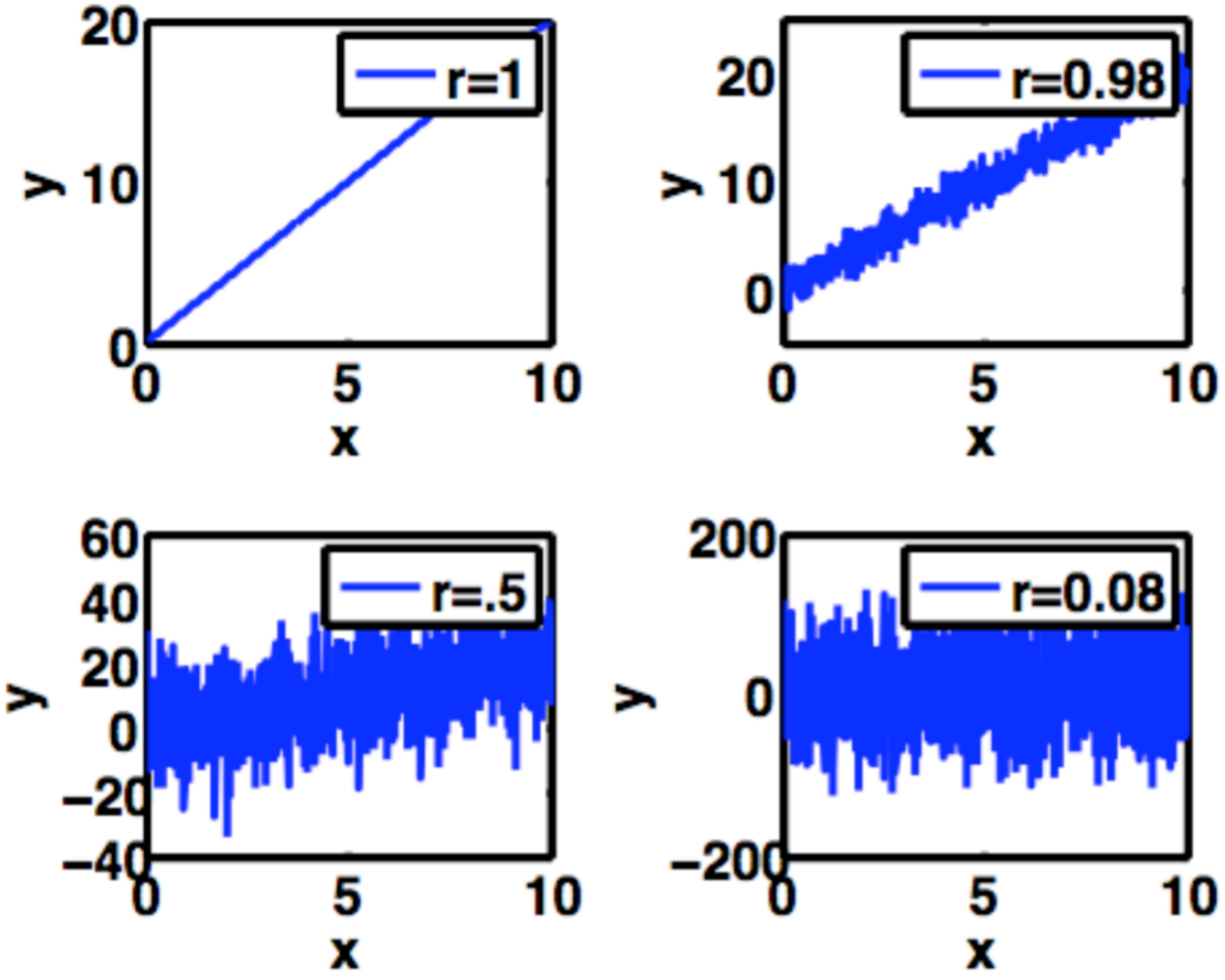
- We want to define a measure of how related our dependent and independent variables are
  - Variance, STD – variation of a single variable
  - Covariance – how two things vary in relation to each other
  - How do we compute the linear dependence of one variable to another?
- Correlation coefficient!

Correlation coefficient

$$\rho(j,k) = \frac{\sum_{i=1}^{N} Z_{ij} Z_{ik}}{N}$$

$$\rho(X,Y) = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}$$

$$\rho(X,Y) = \frac{E((X-\mu_X)(Y-\mu_Y))}{\sigma_X \sigma_Y}$$


r=1
r=0.98
r=.5
r=0.08
x
y
0
5
10
20
60
40
−20
−40
200
−200

Extending central tendency and variability to inference and hypothesis testing

CORRELATION

ASSOCIATION BETWEEN VARIABLES

i.e. Pearson Correlation, Spearman Correlation, chi-square test

COMPARISON OF MEANS

DIFFERENCE IN MEANS BETWEEN VARIABLES

i.e. t-test, ANOVA

REGRESSION

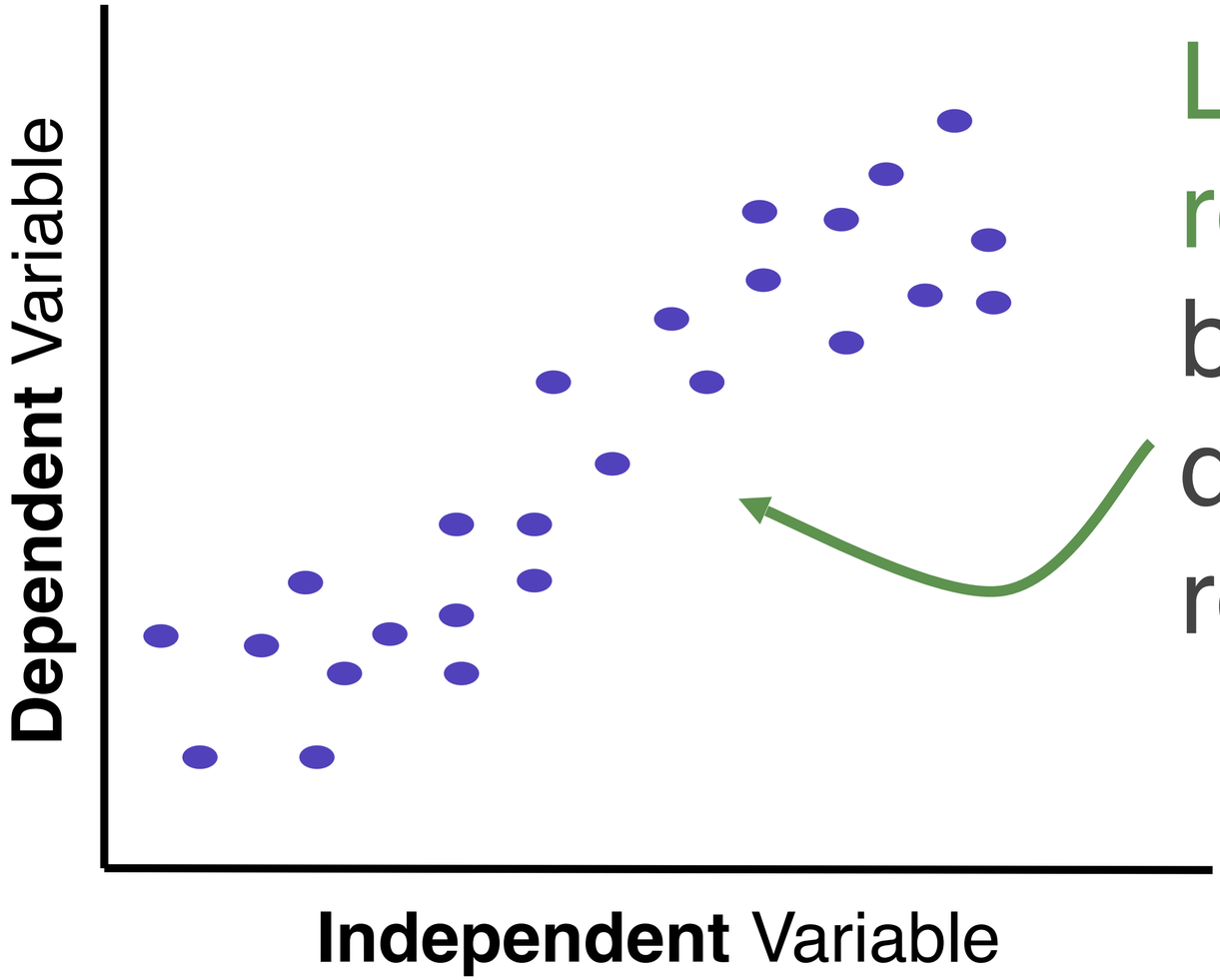
DOES CHANGE IN ONE VARIABLE MEAN CHANGE IN ANOTHER?

I.e. simple regression, multiple regression

NON-PARAMETRIC TESTS

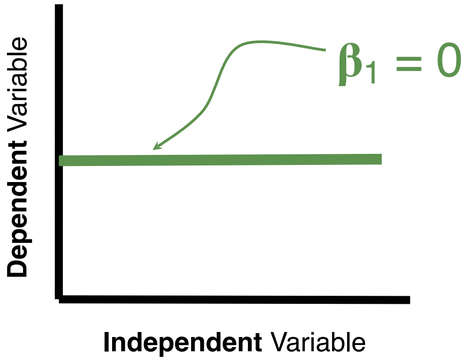
FOR WHEN ASSUMPTIONS IN THESE OTHER 3 CATEGORIES ARE NOT MET

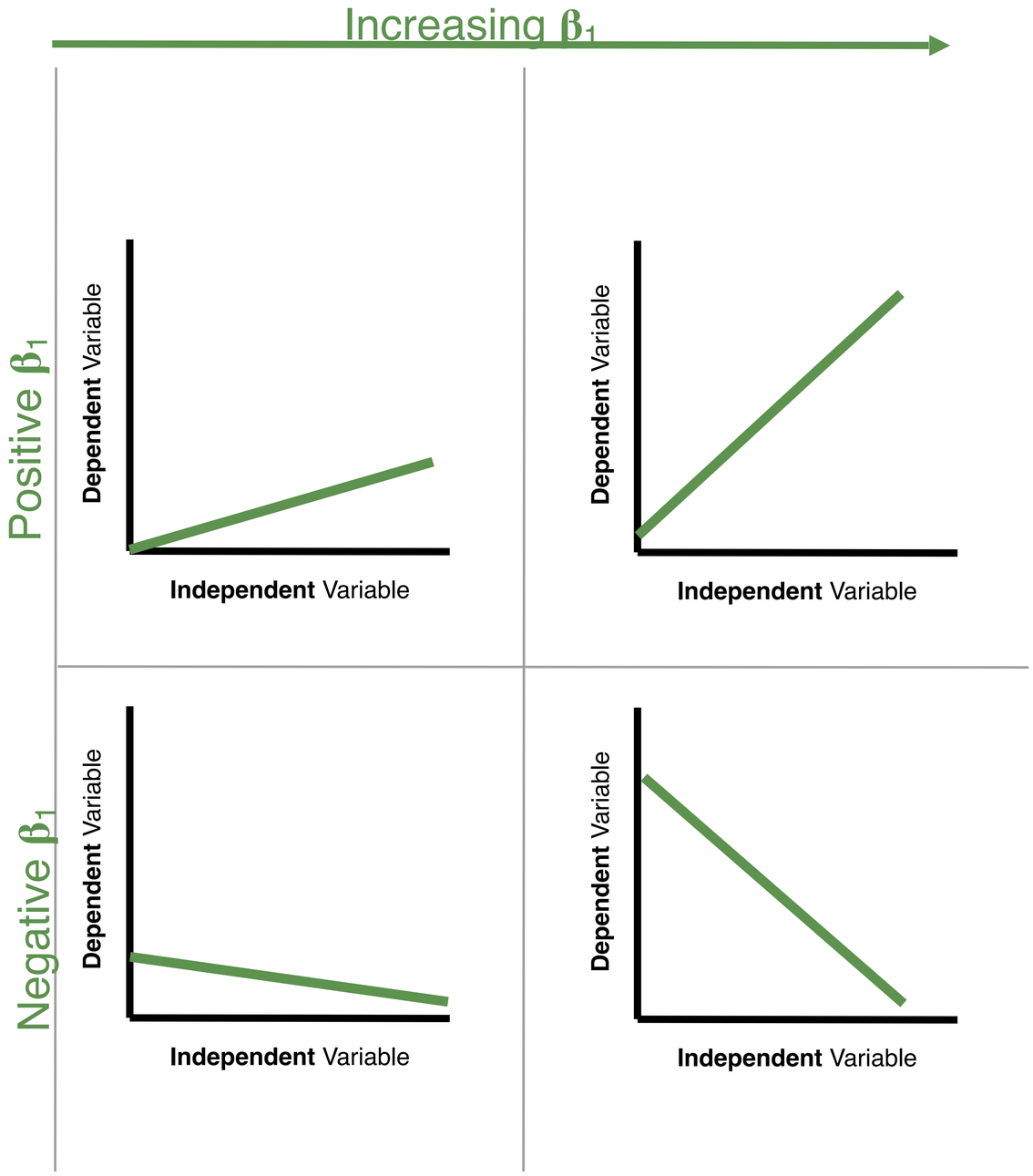
i.e. Wilcoxon rank-sum test, Wilcoxon sign-rank test, sign test



Linear regression can be used to describe this relationship

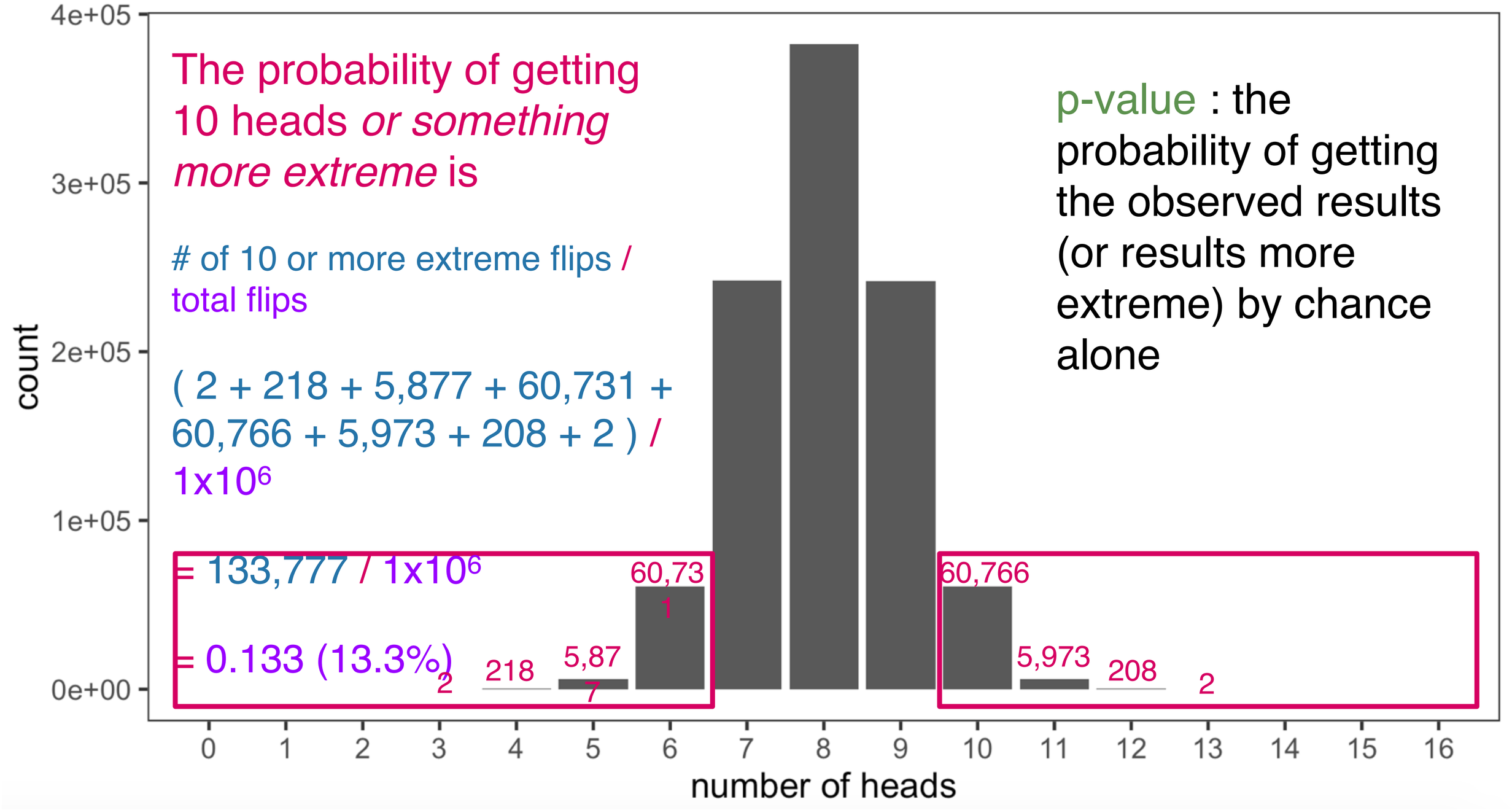
Effect size (β_1) can be estimated using the slope of the line





Assumptions of linear regression

1. Linear relationship
2. No multicollinearity
3. No auto-correlation
4. Homoscedasticity


The probability of getting 10 heads *or something more extreme* is
# of 10 or more extreme flips / total flips
(2 + 218 + 5,877 + 60,731 + 60,766 + 5,973 + 208 + 2) / 1x10^6
= 133,777 / 1x10^6
= 0.133 (13.3%)
p-value : the probability of getting the observed results (or results more extreme) by chance alone
count
4e+05
3e+05
2e+05
1e+05
0e+00
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
number of heads
2
218
5,877
60,731
60,766
5,973
208
2

Takes into account
the effect size (β_1)
and the SE

p-value : the probability of getting the observed results (or results more extreme) by chance alone

Confounding

Models and the modeling process

Linear least squares

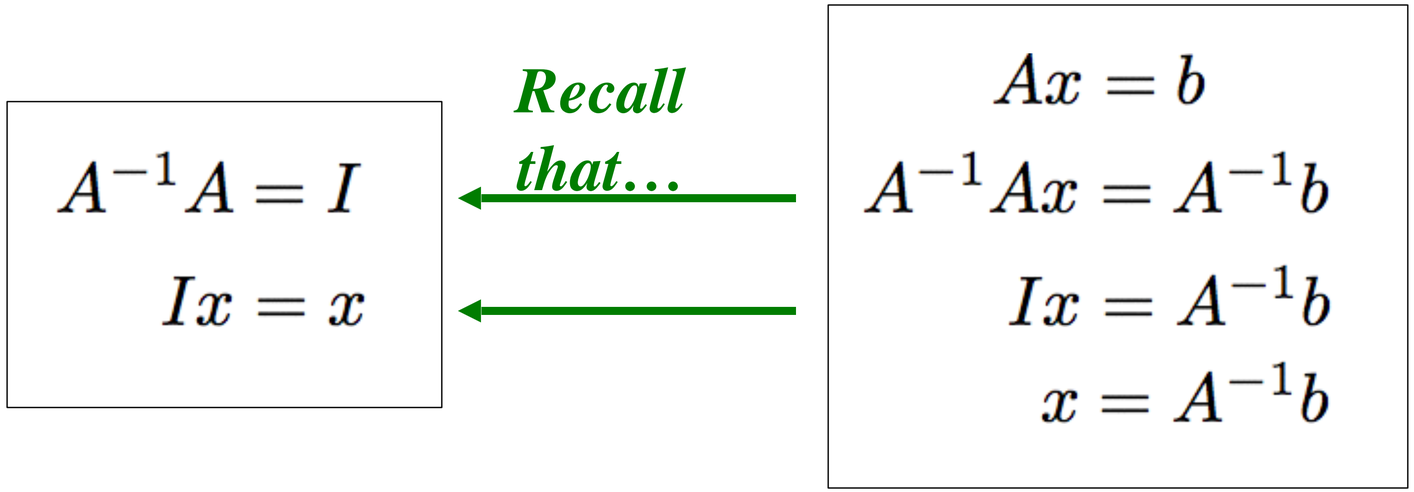
- You're probably all familiar with linear regression -- fitting a line to a bunch of data.
- more formally fitting $y = mx + b$ for paired x,y data (can also do multidimensional)
- Let’s see how it’s done mathematically

Solving Ax=b

- Solving for $\boxed{x = A^{-1}b}$ involves computing the inverse of the A matrix
  - **Insiwhatsitz? Don't worry…inverses are a way to make life easier**
- There are several methods, and you can solve for arbitrarily sized problems (ie what if we want to find 100 variables? Not fun by hand: (Let's use a computer to do it for us!!!:)
  - **Gaussian elimination (what you learned in linear algebra class)**
    - Don't worry you won't have to do it by hand in this class!
  - **Thomas algorithm, etc (and other more efficient methods computationally)**
  - **Python has gaussian elimination (and others) built-in nicely of course through numpy modules**

Solving Ax=b

- We compute the solution of our canonical problem by



Outline for this section

- Colormap implementation
- What is interpolation?
  - **Definition**
  - **Applications, motivation for use**
  - **Orion nebula simulation**
- LERP - Linear interpolation
- BERP - Bilinear interpolation
- TERP - Trilinear interpolation
- SLERP - Spherical linear interpolation in polar coordinates
- Examples

Creating the color map (r,g,b) components

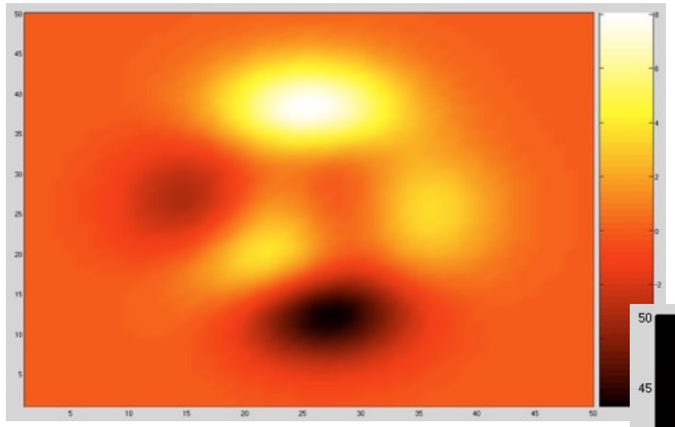
- To create a custom color map we need to make a matrix which is Dim nx3, range [0,1]
- Each column is the range of either red, green, blue
- Writing it by hand:

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

- Typing it into a python variable:

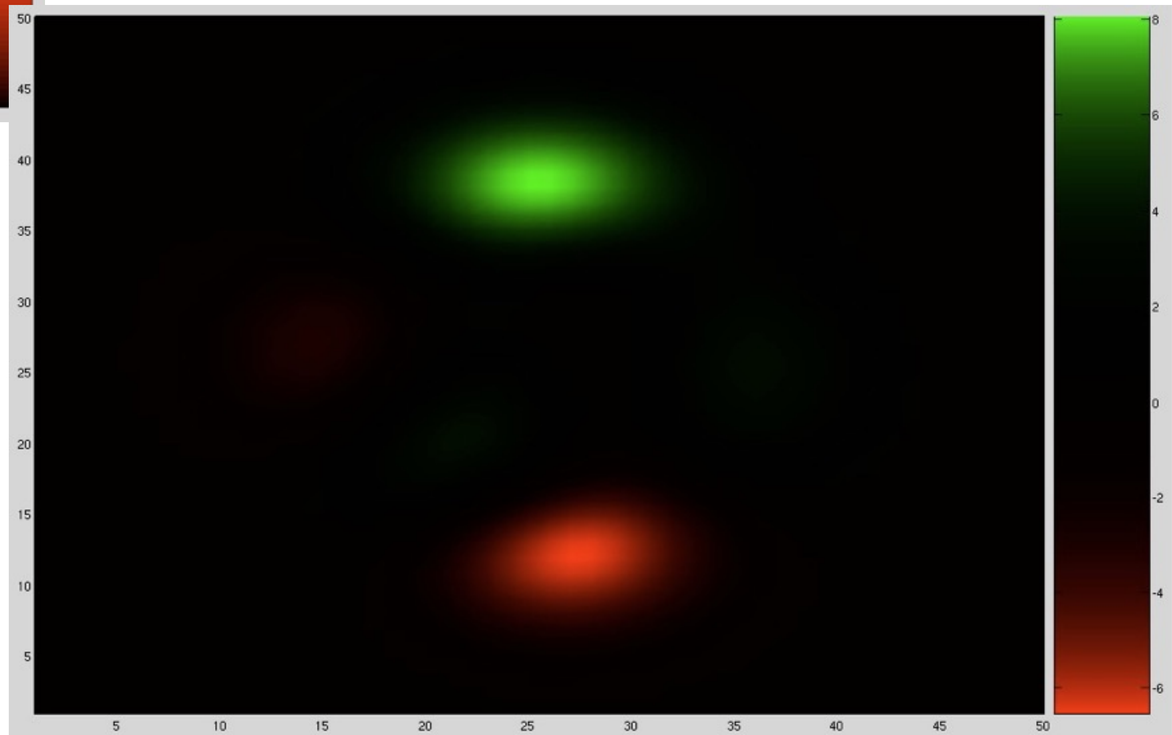
M = ?

Results of our custom color map



<- Using the built-in ‘hot’ color map

Using our color map->



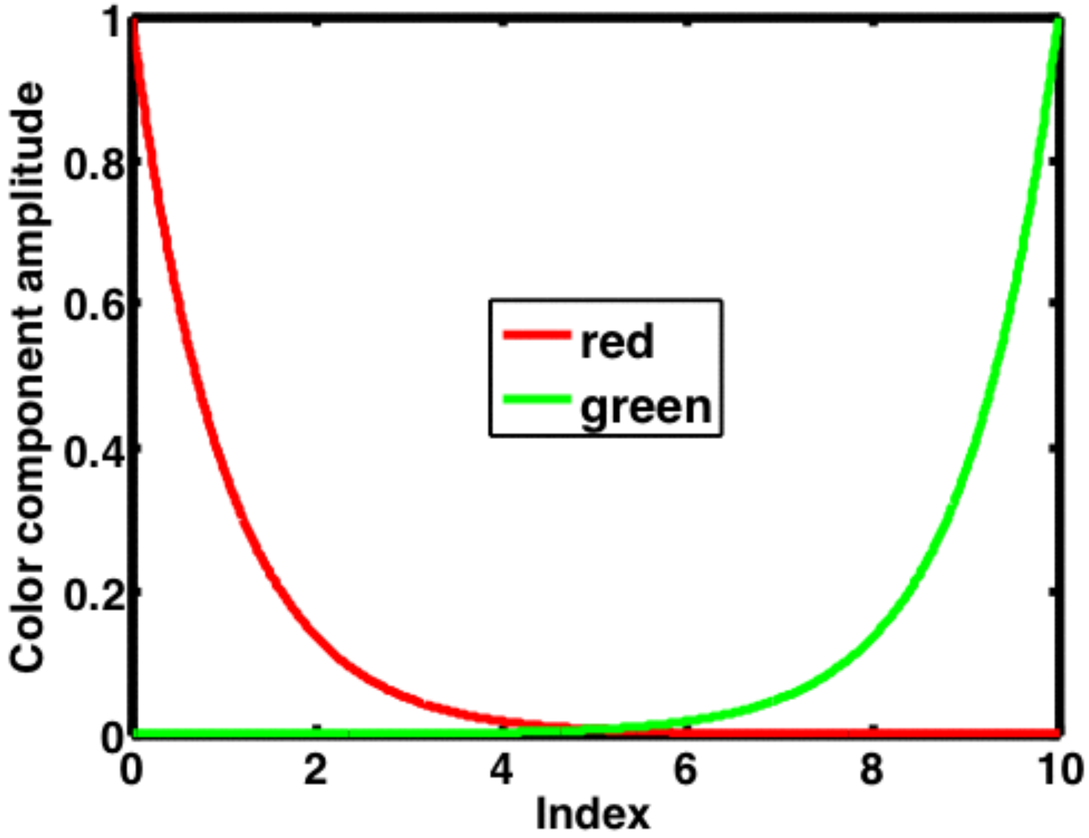
The final smooth color map

- And equations:
  - **Decreasing:**

$$r = \exp(-x)$$
$$x = 0:.01:10$$

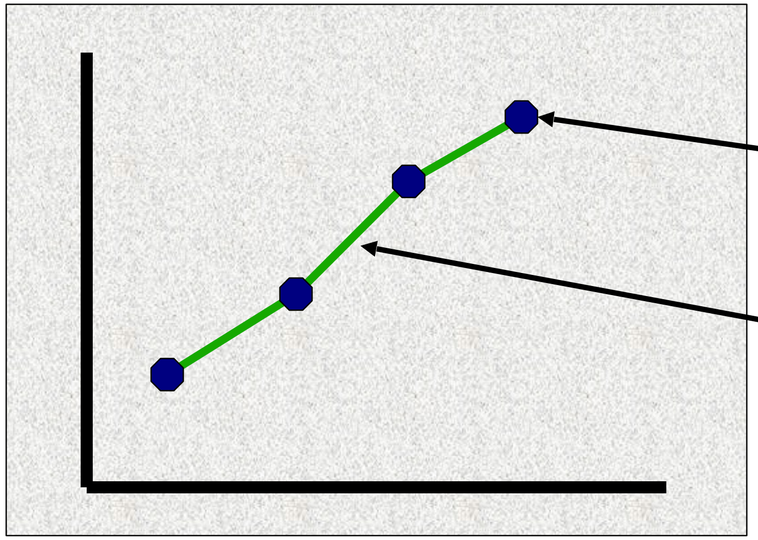
  - **Increasing:**

$$g = \frac{\exp(x)}{\max[\exp(x)]}$$
$$x = 0:.01:10$$

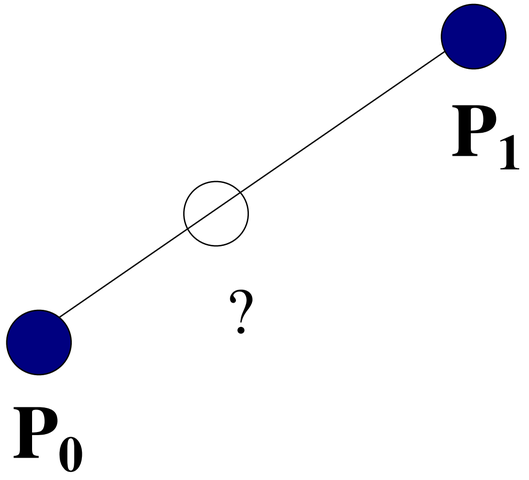


Interpolation defined

- Given a set of data points, we can construct a curve which fits exactly through each datapoint



Linear interpolation ("LERP")



We use a parametric curve to blend between the two points:

$$P(t) = (1-t)P_0 + tP_1$$

Often this is written in the more efficient form:

$$P(t) = P_0 + t(P_1 - P_0)$$

There are less computations, only compute P_1-P_0 once per pair of points

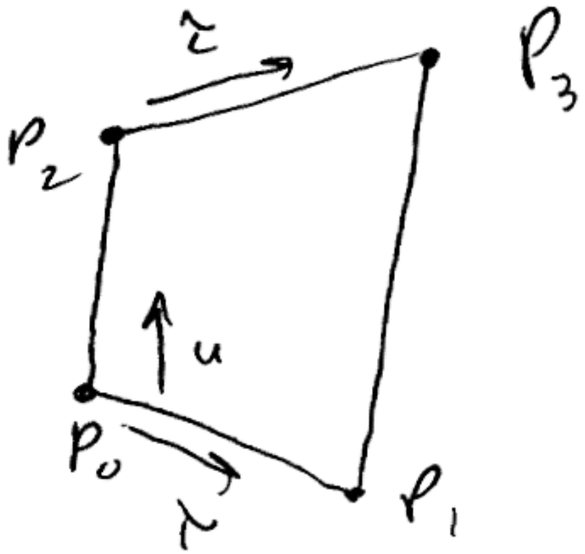
In 3D:

$$x(t) = (1-t)x_0 + tx_1$$

$$y(t) = (1-t)y_0 + ty_1$$

$$z(t) = (1-t)z_0 + tz_1$$

Bilinear interpolation ("BERP")



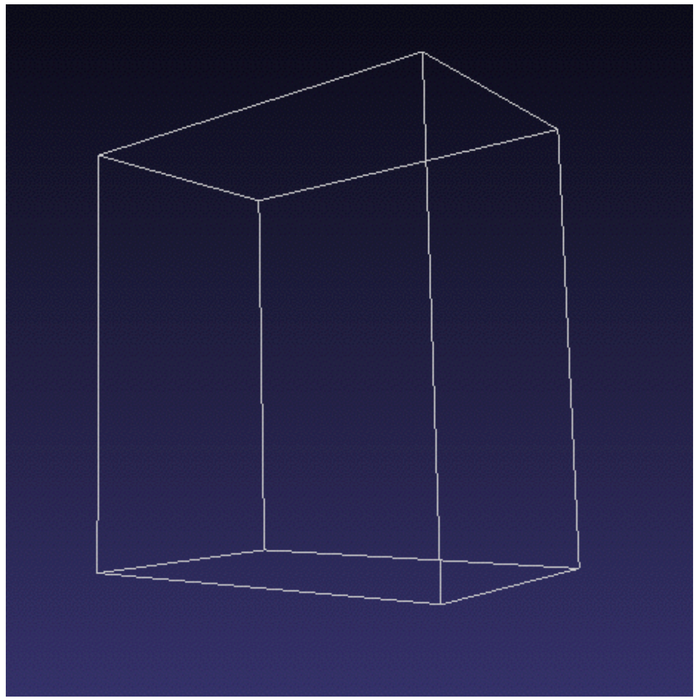
$$P_{01}(t) = (1-\tau)P_0 + \tau P_1$$
$$P_{23}(t) = (1-\tau)P_2 + \tau P_3$$
$$P_{0123}(t) = (1-u)P_{01} + uP_{23}$$

- Substituting the first two into the third:

$$P_{0123}(t) = (1-\tau)(1-u)P_0 + \tau(1-u)P_1 + (1-\tau)uP_2 + \tau uP_3$$

Thus given 4 points, we can find an interpolated point anywhere in the space between them

Trilinear interpolation (“TERP”)

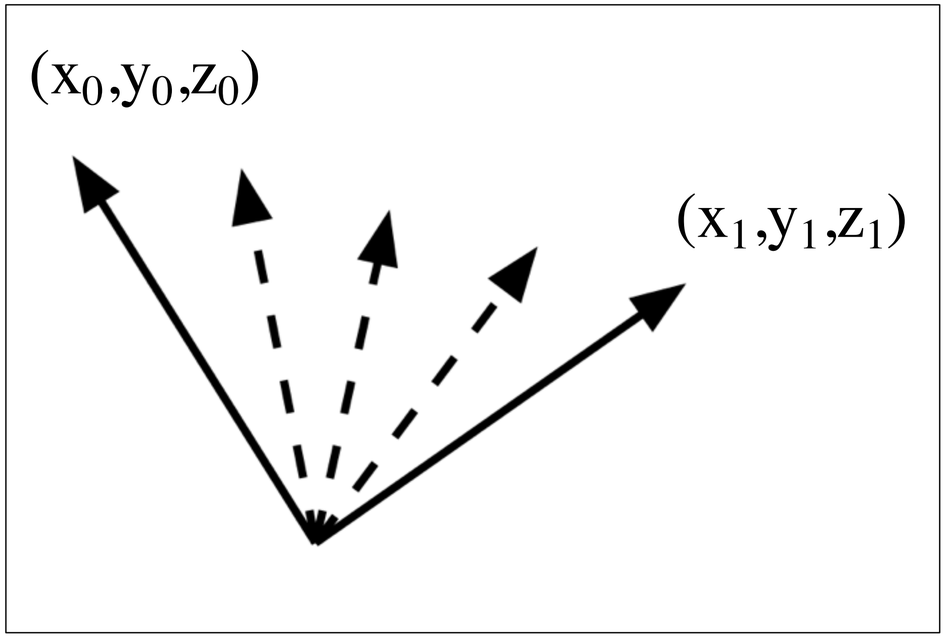


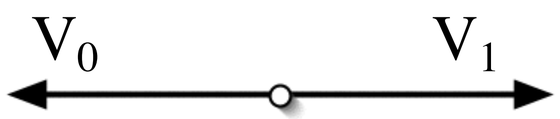
- How might you derive this such that we can interpolate to any point inside this cube?
  - **Same as LERP and BERP but a third interpolation parameter (another dimension)**

The points do NOT have to be evenly spaced

Spherical linear interpolation("SLERP")

- Let's say we have two vectors we want to interpolate between:





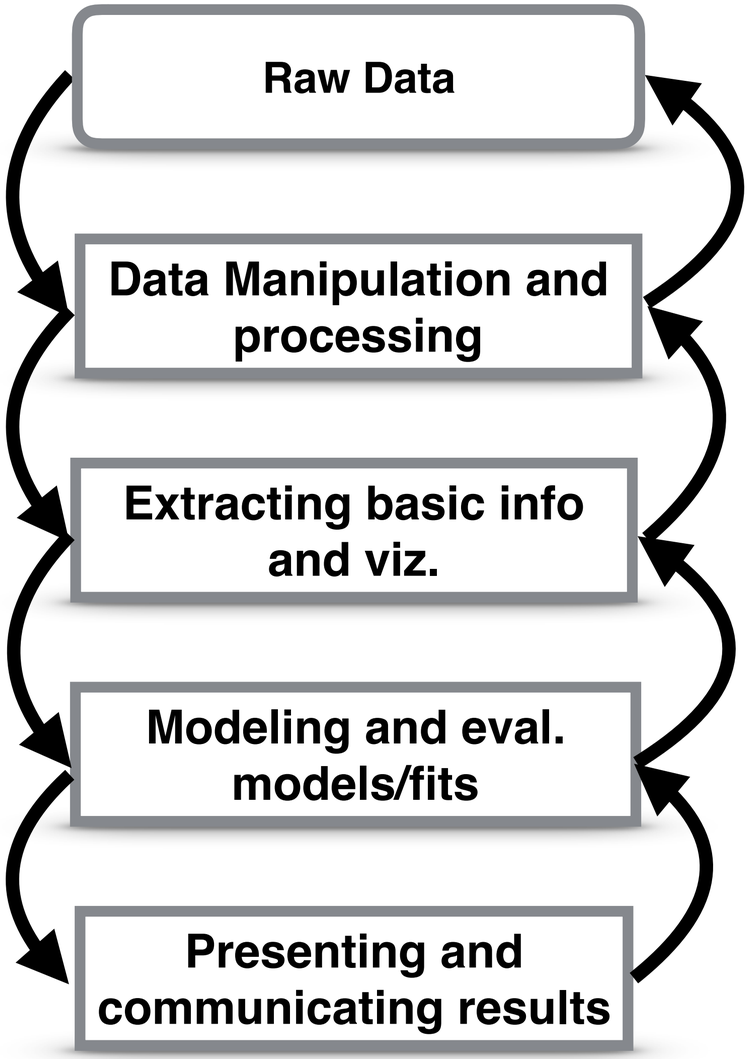
In case the angle = π (180^o) we get a zero length set of vectors by our other interpolation method!

Given a scientific question and hypothesis, what is a typical modeling and data analysis path?

4 main parts to modeling and data analysis

Steps	***M.A.D. Topics***
Step 1	Data manipulation and processing
Step 2	Extracting basic information from data and visualizing that info
Step 3	Modeling the data and evaluating models, data fits
Step 4	Presenting and communicating results

Modeling and Data Analysis pathway



Data manipulation and processing

- We must import the data into our computational package (python, matlab, C code, R, SPSS, etc)
- We have discussed the challenges and strategies of wrangling and cleaning
- Raw data needs to be transformed into a computationally useful structure (rectangular)
- Raw data need to be cleaned and filtered - removing NaNs, missing data, inconsistencies
- At this stage we sometimes discover issues that require new data, or have insights that change the M.D.A. path

Extracting basic info and visualizing: Descriptive analysis

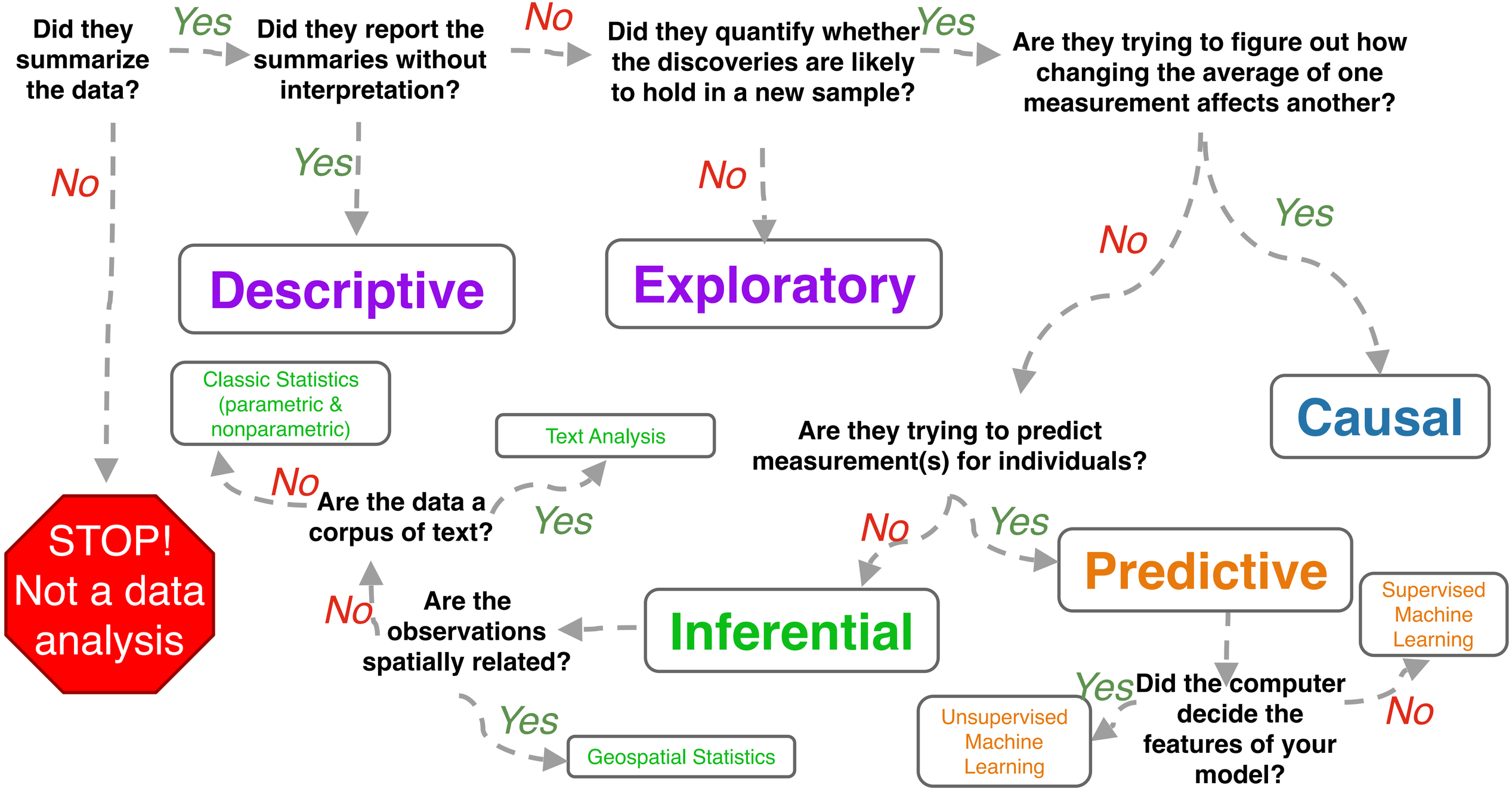
- Understand the data set's characteristics
- Describe what they are
- Explain that description to others so they can understand the data
- Filtering can be here as well
- Classical statistics without inferential conclusions (central tendency and variability), shape of distribution
- Basic visualizations (charts, plots, tables) to visualize the data

Modeling and evaluating model fits

- EDA gives insight into which model set to select - you don't know during EDA what final model you will be choosing
  - Shines the light on the road
- Regression, linear and nonlinear, multivariate regression, curve fits, error analysis (interpolation, approximation)
- Black box, grey box, and white box modeling (approximation, extrapolation)
  - Parameterizing model structures and fitting them (linear/nonlinear, ANN, dynamic/static, kinematic, feedback/feedforward, decision tree
  - ML is one way to fit them
  - Error analysis and assessing goodness of fit (error criteria - RMSE, simple error, etc)

Presenting and communicating results

- You summarize what you did
- You interpret the results of your study -
  - *Do you reject or fail to reject the null?*
  - *What is the significance?*
  - *What are the implications?*
  - *What are the limitations of the data, modeling and analysis?*
  - *Where would you go from here (future work)?*
- Share with the world as papers, software, products, talks, seminars, workshops, courses, etc
- Do you go back up the chain and redo, get new data, perform more modeling, etc? Or do you have an idea for another study?


Did they summarize the data?
Yes
Did they report the summaries without interpretation?
No
Did they quantify whether the discoveries are likely to hold in a new sample?
Yes
Are they trying to figure out how changing the average of one measurement affects another?
No
Yes
Yes
Descriptive
No
Exploratory
No
Yes
Causal
Classic Statistics (parametric & nonparametric)
Text Analysis
Are they trying to predict measurement(s) for individuals?
No
Are the data a corpus of text?
Yes
No
Yes
STOP! Not a data analysis
No
Are the observations spatially related?
Inferential
Predictive
Supervised Machine Learning
Yes
Geospatial Statistics
Yes
Unsupervised Machine Learning
Did the computer decide the features of your model?
No

Summary: Analytical Approaches

1. Descriptive (and Exploratory) Data Analysis are the first step(s) of analysis
2. Inference establishes relationships
    a. Classic Statistics
    b. Geospatial Analysis
    c. Text Analysis
3. Machine Learning and modeling is for prediction
    a. Supervised
    b. Unsupervised
4. Experiments best way to establish the likelihood of causality
    a. Remember you ***cannot*** establish causality with computational methods only correlations along with statistical beliefs
    b. More you are establishing if they are NOT related or ‘NOT NOT’ related

Approximating a curve

- Recall that in regression we want to approximate data with a curve
- The curve can be linear (affine), or nonlinear (polynomial or other shapes)
- Let's go back over regression quickly from the linear perspective then expand to parametrically linear fits of nonlinear functions
- We looked at the linear algebra approach, let's back up to the partial differential equation method (less abbreviated, but might be more clear for those less familiar with linear algebra)

Linear regression revisited

- Let us begin by considering a set of pairs of x and y data points
- The concept of least squares is to fit a linear or nonlinear curve which fits that data the best according to some criterion.
- One such common criterion is the minimization of sum of the squared differences between the actual data and the predicted data

Error criterion

- Thus the error is given by

$$E = \sum_{i=1}^{M} [y_i - y_{LS}(x_i)]^2$$

- Where i = 1,2,...M is the number of data points, and y_{LS} is the approximating curve's predicted y at the point x_i.

- There are several reasons why we square the error and then find the minimum of that function.

  - One of the most notable is to create a function with an attractive bowl shape and a single global minimum with respect to each line fitting parameter

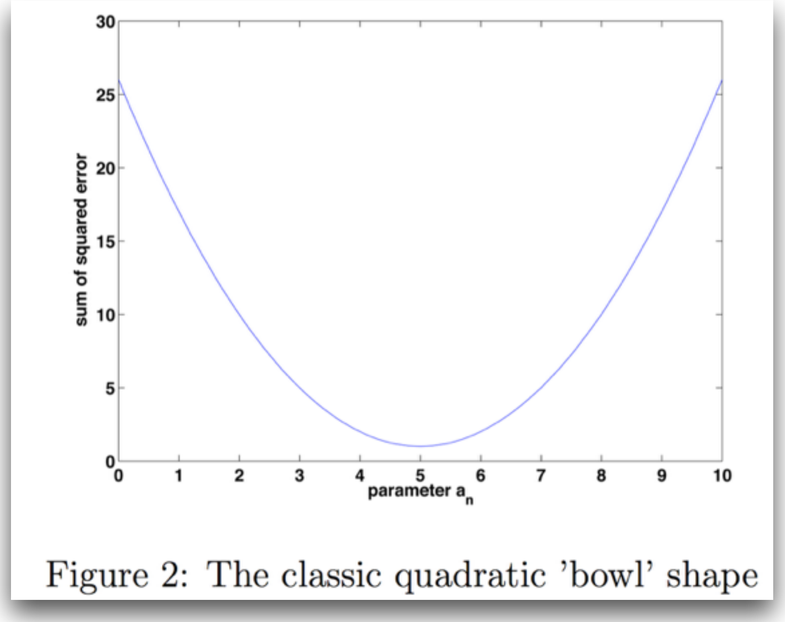


Figure 2: The classic quadratic 'bowl' shape

Error criterion

- This serves to make positive and negative errors all positive
- i.e. if y is bigger than yLS at point i, the un-squared error is positive, and if yLS is bigger than y at point i, the un-squared error would be negative- but if we square the error, negative numbers become positive, so we measure the magnitude of the error
- A larger negative error is still a bigger error!
- Thus by looking at the sum of the squared error we know when we are reducing error (making our fit better) or increasing error (making our fit worse) as we vary our parameters
- We can define a measure of how to find the ’best fit.’

If we want to fit a line, we have only two parameters, a_0 and a_1 to fit the line defined by

$$y_{LS} = a_0 + a_1 x \tag{2}$$

Thus our error equation becomes, after substituting y_{LS} in,

$$E = \sum_{i=1}^{M} [y_i - (a_0 + a_1 x_i)]^2 \tag{3}$$

To find the parameters a_0 and a_1 which minimize the error E, we take the partial derivative with respect to each parameter, a_0 and a_1 and set each resulting equation equal to zero.

$$\frac{\partial E}{\partial a_0} = 0 \tag{4}$$

$$\frac{\partial E}{\partial a_1} = 0 \tag{5}$$

$$\frac{\partial E}{\partial a_0} = \frac{\partial}{\partial a_0} \left\{ \sum_{i=1}^{M} [y_i - (a_0 + a_1 x_i)]^2 \right\} \tag{6}$$

$$\begin{bmatrix} M & \sum_{i=1}^{M} x_i \\ \sum_{i=1}^{M} x_i & \sum_{i=1}^{M} x_i^2 \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix} = \begin{Bmatrix} \sum_{i=1}^{M} y_i \\ \sum_{i=1}^{M} y_i x_i \end{Bmatrix} \tag{16}$$

Which, if we set $A = \begin{bmatrix} M & \sum_{i=1}^{M} x_i \\ \sum_{i=1}^{M} x_i & \sum_{i=1}^{M} x_i^2 \end{bmatrix}$, $x = \begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix}$, and $b = \begin{Bmatrix} \sum_{i=1}^{M} y_i \\ \sum_{i=1}^{M} y_i x_i \end{Bmatrix}$ is equivalent to

$$Ax = b \tag{17}$$

Solving for the parameters

- Note that the A, x, and b are not the same variables as the x's and a's in the above equations.
- We can solve this problem by finding the inverse of A (A^{-1}), and multiplying both sides by A^{-1}, as we did last time
- This gives us the parameters of a_0 and a_1 $$x = A^{-1}b$$

Python implementation

- https://numpy.org/doc/stable/reference/generated/numpy.linalg.lstsq.html
- In discussion we are going to take this further, along with more tools that are even easier and are powerful
- We want you to understand the steps and what is happening

But what about nonlinear systems?

- Not all processes are linear (in fact most real processes are not linear).
- It is therefore more appropriate in such cases to have nonlinear data fitting methods.

Polynomial fits

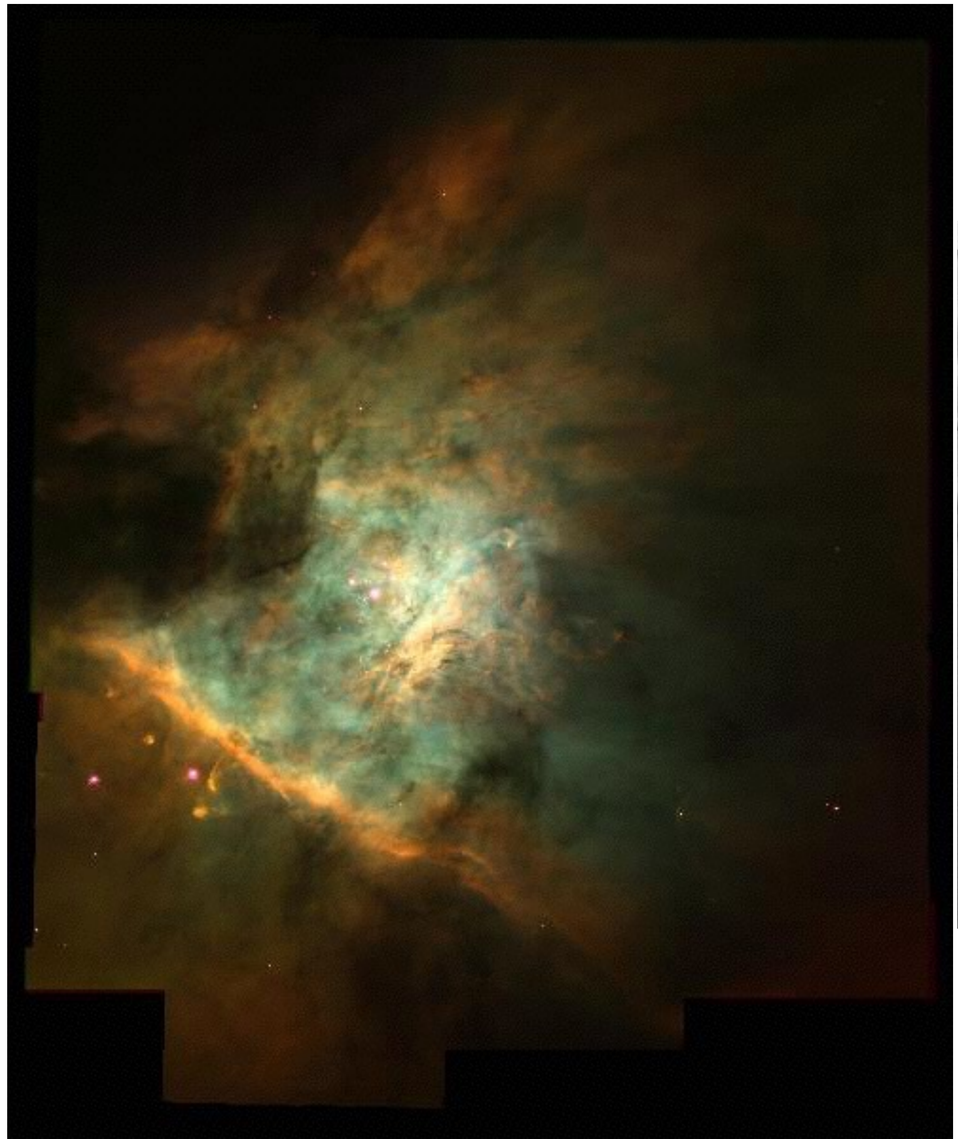
One basic nonlinear function is a polynomial. Consider that we have a set of data $\{x_i, y_i\}$ with $i = 0, 1, ...M$ data points, and we would like to fit a polynomial of order n. The general equation for a polynomial is given by

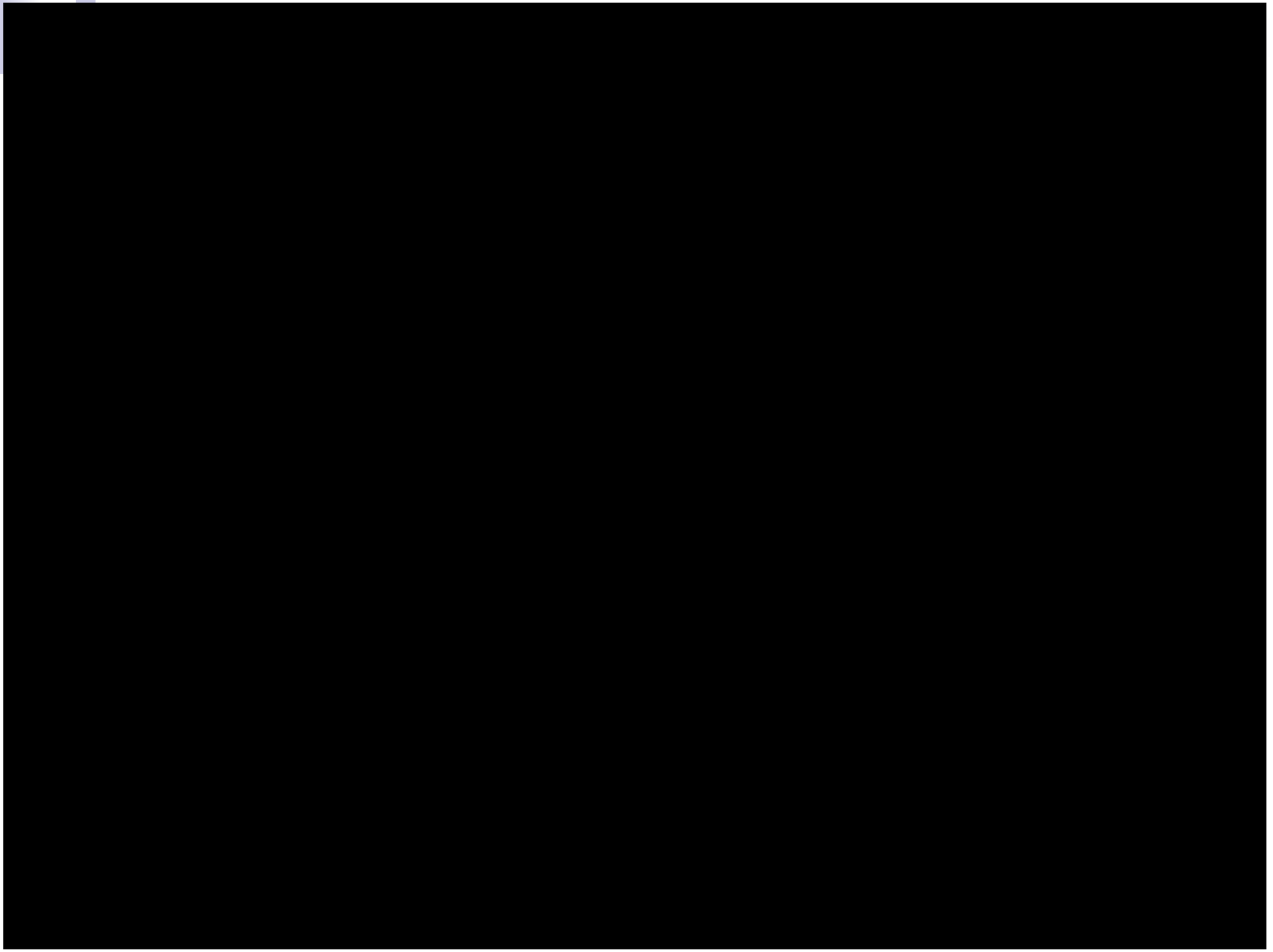
$$P_n(x) = a_0 x^0 + a_1 x^1 + a_2 x^2 + ... + a_n x^n \tag{22}$$

or more compactly written as

$$P_n(x) = \sum_{k=0}^{n} a_k x^k \tag{23}$$

Another SDSC example: the orion nebula animation





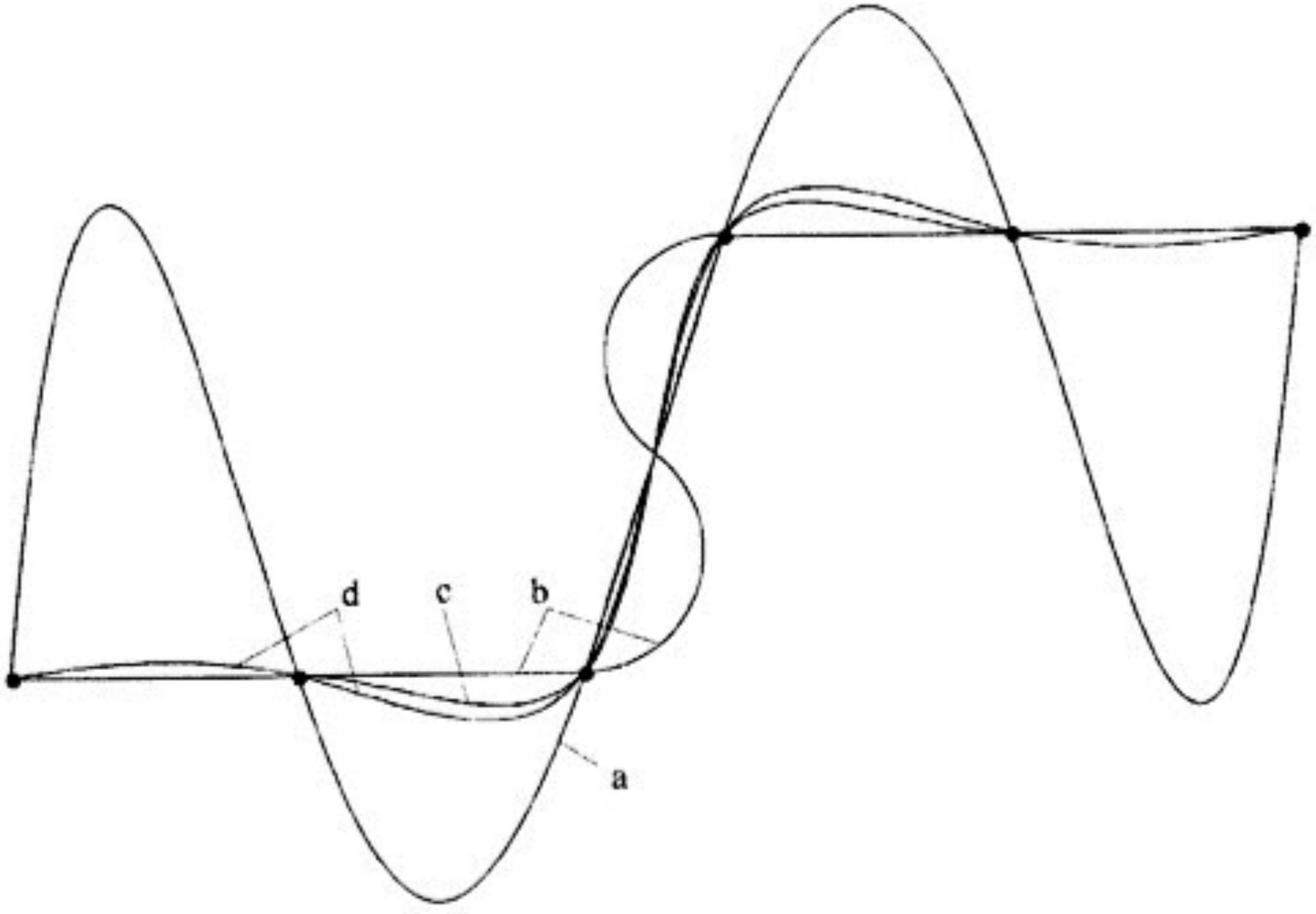


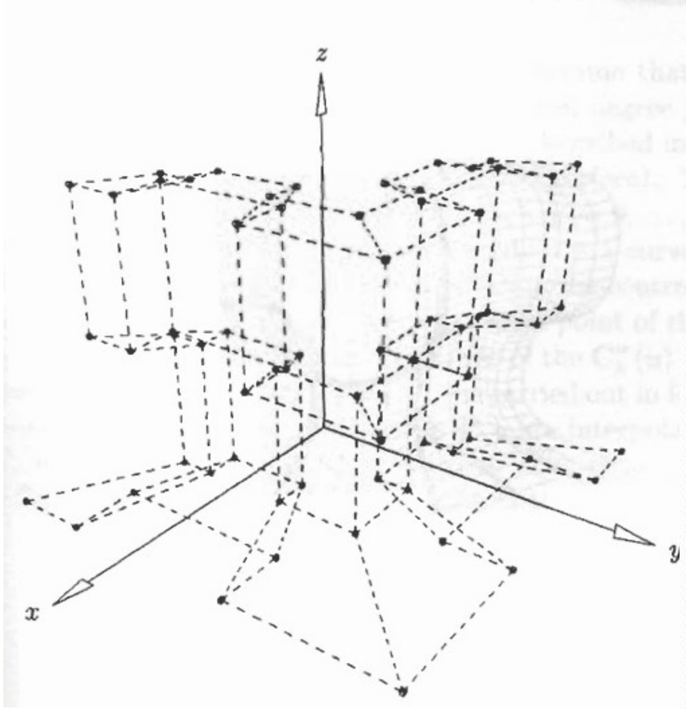
Figure 7.1 Interpolation curves drawn for six vertex points (dots), with y plotted in the vertical and x in the horizontal direction. Curves are shown for (a) a high-order polynomial fit, (b) a circular-arc fit, (c) a parabolic blend, and (d) a natural cubic spline.

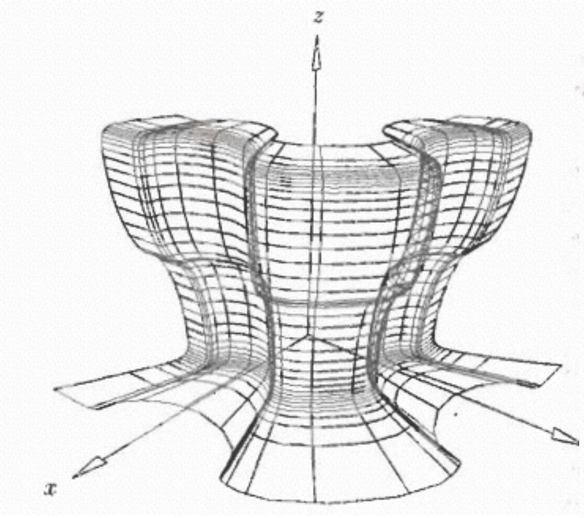
Today we'll develop methods of nonlinear interpolation and extrapolation

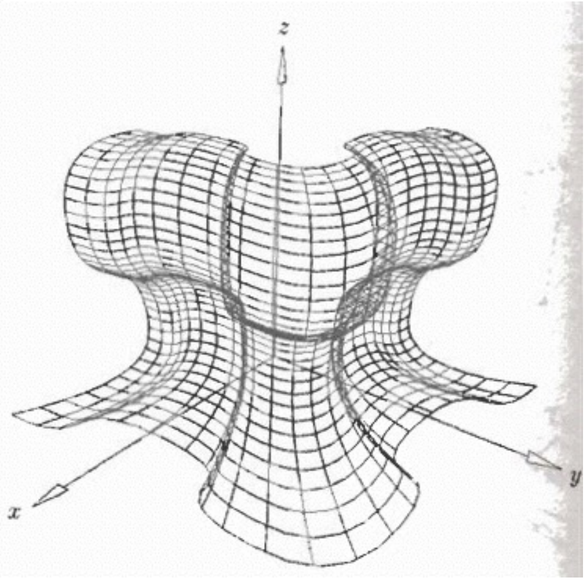
- Lagrange
  - Useful for low number of data points
  - Unstable for high numbers of data points

- Splines
  - There are many kinds discussed in the reading, we'll just discuss one today
  - Good overall method
  - Works with many or few data points

Splines also give you control over the final outcome of the curve

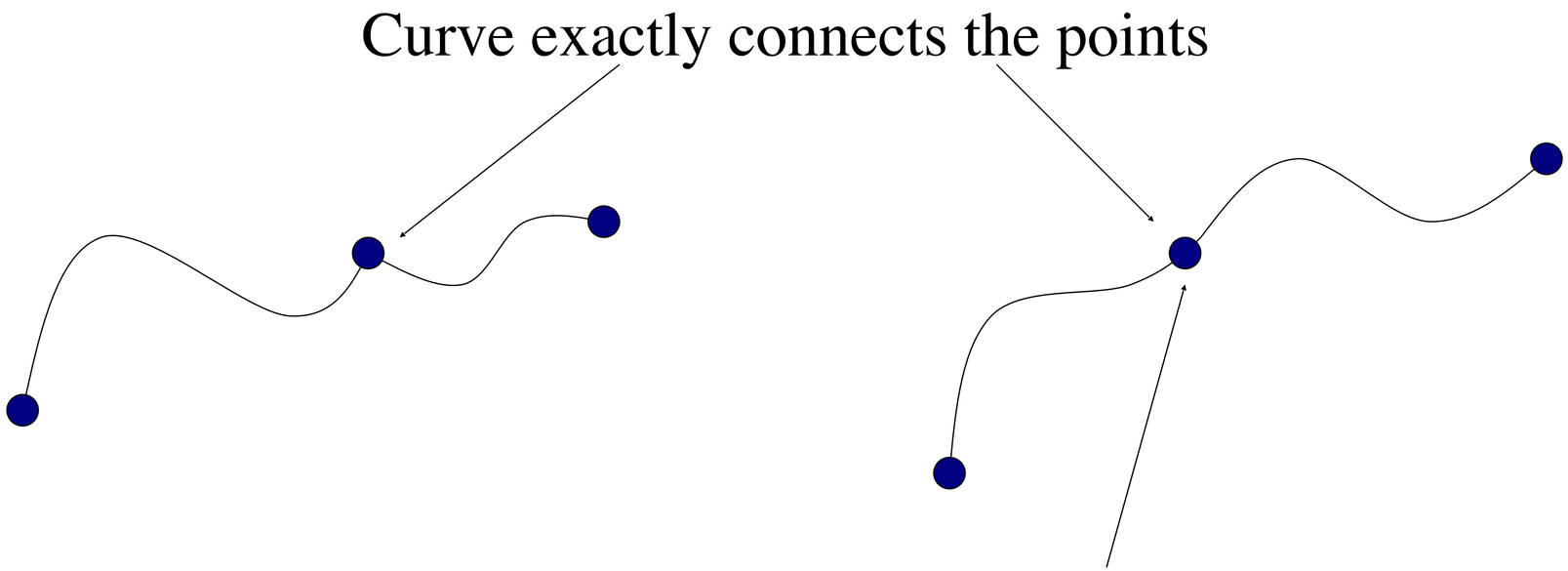






Adding constraints to solve for the unknowns

- Continuity at the joints:



Natural Cubic Splines

- We fit another parametric curve (similar to LERP), with a value of t from 0-1 again and make the ith segment according to

$$Y_i(t) = a_i + b_i t + c_i t^2 + d_i t^3$$

- And we solve for each set of these constants by requiring continuity at the end points (one section smoothly flows into the next, and the slope must match as well)

$$Y_i(0) = y_i = a_i$$

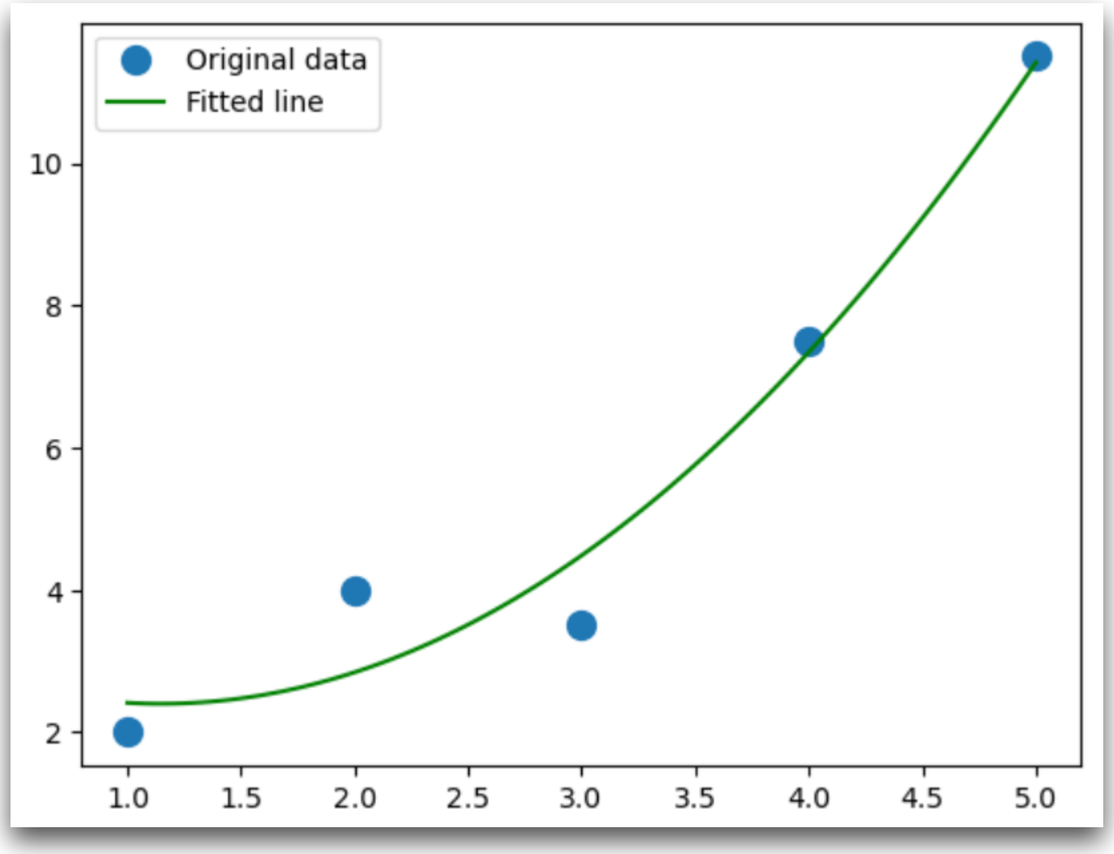
$$Y_i(1) = y_{i+1} = a_i + b_i + c_i + d_i$$

$$Y_i'(0) = D_i = b_i$$

$$Y_i'(1) = D_{i+1} = b_i + 2c_i + 3d_i$$

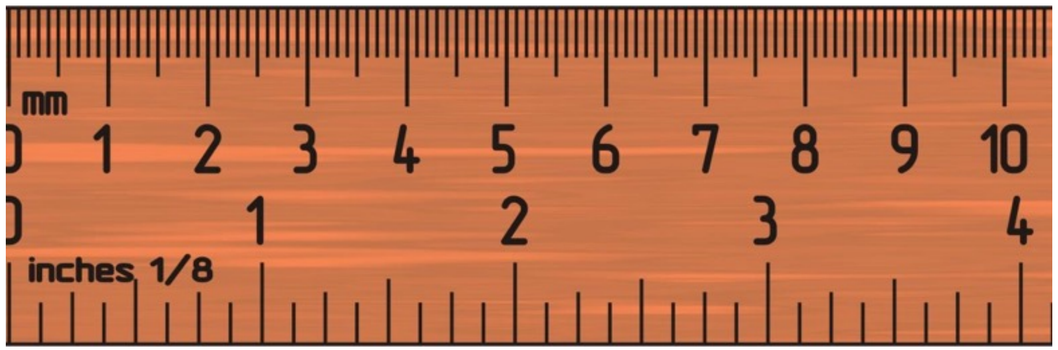
Back to error analysis

- We need to assess the quality of our fit
- Is this any ‘good?’



Uncertainty

- Error does not mean, in science, mistake
  - **It means the level of uncertainty in measurements and calculations**
  - **Can’t eliminate by being careful, must instead minimize them**
- Basically want to have an estimate which is as reliable as possible
  - **‘keep an eye on’ your uncertainty**



The question…

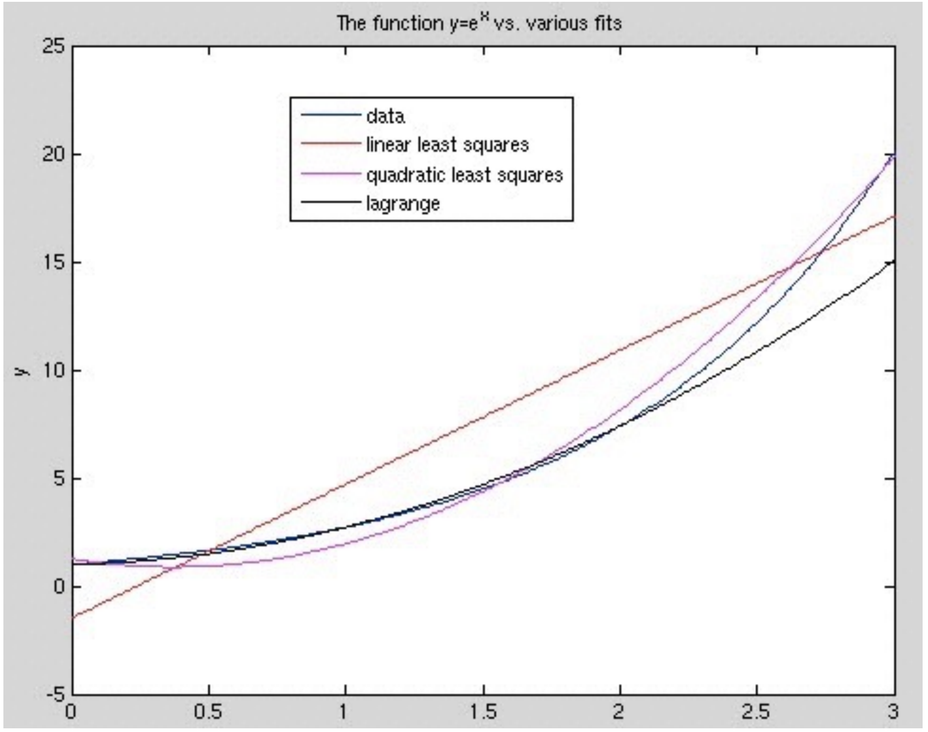
- The question is not whether you are right or not
- The question is whether your approximation is good enough to be useful, dependent on what you consider to be ‘good enough’

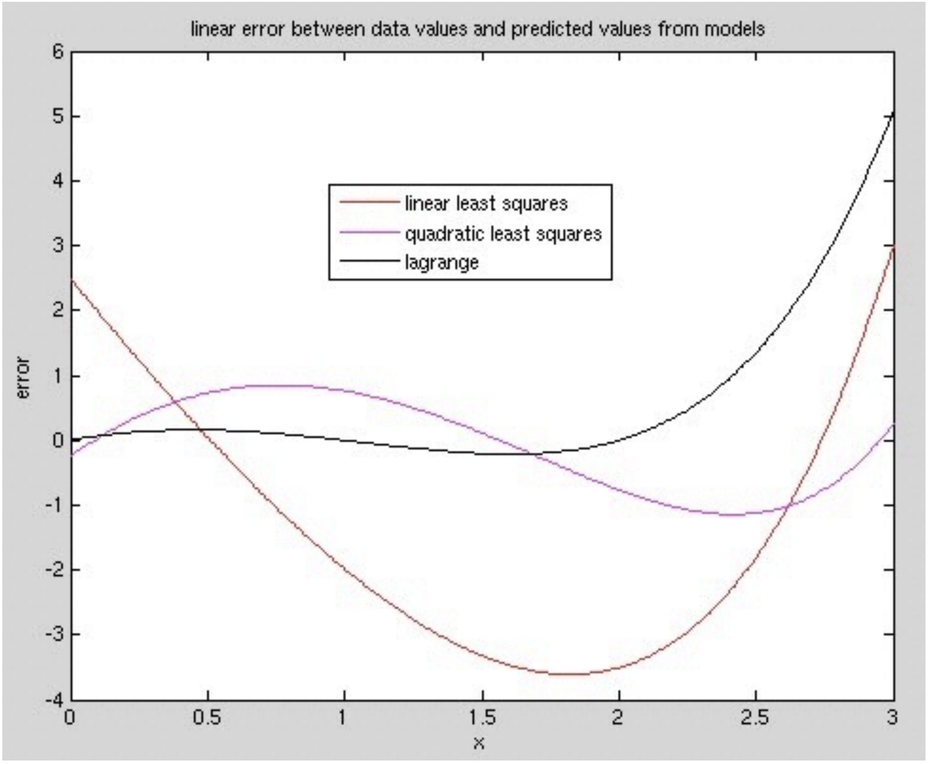
Different error estimates

- There are many ways to estimate errors, here are a couple of common ones
  - To get a single # - can use various norms $\|e\|_2 = \sqrt{\sum_i (y_i - \hat{y}_i)^2}$
    - 2-norm
    - Mean-squared-error $MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$
  - Curve - simple error (for a time dependent signal y(t)) $e(t) = y(t) - \hat{y}(t)$
  - Curve - prediction error $e_p(t) = y(t) - \hat{y}(t \mid t-1)$

Assessing the models

- I assess how well each model fit does by first plotting the error between the data and the different methods
- Then I plot the real function (or data) vs. the different methods along a continuous curve

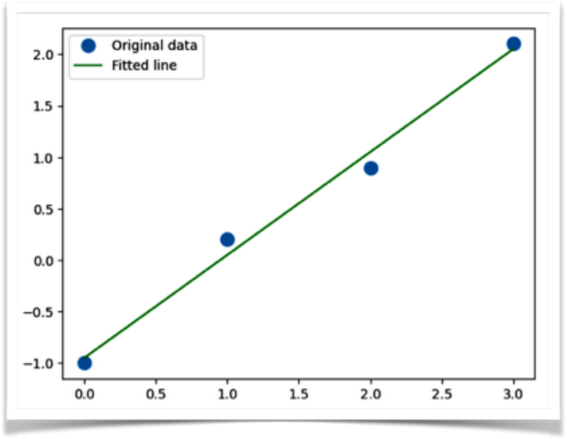

The function y=e^x vs. various fits
data
linear least squares
quadratic least squares
lagrange
y
25
20
15
10
5
0
-5
0
0.5
1
1.5
2
2.5
3

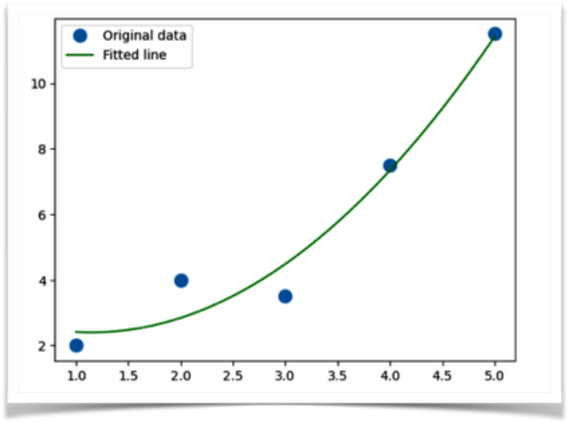

linear error between data values and predicted values from models
linear least squares
quadratic least squares
lagrange
error
x
6
5
4
3
2
1
0
-1
-2
-3
-4
0
0.5
1
1.5
2
2.5
3

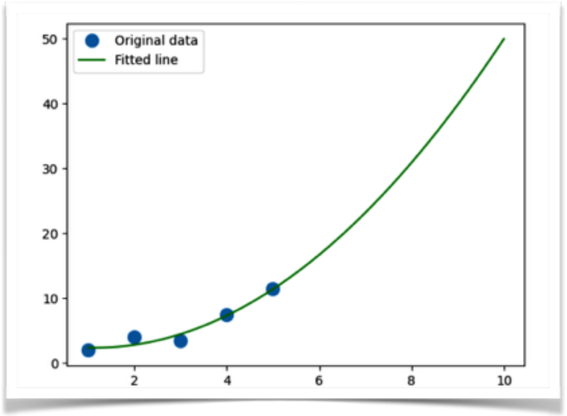
We can also compute the error as a single quantity

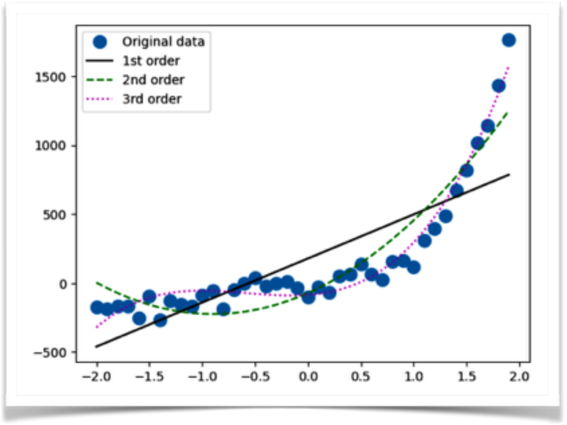
- e2lls = 125.7192
- e2nlls = 10.9367
- e2lag = 86.3331
- From this we see that over this interval, the nonlinear least squares polynomial fits the data the best if we're trying to minimize this error as a criterion for goodness of fit
- Again it depends on our criterion, as the lagrange has the lowest error over the domain of data used for computing the fit
  - **It doesn't extrapolate the future points as well in this case**

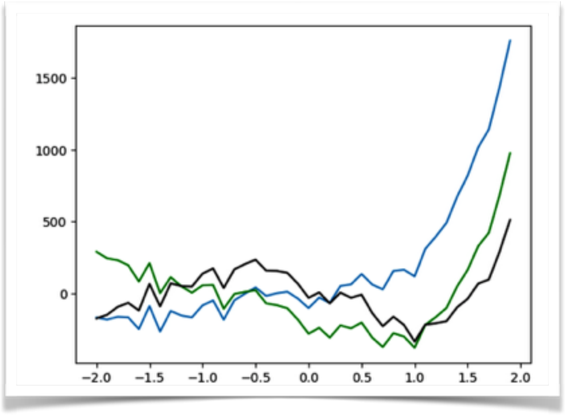
Types of least squares regression - in class development











Optimization is a popular way to study the human brain, behavior and computation

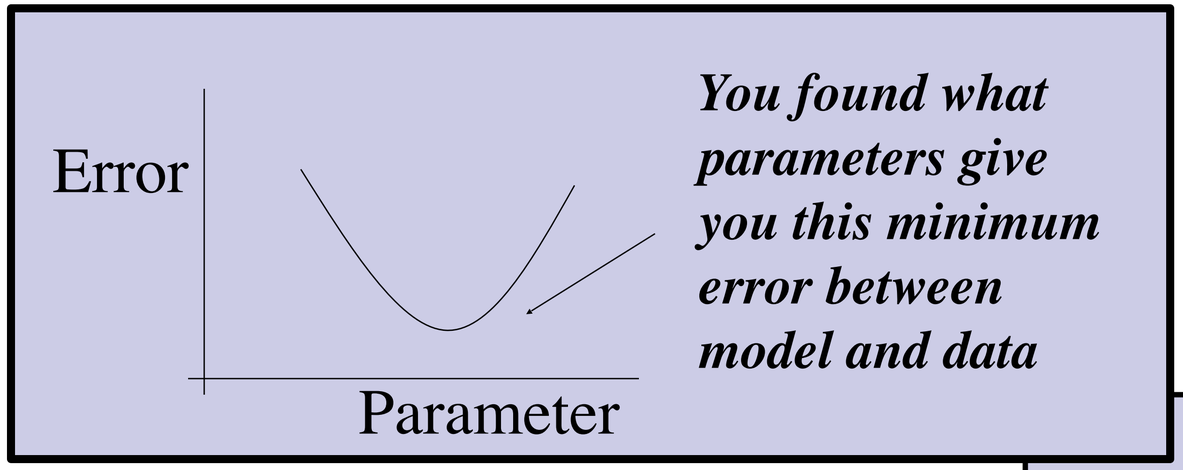
- There is a tremendous amount of interest in optimization and optimality in general in fields studying human cognition and behavior, such as Cognitive Science
  - **For model fitting in general**
  - **But also because it is intuitive to understand many aspects of human behavior in terms of optimization**

Additional practical applications

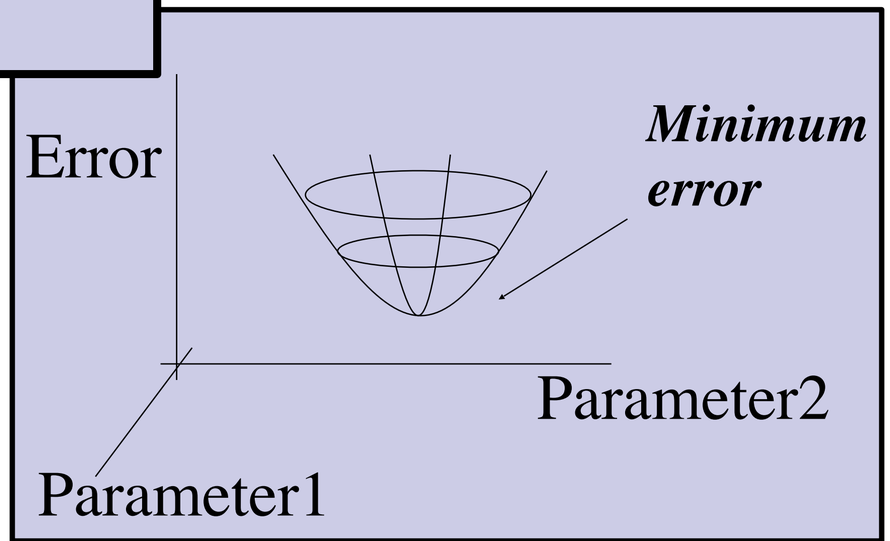
- Optimal control in human movement,
- Optimization of energy usage in society,
- Optimizing storage in hard drives to make information faster to access,
- Optimization in education to improve human learning,
- Optimization in sports to improve performance (power lifting, running, swimming, jumping, throwing, etc)
- Optimization in design to create objects that have reduced wind resistance, are stronger, lighter, less expensive, use cheaper or less impactful materials
- Optimization in network traffic to make cell phones work
- Optimization for traffic flow in vehicles

Graphical view of function minimum

One parameter



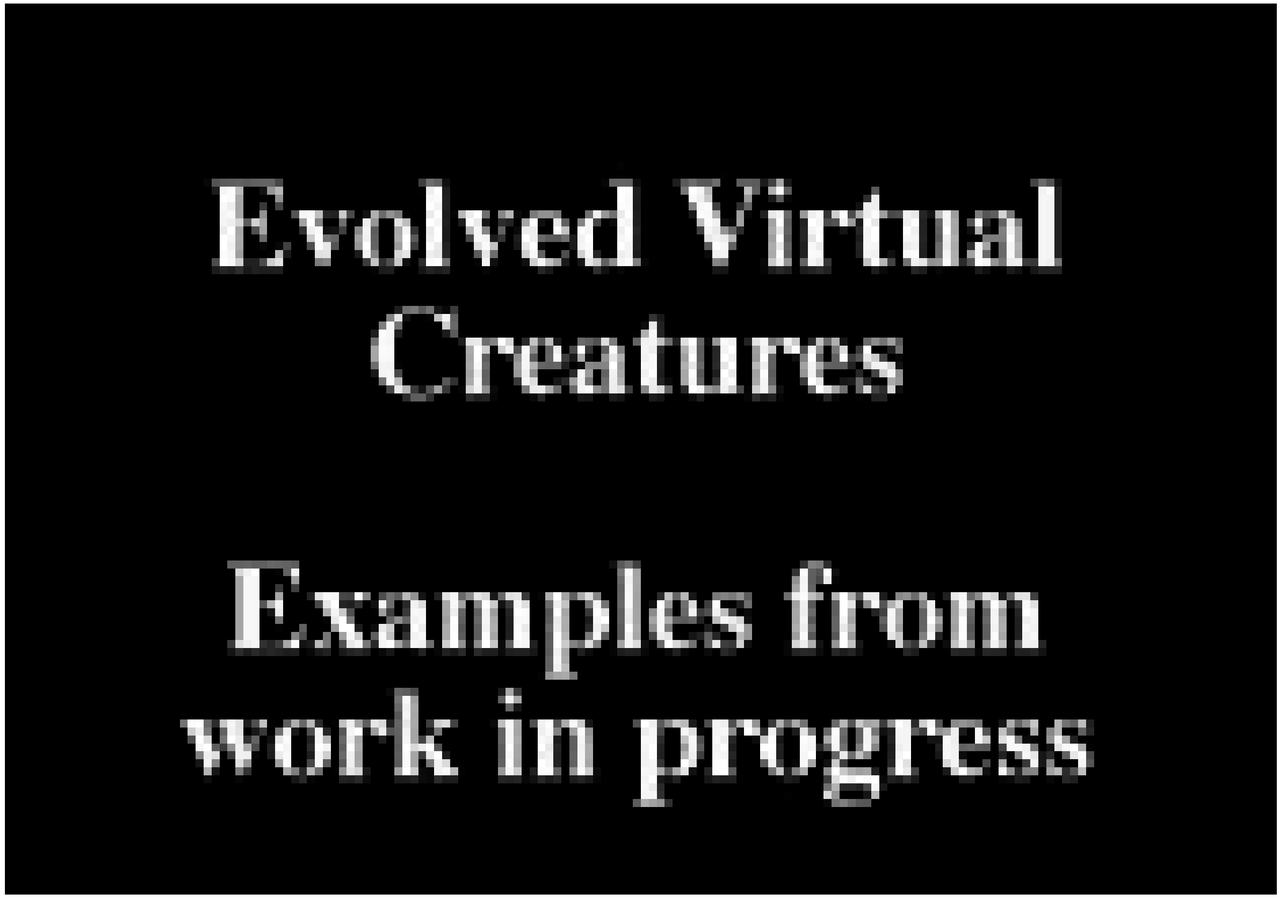
Two parameters



Remind me again, what exactly are we ‘minimizing’ or ‘maximizing?’

- Minimize cost
- Maximize reward
- We decide what that function is (‘cost function’ or ‘reward function’)
  - **Then have some unknown constants**
  - **Then we use these methods to find the constants**
  - **Those constants give us the smallest cost or largest reward function**
    - Can be then interpreted as the ‘best fit’ given a definition of what ‘goodness’ is

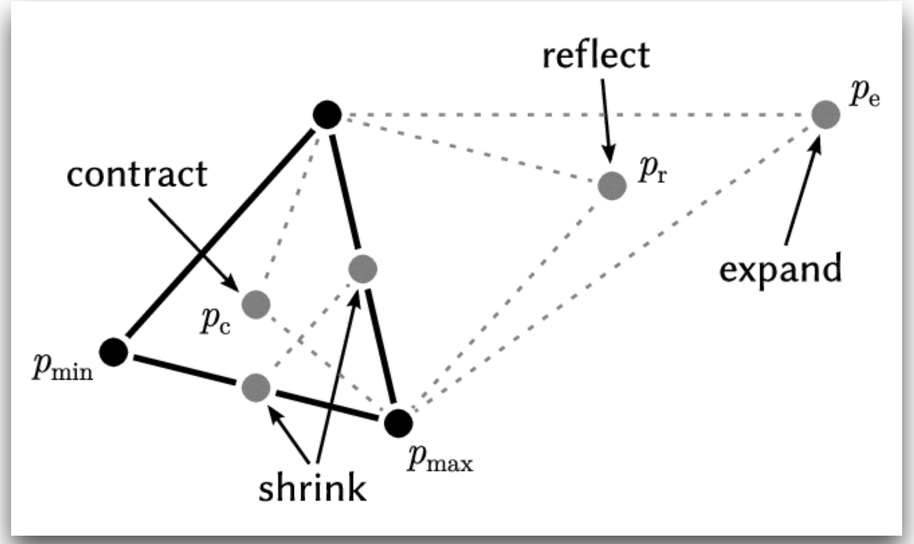
Graphical example - evolving organisms optimize cost, maximize rewards



http://www.karlsims.com/evolved-virtual-creatures.html

Nelder-mead simplex method

- Built into python's scipy, and matlab's optimization toolbox
- Simple to implement
- How does it work?
  - **https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html#scipy.optimize.minimize**
  - **Lagarias, J.C., J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions," SIAM Journal of Optimization, Vol. 9 Number 1, pp. 112-147, 1998.**

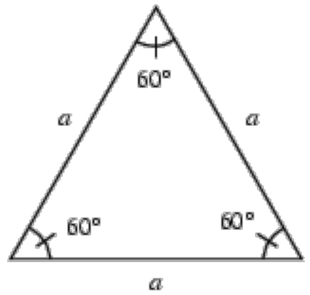


https://upload.wikimedia.org/wikipedia/commons/7/72/An-iteration-of-the-Nelder-Mead-method-over-two-dimensional-space-showing-point-p-min.png

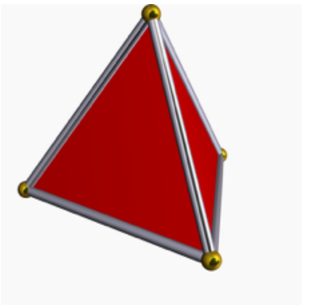
What does a simplex look like?

- Think of it as an N-Dimensional triangle
  - "simplest possible polytope (a **polytope** is a geometric object with flat sides) in any given dimension"[wikipedia]
  - For specifics, start by reading *mathworld* and *wikipedia* definitions of ***simplex*** and related important details like ***convexity*** and ***convex hulls***:
    - ***http://en.wikipedia.org/wiki/Simplex***
    - ***http://mathworld.wolfram.com/Simplex.html***

1D ->line



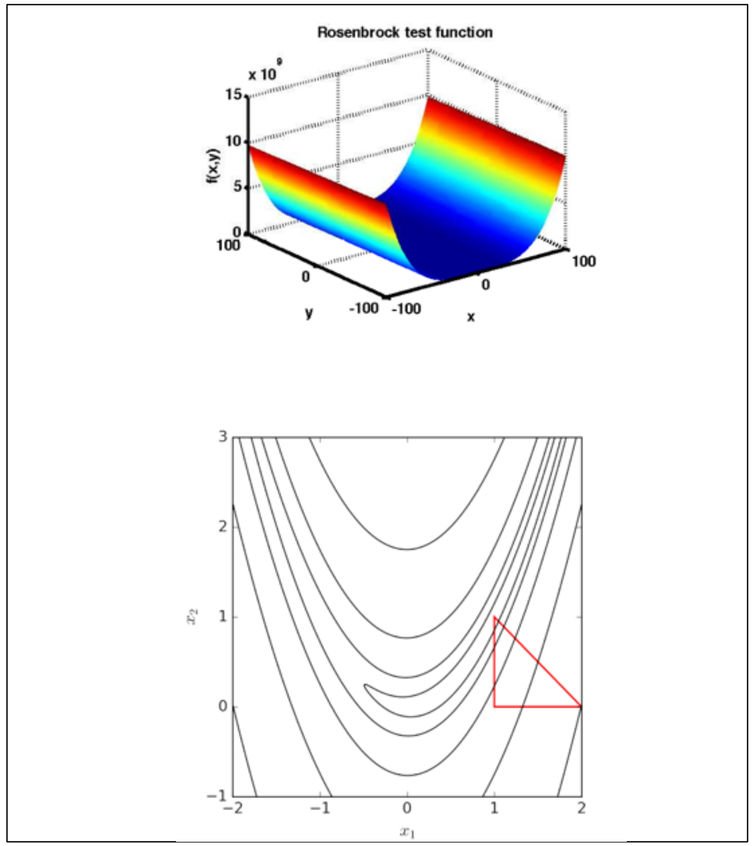
2D ->triangle

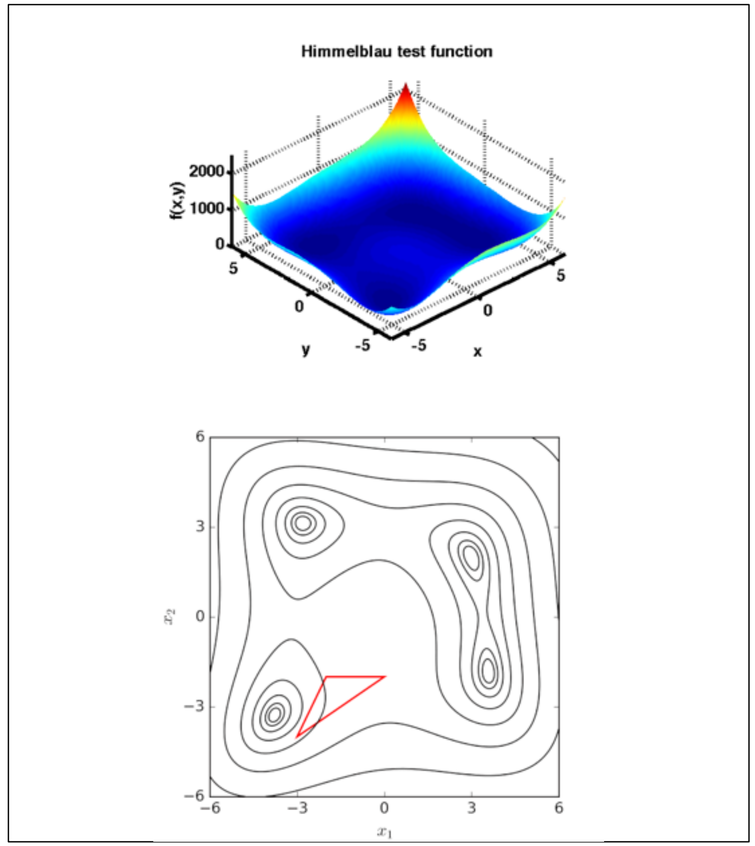


3D ->tetrahedron

About the Nelder-Mead Simplex algorithm

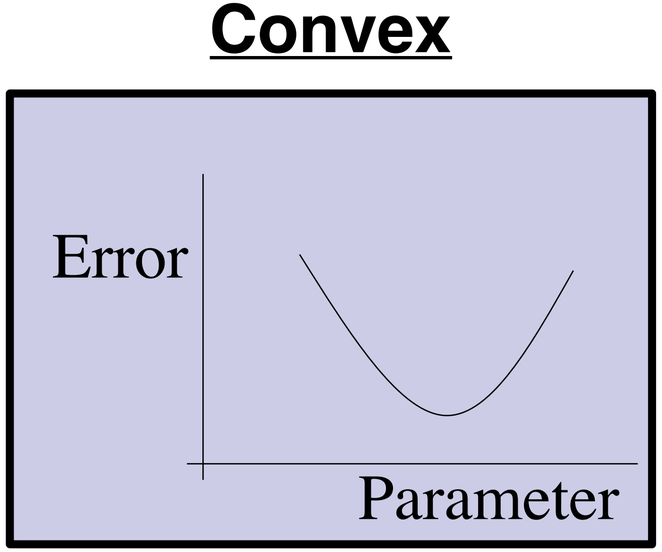
- So we showed examples of the NM algorithm and implementation details in python or matlab





Convexity and convex problems

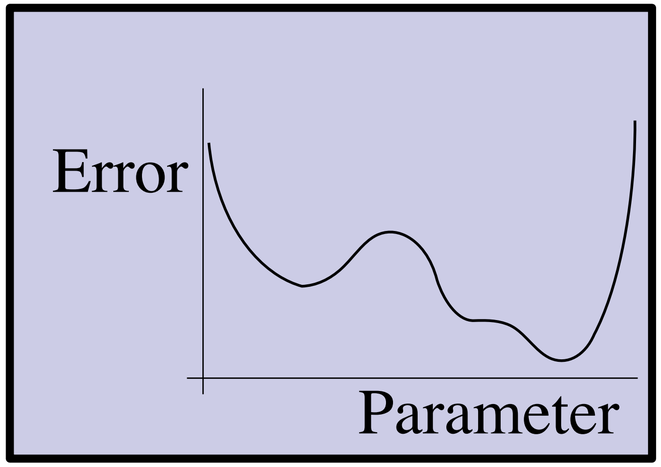
- Convex functions have one global minimum and no additional local minima
  - They can still be hard to minimize though - like for the Rosenbrock function
  - There exist many techniques which rapidly converge to the solution of convex functions



Non-convexity and local minima

- **Non-convex** functions may have multiple ***local minima*** which are not anywhere near the ***global minimum***
  - **For example, the Himmelblau function**
  - **What can we do?**
    - Many strategies - it's hard to know what is the absolute global minimum when you can't explicitly compute it
      - **Can restart with multiple different initial conditions and see if you get the same minima**
      - **Global optimization is a whole branch of mathematics where one attempts to find deterministic algorithms guaranteed to converge to globally optimal solutions in finite time**
- Take home message - use any algorithm with caution and awareness

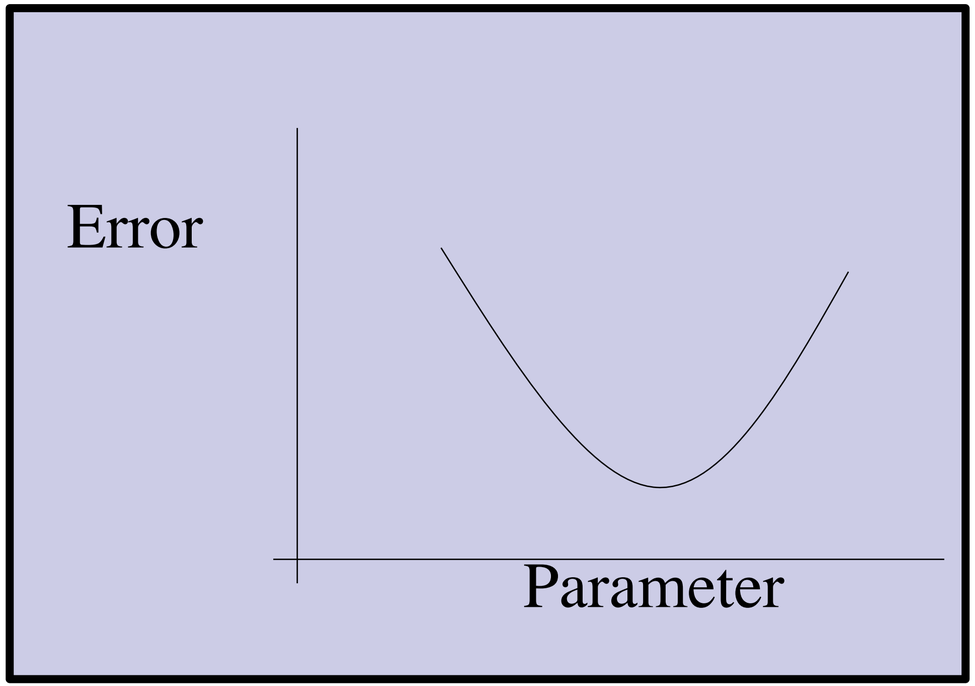
Non convex



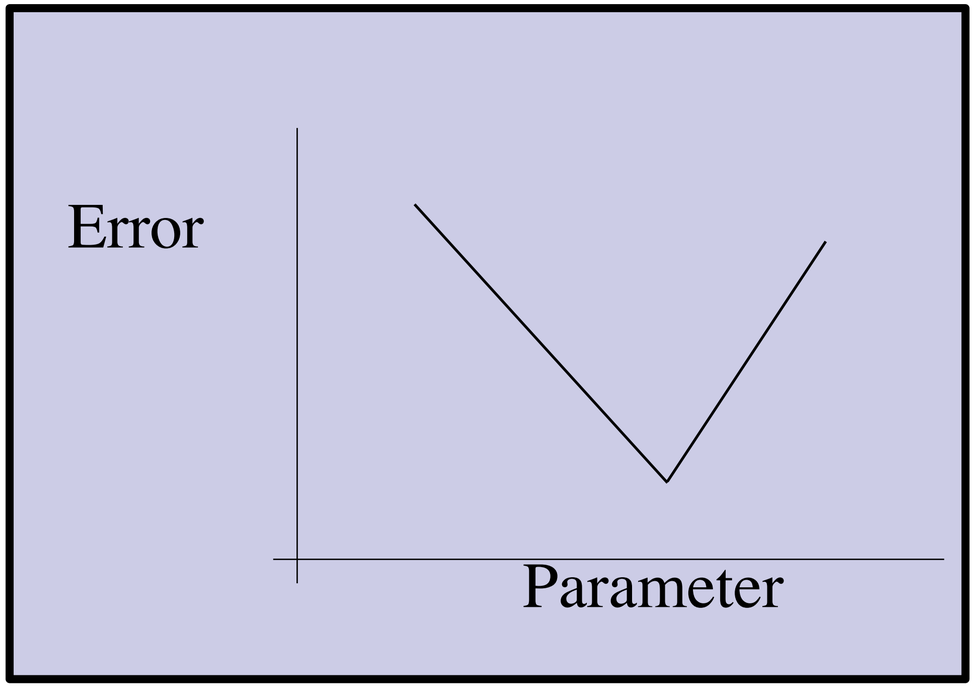
Smooth vs. Non-smooth problems

- Smooth is much easier - derivative is continuous everywhere
- Also

Smooth



Non-smooth



Constrained vs. Unconstrained

- **Constrained optimization** - Process of optimizing some function with respect to some variables subject to constraints on those variables
  - Constraints may be given that we need to satisfy may berange of values, boundary, rules such as shape of differentials, etc
  - Usually boundary - Equality or inequality constraints, such as:
  - (source: https://en.wikipedia.org/wiki/Constrained_optimization#:~:text=In%20mathematical%20optimization%2C%20constrained%20optimization,of%20constraints%20on%20those%20variables.)

A general constrained minimization problem may be written as follows:[2]

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{subject to} \quad & g_i(\mathbf{x}) = c_i \quad \text{for } i = 1, \dots, n \quad \text{Equality constraints} \\ & h_j(\mathbf{x}) \geq d_j \quad \text{for } j = 1, \dots, m \quad \text{Inequality constraints} \end{aligned}$$

where $g_i(\mathbf{x}) = c_i$ for $i = 1, \dots, n$ and $h_j(\mathbf{x}) \geq d_j$ for $j = 1, \dots, m$ are constraints that are required to be satisfied (these are called hard constraints), and $f(\mathbf{x})$ is the objective function that needs to be optimized subject to the constraints.

- **Unconstrained optimization** - solve the optimization function, no constraints or range imposed

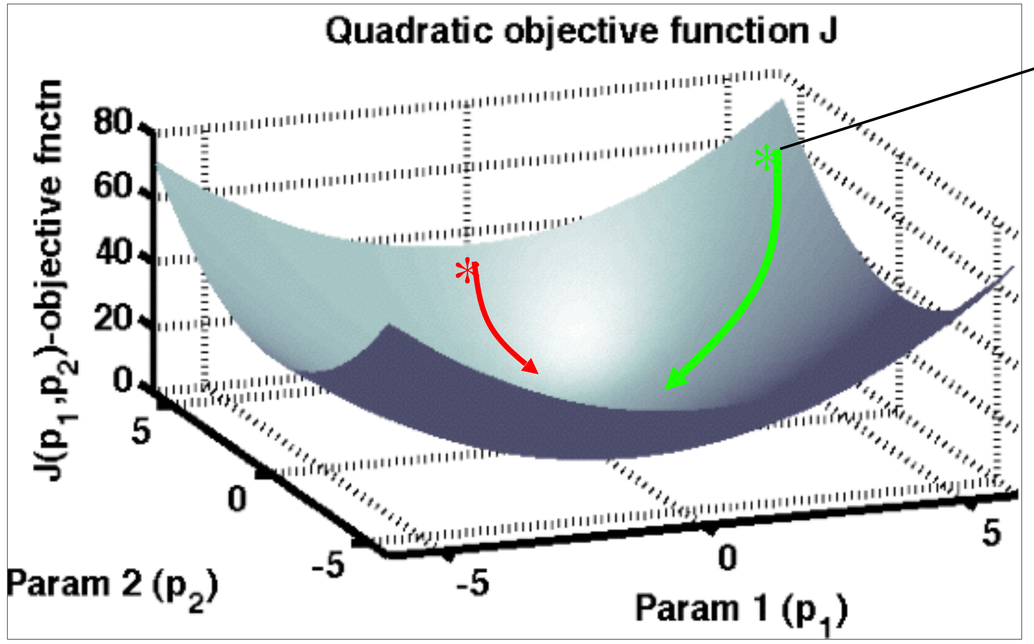
Why not just compute all the minima of a function over all the space of interest?

- You might not know the function!
  - **Think if I told you to find the lowest part of campus blindfolded and with your ears and sense of smell somehow ‘disabled’**
  - **You’d have to feel your way there, you couldn’t predict the final lowest point, if you had no prior knowledge**

How does the gradient descent algorithm work?

- Consider first the objective of gradient descent
  - **You want to get to the bottom of the hill**
  - **Start somewhere, then you ski down the hill**

$$J(\mathbf{x}) = \frac{1}{2}x^T Ax - b^T x$$

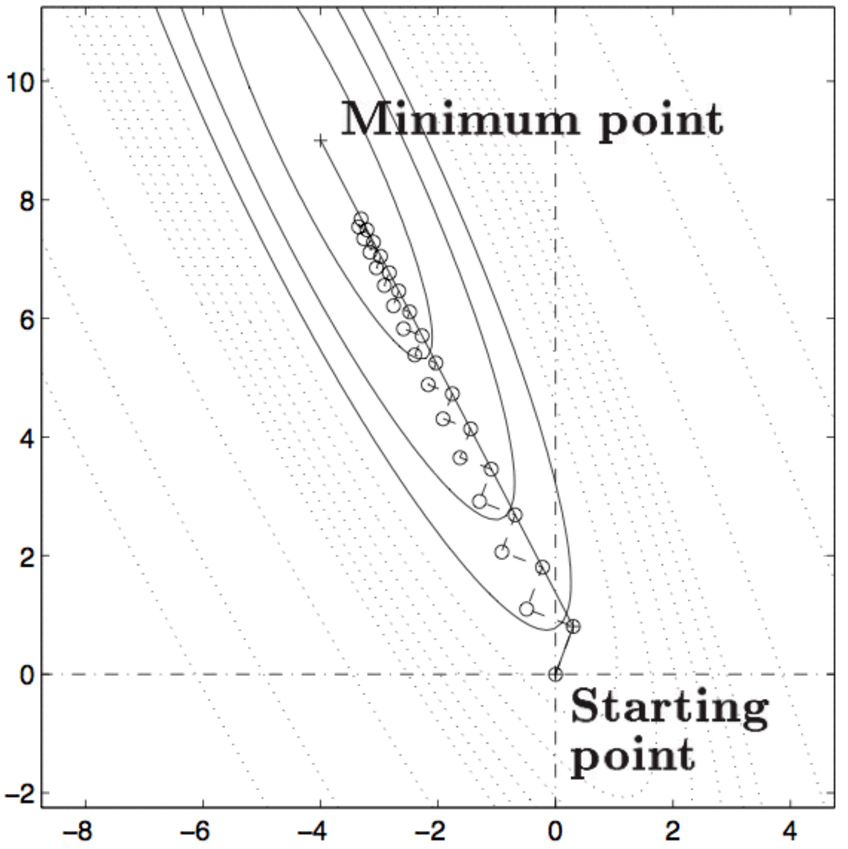


Start here,

Then move towards the minimum

But…

- There are issues with this method when the objective function is more challenging - with very steep sides and long flat valleys (*poorly conditioned*)
- This method also is a bit inefficient since it must ‘tack’ back and forth at 90 degree increments
  - **Due to successive line minimization and lack of momentum from one iteration to the next**
  - **THERE HAS TO BE A BETTER WAY!!!**



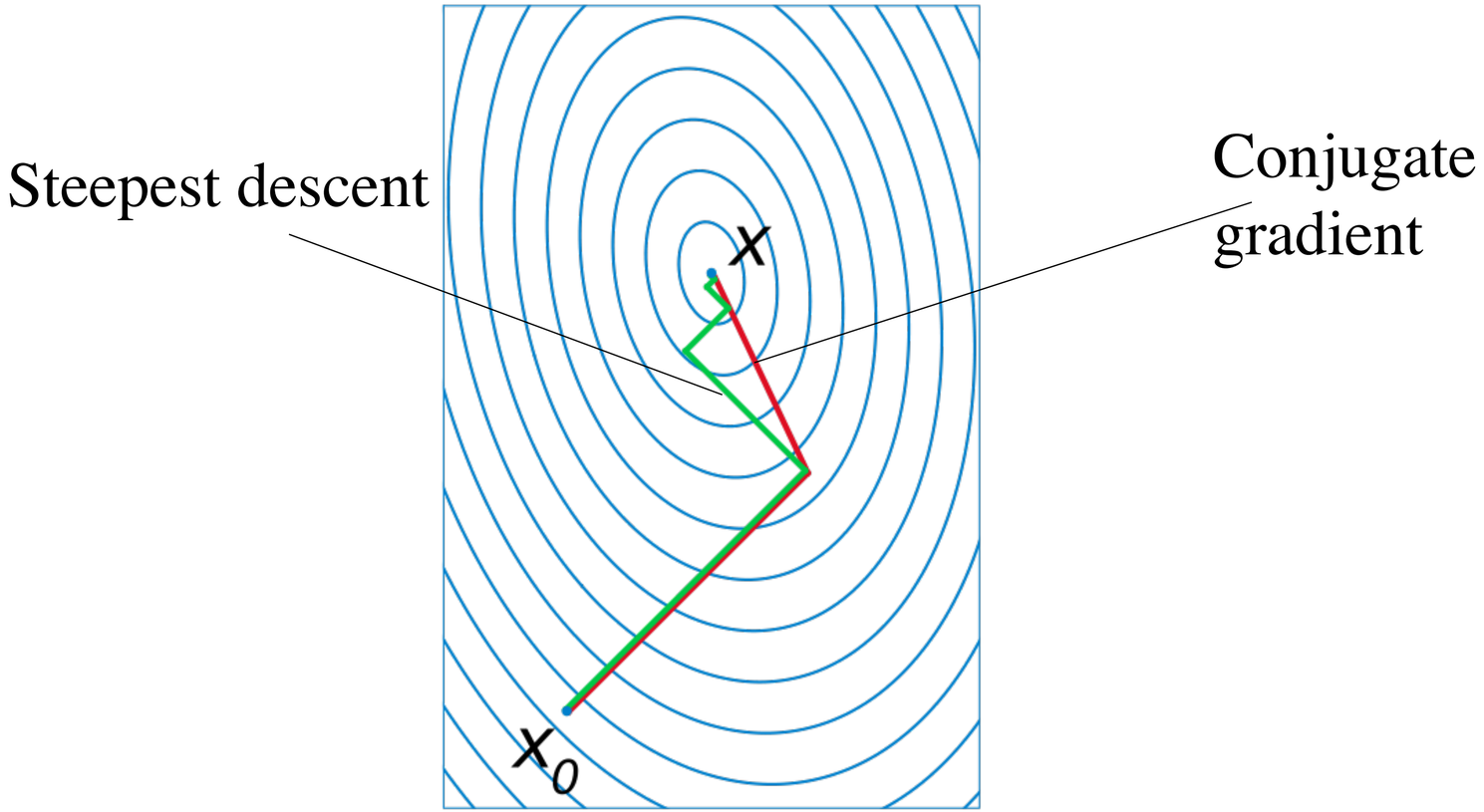
What does it boil down to?

- We compute a sequence of p's which are conjugate
  - **We redefine the descent direction at each iteration after the first to be a linear combination of the direction of steepest descent r and the previous descent direction**

$$\mathbf{p}^{(k)} = \mathbf{r}^{(k)} + \beta\,\mathbf{p}^{(k-1)} \qquad \text{and} \qquad \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha\,\mathbf{p}^{(k)}$$

$$\beta = \frac{{\mathbf{r}^{(k)}}^T \mathbf{r}^{(k)}}{{\mathbf{r}^{(k-1)}}^T \mathbf{r}^{(k-1)}}, \qquad \alpha = \frac{{\mathbf{r}^{(k)}}^T \mathbf{r}^{(k)}}{{\mathbf{p}^{(k)}}^T A\,\mathbf{p}^{(k)}}.$$

The result - an improvement



In summary

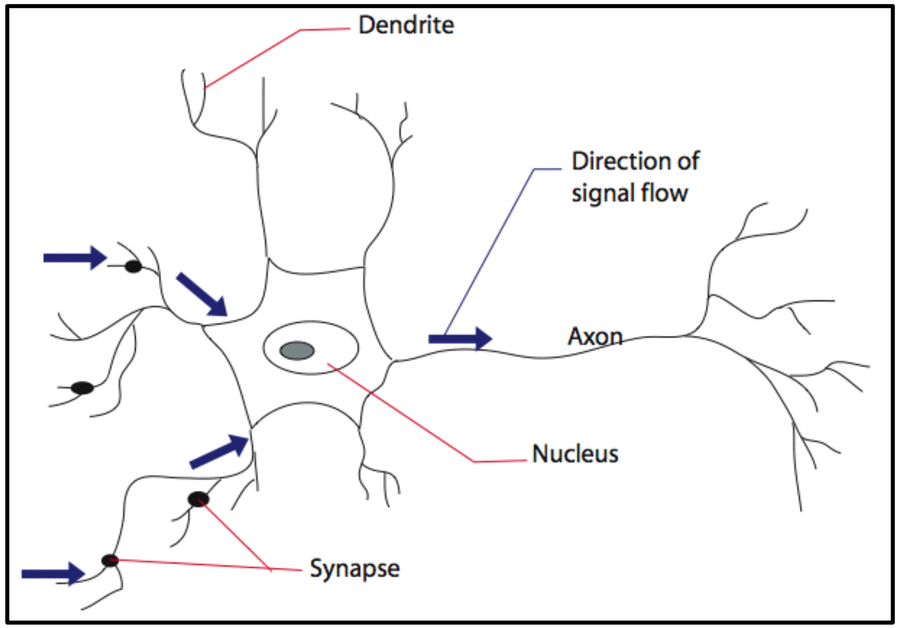
- **Optimization** is an interesting way to understand and model the world as well as solutions to difficult problems
- **Numerical optimization** provides tools for finding parameters for functions we could otherwise not solve for and that fail in simple regression type cases
- At times it provides a tool that is more efficient than solving the problem if solvable
- There are many approaches from simple to complex
- Simple solutions like NM and CGD are very practical
- **Reading: https://scipy-lectures.org/advanced/mathematical_optimization/index.html#smooth-and-non-smooth-problems**

But is there another way?

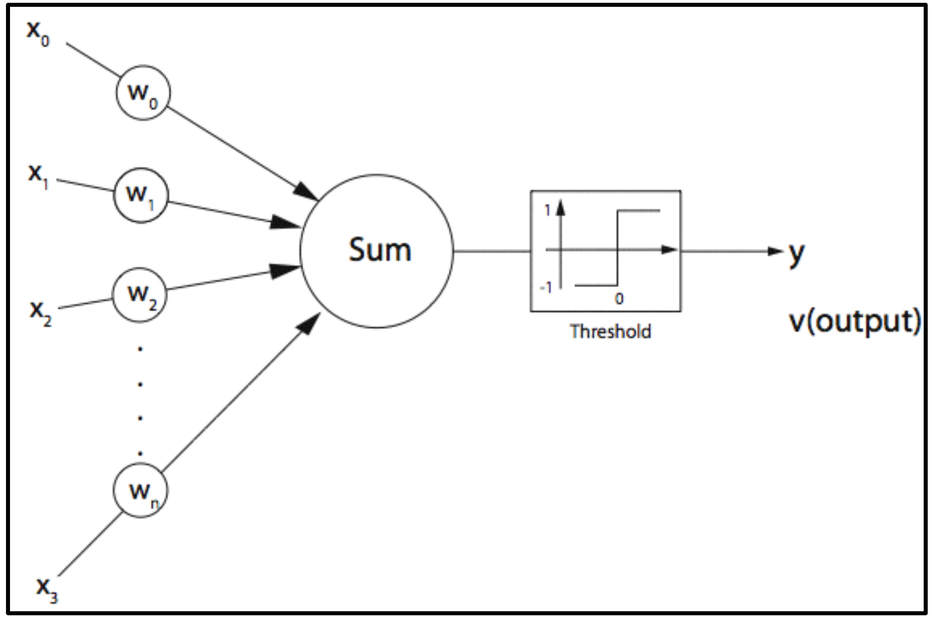
- As cognitive scientists you might want to create a fit to very nonlinear difficult data, and the methods we have used may have difficulty
  - **Or model a system whose properties are not simple, or are difficult to define**
- You may want to model cognition and performance of large groups of structure in the brain rather than just behavior
  - **Gosh I wish there was a model for these sorts of concepts…**

Brief review of neuronal structures and relation to ANNs

A simplified biological neuron



Classic threshold logic unit



A.N.N.’s are best at…

- ANN’s are best at problems where little or nothing is known, so building a mathematical model is difficult, but there happens to be a great deal of data is available
  - **A.N.N.’s are data-driven**
- Some common applications of this type are
  - **pattern classification**
  - **non-linear function approximation and system modeling**
  - **Control**
  - **associative memory**
  - **system prediction**

The threshold logic unit (TLU)

- Takes real-valued inputs (e.g. 0.243 as opposed to 1 or 0 only), x_i, each input associated with a "weight" w_i (or "synaptic weight"), which represents the contact between two nerve cells
- Performs a weighted sum of the x's, and if the sum is larger than a threshold (theta), the neuron outputs a 1, otherwise a 0
- The neuron will 'fire' if the threshold is exceeded, otherwise it does nothing

Artificial Neuron Firing…

- **Neuron Activation** is defined by the weighted sum of

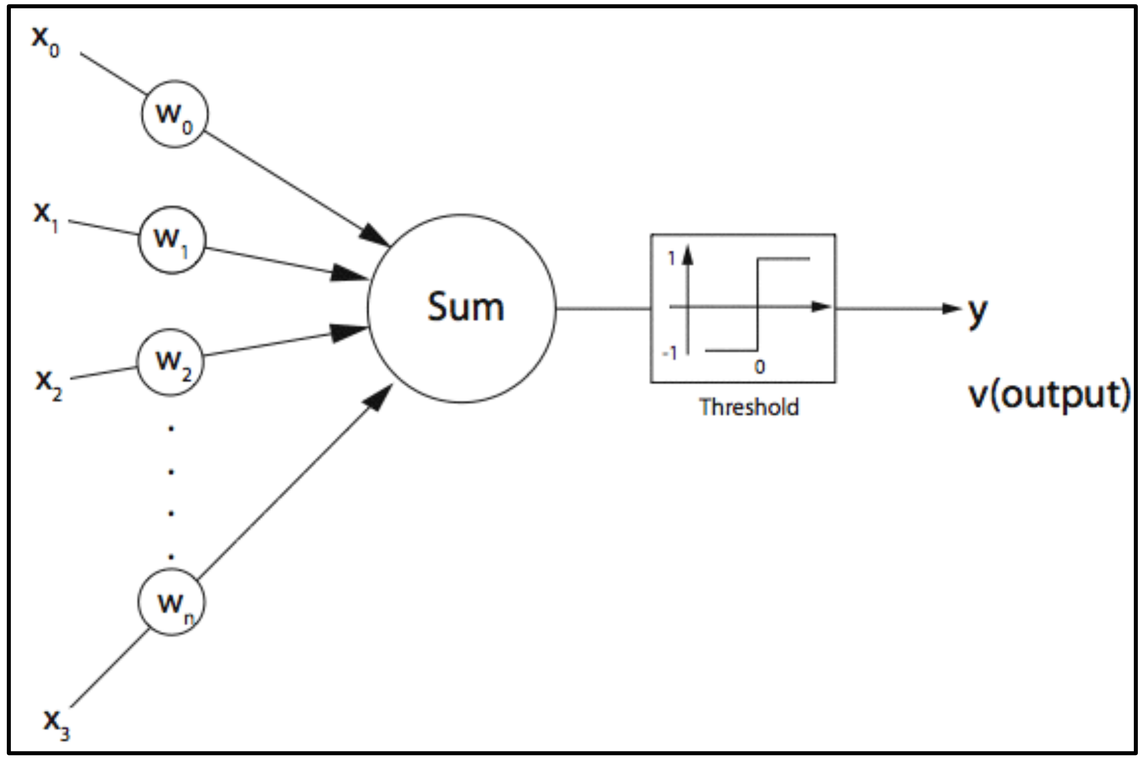
$$Activation = \sum_{i=1}^{n} w_i x_i = w^T x$$

- And whether the neuron fires is determined by

$$y(x) = \begin{cases} 1 & if \sum_{i=1}^{n} w_i x_i \geq \theta, \\ 0 & otherwise \end{cases}$$

Perceptrons are more general than TLU's

- So how is this useful?
  - **Since it can output a 0 or 1, a perceptron alone can perform many logical operations**
  - **AND, OR, NOT**
    - Demos
  - **Combined with more than one TLU, you can have continuous functions, since output of one can be weighted input to another**



How does it ‘learn?’

- The idea is that the perceptron is ‘trained’ by beginning with a guess for the weights, giving it an input, it generates an output (0 or 1), then that is compared with the desired output, and the weights are updated according to some rule
  - **i.e. - if it was wrong, change the weights so next time it will be ‘less wrong’**
  - **think about our discussions of error criteria**
- After the training period, it should respond to certain inputs with reasonable outputs
- Guess what is a popular algorithm for updating the weights?
  - **Yep, gradient descent - usually modified to be conjugate gradient to help with convergence**

Perceptron learning rule

1. Initialize weights and threshold randomly
2. Present an input vector to the neuron
3. Evaluate the output of the neuron
4. Evaluate the error of the neuron and update the weights according to :

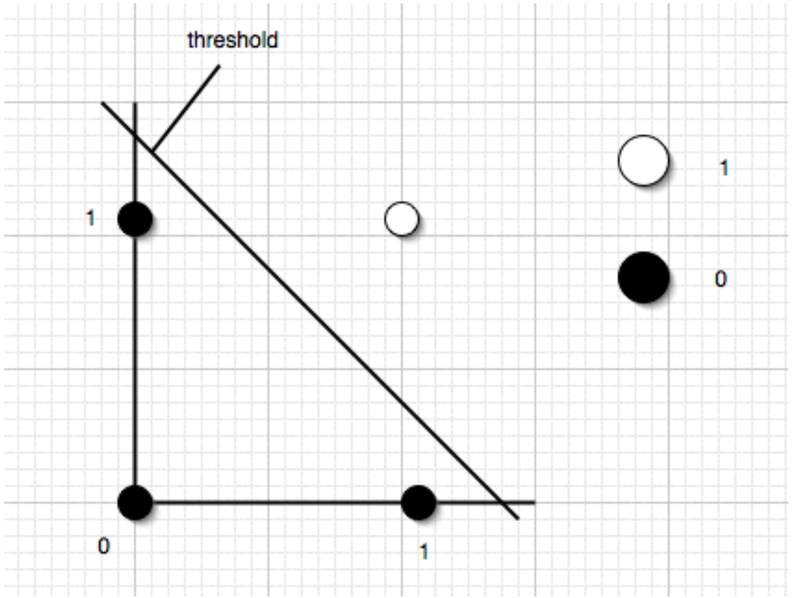
$$w_i^{t+1} = w_i^t + \eta(d - y)x_i$$

   1. **Where d is the desired output, y is the actual output of the neuron, and $\eta(0 < \eta < 1)$ is a parameter called the step size**

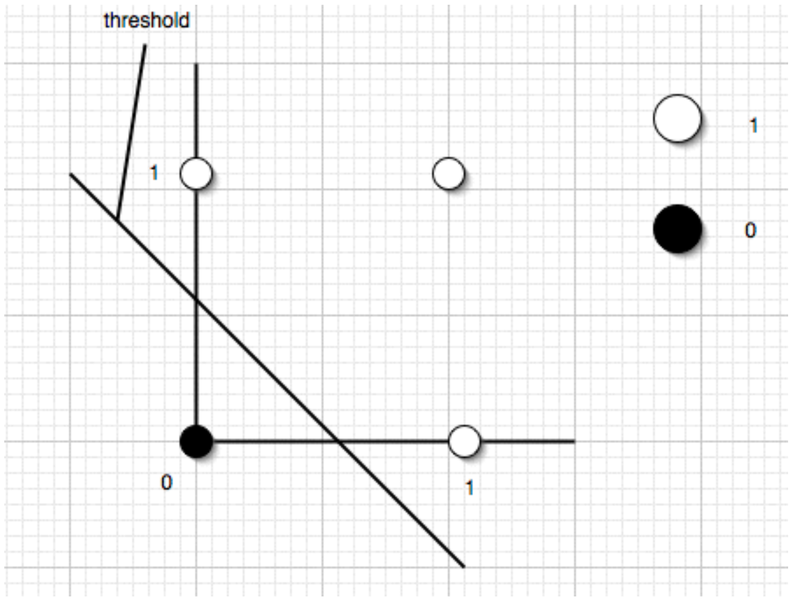
5. Go to step 2 for a certain number of iterations or until the error is less than a pre-specified value

Computing logic with Perceptrons

"And"

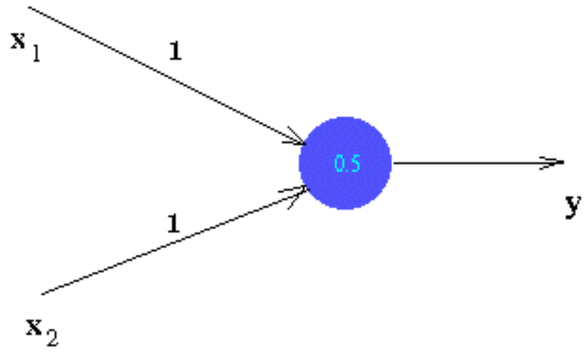


"Or"



Limitations of a single neuron

- XOR problem -
  - build a perceptron which takes 2 boolean inputs and outputs the XOR of them. What we want is a perceptron which will output 1 if the two inputs are different and 0 otherwise.
  - Consider the following perceptron as an attempt to solve the problem



Input	Input	Desired Output
0	0	0
0	1	1
1	0	1
1	1	0

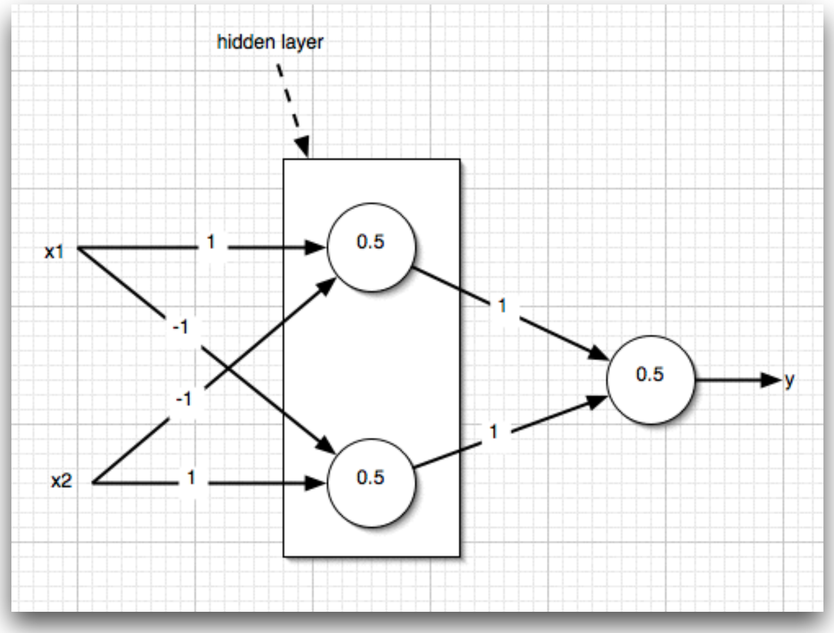
•If the inputs are both 0, then net input is 0 which is less than the threshold (0.5). So the output is 0 - desired output.
•If one of the inputs is 0 and the other is 1, then the net input is 1. This is above threshold, and so the output 1 is obtained.
•But the given perceptron fails for the last case

Limitations of single layer perceptrons (II)

- Widely publicized in the book Perceptrons [MiPa69] by Marvin Minsky and Seymour Papert
- It was not until the 1980s that these limitations were overcome with im- proved (multilayer) perceptron networks and associated learning rules
  - The funding and thus literature for ANN's slowed to a crawl until then!

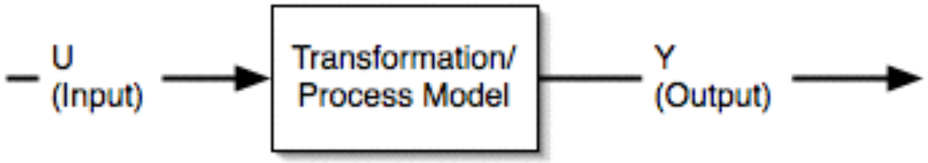
How do we resolve this?

- Feedforward multilayer networks
  - **Simple implementation**
  - **Computational capability**
  - **Input-output data**
  - **No feedback (signals only travel forward)**
- It can be shown that by connecting together multiple TLU's in a two layer network we can solve the XOR problem
  - **Implements two linear decision boundaries**

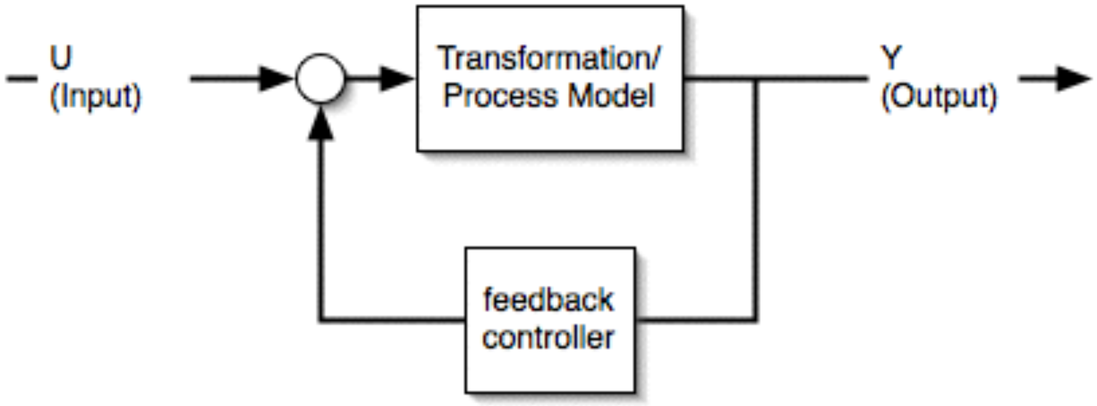


An important concept…

- Feedforward system



- Feedback system

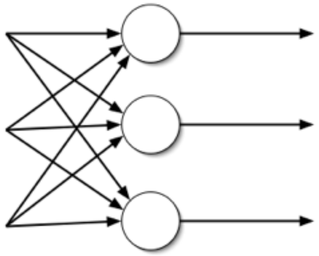


Back to neural networks…

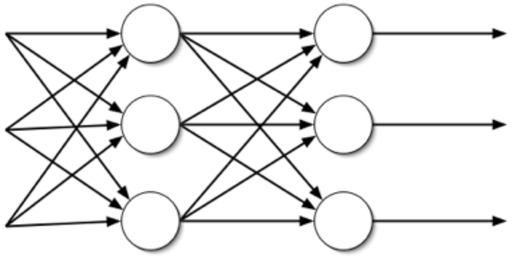
- Now that we have a concept of feedforward and feedback, and how single unit perceptrons work, let's move on to combinations of units to multi-layer networks
- More details next time but main applications of ANN's are
  - **Function fitting**
    - Fit this data without an equation!!!
  - **Classification**
    - blue cat or red cat?

Some typical network topologies

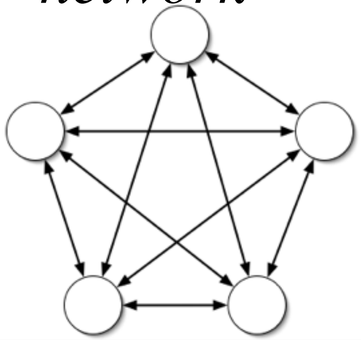
Single layer perceptron



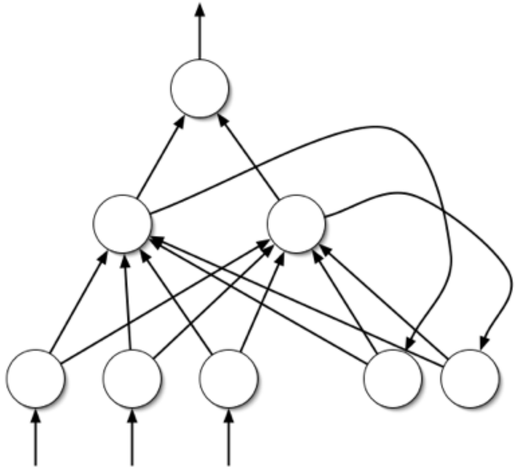
Multi-layer perceptron



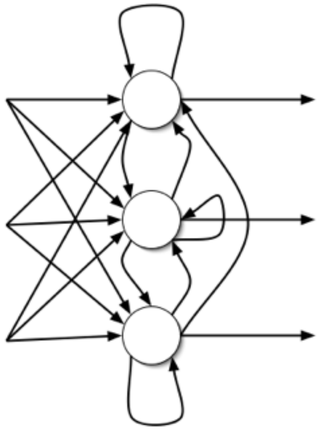
Hopfield network



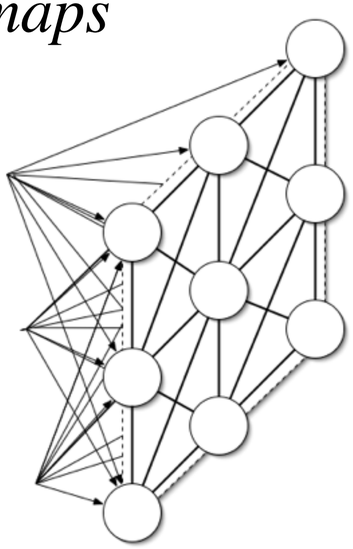
Elman recurrent network



Competitive networks

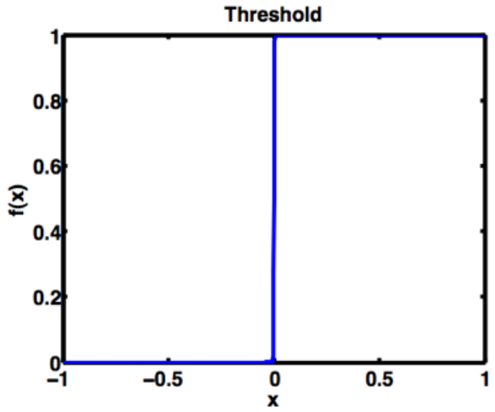


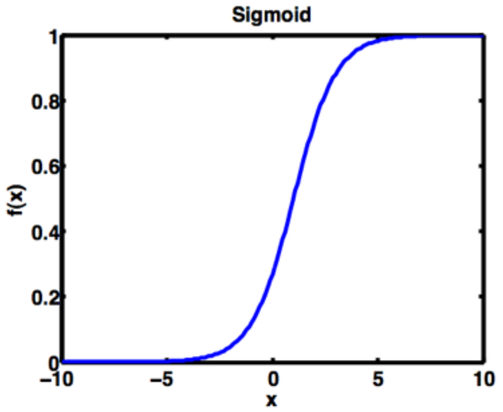
Self-organizing maps

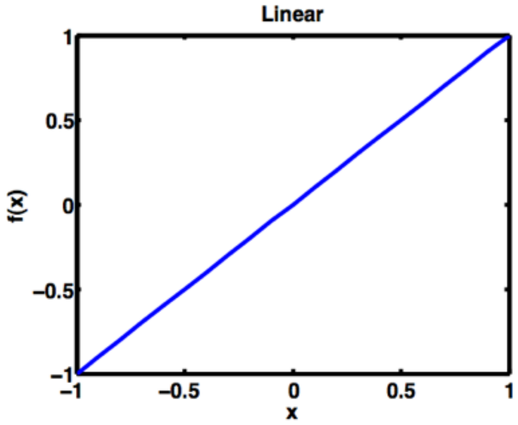


Other activation function concepts

- Threshold
- Sigmoid
- Gaussian
- Hyperbolic tangent
- Sine
- Unit sum
- Square root
- Logistic
- Softmax
- Linear
- Many others







So you can have any shape activation function

- Not just threshold
- Allows you to create real-valued outputs

$$y = f\left(\sum_{i=0}^{n} w_i x_i + b\right)$$

Methods of training networks

- Generally boils down to three learning strategies
  - **Supervised learning**
  - **Unsupervised learning**
  - **Reinforcement learning**
- Many methods with variants, but basically all fall under these above categories

Supervised learning

- *Method of learning whereby an error value is generated from the actual response of the network and the desired response. Following that, the weights are then modified such that the error is gradually reduced*
- **Training set -** A set of known input/output pairs is presented to the network in order to appropriately adjust the weights to produce the desired output given a certain input
- We already saw one example in the perceptron learning algorithm
- We will discuss backpropagation today

Unsupervised learning

- There is still an input/output relationship but no feedback is provided indicating whether network's associations are correct or not
- The network must discover by itself similarities in the patterns of the data
  - **Self-organizing networks - networks that possess the ability to to infer patterns from input-only data**

Reinforcement learning

- Input/output data and a teaching signal
  - **The teaching signal is not a measure of the error, rather an indication of the result as ‘right’ or ‘wrong’ direction**

Back propagation algorithms

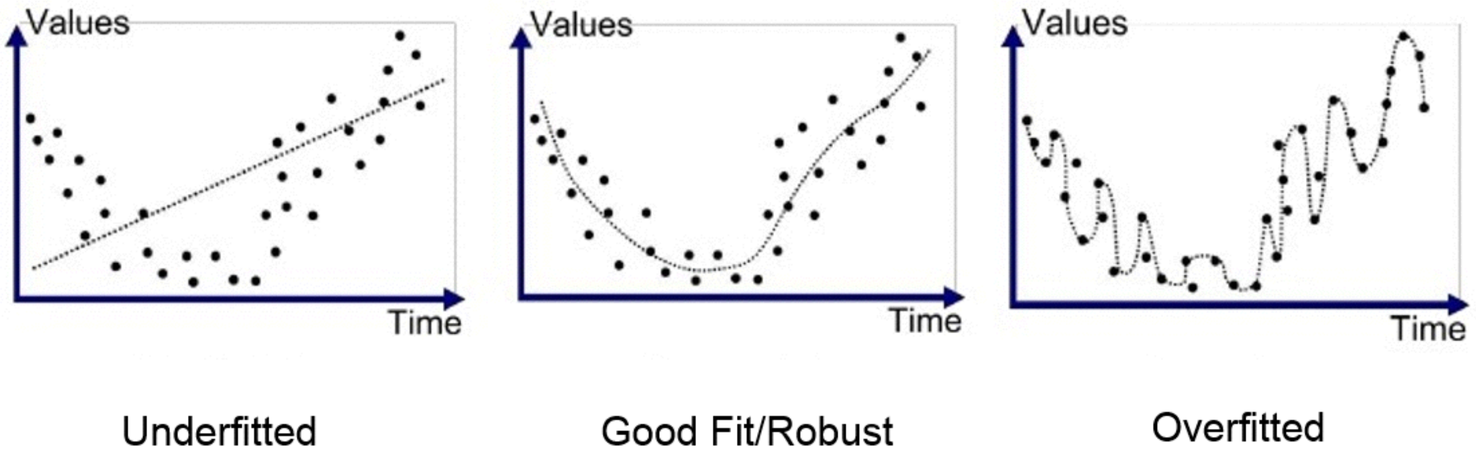
- General algorithm
  - **Present inputs**
  - **Propagate network responses forward**
  - **Compute the error between output and desired output**
  - **Back-propagate deltas**
  - **Update weights**
  - **Repeat for next pattern**
- Matlab demo **- nnd11bc**
- **Coded in python:** https://machinelearninggeek.com/backpropagation-neural-network-using-python/
- **Using scikit-learn in python**
  - https://scikit-learn.org/stable/modules/neural_networks_supervised.html

See scikit-learn for built-in modules on supervised learning

- https://scikit-learn.org/stable/modules/neural_networks_supervised.html

What is overfitting again?

- Happens in machine learning where we have many parameters and limited data, the algorithm begins to fit the noise
- Not truly part of the system, varies from one sample to the next



https://www.google.com/url?sa=i&url=https%3A%2F%2Fmedium.com%2Fgreyatom%2Fwhat-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76&psig=AOvVaw2RnpUlFc9mMXwi3SWPhQRR&ust=1690988870076000&source=images&cd=vfe&opi=89978449&ved=0CA0QjRxqFwoTCKDin8_eu4ADFQAAAAAdAAAAABAD

Techniques to Prevent Overfitting

- Regularization
  - **Reduction of hidden units**
    - Only fit simpler functions
  - **Weight decay**
- Early stopping
  - **Using validation sets**
- Bayesian regularization
  - **(see the MacKay Book)**
- Others
  - Random dropouts
  - etc

Regularization types

$$\omega = \sum_i |w_i|$$

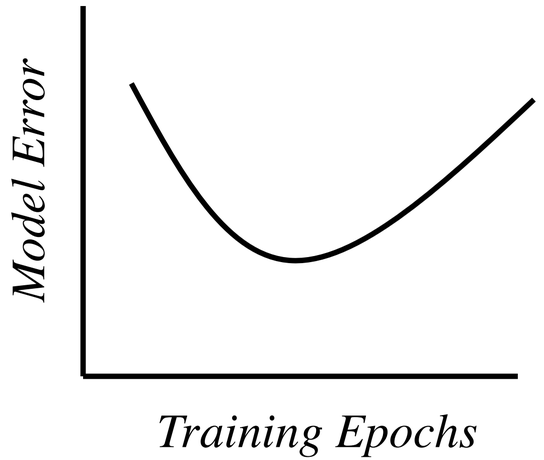
- **L_1 regularization**
  - penalizes sum of absolute value of weights
  - Model becomes simpler, more interpretable (i.e. std. vs. variance)
  - Robust to outliers

$$\omega = \frac{1}{2} \sum_i w_i^2$$

- **L_2 regularization**
  - penalizes sum of square of weights
  - Model becomes complex, able to handle more complex patterns
  - Not robust to outliers

Technique 3: Early stopping to prevent overfitting

- Start the weights very small
  - **Then the neural network starts by behaving fairly linearly**
  - **The weights gradually increase to handle nonlinearities**
- Split the data into a **validation set** and a **training set**
  - **Use the training set to adjust the weights**
  - **Use the validation set to compute model error**
  - **As the fit improves the error will decrease, when the error starts to increase again, you are fitting the noise in the training set**



Unsupervised learning for associative memory

- Hebbian learning (Hebb 1949)
- The weights of neurons whos activities are positively correlated are increased:

$$\frac{dw_{ij}}{dt} \sim Correlation(x_i, x_j)$$

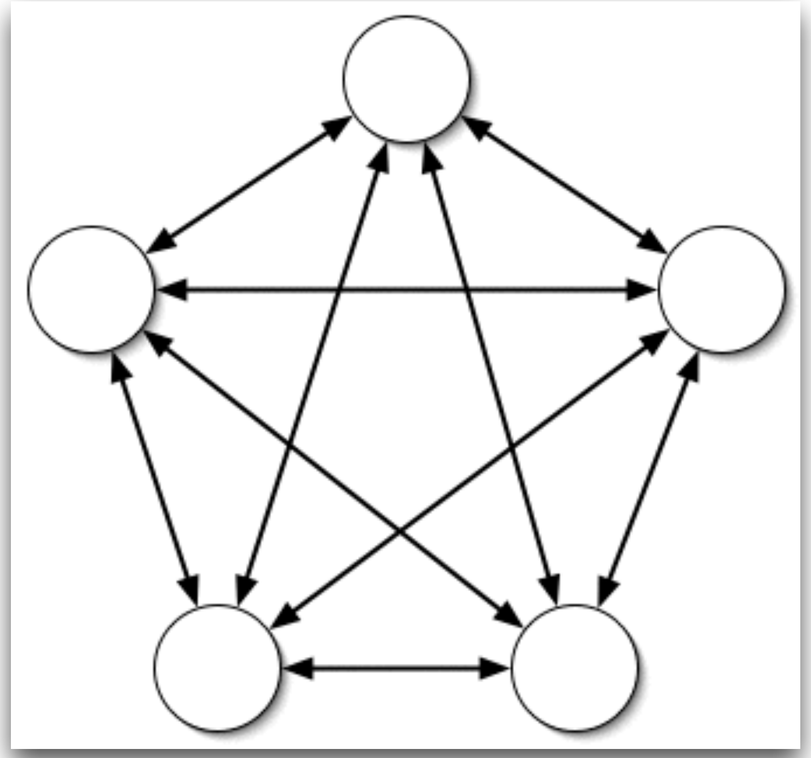
- So when stimulus m is present, the activity of neuron m increases
- Neuron n is associated with another stimulus n
- If these two stimuli co-occur in the environment, the Hebbian learning rule will increase the weights w_{nm} and w_{mn}
  - **Now when stimulus n appears later alone, the positive weight from n->m will cause neuron m to be also activated**

Associative memory with Hopfield networks

- Associative memory sample
  - **(Yellow)--(banana smell)**
- What is a binary Hopfield network?
  - **Weights are constrained to be**
    - Symmetric
    - Bidirectional
    - No self connections (w_ii = 0)
  - **Activity rule**

$$w_{ij} = w_{ji}$$

$$x(a) = \Theta(a) \equiv \begin{cases} 1 & a \geq 0 \\ -1 & a < 0 \end{cases}$$



- We need to specify the order of updates as either
  - **Synchronous**

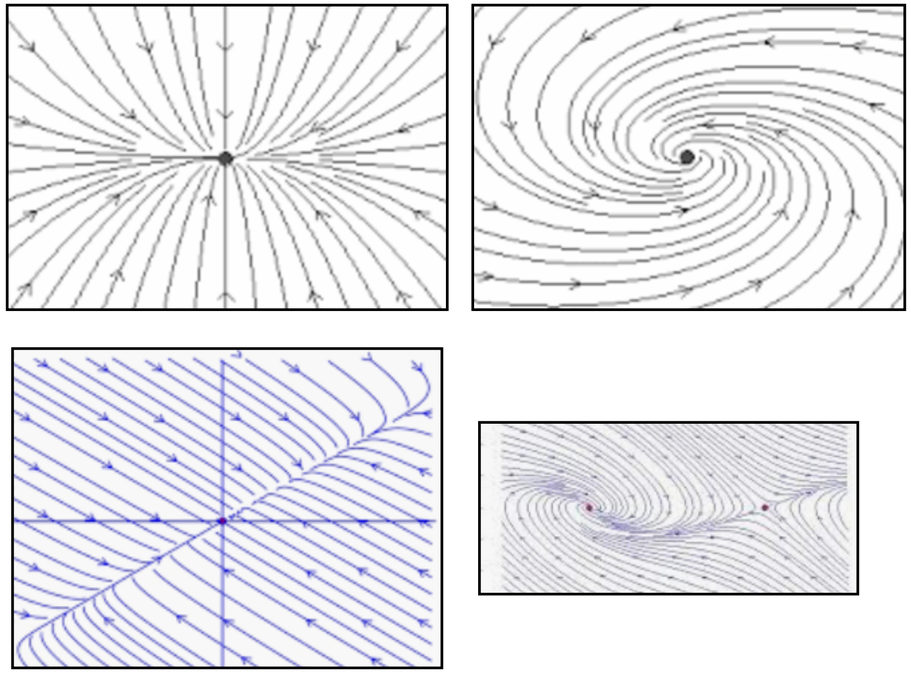
$$a_k = \sum_j w_{kj} x_j$$

$$x_k = \Theta(a_k)$$

  - **Asynchronous -** each neuron sequentially (either fixed or random order) computes its activation then updates its output state and weights

Stability in nonlinear dynamics

- Lyapunov functions
  - **If you can show that a lypapunov function exists for an ANN, then it's dynamics converge rather than diverge**
  - **Look up lyapunov functions for more info, there is not time to cover them here**



Introducing a 1-bit error is corrected in 1 iteration

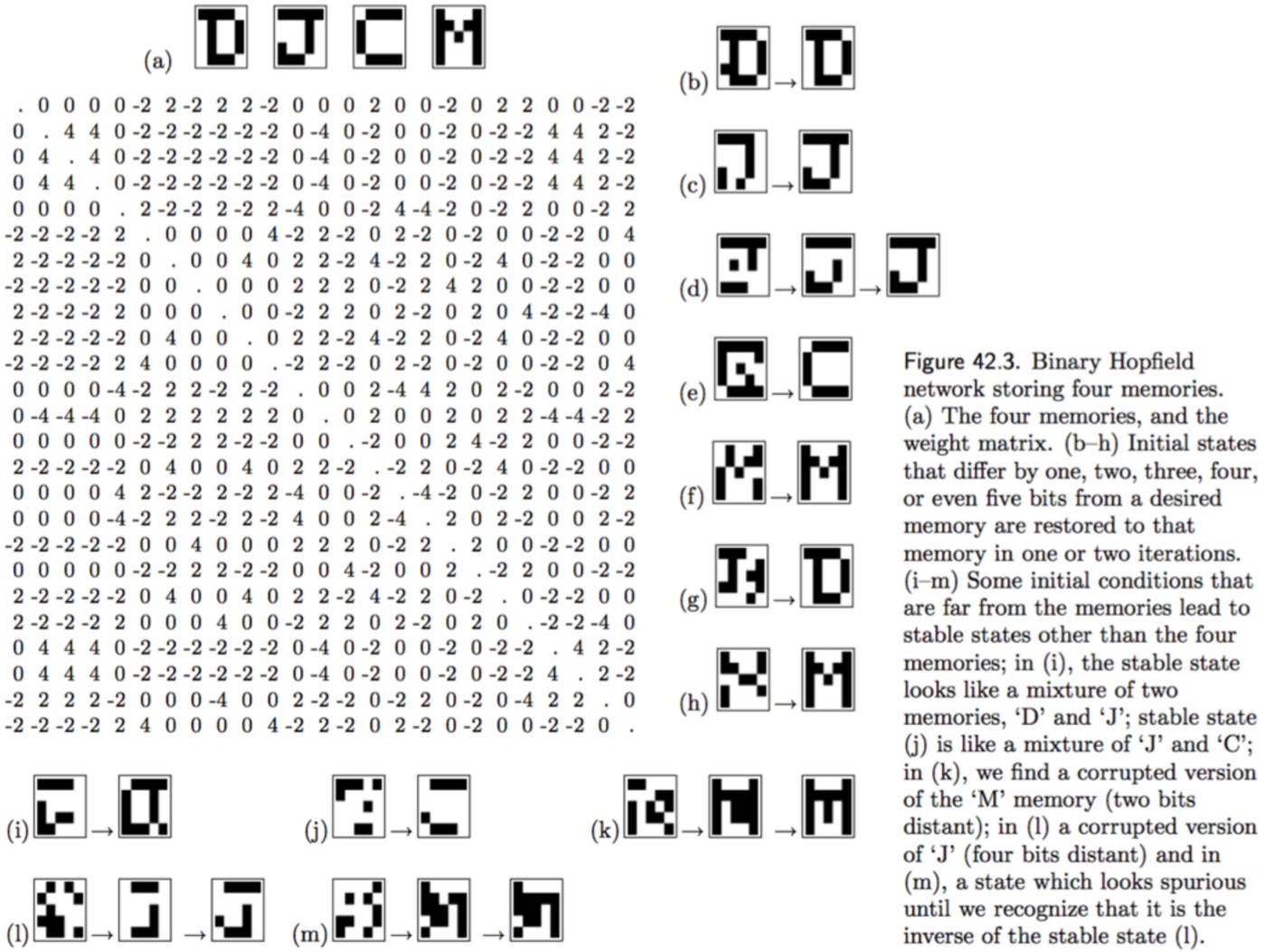


Figure 42.3. Binary Hopfield network storing four memories. (a) The four memories, and the weight matrix. (b–h) Initial states that differ by one, two, three, four, or even five bits from a desired memory are restored to that memory in one or two iterations. (i–m) Some initial conditions that are far from the memories lead to stable states other than the four memories; in (i), the stable state looks like a mixture of two memories, 'D' and 'J'; stable state (j) is like a mixture of 'J' and 'C'; in (k), we find a corrupted version of the 'M' memory (two bits distant); in (l) a corrupted version of 'J' (four bits distant) and in (m), a state which looks spurious until we recognize that it is the inverse of the stable state (l).

Imagine a computer where you destroy 20% of the components and it still works!

Thank you to Sagarika

- **Please take a moment to thank Sagarika for her efforts in the course!**

Thesis

- We are increasingly producing and needing to process, analyze and model data
- There are a wide variety of analysis and modeling tools
- The skilled M.A.D. scientist is adept at accessing then gaining an intuition into their data in order to determine the path they want to take to answer their question

Thesis II

- This involves the steps we have laid out in the course

Steps	***M.A.D. Topics***
Step 1	Data manipulation and processing
Step 2	Extracting basic information from data and visualizing that info
Step 3	Modeling the data and evaluating models, data fits
Step 4	Presenting and communicating results

Thesis III

- The specific analysis follows a different path depending on the data and question
- The model and assessment really depends on what your goal is
  - Are you trying to make predictions?
  - Are you trying to classify something?
  - Are you trying to build a memory structure?

Thesis IV

- It ultimately boils down to using data to clarify the unknown
- The problem is data, statistics, analysis and modeling can be used to argue both directions
  - It’s not infallible and doesn’t provide a certain answer
  - So what can you do? Is it useless then?

Thesis V

- Answer: No it’s not useless, it’s a tool
  - Use it to gain insight, but not as the answer
  - The person always has to make the cognitive connection to the insight
- Practicing these tools will help you to know how to choose the path to take when faced with a M.A.D. problem

Final thoughts

- You have seen many of the fundamental tools here in various pieces throughout your education
- This course has shown you how to put it all together into insight
- It is a beginning, and I hope you will continue on this path, take it further, develop your abilities and adapt to the ever developing tools and environment
- Do not fear the new technologies being developed in AI and elsewhere or get locked into one approach - use these tools to make a difference in the world

Final thoughts II

- These methods are powerful and continue to get more powerful
  - Memory and computation
- You are the future, use these materials to enable and empower you to solve the problems of the world
- Have faith in yourself
- Learn from all your experiences both positive and negative and then you can never fail
- Remember, just because nobody has ever done something it does not mean it’s impossible, it just means you haven’t done it yet

Final thoughts III

- *It has been an honor to teach you, a pleasure to be here, and I have learned a great deal.*
- *I hope you always learn your whole life, always feel free to reach out in the future*
- *I'll be in touch with those who signed up in terms of research, industry etc*

As always, thank you for your attention.

It’s a small world, I’ll see you next time!