

ME 536

— Weeks 2 ...
Some math 2 *remember*

Adopted Material

Several of the slides in this presentation are adopted from **Dr. Sekmen's** workshop at TSU (Tennessee State University, Computer Science Department) where the videos and powerpoint slides are accessible at:

http://www.tnstate.edu/computer_science/datascience/mini_courses.aspx

Adopted slides will have a  logo at that slide.

Collections - sets of things:

A subset of Real numbers

Readings from a sensor array

Elements in a set - people around a table

Grades you get in a semester, or all of them in your transcript

Set of movies you have seen

Association of Meaning to Things based on:

Relationship of elements in a set:

Are there meaningful subsets ?

Change in the values:

Is the sensor trying to tell something ?

Past trend to guess:

Get ready for what is next...

But after all, these are all numbers... in a sense...

Values - in this context scalars

Will mostly come from the set of ***Real Numbers***

Other possible alternatives are:

- Integers
- Complex numbers
- ...

Collection of values form:

- Vectors,
- Matrices,
- In general, Tensors of different degrees (orders)

Vectors

- Has a magnitude and a direction (meaningful mostly in 2-3D) - *ME intuition*
- Represents collection of values / numbers that represent:
 - Direction and magnitude :)
 - Location
 - Area
 - Volume
 - Hyper-planes, -volumes

Vectors

A vector:

- Number of elements → Dimension of the vector
- Dimension of the vector → Dimension of the ambient **space** it lives in
- Might have finite or infinite elements

$\mathbf{r}_i^{\mathbf{M}}$ indicates the i^{th} row of matrix \mathbf{M}

$\mathbf{c}_j^{\mathbf{M}}$ indicates the j^{th} column of matrix \mathbf{M}

In the presence of a single matrix of interest,
 i^{th} row and j^{th} column is represented as \mathbf{r}_i and \mathbf{c}_j respectively.

Given that we assume data is in the columns,
 j^{th} column of matrix \mathbf{M} as \mathbf{m}_j

Note that matrix is upper-case and its column being a vector
it is lower-case, yet both bold.

m_{ij} or $m_{i,j}$ or $\mathbf{M}(i, j)$ indicates
the element of \mathbf{M} at location (i, j)

Note that, the letter 'm' is due to it being an element of \mathbf{M}

Similarly for some matrix \mathbf{Q} , element at $(3, 4)$ is $q_{3,4}$

$\tilde{\mathbf{M}}$ is an approximation of \mathbf{M} , i.e. $\tilde{\mathbf{M}} \approx \mathbf{M}$

$\tilde{\mathbf{M}}_k$ is a rank- k approximation of \mathbf{M}

Finally, unit vectors will be given with a hat on top.

Ex: If vector \mathbf{b} with some random length is normalized, we will get $\hat{\mathbf{b}}$

Convention on Notation

In our lecture notes and assignments:

scalars will be represented with lowercase italic letters: s

vectors will be represented with bold lowercase letters: \mathbf{v}

MATRICES will be represented with bold capital letters: \mathbf{M}

Vector Space - yes they live in spaces

- ◆ A vector space **is a set of objects** that may be added together or multiplied by numbers (called scalars)
 - ▶ Scalars are typically real numbers
 - ♦ But can be complex numbers, rational numbers, or generally any field
 - ▶ Vector addition and scalar multiplication must satisfy certain requirements (called axioms)

Example: \mathbb{R}^2

- ① You can add $x \in \mathbb{R}^2$ and $y \in \mathbb{R}^2$ and $x + y \in \mathbb{R}^2$
- ② You can multiple $x \in \mathbb{R}^2$ by $\alpha \in \mathbb{R}$ and $\alpha x \in \mathbb{R}^2$

Vector Space

A set \mathcal{V} is called a vector space over field \mathcal{F} (often \mathbb{R} or \mathbb{C}) when vector addition and scalar multiplication satisfies:

Vector Addition

1. $v + w \in \mathcal{V}$ for all $v, w \in \mathcal{V}$
2. $(v + w) + u = v + (w + u) \in \mathcal{V}$ for all $v, w, u \in \mathcal{V}$ Associativity
3. $v + w = w + v$ for all $v, w \in \mathcal{V}$ Commutativity
4. There is an element $0 \in \mathcal{V}$ such that $v + 0 = v$ for all $v \in \mathcal{V}$ Identity Element
5. For every $v \in \mathcal{V}$, there exist $-v \in \mathcal{V}$ such that $v + (-v) = 0$ Inverse Element

Scalar Multiplication

1. $\alpha v \in \mathcal{V}$ for all $\alpha \in \mathcal{F}$ and $v \in \mathcal{V}$
2. $(\alpha\beta)v = \alpha(\beta v)$ for all $\alpha, \beta \in \mathcal{F}$ and $v \in \mathcal{V}$ Compatibility
3. $\alpha(v + w) = \alpha v + \alpha w$ for all $\alpha \in \mathcal{F}$ and $v, w \in \mathcal{V}$
4. $(\alpha + \beta)v = \alpha v + \beta v$ for all $\alpha, \beta \in \mathcal{F}$ and $v \in \mathcal{V}$ Distributivity
5. $1v = v$ for all $v \in \mathcal{V}$ Identity Element

Vector Space - yes they are a set of objects

A set \mathcal{V} is called a vector space over field \mathcal{F} (often \mathbb{R} or \mathbb{C}) when vector addition and scalar multiplication satisfies:

Vector Addition

1. $v + w \in \mathcal{V}$ for all $v, w \in \mathcal{V}$
2. $(v + w) + u = v + (w + u) \in \mathcal{V}$ for all $v, w, u \in \mathcal{V}$ Associativity
3. $v + w = w + v$ for all $v, w \in \mathcal{V}$ Commutativity
4. There is an element $0 \in \mathcal{V}$ such that $v + 0 = v$ for all $v \in \mathcal{V}$ Identity Element
5. For every $v \in \mathcal{V}$, there exist $-v \in \mathcal{V}$ such that $v + (-v) = 0$ Inverse Element

Scalar Multiplication

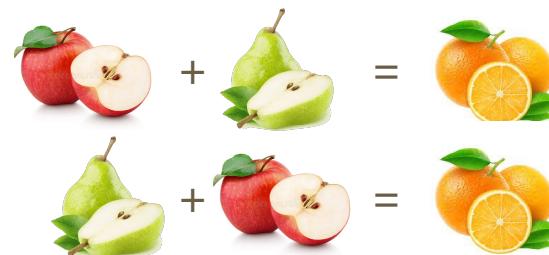
1. $\alpha v \in \mathcal{V}$ for all $\alpha \in \mathcal{F}$ and $v \in \mathcal{V}$
2. $(\alpha\beta)v = \alpha(\beta v)$ for all $\alpha, \beta \in \mathcal{F}$ and $v \in \mathcal{V}$ Compatibility
3. $\alpha(v + w) = \alpha v + \alpha w$ for all $\alpha \in \mathcal{F}$ and $v, w \in \mathcal{V}$
4. $(\alpha + \beta)v = \alpha v + \beta v$ for all $\alpha, \beta \in \mathcal{F}$ and $v \in \mathcal{V}$ Distributivity
5. $1v = v$ for all $v \in \mathcal{V}$ Identity Element

With:

a set of objects



- 2 operators: addition and multiplication
- Some properties for each operation



...

In general - Even if we are interested in objects

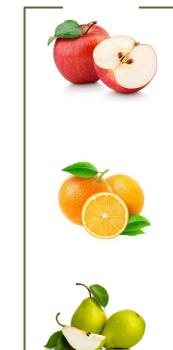
They are translated into numbers

- Therefore a vector is not necessarily an arrow :) might have a corresponding meaning
- We might need to assess **how close** 2 vectors are



$$+ \begin{bmatrix} 2.5 \\ 1.25 \\ 1.5 \end{bmatrix}$$

$$\begin{bmatrix} 2.5 \\ 1.25 \\ 1.5 \end{bmatrix} > ? < \begin{bmatrix} 1.5 \\ 2.5 \\ 2.25 \end{bmatrix}$$



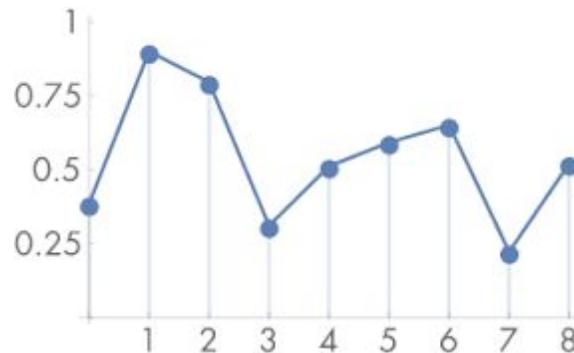
$$+ \begin{bmatrix} 1.5 \\ 2.5 \\ 2.25 \end{bmatrix}$$

Vector Space

- ◆ A vector space may have additional structures such as a norm or inner product
 - ▶ This is typical for infinite dimensional function spaces whose vectors are functions.
- ◆ Many practical problems require ability to decide whether a sequence of vectors converges to a given vector
 - ▶ In order to allow proximity and continuity considerations, most vector spaces are endowed with a suitable topology
 - A topology is a structure that allows to define “being close to each other”
 - Such topological vector spaces have richer theory
 - Banach space topology is given by a norm
 - Hilbert space topology is given by an inner product

Vectors to represents functions

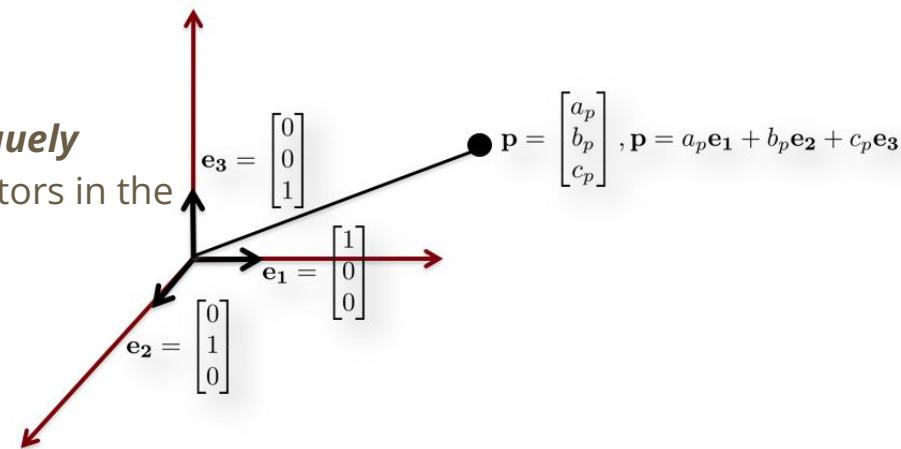
Just recall ME310



But let the number of elements $\rightarrow \infty$

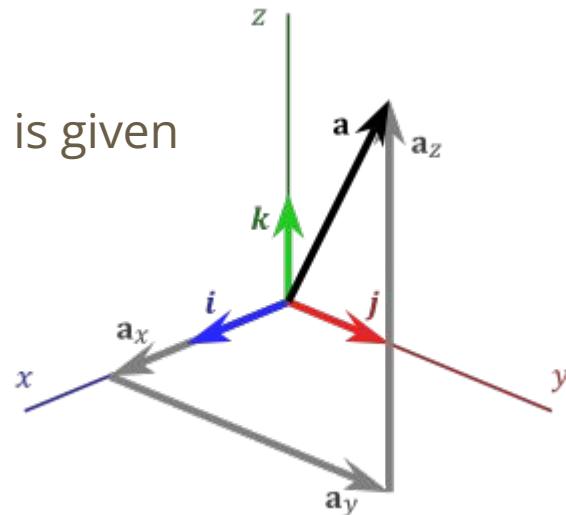
On what basis?

- For every vector space **bases** (i.e. more than 1 basis) can be written
- A basis is formed by just sufficient number of vectors to **span** that **space**
- Hence:
 - All vectors in this vector space can be **uniquely** defined as a linear combination of the vectors in the selected basis
 - Vector \rightarrow **Matrix E** times a vector **p**: Ep
 - **Matrix** contains e_i in its columns



Standard basis

- Standard basis is implied if no basis is given
- Invisibly there all the time
- Simple, but not unique



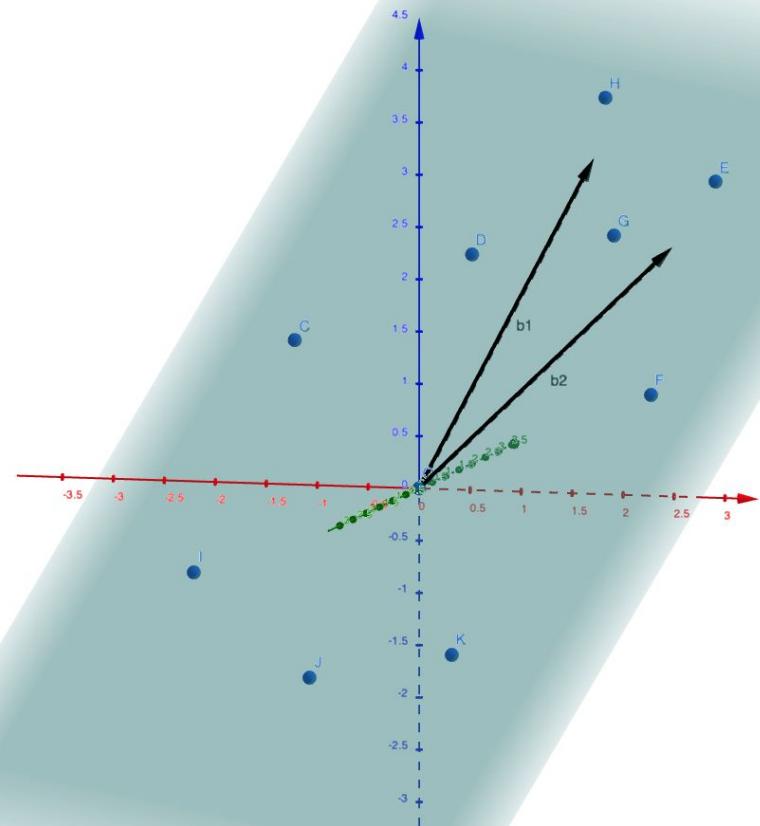
$$\mathbf{e}_1 = (1, 0, 0), \quad \mathbf{e}_2 = (0, 1, 0), \quad \mathbf{e}_3 = (0, 0, 1).$$

After a certain number of dimensions letters will run out!!!

Some 2 - 3D exercises

Standard vs other bases

Why do we need them?



Exercise

Write the following with respect to standard basis e_i :

$$\begin{bmatrix} 2 \\ 5 \end{bmatrix} =$$

Exercise

Write the following with respect to basis: $\mathbf{b}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \mathbf{b}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

$$\begin{bmatrix} 2 \\ 5 \end{bmatrix} = x\mathbf{b}_1 + y\mathbf{b}_2 = [\mathbf{b}_1 \quad \mathbf{b}_2] \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2 \\ 5 \end{bmatrix}$$

$$x = ? \quad y = ?$$

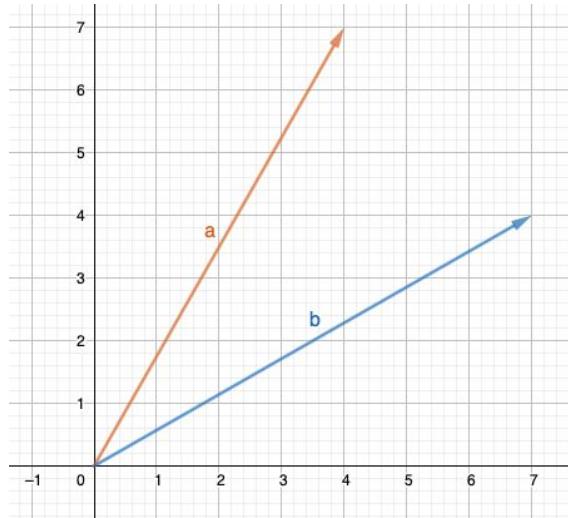
$$[\mathbf{b}_1 \quad \mathbf{b}_2] \begin{bmatrix} x_1 & x_2 \\ y_1 & y_2 \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 5 & 1 \end{bmatrix}$$

$$x_i = ? \quad y_i = ?$$

Recall: Dot Product

We will later refer to Dot Product as *Inner Product*:

$$\langle \mathbf{a}, \mathbf{b} \rangle = \|\mathbf{a}\| \|\mathbf{b}\| \cos(\theta)$$

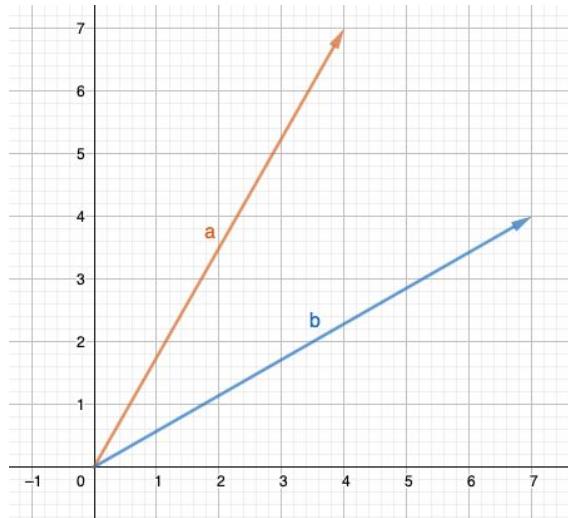


Recall: Dot Product

We will later refer to Dot Product as *Inner Product*:

$$\langle \mathbf{a}, \mathbf{b} \rangle = \|\mathbf{a}\| \|\mathbf{b}\| \cos(\theta)$$

$$\langle \mathbf{a}, \mathbf{b} \rangle =$$



$$\mathbf{a} = \begin{bmatrix} 4 \\ 7 \end{bmatrix}$$

$$\mathbf{b} = \begin{bmatrix} 7 \\ 4 \end{bmatrix}$$

Exercise:

On colab, help me write a function that generates N random data points on a plane spanned by v1, v2 in 3D.

If spanning vectors are not given, they should be random. The function should also allow addition of noise to the data

Recall: Orthogonal decomposition & Projection

Let $\mathbf{a}, \mathbf{b} \in \mathbb{R}^2$ and \mathbf{a}_b is the component of \mathbf{a} along \mathbf{b} , and $\mathbf{a}_{b\perp}$ is the component of \mathbf{a} that is perpendicular to \mathbf{b} .

Question is, given \mathbf{a} and \mathbf{b} how can you find \mathbf{a}_b and $\mathbf{a}_{b\perp}$?

Note that:

$$\mathbf{a} = \mathbf{a}_b + \mathbf{a}_{b\perp}$$

Therefore, if we find \mathbf{a}_b , then,

$$\mathbf{a}_{b\perp} = \mathbf{a} - \mathbf{a}_b$$

Finding \mathbf{a}_b then simply is a unit projection vector of \mathbf{a} along \mathbf{b} and multiplying it with the $\|\mathbf{a}_b\|$.

Note that:

$$\|\mathbf{a}_b\| = \|\mathbf{a}\| \cos(\theta)$$

Recall by definition:

$$\langle \mathbf{a}, \mathbf{b} \rangle = \|\mathbf{a}\| \|\mathbf{b}\| \cos(\theta)$$

Hence,

$$\|\mathbf{a}_b\| = \frac{\langle \mathbf{a}, \mathbf{b} \rangle}{\|\mathbf{b}\|}$$

If this magnitude is multiplied by a unit vector along \mathbf{b} we are done with \mathbf{a}_b as:

$$\mathbf{a}_b = \frac{\langle \mathbf{a}, \mathbf{b} \rangle}{\|\mathbf{b}\|} \frac{\mathbf{b}}{\|\mathbf{b}\|} = \frac{\langle \mathbf{a}, \mathbf{b} \rangle}{\|\mathbf{b}\|^2} \mathbf{b} = \frac{\langle \mathbf{a}, \mathbf{b} \rangle}{\langle \mathbf{b}, \mathbf{b} \rangle} \mathbf{b}$$

Finally:

$$\mathbf{a}_{b\perp} = \mathbf{a} - \mathbf{a}_b = \mathbf{a} - \frac{\langle \mathbf{a}, \mathbf{b} \rangle}{\|\mathbf{b}\|^2} \mathbf{b}$$

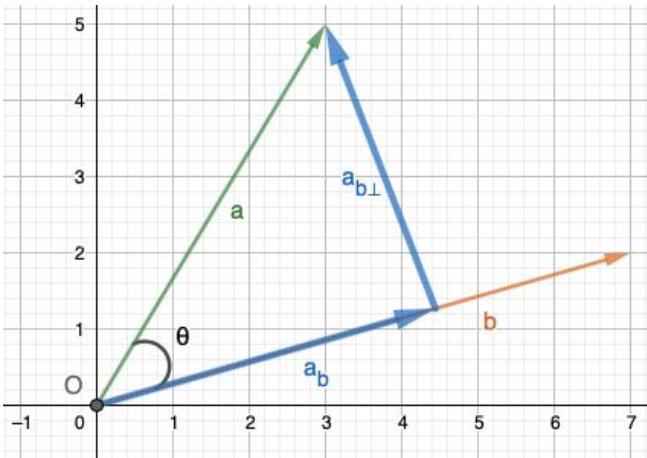
Also note that:

$$\mathbf{a}_{b\perp} = \mathbf{a} - \frac{\langle \mathbf{a}, \mathbf{b} \rangle}{\langle \mathbf{b}, \mathbf{b} \rangle} \mathbf{b}$$

If \mathbf{b} was already unit, things would have been easier:

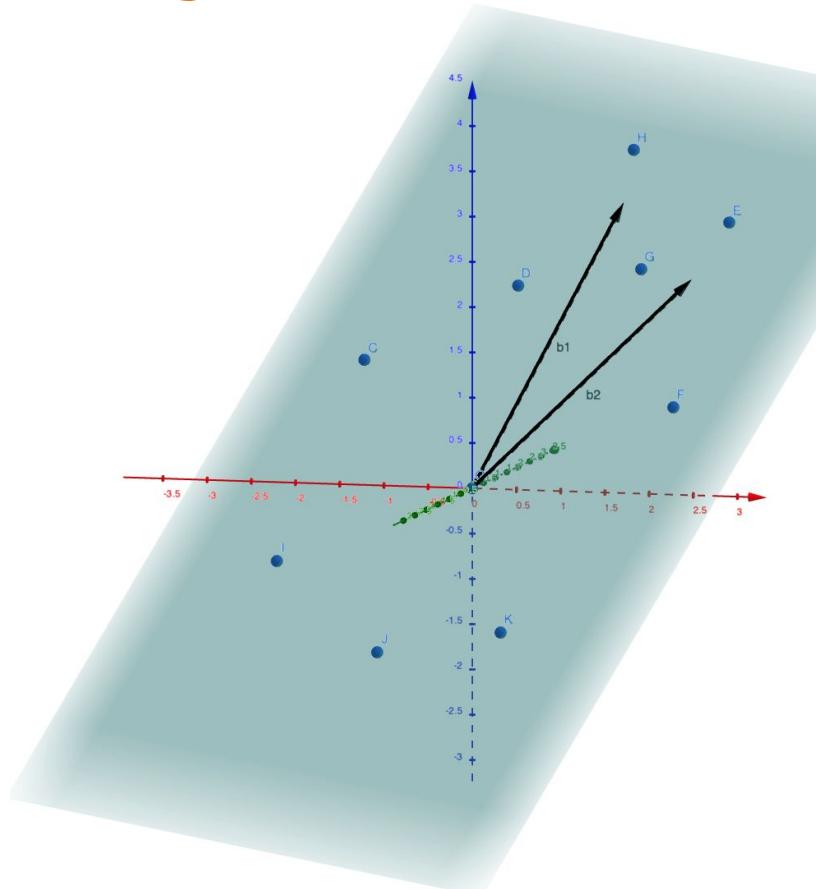
$$\mathbf{a}_b = \langle \mathbf{a}, \mathbf{b} \rangle \mathbf{b}$$

$$\mathbf{a}_{b\perp} = \mathbf{a} - \langle \mathbf{a}, \mathbf{b} \rangle \mathbf{b}$$



How to find an orthonormal basis given data

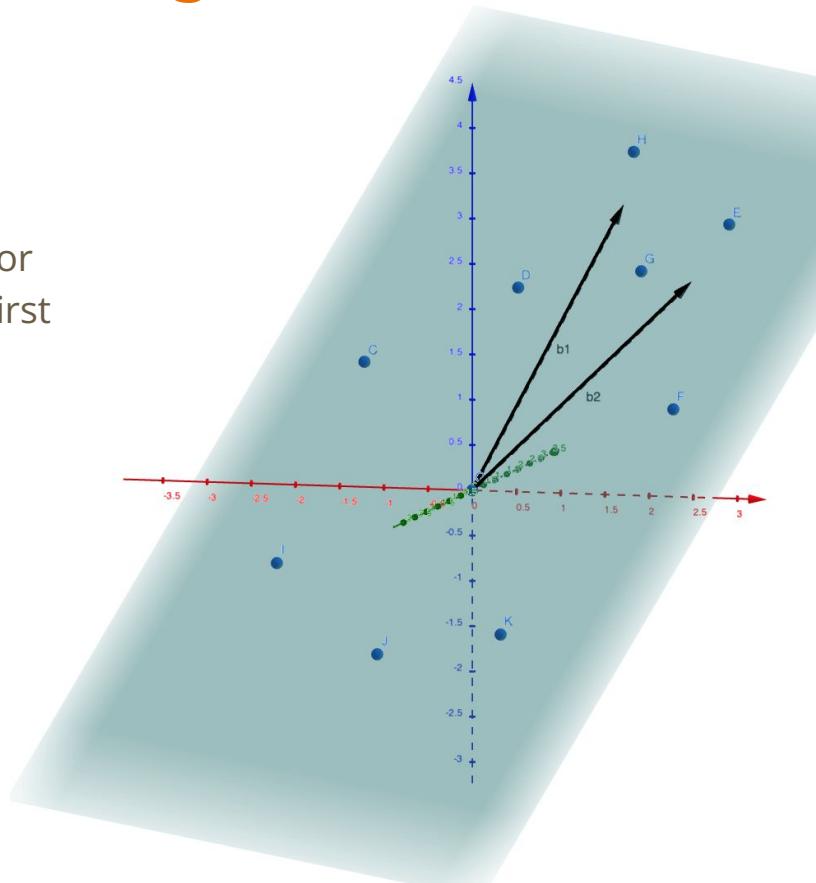
What do you think?



How to find an orthonormal basis given data

Gram-Schmidt says:

- Take one of the data point to be the first basis vector
- Take another, find the perpendicular comp to the first
- Take another, find the perpendicular comp to the previous two
- Take another, find the perpendicular comp to the previous three
- ...



How to find an orthonormal basis given data: Gram-Schmidt way - w/ geogebra example

Let the data matrix $\mathbf{M}_{n \times m} = [\mathbf{c}_1 \mathbf{c}_2 \cdots \mathbf{c}_m]$ where \mathbf{c}_i , i.e. columns of \mathbf{M} , are our data points, and $\mathbf{c}_i \in \mathbb{R}^n$. Let's assume that data lives in a d dimensional subspace in \mathbb{R}^n where $d < n$ and for the sake of argument assume that we have enough or more than enough data, i.e. $d \leq m$.

To find an **orthogonal** basis for the data:

1. select one of the data points as the first basis vector, without loss of generality, we can choose the first data point

$$\mathbf{v}_1 = \mathbf{c}_1$$

2. find a perpendicular vector to \mathbf{v}_1 using \mathbf{c}_2 :

$$\mathbf{v}_2 = \mathbf{c}_2 - \frac{\mathbf{c}_2 \cdot \mathbf{v}_1}{\mathbf{v}_1 \cdot \mathbf{v}_1} \mathbf{v}_1$$

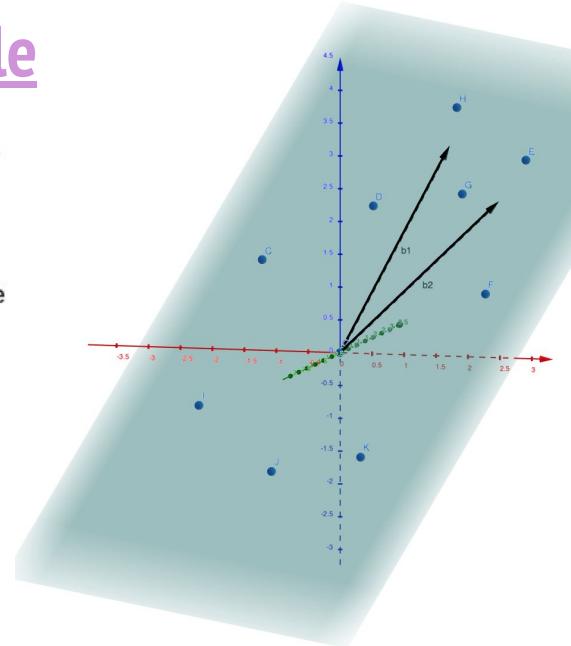
3. find a perpendicular vector to $\text{Span}\{\mathbf{v}_1, \mathbf{v}_2\}$ using \mathbf{c}_3 :

$$\mathbf{v}_3 = \underbrace{\mathbf{c}_3 - \frac{\mathbf{c}_3 \cdot \mathbf{v}_1}{\mathbf{v}_1 \cdot \mathbf{v}_1} \mathbf{v}_1}_{\perp \{\mathbf{v}_1, \mathbf{v}_2\}} - \frac{\mathbf{c}_3 \cdot \mathbf{v}_2}{\mathbf{v}_2 \cdot \mathbf{v}_2} \mathbf{v}_2$$

... continue until you find all basis vectors i.e. $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_d\}$

If an **orthonormal** basis is required, normalize \mathbf{v}_i :

$$\mathbf{v}_i = \frac{1}{\sqrt{\mathbf{v}_i \cdot \mathbf{v}_i}} \mathbf{v}_i$$

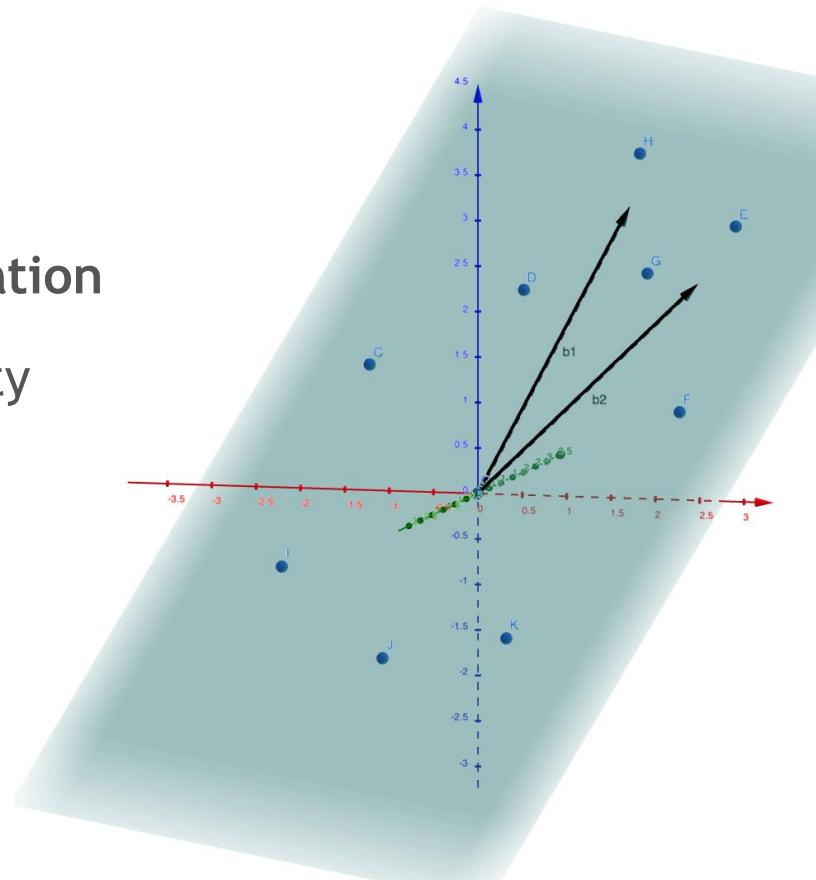


RECALL: $\mathbf{a}_{b\perp} = \mathbf{a} - \mathbf{a}_b = \mathbf{a} - \frac{\langle \mathbf{a}, \mathbf{b} \rangle}{\|\mathbf{b}\|^2} \mathbf{b}$

How to find an orthonormal basis given data: Alternative to Gram-Schmidt

Check out:

Householder Orthogonalization
for better numerical stability



How to find an orthonormal basis given data: Python way

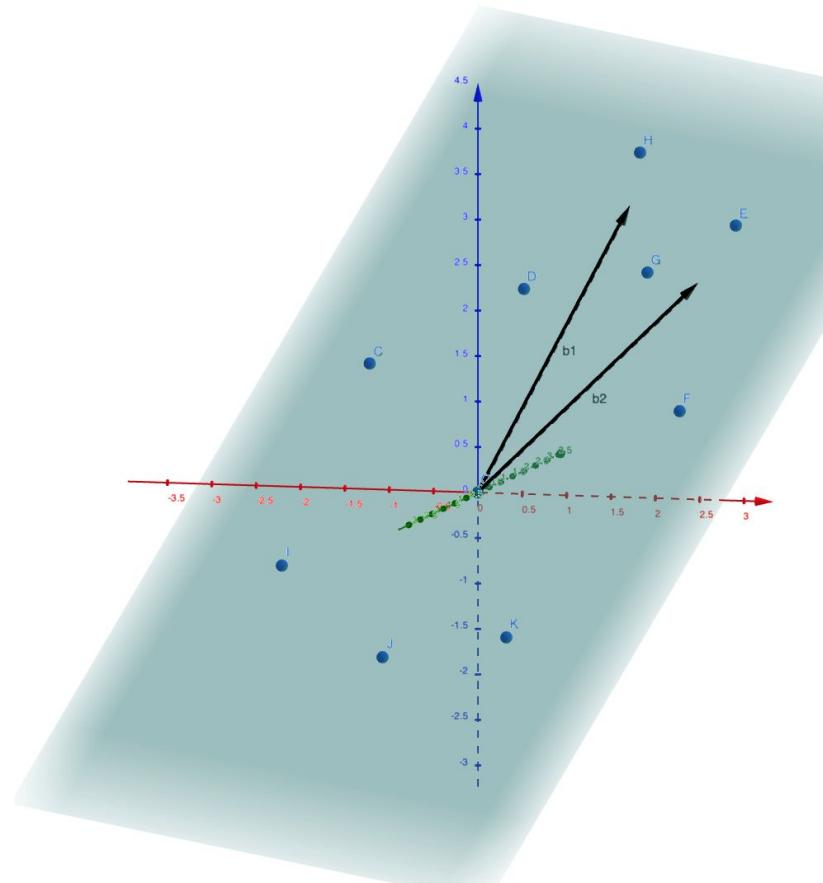
Check out:

`scipy.linalg.orth()`

and

`numpy.linalg.qr()`

To get an insight about each



Recall: Orthogonal Matrices (a.k.a. Orthogonal)

`scipy.orth` just returns an orthonormal basis:

Data Matrix

| | | | | |
|------|------|------|------|------|
| 9.00 | 0.00 | 4.00 | 8.00 | 8.00 |
| 3.00 | 2.00 | 8.00 | 8.00 | 2.00 |
| 8.00 | 0.00 | 2.00 | 6.00 | 0.00 |

$$\mathbf{B} = \text{orth}(\mathbf{D})$$

Why do we want and like orthogonal matrices?

\mathbf{B}

| | | |
|-------|-------|-------|
| -0.72 | 0.45 | -0.54 |
| -0.53 | -0.85 | 0.00 |
| -0.45 | 0.28 | 0.84 |

$\mathbf{B} @ \mathbf{B.T}$

| | | |
|-------|-------|------|
| 1.00 | -0.00 | 0.00 |
| -0.00 | 1.00 | 0.00 |
| 0.00 | 0.00 | 1.00 |

$\mathbf{B.T} @ \mathbf{B}$

| | | |
|-------|-------|-------|
| 1.00 | -0.00 | 0.00 |
| -0.00 | 1.00 | -0.00 |
| 0.00 | -0.00 | 1.00 |

Example: `scipy.orth(D)` just returns the basis: B

Hence we have D , and $B = \text{orth}(D)$

Data Matrix

| | | | | |
|------|------|------|------|------|
| 9.00 | 0.00 | 4.00 | 8.00 | 8.00 |
| 3.00 | 2.00 | 8.00 | 8.00 | 2.00 |
| 8.00 | 0.00 | 2.00 | 6.00 | 0.00 |

B

| | | |
|-------|-------|-------|
| -0.72 | 0.45 | -0.54 |
| -0.53 | -0.85 | 0.00 |
| -0.45 | 0.28 | 0.84 |

$B @ B.T$

| | | |
|-------|-------|------|
| 1.00 | -0.00 | 0.00 |
| -0.00 | 1.00 | 0.00 |
| 0.00 | 0.00 | 1.00 |

$B.T @ B$

| | | |
|-------|-------|-------|
| 1.00 | -0.00 | 0.00 |
| -0.00 | 1.00 | -0.00 |
| 0.00 | -0.00 | 1.00 |

But what are the coordinates C of D with respect to B ?

$$D = BC$$

$$C = B^{-1}D$$

Is B always invertible?

Example: Find an orthogonal basis using QR

Data Matrix

| | | | | |
|------|------|------|------|------|
| 9.00 | 0.00 | 4.00 | 8.00 | 8.00 |
| 3.00 | 2.00 | 8.00 | 8.00 | 2.00 |
| 8.00 | 0.00 | 2.00 | 6.00 | 0.00 |

What will be the shape and structure of these matrices where $D = QR$?

| Q | Q @ Q.T | | |
|---------|---------|-------|-------|
| | -0.73 | 0.18 | -0.66 |
| -0.24 | -0.97 | -0.00 | |
| -0.64 | 0.16 | 0.75 | |
| Q.T @ Q | | | |
| 1.00 | 0.00 | -0.00 | |
| 0.00 | 1.00 | 0.00 | |
| -0.00 | 0.00 | 1.00 | |

| R | -12.41 | -0.48 | -6.12 | -11.60 | -6.29 |
|------|--------|-------|-------|--------|-------|
| 0.00 | -1.94 | -6.72 | -5.35 | -0.50 | |
| 0.00 | 0.00 | -1.16 | -0.83 | -5.31 | |

Example: change of basis using $D = QR$

$$\begin{array}{c|ccccc} \text{Data Matrix} & | & 9.00 & 0.00 & 4.00 & 8.00 & 8.00 \\ \hline & | & 3.00 & 2.00 & 8.00 & 8.00 & 2.00 \\ \hline & | & 8.00 & 0.00 & 2.00 & 6.00 & 0.00 \end{array} \quad = \quad \begin{array}{c|ccccc} Q & | & -0.73 & 0.18 & -0.66 & | & R \\ \hline & | & -0.24 & -0.97 & -0.00 & | & | \\ & | & -0.64 & 0.16 & 0.75 & | & | \end{array} \quad \begin{array}{c|ccccc} & | & -12.41 & -0.48 & -6.12 & -11.60 & -6.29 \\ \hline & | & 0.00 & -1.94 & -6.72 & -5.35 & -0.50 \\ \hline & | & 0.00 & 0.00 & -1.16 & -0.83 & -5.31 \end{array}$$

Coordinates with respect to:

$$\begin{array}{c|ccccc} \text{Data Matrix} & | & 9.00 & 0.00 & 4.00 & 8.00 & 8.00 \\ \hline & | & 3.00 & 2.00 & 8.00 & 8.00 & 2.00 \\ \hline & | & 8.00 & 0.00 & 2.00 & 6.00 & 0.00 \end{array}$$

Standard basis $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$

$$\begin{array}{c|ccccc} R & | & -12.41 & -0.48 & -6.12 & -11.60 & -6.29 \\ \hline & | & 0.00 & -1.94 & -6.72 & -5.35 & -0.50 \\ \hline & | & 0.00 & 0.00 & -1.16 & -0.83 & -5.31 \end{array}$$

Columns of Q i.e. $\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3$

Exercise: Is it always good to go by the book?

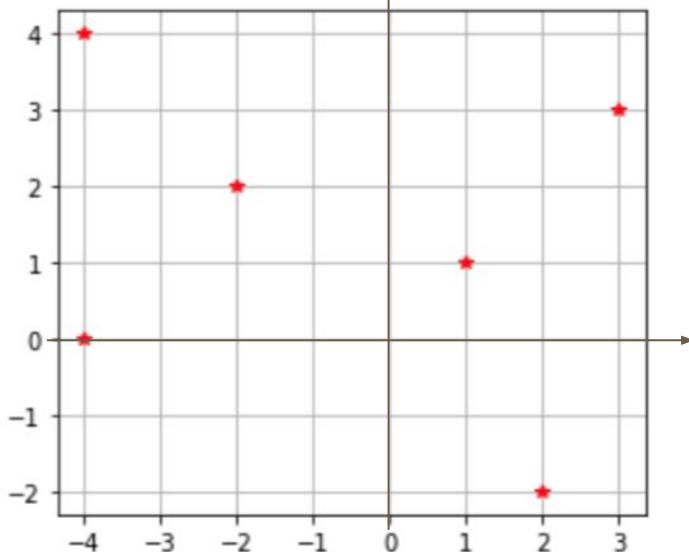
Find an orthogonal basis for the data points on the columns of

$$\begin{bmatrix} -4 & -2 & 1 & 3 & -4 & -2 \\ 0 & 2 & 1 & 3 & 4 & -2 \end{bmatrix}$$

Exercise:

Should we rush into the Gram-Schmidt right away or first analyze data?

$$\begin{bmatrix} -4 & -2 & 1 & 3 & -4 & -2 \\ 0 & 2 & 1 & 3 & 4 & -2 \end{bmatrix}$$



Note that choice of the first column as the first basis is OK but not optimal

Write the coordinates of given points w.r.t. the following bases and observe the difference: *colab might help*

- a. column 1 and column 2
- b. column 2 and column 3

Exercise: Find a *basis*

Is orthogonal always the best?

$$\begin{bmatrix} 4 & 2 & -1 & -8 & 12 & 1 \\ 2 & 6 & -3 & -4 & 6 & 3 \\ 5 & 4 & -2 & -10 & 15 & 2 \end{bmatrix}$$

Find an orthogonal basis and write down the coordinates wrt to the this basis

Exercise: Find a *basis*

Is orthogonal always the best?

$$\begin{bmatrix} 4 & 2 & -1 & -8 & 12 & 1 \\ 2 & 6 & -3 & -4 & 6 & 3 \\ 5 & 4 & -2 & -10 & 15 & 2 \end{bmatrix}$$

Find an orthogonal basis and write down the coordinates wrt to the this basis

First attempt: $\mathbf{M} = \mathbf{Mb}^{-1} * \mathbf{M}_{\text{new}} \rightarrow \mathbf{M}_{\text{new}} = \mathbf{Mb}^{-1} * \mathbf{M}$

```
M = np.array([[4,2,-1,-8,12,1], [2,6,-3,-4,6,3],[5,4,-2,-10,15,2]])
Mb = orth(M)
Mnew = inv(Mb) @ M
```

```
ValueError                                     Traceback (most recent call last)
<ipython-input-11-f7786f902e17> in <cell line: 6>()
      4 M = np.array([[4,2,-1,-8,12,1], [2,6,-3,-4,6,3],[5,4,-2,-10,15,2]])
      5 Mb = orth(M)
----> 6 Mnew = inv(Mb) @ M
      7 Mnew = inv(Mb.T@Mb)@Mb.T@ M
      8 #print(f'new basis:\n{n{np.around(Mb,2)}}')

/usr/local/lib/python3.10/dist-packages/scipy/linalg/_basic.py in inv(a, overwrite_a, check_finite)
    940     al = _asarray_validated(a, check_finite=check_finite)
    941     if len(al.shape) != 2 or al.shape[0] != al.shape[1]:
--> 942         raise ValueError('expected square matrix')
    943     overwrite_a = overwrite_a or _datacopied(al, a)
    944     getrf, getri, getri_lwork = get_lapack_funcs(('getrf', 'getri',
```

ValueError: expected square matrix



Exercise: Find a *basis*

Is orthogonal always the best?

$$\begin{bmatrix} 4 & 2 & -1 & -8 & 12 & 1 \\ 2 & 6 & -3 & -4 & 6 & 3 \\ 5 & 4 & -2 & -10 & 15 & 2 \end{bmatrix}$$

Find an orthogonal basis and write down the coordinates wrt to the this basis

```
M = np.array([[4,2,-1,-8,12,1], [2,6,-3,-4,6,3],[5,4,-2,-10,15,2]])  
Mb = orth(M)  
Mnew = inv(Mb.T@Mb)@Mb.T@ M
```

We know $(M \circ)^{-1}$ did not work

New coordinates of data matrix M
| 4.00 2.00 -1.00 -8.00 12.00 1.00 |
| 2.00 6.00 -3.00 -4.00 6.00 3.00 |
| 5.00 4.00 -2.00 -10.00 15.00 2.00 |

with respect to basis

| -0.57 0.38 |
| -0.36 -0.91 |
| -0.74 0.14 |

are

| -6.70 -6.23 3.12 13.39 -20.09 -3.12 |
| 0.41 -4.15 2.07 -0.83 1.24 -2.07 |

What is interesting here?

Pay attention to the dimensions of the new coordinate matrix!!!

Exercise: Find a *basis*

Is orthogonal always the best?

Find a better basis but how?

$$\begin{bmatrix} 4 & 2 & -1 & -8 & 12 & 1 \\ 2 & 6 & -3 & -4 & 6 & 3 \\ 5 & 4 & -2 & -10 & 15 & 2 \end{bmatrix}$$

Exercise: Find a *basis*

$$\begin{bmatrix} 4 & 2 & -1 & -8 & 12 & 1 \\ 2 & 6 & -3 & -4 & 6 & 3 \\ 5 & 4 & -2 & -10 & 15 & 2 \end{bmatrix}$$

May be it is just in front of your eyes?

Find a better basis but how?

```
new coordinates of
[[ 4   2   -1   -8   12   1]
 [ 2   6   -3   -4   6   3]
 [ 5   4   -2  -10   15   2]]
wrt
[[4 2]
 [2 6]
 [5 4]]
are:
[[ 1.   -0.    0.   -2.    3.   -0. ]
 [-0.    1.   -0.5   0.   -0.   0.5]]
```

Exercise: Find a *basis*

Is orthogonal always the best?

$$\begin{bmatrix} 4 & 2 & -1 & -8 & 12 & 1 \\ 2 & 6 & -3 & -4 & 6 & 3 \\ 5 & 4 & -2 & -10 & 15 & 2 \end{bmatrix}$$

Find a better basis but how?

```
[[ -6.7    -6.23     3.12   13.39  -20.09   -3.12]
 [  0.41   -4.15     2.07   -0.83    1.24   -2.07]]
```

vs

```
[[ 1.   -0.    0.   -2.    3.   -0. ]
 [-0.    1.   -0.5   0.   -0.    0.5]]
```

STILL HOW if this was a higher dimensional much bigger matrix?

Recall: Orthogonal decomposition & Projection

Let $\mathbf{a}, \mathbf{b} \in \mathbb{R}^2$ and \mathbf{a}_b is the component of \mathbf{a} along \mathbf{b} , and $\mathbf{a}_{b\perp}$ is the component of \mathbf{a} that is perpendicular to \mathbf{b} .

Question is, given \mathbf{a} and \mathbf{b} how can you find \mathbf{a}_b and $\mathbf{a}_{b\perp}$?

Note that:

$$\mathbf{a} = \mathbf{a}_b + \mathbf{a}_{b\perp}$$

Therefore, if we find \mathbf{a}_b , then,

$$\mathbf{a}_{b\perp} = \mathbf{a} - \mathbf{a}_b$$

Finding \mathbf{a}_b then simply is a unit projection vector of \mathbf{a} along \mathbf{b} and multiplying it with the $\|\mathbf{a}_b\|$.

Note that:

$$\|\mathbf{a}_b\| = \|\mathbf{a}\| \cos(\theta)$$

Recall by definition:

$$\langle \mathbf{a}, \mathbf{b} \rangle = \|\mathbf{a}\| \|\mathbf{b}\| \cos(\theta)$$

Hence,

$$\|\mathbf{a}_b\| = \frac{\langle \mathbf{a}, \mathbf{b} \rangle}{\|\mathbf{b}\|}$$

If this magnitude is multiplied by a unit vector along \mathbf{b} we are done with \mathbf{a}_b as:

$$\mathbf{a}_b = \frac{\langle \mathbf{a}, \mathbf{b} \rangle}{\|\mathbf{b}\|} \frac{\mathbf{b}}{\|\mathbf{b}\|} = \frac{\langle \mathbf{a}, \mathbf{b} \rangle}{\|\mathbf{b}\|^2} \mathbf{b} = \frac{\langle \mathbf{a}, \mathbf{b} \rangle}{\langle \mathbf{b}, \mathbf{b} \rangle} \mathbf{b}$$

Finally:

$$\mathbf{a}_{b\perp} = \mathbf{a} - \mathbf{a}_b = \mathbf{a} - \frac{\langle \mathbf{a}, \mathbf{b} \rangle}{\|\mathbf{b}\|^2} \mathbf{b}$$

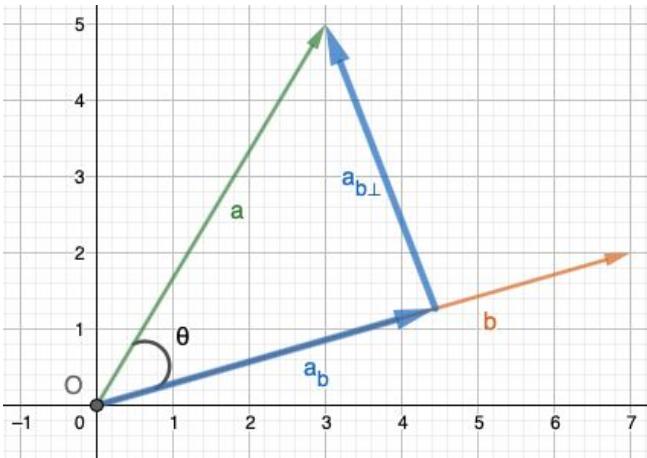
Also note that:

$$\mathbf{a}_{b\perp} = \mathbf{a} - \frac{\langle \mathbf{a}, \mathbf{b} \rangle}{\langle \mathbf{b}, \mathbf{b} \rangle} \mathbf{b}$$

If \mathbf{b} was already unit, things would have been easier:

$$\mathbf{a}_b = \langle \mathbf{a}, \mathbf{b} \rangle \mathbf{b}$$

$$\mathbf{a}_{b\perp} = \mathbf{a} - \langle \mathbf{a}, \mathbf{b} \rangle \mathbf{b}$$

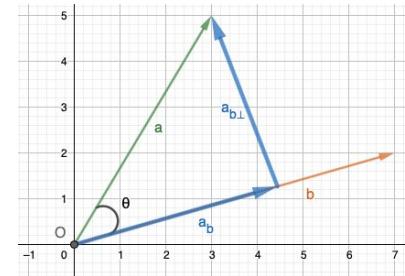
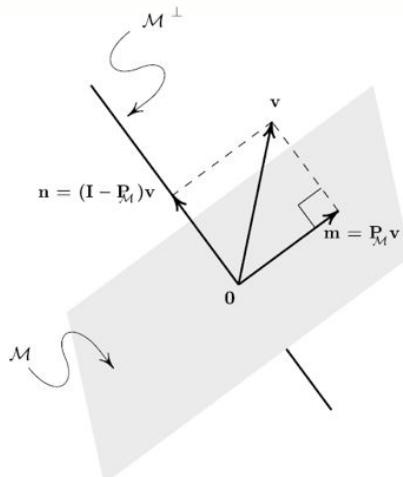


Orthogonal Projection

Orthogonal Projection

For $\mathbf{v} \in \mathcal{V}$, let $\mathbf{v} = \mathbf{m} + \mathbf{n}$, where $\mathbf{m} \in \mathcal{M}$ and $\mathbf{n} \in \mathcal{M}^\perp$.

- \mathbf{m} is called the *orthogonal projection* of \mathbf{v} onto \mathcal{M} .
- The projector $\mathbf{P}_{\mathcal{M}}$ onto \mathcal{M} along \mathcal{M}^\perp is called the *orthogonal projector* onto \mathcal{M} .
- $\mathbf{P}_{\mathcal{M}}$ is the unique linear operator such that $\mathbf{P}_{\mathcal{M}}\mathbf{v} = \mathbf{m}$



What if you project \mathbf{m} again with $\mathbf{P}_{\mathcal{M}}$?

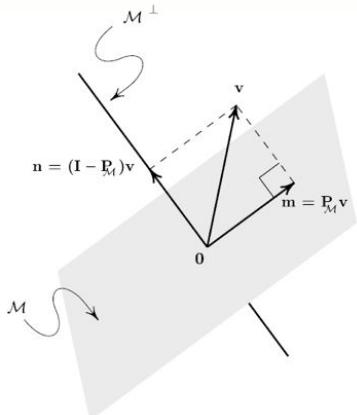
Again and again and ...

Orthogonal Projection

Orthogonal Projection

For $\mathbf{v} \in \mathcal{V}$, let $\mathbf{v} = \mathbf{m} + \mathbf{n}$, where $\mathbf{m} \in \mathcal{M}$ and $\mathbf{n} \in \mathcal{M}^\perp$.

- \mathbf{m} is called the *orthogonal projection* of \mathbf{v} onto \mathcal{M} .
- The projector $\mathbf{P}_{\mathcal{M}}$ onto \mathcal{M} along \mathcal{M}^\perp is called the *orthogonal projector* onto \mathcal{M} .
- $\mathbf{P}_{\mathcal{M}}$ is the unique linear operator such that $\mathbf{P}_{\mathcal{M}}\mathbf{v} = \mathbf{m}$



Finding the projection matrix \mathbf{P}_M

Recall from previous slides (with a bit of name swapping) that a vector \mathbf{v} can be projected on another vector \mathbf{a}_i as:

$$\mathbf{v}_{a_i} = \frac{\langle \mathbf{v}, \mathbf{a}_i \rangle}{\langle \mathbf{a}_i, \mathbf{a}_i \rangle} \mathbf{a}_i$$

If the columns of matrix \mathbf{A} are orthonormal and span the subspace \mathbf{M} , then the projection matrix \mathbf{P}_M can be written as follows:

$$\mathbf{P}_M = \mathbf{AA}^T$$

Without columns of \mathbf{A} satisfying orthonormality condition, the projection matrix \mathbf{P}_M can be written in general as follows:

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

Proof (of at least the special case) is left to you as an exercise or is it?

Note that:

If \mathbf{v} was already in span of \mathbf{A} , no information is lost, or in other words, $\mathbf{v}_{M^\perp} = \mathbf{0}$

ME 536

— Weeks 4: Some more math
2 remember —

Matrices

Collection of vectors, where a vector generally represents a data point.

Vectors can be in the rows or columns of the **matrix**.

Convention for the rest of the course:

*Columns of a data matrix represent **data points** in the rest of the class unless otherwise mentioned*

*Let's refresh our memories on **Matrices** and **Vectors** in action*

Matrix multiplied by a Vector

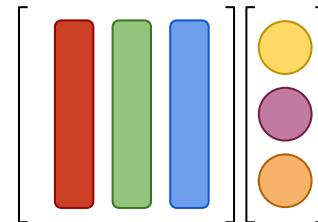
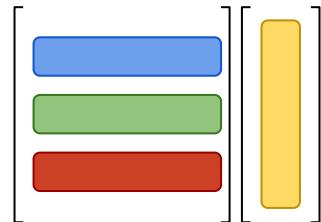
$$\mathbf{M}\mathbf{x} = \mathbf{a}$$

Vector multiplies the matrix on the right.

Recall: \mathbf{M} is the matrix and \mathbf{x} , \mathbf{a} are vectors.

Matrix multiplied by a Vector: 2 Main Interpretations

$$\mathbf{M}\mathbf{x} = \mathbf{a}$$



$$\mathbf{a}^T = \begin{bmatrix} \text{Blue Bar} & \cdot & \text{Yellow Bar} \\ \text{Green Bar} & \cdot & \text{Yellow Bar} \\ \text{Red Bar} & \cdot & \text{Yellow Bar} \end{bmatrix}$$

Dot (or inner) product version

$$\mathbf{a} = \text{Yellow Circle} + \text{Purple Circle} + \text{Orange Circle} \times \text{Blue Bar}$$

Linear Combination version

Generally we are used to: square \mathbf{M} with no rank deficiency, i.e. \mathbf{M} is full rank

Row and Column Space of a Matrix

$$\begin{bmatrix} 4 & 2 & -1 & -8 & 12 & 1 \\ 2 & 6 & -3 & -4 & 6 & 3 \\ 5 & 4 & -2 & -10 & 15 & 2 \end{bmatrix}$$

Given \mathbf{M}

$$\begin{bmatrix} -4 & -2 & 1 & 3 & -4 & -2 \\ 0 & 2 & 1 & 3 & 4 & -2 \end{bmatrix}$$

Column space of \mathbf{M} : $\mathbf{C}(\mathbf{M})$

Spanned by columns of \mathbf{M}

Row space of \mathbf{M} : $\mathbf{R}(\mathbf{M})$

Spanned by rows of \mathbf{M}

Row and Column Space of a Matrix

Column space of \mathbf{M} : $\mathbf{C}(\mathbf{M}) \rightarrow$ Spanned by columns of \mathbf{M}

Row space of \mathbf{M} : $\mathbf{R}(\mathbf{M}) \rightarrow$ Spanned by rows of \mathbf{M}

$$\mathbf{C}(\mathbf{M}) \stackrel{?}{=} \mathbf{R}(\mathbf{M})$$

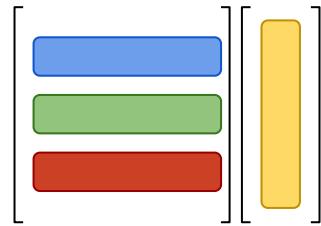
1- Fix for general case if it is not always True

2- Special case when this is correct?

Matrix multiplied by a Vector: 2 Main Interpretations

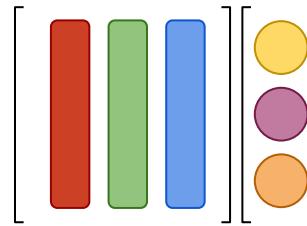
$$\mathbf{Mx} = \mathbf{0}$$

What if the trivial is not the only alternative: $\mathbf{x} \neq \mathbf{0}$



Dot (or inner) product version

$$0 = \begin{array}{|c|} \hline \text{blue} \\ \hline \end{array} \cdot \begin{array}{|c|} \hline \text{yellow} \\ \hline \end{array} = \begin{array}{|c|} \hline \text{green} \\ \hline \end{array} \cdot \begin{array}{|c|} \hline \text{yellow} \\ \hline \end{array} = \begin{array}{|c|} \hline \text{red} \\ \hline \end{array} \cdot \begin{array}{|c|} \hline \text{yellow} \\ \hline \end{array}$$



Linear Combination version

$$0 = \begin{array}{|c|} \hline \text{yellow} \\ \hline \end{array} + \begin{array}{|c|} \hline \text{purple} \\ \hline \end{array} + \begin{array}{|c|} \hline \text{orange} \\ \hline \end{array} + \begin{array}{|c|} \hline \text{blue} \\ \hline \end{array}$$
$$- \begin{array}{|c|} \hline \text{red} \\ \hline \end{array} = \frac{\begin{array}{|c|} \hline \text{yellow} \\ \hline \end{array}}{\cancel{\text{yellow}}} + \frac{\begin{array}{|c|} \hline \text{purple} \\ \hline \end{array}}{\cancel{\text{yellow}}} + \frac{\begin{array}{|c|} \hline \text{orange} \\ \hline \end{array}}{\cancel{\text{yellow}}} + \frac{\begin{array}{|c|} \hline \text{blue} \\ \hline \end{array}}{\cancel{\text{yellow}}}$$

$\mathbf{Mx} = \mathbf{0}$: Different Cases

Let:

$$\text{rank } (\mathbf{M}_{3 \times 3}) = 2$$

$$\begin{bmatrix} \text{Red} & \text{Green} & \text{Blue} \end{bmatrix} \begin{bmatrix} \text{Yellow} \\ \text{Purple} \\ \text{Orange} \end{bmatrix}$$

$$\text{Red} = \text{Purple} + \text{Red}$$

$$\text{rank } (\mathbf{M}_{3 \times 3}) = 1$$

$$\begin{bmatrix} \text{Red} & \text{Green} & \text{Blue} \end{bmatrix} \begin{bmatrix} \text{Yellow} \\ \text{Purple} \\ \text{Orange} \end{bmatrix}$$

$$\begin{array}{l} \text{Red} = \text{Purple} \\ \text{Blue} = \text{Red} \end{array}$$

What does $C(\mathbf{M})$ represent in each case?

What about the basis of \mathbf{M} ?

$\mathbf{Mx} = \mathbf{0}$: NULL Space - $N(\mathbf{M})$

Let:

$$\text{rank } (\mathbf{M}_{3 \times 3}) = 2$$

$$\begin{bmatrix} \text{Red} & \text{Green} & \text{Blue} \end{bmatrix} \begin{bmatrix} \text{Yellow} \\ \text{Purple} \\ \text{Orange} \end{bmatrix}$$

$$\text{Red} = \text{Purple} + \text{Red}$$

$$\text{rank } (\mathbf{M}_{3 \times 3}) = 1$$

$$\begin{bmatrix} \text{Red} & \text{Green} & \text{Blue} \end{bmatrix} \begin{bmatrix} \text{Yellow} \\ \text{Purple} \\ \text{Orange} \end{bmatrix}$$

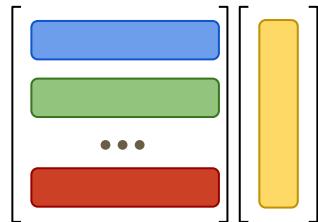
$$\begin{aligned} \text{Red} &= \text{Purple} \\ \text{Blue} &= \text{Red} \end{aligned}$$

Where does \mathbf{x} come from? Where do all \mathbf{x} live?

Dimension of the $N(\mathbf{M})$? Number of free Parameters?

WRAP UP for $N(\mathbf{M}_{m \times n})$

Let: $\mathbf{M}_{m \times n}$



$$N(\mathbf{M}) \perp R(\mathbf{M})$$

similarly

$$N(\mathbf{M}) \perp C(\mathbf{M}^T)$$

$$0 = \begin{array}{c|c} \text{blue} & \text{yellow} \end{array} \cdot \begin{array}{c|c} \text{green} & \text{yellow} \end{array} = \begin{array}{c|c} \text{green} & \text{yellow} \end{array} \cdot \dots = \begin{array}{c|c} \text{red} & \text{yellow} \end{array} \cdot \dots = \dots$$

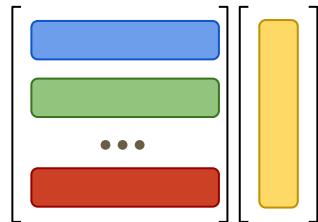
where

$$\dim(R(\mathbf{M})) = \dim(C(\mathbf{M}^T)) = \text{rank}(\mathbf{M}) = r$$

$$\dim(N(\mathbf{M})) = ?$$

WRAP UP for $N(M_{mxn})$

Similarly for M^T_{nxm}



$$N(M^T) \perp R(M^T)$$

hence

$$N(M^T) \perp C(M)$$

$$0 = \begin{matrix} \text{blue bar} \\ \cdot \\ \text{orange bar} \end{matrix} = \begin{matrix} \text{green bar} \\ \cdot \\ \text{orange bar} \end{matrix} = \cdots = \begin{matrix} \text{red bar} \\ \cdot \\ \text{orange bar} \end{matrix}$$

where

$$\dim(R(M^T)) = \dim(C(M)) = \text{rank}(M) = r$$

$$\dim(N(M^T)) = ?$$

Time Capsule: note to the Future

Recall that choosing a basis **from the data** matrix \mathbf{M} is:

- Practical
- Yet subject to *ill-conditioned* cases!
 - For a square matrix, you can check for very small $|\cdot|$
 - But what if the Basis vectors do not form a square matrix?

Time Capsule: note to the Future

Given a full rank square matrix $\mathbf{A}_{n \times n}$

We can consider this as a mapping from \mathbb{R}^n to \mathbb{R}^n , i.e. $\mathbf{A} : \mathbb{R}^n \rightarrow \mathbb{R}^n$

| For $\mathbf{Ax} = \mathbf{a}$, we have $\mathbf{x}, \mathbf{a} \in \mathbb{R}^n$

What if $\mathbf{A}_{n \times m}$ is not square?

Then we have a mapping from \mathbb{R}^m to \mathbb{R}^n , i.e. $\mathbf{A} : \mathbb{R}^m \rightarrow \mathbb{R}^n$

| For $\mathbf{Ax} = \mathbf{a}$, we have $\mathbf{x} \in \mathbb{R}^m$, and $\mathbf{a} \in \mathbb{R}^n$

Consider cases:

1. $\mathbf{A}_{3 \times 2}$
2. $\mathbf{A}_{200 \times 10}$
3. $\mathbf{A}_{2 \times 3}$

Can you move back and forth?

Try to think about it, drawing might help

A famous Space: \mathbb{R}^n

- ◆ All of us know a very well-known vector space: \mathbb{R}^n

For $x = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$, we know that the length of vector x

$$\|x\| = \left(\sum_{k=1}^n |x_k|^2 \right)^{1/2}$$

- ◆ For general vector spaces, we need a concept that corresponds to length in \mathbb{R}^n
- ◆ We use “norm” instead of “length”

Good news:

Our data will almost exclusively live in \mathbb{R}^n

So no weird brain freezing spaces

A famous Norm: length

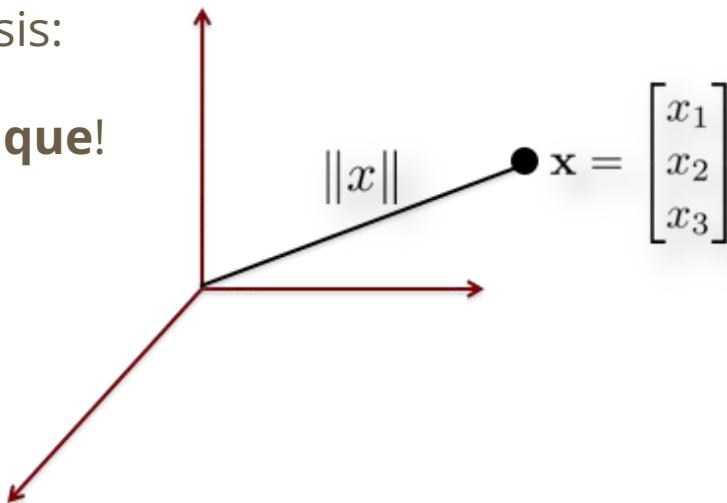
Euclidean Vector Norm

For a vector $x \in \mathbb{R}^n$, the euclidean norm is defined by:

$$\|x\| = (x_1^2 + x_2^2 + \dots + x_n^2)^{1/2}$$

Just like the standard basis:

- *Famous, but not unique!*



In general: Norm

A norm for a real or complex vector space \mathcal{V} is a function

$$\| \cdot \| : \mathcal{V} \rightarrow \mathbb{R}$$

with three properties:

1. $\|x\| \geq 0$ for all $x \in V$ and $\|x\| = 0 \iff x = 0$
2. $\|\alpha x\| = |\alpha| \|x\|$ for all scalars α and for all $x \in \mathcal{V}$
3. $\|x + y\| \leq \|x\| + \|y\|$ for all $x, y \in \mathcal{V}$

p-Norms

p-Norms

For $p \geq 1$, the p-norm of a vector $x \in \mathbb{R}^n$ is defined by:

$$\|x\|_p = (\|x_1\|^p + \|x_2\|^p + \dots + \|x_n\|^p)^{1/p}$$

$\|x\|_1$ norm → • $\|x\|_1$ - a.k.a Taxicab or Manhattan norm

$\|x\|_2$ norm → • $\|x\|_2$ - Euclidean norm

...

$\|x\|_\infty$ norm → • $\|x\|_\infty$ - $\max(|x_1|, \dots, |x_n|)$

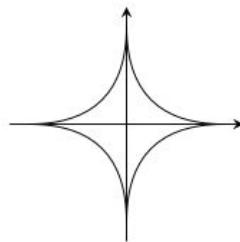
Even though value of p determines the norm, *in general* they are referred to as $\|x\|_p$ norm. (with a *lowercase L*)

NOTE: shape of $\|\cdot\|_p$ norms:

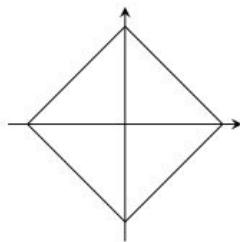
What are the set of **points with unit norm** look like:

$$\|\cdot\|_p = 1$$

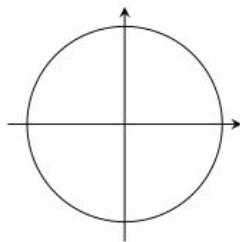
for different values of $p = \{\frac{1}{2}, 1, 2, \infty\}$ in 2D.



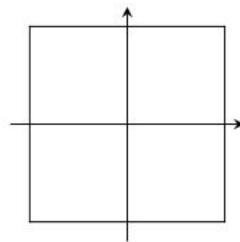
$$p = \frac{1}{2}$$



$$p = 1$$

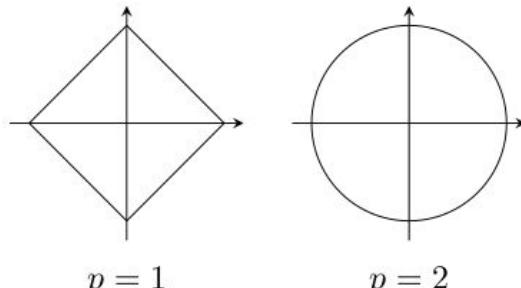


$$p = 2$$



$$p = \infty$$

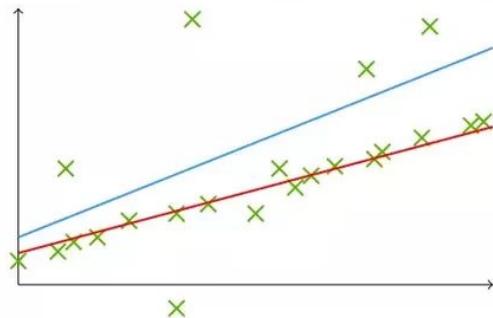
NOTE: ℓ_1 over ℓ_2 ?



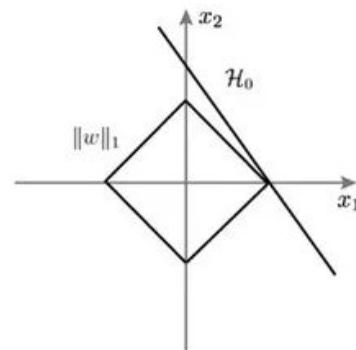
Given : A set of **points** in 2-dimension

Goal : Find a line to fit those points

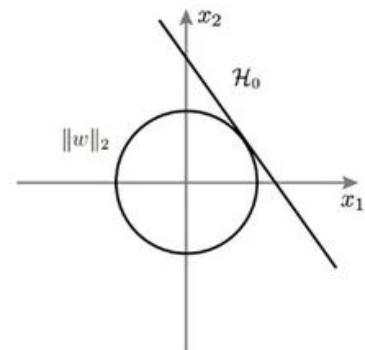
Output : ℓ_2 minimizer **line**, ℓ_1 minimizer **line**



A L1 regularization



B L2 regularization



NOTE: l_0 norm

Despite the fact that l_0 norm is **not a norm by definition** but it is very often *used in practice*.

l_0 norm is defined by the **number of non-zero elements** in the vector.

→ i.e. minimizing l_0 results in sparse solution

NOTE: yet another norm

Note that if \mathbf{D} is a symmetric matrix with positive diagonal terms, then for the vector \mathbf{v} :

$$\mathbf{v}^T \mathbf{D} \mathbf{v}$$

defines a norm in the form of a weighted sum of squares.

Example: a 2x2 case

$$\mathbf{D} = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}, \mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

$$\mathbf{v}^T \mathbf{D} \mathbf{v} = av_1^2 + bv_2^2$$

Good news:

Every finite dimensional real or complex topological vector space has a norm

So no hunt for finding norms, well mostly...

A famous Inner Product in: \mathbb{R}^n a.k.a *DOT Product*

\mathbb{R}^n : Collection of all finite sequences

$$x = (x_1, x_2, \dots, x_n), \quad x_k \in \mathbb{R} \text{ for } k = 1, 2, \dots, n$$

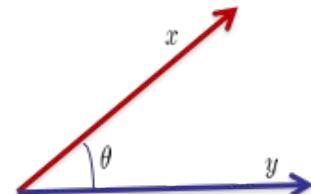
We can define norm for $\|x\| = (\sum_{k=1}^n x_k^2)^{1/2}$

Recall: inner product of $x, y \in \mathbb{R}^n$, $\langle x, y \rangle = \sum_{k=1}^n x_k y_k$

Clearly: $\langle x, x \rangle = \|x\|^2$

Inner product is a very important tool for analysis in \mathbb{R}^n .

It is a measure of angle between vectors



$$\langle x, y \rangle = \|x\| \|y\| \cos \theta$$

In general: Inner Product

Definition: An *inner product* on a real (or complex) vector space \mathcal{V} is a function that maps each ordered pairs of vectors v and w to a real (or complex) scalar $\langle v, w \rangle$ such that the following properties hold:

1. $\langle v, v \rangle$ is real with $\langle v, v \rangle \geq 0$ for all $v \in V$ and $\langle v, v \rangle = 0 \iff v = 0$
2. $\langle v, \alpha w \rangle = \bar{\alpha} \langle v, w \rangle$ for all scalars α and for all $v, w \in \mathcal{V}$
3. $\langle v, w + u \rangle = \langle v, w \rangle + \langle v, u \rangle$ for all $v, u, w \in \mathcal{V}$
4. $\langle v, w \rangle = \overline{\langle w, v \rangle}$ for all $v, w \in \mathcal{V}$ (note that $\langle v, w \rangle = \langle w, v \rangle$ for real spaces)

Brain teaser:

Try to come up with *your custom*:

- norm for vectors in \mathbb{R}^n
with three properties:
 1. $\|x\| \geq 0$ for all $x \in V$ and $\|x\| = 0 \iff x = 0$
 2. $\|\alpha x\| = |\alpha| \|x\|$ for all scalars α and for all $x \in \mathcal{V}$
 3. $\|x + y\| \leq \|x\| + \|y\|$ for all $x, y \in \mathcal{V}$
- inner product for vector pairs in \mathbb{R}^n
 1. $\langle v, v \rangle$ is real with $\langle v, v \rangle \geq 0$ for all $v \in V$ and $\langle v, v \rangle = 0 \iff v = 0$
 2. $\langle v, \alpha w \rangle = \bar{\alpha} \langle v, w \rangle$ for all scalars α and for all $v, w \in \mathcal{V}$
 3. $\langle v, w + u \rangle = \langle v, w \rangle + \langle v, u \rangle$ for all $v, u, w \in \mathcal{V}$
 4. $\langle v, w \rangle = \overline{\langle w, v \rangle}$ for all $v, w \in \mathcal{V}$ (note that $\langle v, w \rangle = \langle w, v \rangle$ for real spaces)

Inner Product Spaces

Simple Recipe: *A vector space with an Inner Product*

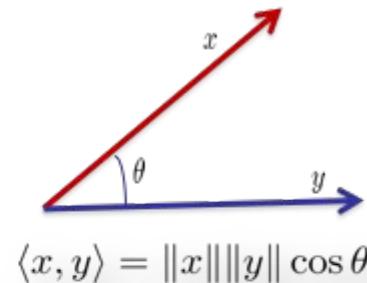
Examples: \mathbb{R}^n with $\langle x, y \rangle = \sum_{k=1}^n x_k y_k$

\mathbb{C}^n with $\langle x, y \rangle = \sum_{k=1}^n x_k \overline{y_k}$

Cauchy-Schwarz Inequality

Inner product for \mathbb{R}^2

$$\langle x, y \rangle = \|x\| \|y\| \cos(\theta)$$



$$|\langle x, y \rangle| \leq \|x\| \|y\| = \langle x, x \rangle^{1/2} \langle y, y \rangle^{1/2}$$

For a general normed vector space \mathcal{V}

Theorem. $|\langle v, w \rangle| \leq \langle v, v \rangle^{1/2} \langle w, w \rangle^{1/2}, \quad \forall v, w \in \mathcal{V}$

Good to know

- ◆ In any inner product vector space, regardless of the inner product we can always define a norm

Lemma. *If \mathcal{V} has the inner product $\langle ., . \rangle$, then*

$$\|v\| = |\langle v, v \rangle|^{1/2} \text{ is a norm}$$

- ◆ But opposite is not true. We may not always define an inner product from a given norm

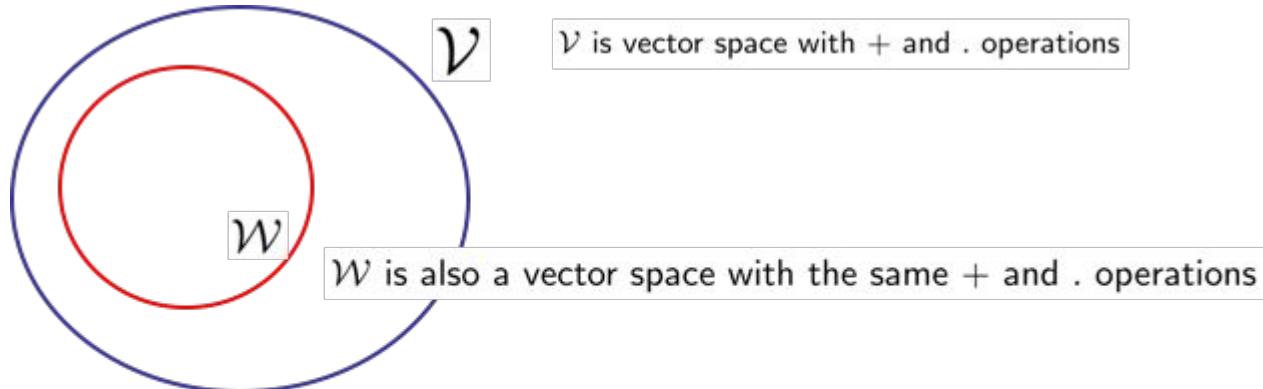
But...

- ◆ We may define an inner product from a given norm if the parallelogram law holds for the norm

$$\|v + w\|^2 + \|v - w\|^2 = 2 (\|v\|^2 + \|w\|^2)$$

for all $v, w \in \mathcal{V}$

Subspaces

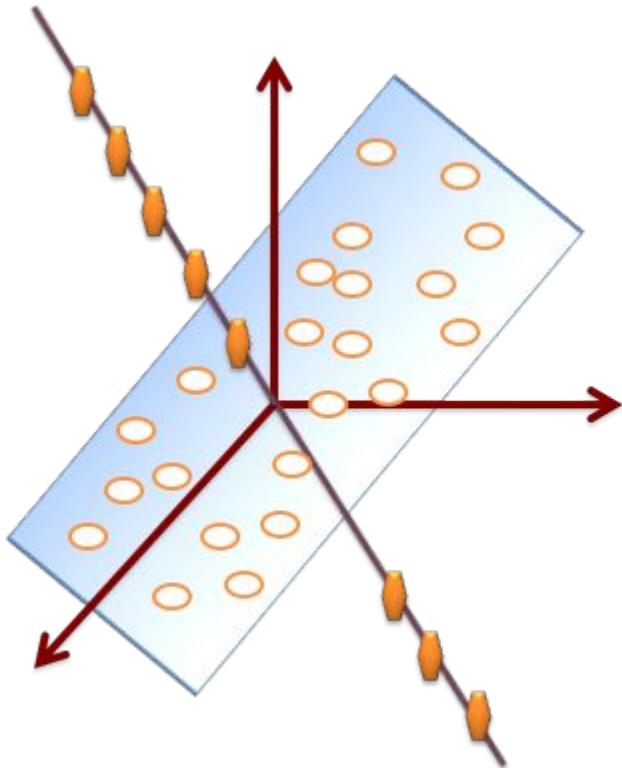


Lemma. *A subset $\mathcal{W} \subset \mathcal{V}$ is a subspace if $\alpha, \beta \in \mathcal{F}$ and $v, w \in \mathcal{W} \Rightarrow \alpha v + \beta w \in \mathcal{W}$*

Practical consequence:

Any *line, plane, etc* are **subspaces** if they **include the origin**

Subspaces: Examples



A line through
origin in \mathbb{R}^3 is a 1-
dimensional
subspace of \mathbb{R}^3

A plane through
origin in \mathbb{R}^3 is a 2-
dimensional
subspace of \mathbb{R}^3

Sum of Subspaces

If \mathcal{X} and \mathcal{Y} are subspaces of a vector space \mathcal{V} , then the *sum* of \mathcal{X} and \mathcal{Y} is defined as all possible sums of vectors from \mathcal{X} and \mathcal{Y} :

$$\mathcal{X} + \mathcal{Y} = \{x + y : x \in \mathcal{X} \text{ and } y \in \mathcal{Y}\}$$

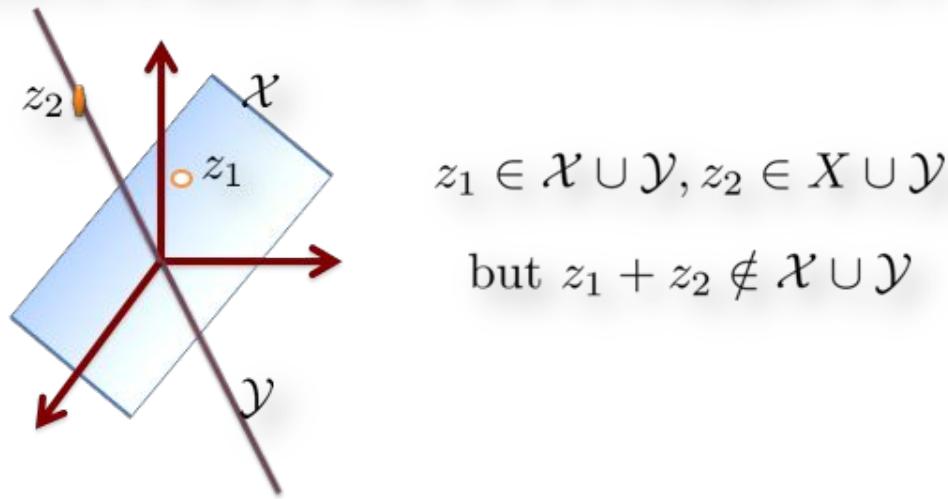
The sum of \mathcal{X} and \mathcal{Y} is a subspace of \mathcal{V}

Union of Subspaces

If \mathcal{X} and \mathcal{Y} are subspaces of a vector space \mathcal{V} , then the *union* of \mathcal{X} and \mathcal{Y} is defined:

$$\mathcal{X} \cup \mathcal{Y} = \{z : z \in \mathcal{X} \text{ or } z \in \mathcal{Y}\}$$

The union of \mathcal{X} and \mathcal{Y} may not be a subspace of \mathcal{V}

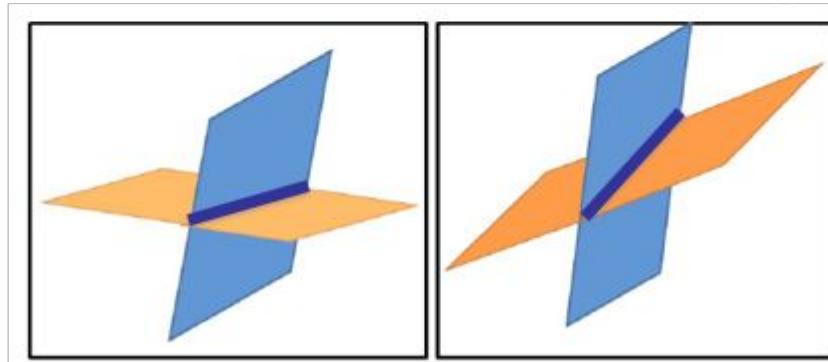


Intersection of Subspaces

If \mathcal{X} and \mathcal{Y} are subspaces of a vector space \mathcal{V} , then the *intersection* of \mathcal{X} and \mathcal{Y} is defined:

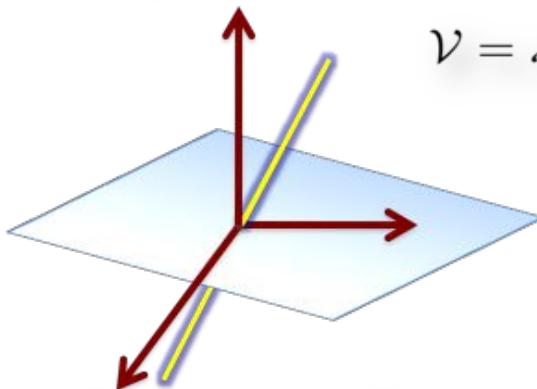
$$\mathcal{X} \cap \mathcal{Y} = \{z : z \in \mathcal{X} \text{ and } z \in \mathcal{Y}\}$$

The intersection of \mathcal{X} and \mathcal{Y} is a subspace of \mathcal{V}



Complementary Subspaces

Subspaces \mathcal{X} and \mathcal{Y} a space \mathcal{V} are *complementary* if



$$\mathcal{V} = \mathcal{X} + \mathcal{Y} \text{ and } \mathcal{X} \cap \mathcal{Y} = \emptyset$$

In this case, \mathcal{V} is said to be the *direct sum* of \mathcal{X} and \mathcal{Y}

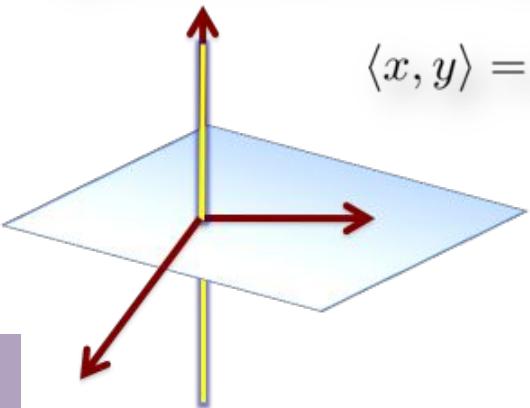
$$\mathcal{V} = \mathcal{X} \oplus \mathcal{Y}$$

For each $v \in \mathcal{V}$, there are *unique* vectors $x \in \mathcal{X}$ and $y \in \mathcal{Y}$
such that $v = x + y$

Orthogonal Subspaces

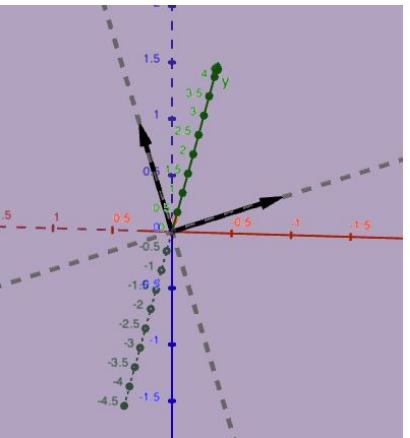
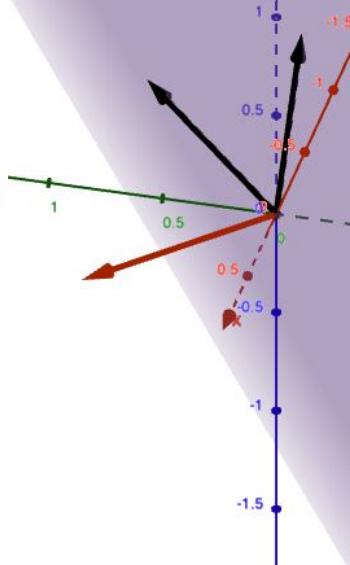
If \mathcal{X} and \mathcal{Y} are subspaces of an *inner-product* vector space \mathcal{V} , then \mathcal{X} and \mathcal{Y} are orthogonal $\mathcal{X} \perp \mathcal{Y}$ if

$$\langle x, y \rangle = 0 \text{ for all } x \in \mathcal{X} \text{ and } y \in \mathcal{Y}$$



If \mathcal{X} is a subspace of a finite dimensional inner-product space \mathcal{V} , then \mathcal{X}^\perp is its orthogonal complement if

$$\mathcal{V} = \mathcal{X} \bigoplus \mathcal{X}^\perp$$



Independent vs Disjoint

Linear subspaces $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_k$ of \mathbb{R}^n are independent

if and only if

$$\dim(S_1 + S_2 + \dots + S_k) = \dim(S_1) + \dim(S_2) + \dots + \dim(S_k)$$

Linear subspaces $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_k$ of \mathbb{R}^n are disjoint

if they intersect only at the origin

Independence is stronger than disjointedness

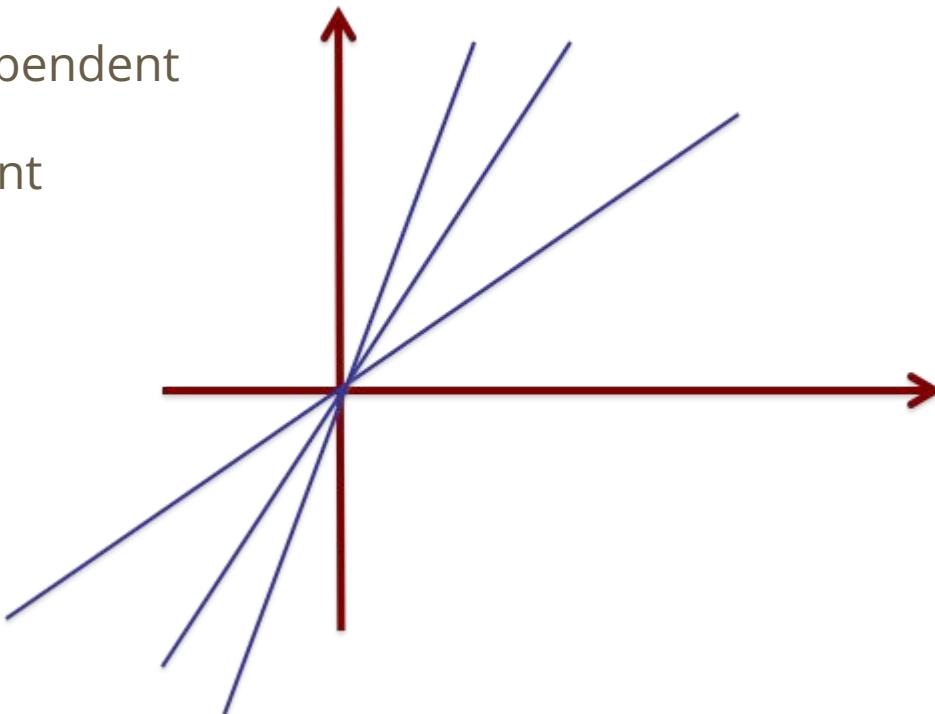
Three lines in \mathbb{R}^2 intersecting at 0 are disjoint

But they are not independent

Independent vs Disjoint

In \mathbb{R}^2 any 2 lines are independent

3 or more lines are disjoint

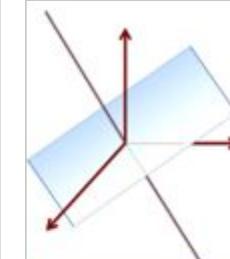


Minimal Angle

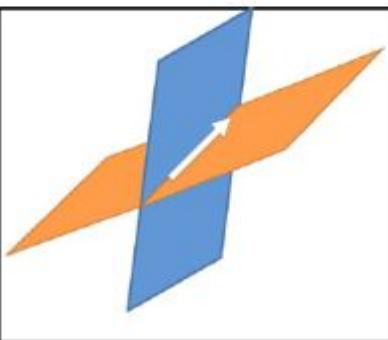
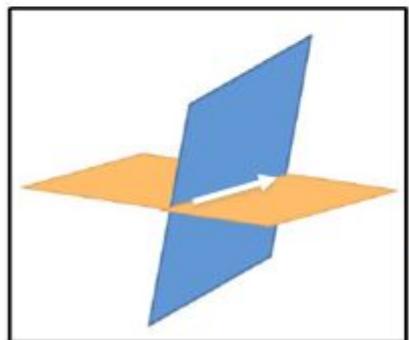
Minimum Angle

Let \mathcal{F} and \mathcal{G} be subspaces of \mathbb{R}^D . The minimal angle between \mathcal{F} and \mathcal{G} is defined as

$$\theta_{\min} = \arccos \left[\max_{\substack{f \in \mathcal{F} \\ g \in \mathcal{G} \\ \|f\|_2 = \|g\|_2 = 1}} f^T g \right]$$



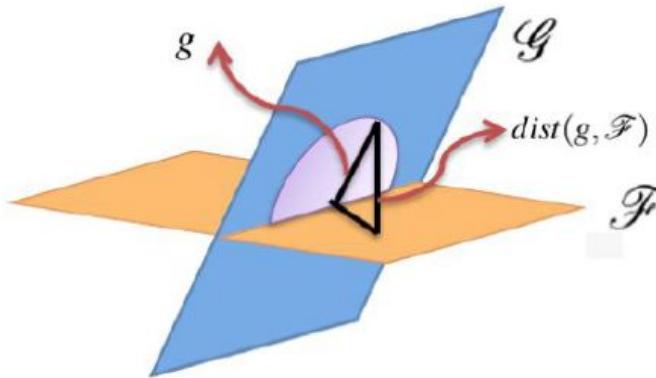
It is a good measure for complementary subspaces



It is not a good measure for non-complementary subspaces

Gap between Subspaces

$$d(\mathcal{F}, \mathcal{G}) = \max_{\substack{g \in \mathcal{G} \\ \|g\|_2=1}} \text{dist}(g, \mathcal{F}) = \max_{\substack{g \in \mathcal{G} \\ \|g\|_2=1}} \|(I - P_{\mathcal{F}})g\|_2$$



$$\text{gap}(\mathcal{F}, \mathcal{G}) = \min(d(\mathcal{F}, \mathcal{G}), d(\mathcal{G}, \mathcal{F}))$$

Maximal Angle

Maximum Angle

The maximal angle between \mathcal{F} and \mathcal{G} is defined as

$$\theta_{\max} = \arcsin(\text{gap}(\mathcal{F}, \mathcal{G})),$$

where $0 \leq \theta_{\max} \leq \pi/2$.

It is useful for subspaces
of equal dimension

Principal Angles: An iterative process

Principle Angles

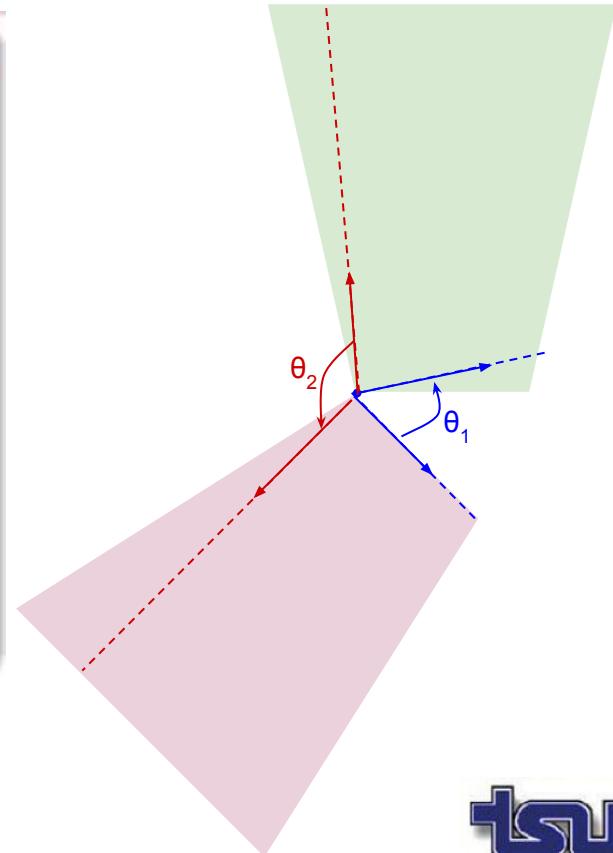
Let \mathcal{F} and \mathcal{G} be subspaces of \mathbb{R}^D . Let $k = \min(\dim \mathcal{F}, \dim \mathcal{G})$.

Then, the principle angles $\theta_1, \theta_2, \dots, \theta_k$ are the numbers

$0 \leq \theta_i \leq \pi/2$ and they are defined as

$$\cos \theta_i = \max_{\substack{f \in \mathcal{F}_i \\ g \in \mathcal{G}_i}} f^t g = f_i^t g_i \quad i = 1, \dots, k$$
$$f_i \|_2 = \|g_i\|_2 = 1$$

where $\mathcal{F}_1 = \mathcal{F}$ and $\mathcal{G}_1 = \mathcal{G}$, $\|f_i\|_2 = 1$, $\|g_i\|_2 = 1$,
 $\mathcal{F}_i = f_{i-1}^\perp \cap \mathcal{F}_{i-1}$, and $\mathcal{G}_i = g_{i-1}^\perp \cap \mathcal{G}_{i-1}$. Note that
 $\theta_1 \leq \theta_2 \leq \dots \leq \theta_k$.



Principal Angles

Principle Angles

Let \mathcal{F} and \mathcal{G} be subspaces of \mathbb{R}^D . Let $k = \min(\dim \mathcal{F}, \dim \mathcal{G})$. Then, the principle angles $\theta_1, \theta_2, \dots, \theta_k$ are the numbers $0 \leq \theta_i \leq \pi/2$ and they are defined as

$$\cos \theta_i = \max_{\substack{f \in \mathcal{F}_i \\ g \in \mathcal{G}_i}} f^t g = f_i^t g_i \quad i = 1, \dots, k$$
$$f_i \|_2 = \|g_i\|_2 = 1$$

where $\mathcal{F}_1 = \mathcal{F}$ and $\mathcal{G}_1 = \mathcal{G}$, $\|f_i\|_2 = 1$, $\|g_i\|_2 = 1$, $\mathcal{F}_i = f_{i-1}^\perp \cap \mathcal{F}_{i-1}$, and $\mathcal{G}_i = g_{i-1}^\perp \cap \mathcal{G}_{i-1}$. Note that $\theta_1 \leq \theta_2 \leq \dots \leq \theta_k$.

Time Capsule: note to the Future

Given two subspaces \mathcal{F} and \mathcal{G} let \mathbf{F}, \mathbf{G} form orthogonal bases for subspaces \mathcal{F} and \mathcal{G} respectively.

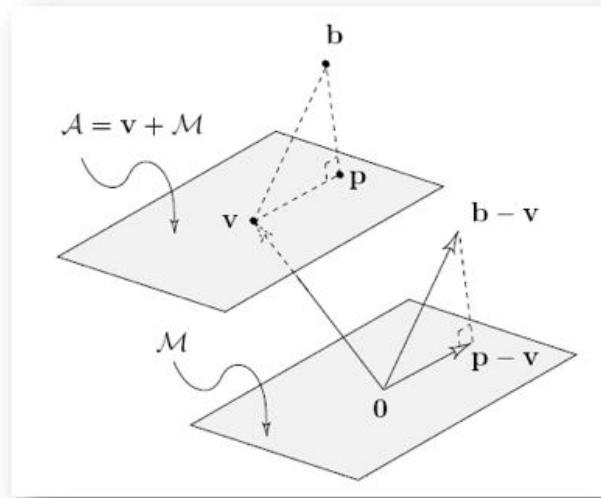
$$\cos \theta_i = S(\mathbf{F}^T \mathbf{G})$$

where $S(\cdot)$ correspond to the *singular values* of \cdot



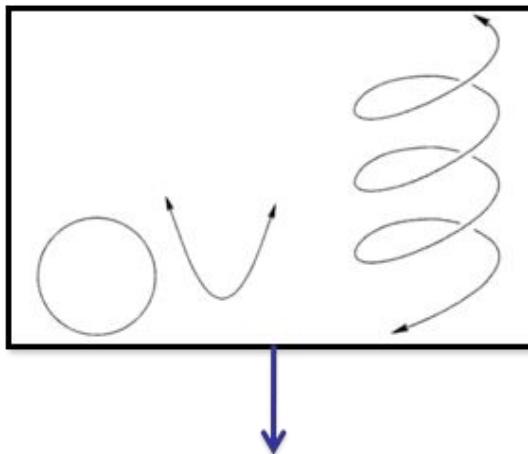
Affine Space

Affine Projections. If $\mathbf{v} \neq \mathbf{0}$ is a vector in a space \mathcal{V} , and if \mathcal{M} is a subspace of \mathcal{V} , then the set of points $\mathcal{A} = \mathbf{v} + \mathcal{M}$ is called an *affine space* in \mathcal{V} .

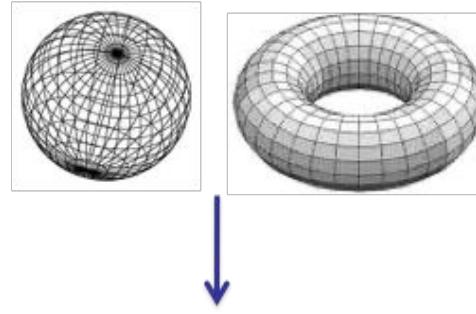


Manifolds

- ◆ A manifold is a mathematical space that (on a sufficiently small scale) resembles to the Euclidean space of a specific dimension



1-dimensional



2-dimensional

Manifolds

- ◆ Manifolds are like curves and surfaces, except that they might be of higher dimension
- ◆ Every manifold has a dimension
 - ▶ The number of independent parameters to specify a point
- ◆ n dimensional manifold is an object modeled *locally* on
 - ▶ It takes exactly n numbers to specify a point



Recall ME210:
Parametric definition
of curves and surfaces

to be continued...

With matrix decomposition methods

ME 536

Week 5: SVD, ...

Matrix Notation Conventions

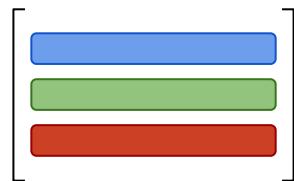
Given a matrix \mathbf{M} :

$\tilde{\mathbf{M}}$ is an approximation of \mathbf{M} ,
i.e. $\tilde{\mathbf{M}} \approx \mathbf{M}$ hence, $\tilde{\mathbf{M}} - \mathbf{M} \neq 0$

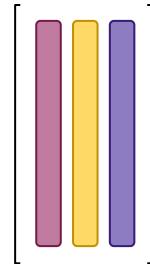
$\tilde{\mathbf{M}}_k$ is a *rank-k* approximation of \mathbf{M}

Matrix Notation Conventions

Given a matrix \mathbf{M} :



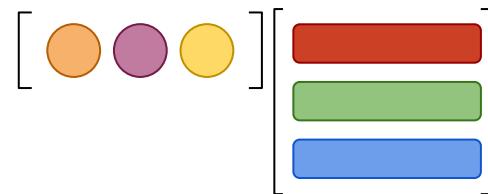
A **fat / wide / short** matrix refers to \mathbf{M} that has more columns than rows



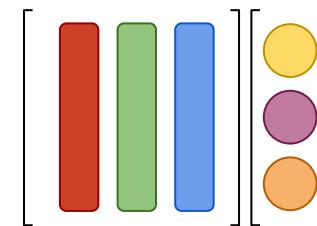
A **skinny / tall** matrix refers to \mathbf{M} that has more rows than columns

Matrix and a Vector multiplied: 2 Alternatives

$$\mathbf{rM} = \mathbf{b}$$



$$\mathbf{Mc} = \mathbf{a}$$



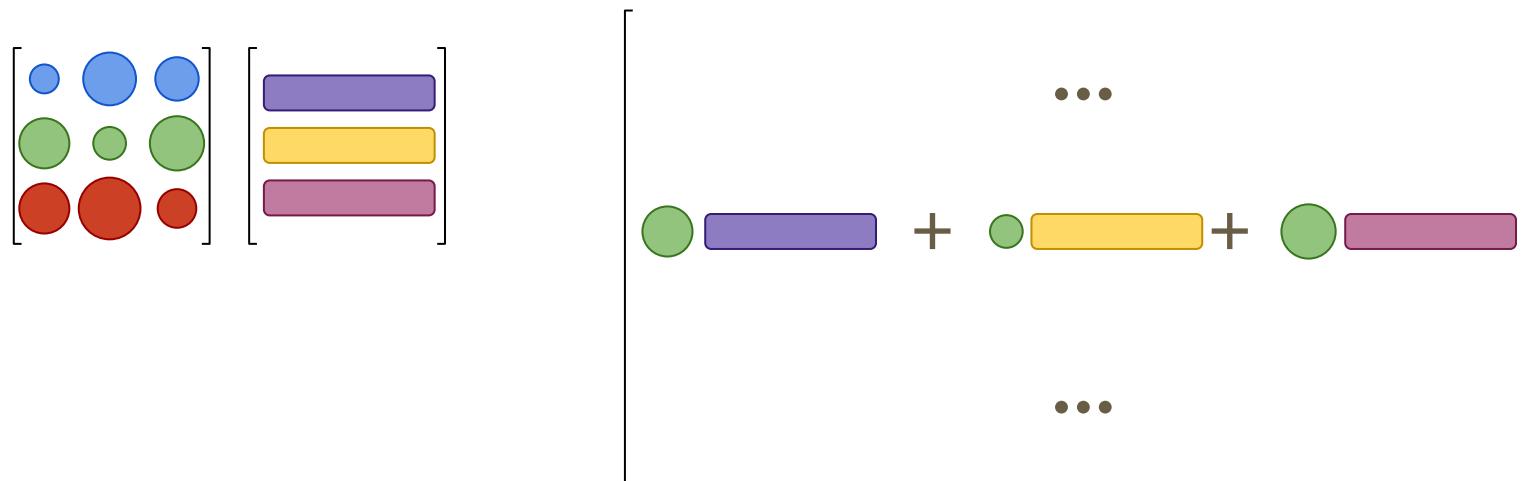
$$\mathbf{b} = \text{orange circle} \cdot \text{red bar} + \text{purple circle} \cdot \text{green bar} + \text{yellow circle} \cdot \text{blue bar}$$

$$\mathbf{a} = \text{yellow circle} \cdot \text{red bar} + \text{purple circle} \cdot \text{green bar} + \text{orange circle} \cdot \text{blue bar}$$

Matrix multiplied by a Matrix: $MN = Q$

Linear combination of Rows

$$M \quad N = Q$$



$$r_i^Q = \sum_j m_{i,j} r_j^N \rightarrow r_i^Q \in \mathbf{R}(N), \forall i$$

Matrix multiplied by a Matrix: $\mathbf{M}\mathbf{N} = \mathbf{Q}$

Linear combination of Columns

$$\mathbf{M} \quad \mathbf{N} = \mathbf{Q}$$

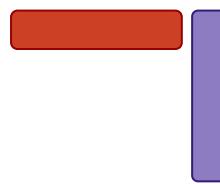
The diagram illustrates the multiplication of two matrices, \mathbf{M} and \mathbf{N} , to produce matrix \mathbf{Q} . Matrix \mathbf{M} is shown as a vertical stack of three colored bars: purple, yellow, and maroon. Matrix \mathbf{N} is shown as a grid of colored circles (red, green, blue) in three rows and three columns. The resulting matrix \mathbf{Q} is shown as a vertical stack of three colored bars, each representing a linear combination of the columns of \mathbf{N} weighted by the entries of the corresponding row of \mathbf{M} . The first bar of \mathbf{Q} is the sum of the first column of \mathbf{N} (red, green, blue circles) weighted by the entries of the first row of \mathbf{M} (purple, yellow, maroon bars). Ellipses indicate that this pattern continues for the remaining columns of \mathbf{Q} .

$$c_i^{\mathbf{Q}} = \sum_j n_{j,i} c_j^{\mathbf{M}} \rightarrow c_j^{\mathbf{Q}} \in \mathbf{C}(\mathbf{M}), \forall j$$

Recall: yet another alternative perspective

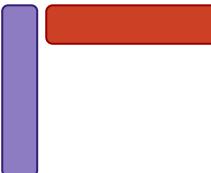
It was also possible to use *inner-product* perspective:

Given two column vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$



$$\mathbf{a}^T \mathbf{b} = s$$

How about the other way around: *outer-product*?

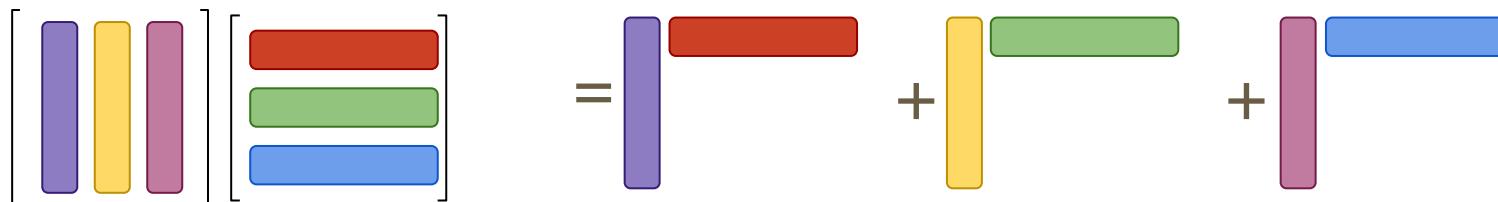


$$\mathbf{a} \mathbf{b}^T = \mathbf{S}$$

Matrix multiplied by a Matrix: $\mathbf{MN} = \mathbf{Q}$

Sum of Outer Products

$$\mathbf{M} \quad \mathbf{N} \quad = \quad \mathbf{Q} = \sum_i \mathbf{c}_i^{\mathbf{M}} \mathbf{r}_i^{\mathbf{N}}$$



$\mathbf{c}_i^{\mathbf{M}} \mathbf{r}_i^{\mathbf{N}}$ is the outer product between the i^{th} column of \mathbf{M} and the i^{th} row of \mathbf{N} .

Also note that: $\text{rank}(\mathbf{c}_i^{\mathbf{M}} \mathbf{r}_i^{\mathbf{N}}) = 1, \forall i$

EXERCISE:

Find the result as a sum of outer products

$$\begin{bmatrix} 3 & 5 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix} \begin{bmatrix} 2 & 1 \end{bmatrix} + \begin{bmatrix} 5 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 3 \end{bmatrix}$$
$$= \begin{bmatrix} 6 & 3 \\ 2 & 1 \end{bmatrix} + \begin{bmatrix} 5 & 15 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 11 & 18 \\ 3 & 4 \end{bmatrix}$$

Note that each outer product is a **Rank-1** matrix

whereas their sum is a **Rank-2** matrix

What if *first outer-product matrix was more important than the second?*

Matrix multiplied by a Matrix: $MN = Q$

By sub-blocks

M

N

=

Q

$$\left[\begin{array}{c|c} M_1 & M_2 \\ \hline M_3 & M_4 \end{array} \right] \left[\begin{array}{c|c} N_1 & N_2 \\ \hline N_3 & N_4 \end{array} \right] = \left[\begin{array}{c|c} M_1 N_1 + M_2 N_3 & M_1 N_2 + M_2 N_4 \\ \hline M_3 N_1 + M_4 N_3 & M_3 N_2 + M_4 N_4 \end{array} \right]$$

If sub-matrices (blocks) in M and N are **compatible**, Q can be found by treating blocks as matrix elements.

Useful when some blocks are all-zeros, all-ones or Identity

Matrix Decomposition

| <u>Method</u> | <u>Desired forms, features</u> |
|---------------|--------------------------------|
| • Eigen- | Diagonal |
| • LU | Triangular |
| • QR | Orthonormal, triangular |
| • CUR | Basis from data |
| • SVD | Orthonormal, diagonal |
| • ... | |

Recall: Eigen Decomposition

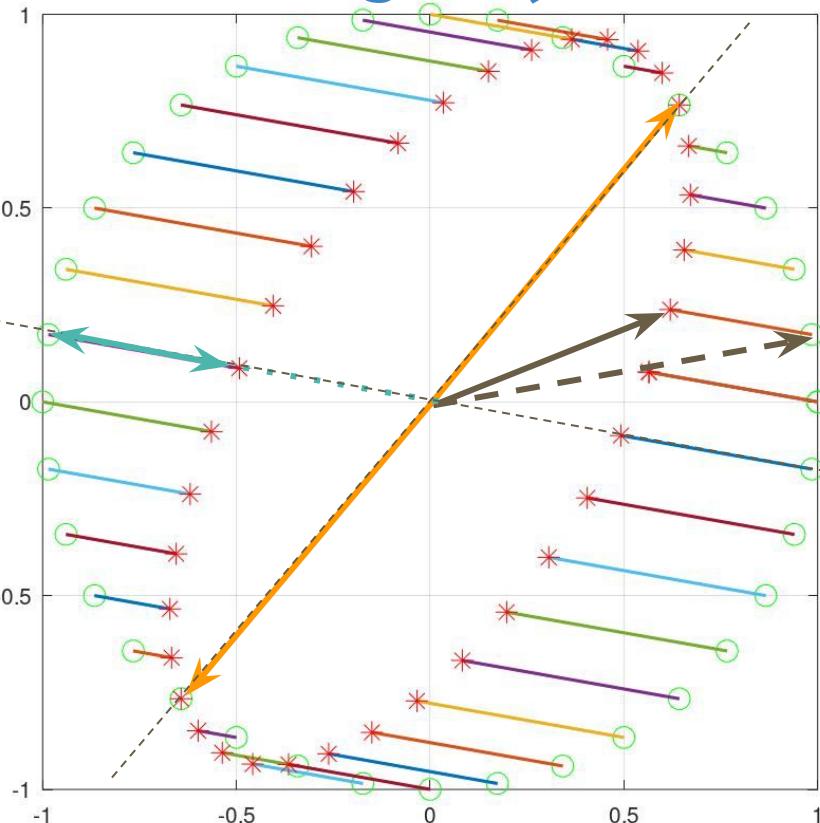
Refresh your Memory:

Eigen Decomposition starts with:

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v} \text{ where } \mathbf{A} \text{ is } n \times n$$

to yield $\mathbf{A} = \mathbf{V}\Lambda\mathbf{V}^{-1}$

Recall: Eigen-(Values, Vectors) $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$



Let columns of \mathbf{C} be the green circles (○),
and let there be a 2×2 matrix \mathbf{A} where:

$$\mathbf{S} = \mathbf{A} \cdot \mathbf{C}$$

Stars (*) are the columns of \mathbf{S}

What are the eigenvalues and eigenvectors
of \mathbf{A} ?

Orthogonal Matrices

A **matrix is M** referred to as **orthonormal** (*orthogonal by some other*) if its *columns and rows* are orthogonal unit vectors (i.e. *orthonormal vectors*)

$$\begin{bmatrix} \text{[3 colored bars]} \\ \times \\ \text{[3 colored bars]} \end{bmatrix} = \begin{bmatrix} 1 & \dots & 0 \\ 0 & \dots & 0 \\ 0 & \dots & 1 \end{bmatrix}$$

$$\mathbf{M}^T \mathbf{M} = \mathbf{M} \mathbf{M}^T = \mathbf{I}$$

Therefore:

$$\mathbf{M}^{-1} = \mathbf{M}^T$$

Note that: $\|\mathbf{M}\mathbf{v}\| = \|\mathbf{v}\|$ but WHY? and what does it imply?

Properties of Symmetric Matrices

Symmetric → Square implied

Eigen(.)s of a symmetric matrix:

- Eigenvalues are real
- Eigenvectors are **orthogonal**

Square and symmetric ??? too specific

We were getting ready to deal with some generic matrix: $\mathbf{M}_{d \times n}$

Properties of Orthogonal¹ Matrices

- Transpose is inverse
- They do **not scale** but **just rotate**
- Multiplication of 2 orthogonal matrices are orthogonal

1: Unitary if elements are not necessarily real

How do you get a symmetric matrix from: $\mathbf{M}_{d \times n}$

$$\mathbf{M}^T \mathbf{M} = \mathbf{V}_{n \times n} \quad \rightarrow \text{rank}(\mathbf{V})$$

$$\mathbf{M} \mathbf{M}^T = \mathbf{U}_{d \times d} \quad \rightarrow \text{rank}(\mathbf{U})$$

How do you get a symmetric matrix from: $\mathbf{M}_{d \times n}$

$$\mathbf{M}^T \mathbf{M} = \mathbf{V}_{n \times n}$$

$$\mathbf{M} \mathbf{M}^T = \mathbf{U}_{d \times d}$$

→ eigen(values/vectors) of (\mathbf{U})

→ eigen(values/vectors) of (\mathbf{V})

SVD: Singular Value Decomposition

Any matrix $\mathbf{M}_{d \times n}$ can be decomposed as:

Left
Singular
Vectors

$$\mathbf{M} = \mathbf{U} \Sigma \mathbf{V}^T$$

Singular
Values

Right
Singular
Vectors

Orthogonal

Diagonal

??? dimensions ???

SVD: Singular Value Decomposition

$$\mathbf{M}_{d \times n} = \mathbf{U}_{d \times d} \ \boldsymbol{\Sigma}_{d \times n} \ \mathbf{V}^T_{n \times n}$$

Orthogonal

$$\mathbf{U}_{d \times d} \rightarrow \mathbf{U}\mathbf{U}^T = \mathbf{U}^T\mathbf{U} = \mathbf{I}$$

Orthogonal

$$\mathbf{V}_{n \times n} \rightarrow \mathbf{V}\mathbf{V}^T = \mathbf{V}^T\mathbf{V} = \mathbf{I}$$

SVD: Singular Value Decomposition

$$\mathbf{M}_{d \times n} = \mathbf{U}_{d \times d} \Sigma_{d \times n} \mathbf{V}^T_{n \times n} = \begin{bmatrix} \text{purple bar} & \text{yellow bar} & \text{purple bar} \end{bmatrix} \begin{bmatrix} \text{red circle} & & \\ & \text{green circle} & \\ & & \text{blue circle} \end{bmatrix} \begin{bmatrix} \text{red bar} \\ \text{green bar} \\ \text{blue bar} \end{bmatrix}$$

$$d = n$$

$$\Sigma_{d \times d} = \begin{bmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ \dots & \dots & \ddots & \dots \\ 0 & 0 & \dots & \sigma_d \end{bmatrix}_{d \times d}$$

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_{d-1} \geq \sigma_d$$

SVD: Singular Value Decomposition - Tall M

$$\mathbf{M}_{d \times n} = \mathbf{U}_{d \times d} \ \boldsymbol{\Sigma}_{d \times n} \ \mathbf{V}^T_{n \times n}$$

$d > n$

$$\boldsymbol{\Sigma}_{d \times n} = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & \sigma_n \\ 0 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}_{d \times n}$$

SVD: Singular Value Decomposition - Wide M

$$\mathbf{M}_{d \times n} = \mathbf{U}_{d \times d} \ \boldsymbol{\Sigma}_{d \times n} \ \mathbf{V}^T_{n \times n}$$

$d < n$

$$\boldsymbol{\Sigma}_{d \times n} = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & \sigma_d & 0 & \cdots & 0 \end{bmatrix}_{d \times n}$$

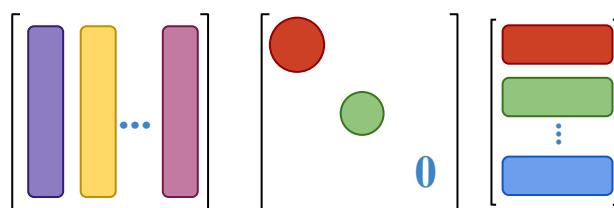
Show that:

- The singular values of matrix \mathbf{M} are the square root of the eigenvalues of $\mathbf{M}^T\mathbf{M}$ or $\mathbf{M}\mathbf{M}^T$
- Left Singular vectors are eigenvectors of $\mathbf{M}\mathbf{M}^T$
- Right Singular vectors are eigenvectors of $\mathbf{M}^T\mathbf{M}$

SVD: Low Rank Data matrix $\rightarrow \text{rank}(\mathbf{M}) = r < \min(d, n)$

Let $\text{rank}(\mathbf{M}) = r \rightarrow \Sigma_{d \times n} =$

$$\begin{bmatrix} \sigma_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & \sigma_r & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{bmatrix}_{d \times n}$$



$$\mathbf{M}_{d \times n} = \mathbf{U}_{d \times r} \Sigma_{r \times r} \mathbf{V}^T_{r \times n}$$

SVD: Singular Value Decomposition - columns

*columns are important, columns are meaningful, oooh those columns
how about the rows?*

$$\mathbf{M}_{d \times n} = \mathbf{U}_{d \times d} \Sigma_{d \times n} \mathbf{V}^T_{n \times n}$$

$\mathbf{u}_1, \dots, \mathbf{u}_r$ is an orthonormal basis for $\mathbf{C}(\mathbf{M})$

$\mathbf{u}_{r+1}, \dots, \mathbf{u}_d$ is an orthonormal basis for $\mathbf{N}(\mathbf{M}^T)$, i.e. **left null space**

$\mathbf{v}_1, \dots, \mathbf{v}_r$ is an orthonormal basis for $\mathbf{C}(\mathbf{M}^T)$

$\mathbf{v}_{r+1}, \dots, \mathbf{v}_n$ is an orthonormal basis for $\mathbf{N}(\mathbf{M})$, i.e. **null space**

where $r = \text{rank}(\mathbf{M})$

SVD: Low Rank Data matrix $\rightarrow \text{rank}(\mathbf{M}) < n$

$$\text{rank}(\mathbf{M}) = r < n$$

$$\hat{\mathbf{M}}_{d \times n} = \mathbf{U}_{d \times r} \boldsymbol{\Sigma}_{r \times r} \mathbf{V}_{r \times n}^T$$

\mathbf{M} can be reconstructed from $\mathbf{U}_{d \times r} \boldsymbol{\Sigma}_{r \times r} \mathbf{V}_{r \times n}^T$ without any loss

$$\hat{\mathbf{M}} = \mathbf{M}$$

Referred to with names such as: *skinny* SVD, *economy* SVD, truncated SVD

SVD: Matrix Approximation - who get to die first?

$$\mathbf{M}_{d \times n} = \mathbf{U}_{d \times d} \mathbf{\Sigma}_{d \times n} \mathbf{V}^T_{n \times n}$$

For some $0 < k < r = \text{rank}(\mathbf{M})$ approximation of \mathbf{M} :

$$\tilde{\mathbf{M}}_{d \times n} = \mathbf{U}_{d \times k} \mathbf{\Sigma}_{k \times k} \mathbf{V}^T_{k \times n}$$

$$\tilde{\mathbf{M}}_k \approx \mathbf{M}$$

$$\mathbf{M}_{error} = \tilde{\mathbf{M}}_k - \mathbf{M}$$

How good is the approximation: $\tilde{\mathbf{M}}_{d \times n} = \mathbf{U}_{d \times k} \boldsymbol{\Sigma}_{k \times k} \mathbf{V}_{k \times n}^T$

Best in some sense, i.e. Frobenius

Assume that $k < \text{rank}(\mathbf{M}) = r$

SVD: Sum of Outer Products Perspective

$$\mathbf{M}_{d \times n} = \mathbf{U}_{d \times d} \mathbf{\Sigma}_{d \times n} \mathbf{V}^T_{n \times n}$$

$$\mathbf{M} = \sum \sigma_i \mathbf{c}_i^U \mathbf{r}_i^{V^T}$$

$$\mathbf{M} = \sigma_1 \mathbf{c}_1^U \mathbf{r}_1^{V^T} + \sigma_2 \mathbf{c}_2^U \mathbf{r}_2^{V^T} + \dots + \sigma_r \mathbf{c}_r^U \mathbf{r}_r^{V^T}$$

where \mathbf{c}_i^U is $d \times 1$ and $\mathbf{r}_i^{V^T}$ is $1 \times n$

hence, $\mathbf{c}_i^U \mathbf{r}_i^{V^T}$ is $d \times n$ where the *importance* is determined by σ_i

SVD: Sum of Outer Products Perspective

$$\mathbf{M}_{d \times n} = \mathbf{U}_{d \times d} \ \boldsymbol{\Sigma}_{d \times n} \ \mathbf{V}^T_{n \times n}$$

$$\mathbf{M} = \sum \sigma_i \mathbf{c}_i^U \mathbf{r}_i^{V^T}$$

$$\mathbf{M} = \sigma_1 \mathbf{c}_1^U \mathbf{r}_1^{V^T} + \sigma_2 \mathbf{c}_2^U \mathbf{r}_2^{V^T} + \dots + \sigma_r \mathbf{c}_r^U \mathbf{r}_r^{V^T}$$

$$\mathbf{M} = \sigma_1 \left| \begin{array}{c} \text{purple bar} \\ \text{red bar} \end{array} \right| + \sigma_2 \left| \begin{array}{c} \text{yellow bar} \\ \text{green bar} \end{array} \right| + \dots + \sigma_r \left| \begin{array}{c} \text{purple bar} \\ \text{blue bar} \end{array} \right|$$

How good is the approximation: $\tilde{\mathbf{M}}_{d \times n} = \mathbf{U}_{d \times k} \Sigma_{k \times k} \mathbf{V}_{k \times n}^T$

Best $\text{rank}(k)$ approximation in *Nuclear, Spectral and Frobenious norm sense*^{1,2}

$$\Sigma_{d \times n} = \begin{bmatrix} \sigma_1 & \cdots & 0 & 0 & \cdots & 0 & \cdots \\ \cdots & \ddots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & \sigma_k & 0 & \cdots & 0 & \cdots \\ 0 & \cdots & 0 & \sigma_{k+1} & \cdots & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \ddots & \cdots & \cdots \\ 0 & \cdots & 0 & 0 & \cdots & \sigma_r & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \ddots \end{bmatrix}_{d \times n}$$

Note that: $k < \text{rank}(\mathbf{M}) = r$

$$\mathbf{M} = \sigma_1 \mathbf{c}_1^U \mathbf{r}_1^{V^T} + \sigma_2 \mathbf{c}_2^U \mathbf{r}_2^{V^T} + \dots + \sigma_r \mathbf{c}_r^U \mathbf{r}_r^{V^T}$$

$$\tilde{\mathbf{M}} = \sigma_1 \mathbf{c}_1^U \mathbf{r}_1^{V^T} + \sigma_2 \mathbf{c}_2^U \mathbf{r}_2^{V^T} + \dots + \sigma_k \mathbf{c}_k^U \mathbf{r}_k^{V^T}$$

$$\mathbf{M} = \tilde{\mathbf{M}} + \sigma_{k+1} \mathbf{c}_{k+1}^U \mathbf{r}_{k+1}^{V^T} + \dots + \sigma_r \mathbf{c}_r^U \mathbf{r}_r^{V^T}$$

$$\mathbf{M} - \tilde{\mathbf{M}} = \sigma_{k+1} \mathbf{c}_{k+1}^U \mathbf{r}_{k+1}^{V^T} + \dots + \sigma_r \mathbf{c}_r^U \mathbf{r}_r^{V^T}$$

1- Check out [Eckart-Young-Mirsky theorem](#)

2- What sense? Do matrices have norms too, now I have seen everything

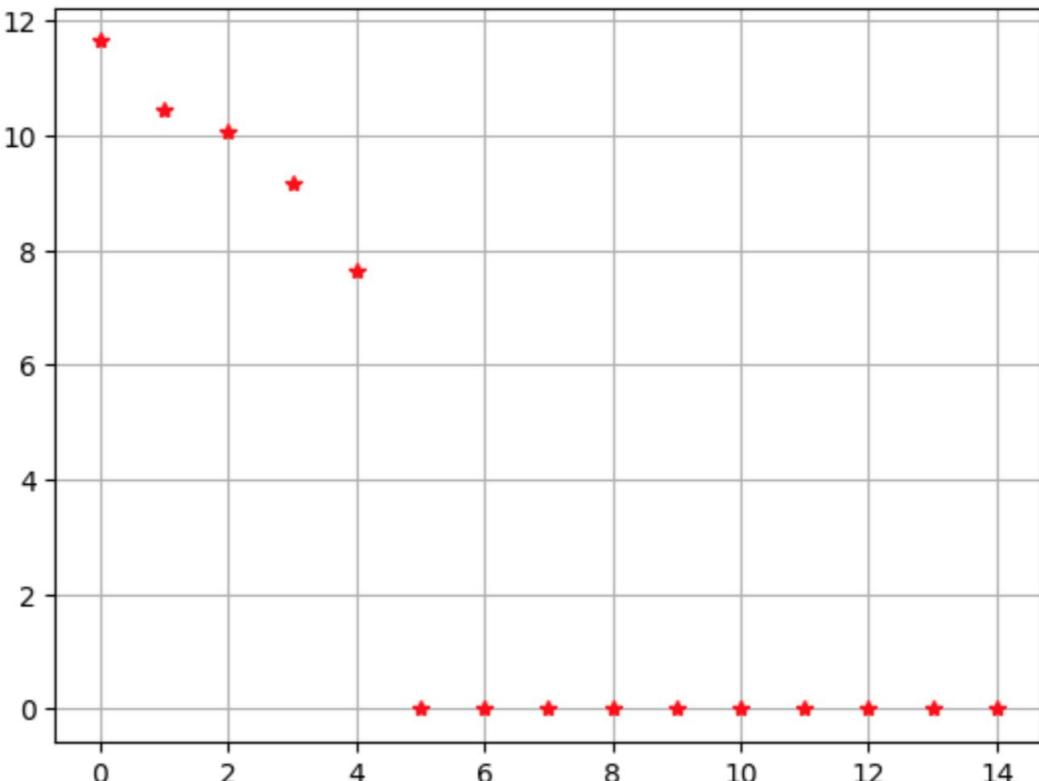
Python:

Recall my python function:

```
>> DataInSubspace()
```

```
1 M = DataInSubspace(15, 100, 5)
2 print(f'rank(M)={np.linalg.matrix_rank(M)}')
3 U,S,VT = np.linalg.svd(M, full_matrices=False)
4
5 M5 = U @ (np.diag(S) @ VT)
6 print(f'rank(M5)={np.linalg.matrix_rank(M5)}')
7 print(f'M5 - M = {abs(M5-M).sum()}')
8 plt.plot(S, 'r*')
9 plt.grid()
10 plt.title('Singular values')
```

```
rank(M)=5
rank(M5)=5
abs(M5 - M).sum = 6.899799741971746e-13
```



Python:

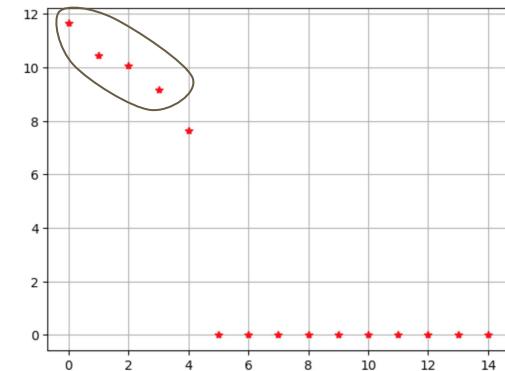
Rank-4 approximation of M

$$M = \sigma_1 c_1^U r_1^{V^T} + \sigma_2 c_2^U r_2^{V^T} + \sigma_3 c_3^U r_3^{V^T} + \sigma_4 c_4^U r_4^{V^T} + \sigma_5 c_5^U r_5^{V^T}$$

$$M - M_a = M_{diff} = \sigma_5 c_5^U r_5^{V^T}$$

```
1 M = DataInSubspace(15, 100, 5)
2 print(f'rank(M)={np.linalg.matrix_rank(M)}')
3 U,S,VT = np.linalg.svd(M, full_matrices=False)
4
5 M5 = U @ (np.diag(S) @ VT)
6 print(f'rank(Ma)={np.linalg.matrix_rank(M5)}')
7 print(f'Ma - M = {abs(M5-M).sum()}')
8 plt.plot(S, 'r*')
9 plt.grid()
10 plt.title('Singular values')
```

```
rank(M)=5
rank(M5)=5
abs(M5 - M).sum = 6.899799741971746e-13
```



```
1 # reconstruct a low-rank approximation of M
2 RankApp = 4
3 Ma = U[:, :RankApp] @ (np.diag(S[:RankApp])) @ VT[:RankApp, :]
4 print(f'abs(Ma-M).sum()={abs(Ma-M).sum()}')
```

abs(Ma-M).sum()=171.7903369122311

```
1
2 #verify result
3 Mdiff = U[:, RankApp:] @ (np.diag(S[RankApp:])) @ VT[RankApp:, :]
4 print(f'abs(Mdiff).sum()={abs(Mdiff).sum()}')
```

abs(Mdiff).sum()=171.7903369122311

Expectation

When we approximate a matrix we expect that the difference between them is small.

How do we define **small for matrices?**

Induced Matrix Norms - *induced by Vector norms*

Focus on the transformation by the matrix:

$$\mathbf{M} : \mathbb{R}^n \rightarrow \mathbb{R}^m$$

$$\|\mathbf{M}\|_{q,p} = \sup_{\mathbf{x} \neq \mathbf{0}} \frac{\|\mathbf{Mx}\|_p}{\|\mathbf{x}\|_q}$$

common usage:

$$\|\mathbf{M}\|_{p,p} = \|\mathbf{M}\|_p = \sup_{\mathbf{x} \neq \mathbf{0}} \frac{\|\mathbf{Mx}\|_p}{\|\mathbf{x}\|_p}$$

Induced Matrix Norms

Commonly used induced p-norms

$$\|\mathbf{M}\|_p = \sup_{\mathbf{x} \neq \mathbf{0}} \frac{\|\mathbf{Mx}\|_p}{\|\mathbf{x}\|_p}$$

$$\|\mathbf{M}\|_1 = \max_{j=1, \dots, n} \left\{ \sum_{i=1}^d |m_{ij}| \right\} \rightarrow \text{max. column sum}$$

$$\|\mathbf{M}\|_2 = \max_{j=1, \dots, n} \left\{ \lambda_j(\mathbf{M}^T \mathbf{M}) \right\}^{\frac{1}{2}} = \sigma_{max}(\mathbf{M}) \rightarrow \text{max. singular value}$$

a.k.a **spectral norm**

$$\|\mathbf{M}\|_\infty = \sup_{\mathbf{x} \neq \mathbf{0}} \frac{\|\mathbf{Mx}\|_\infty}{\|\mathbf{x}\|_\infty} = \max_{i=1, \dots, d} \left\{ \sum_{j=1}^n |m_{ij}| \right\} \rightarrow \text{max. row sum}$$

Elementwise Matrix Norms

$$\|A\|_* = \text{trace}(\sqrt{A^* A}) = \sum_{i=1}^{\min\{m,n\}} \sigma_i(A),$$

Elements of an **$d \times n$ matrix** is treated as a **vector of length ($d * n$)**, and vector norms are used

Check out [Schatten norms](#) as well

Nuclear Norm a.k.a. Trace Norm: $\|\mathbf{M}\|_N = \text{trace}(\sqrt{\mathbf{M}^T \mathbf{M}}) = \sum_{i=1}^{\text{rank}(\mathbf{M})} \sigma_i(\mathbf{M})$

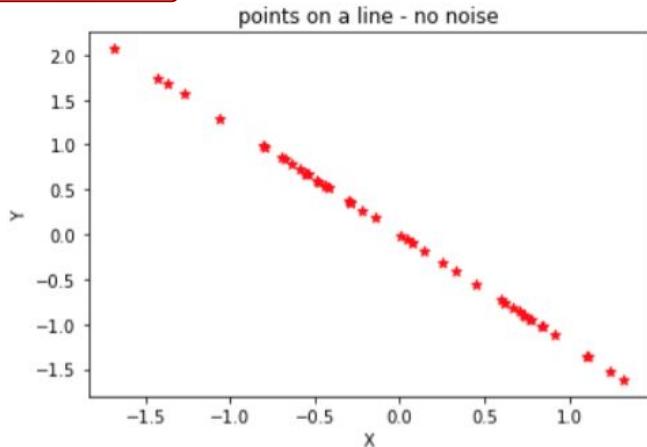
Frobenius Norm: $\|\mathbf{M}\|_F = \sqrt{\sum_{i=1}^d \sum_{j=1}^n |m_{ij}|^2} = \sqrt{\text{trace}(\mathbf{M}^T \mathbf{M})} = \sqrt{\sum_{i=1}^{\text{rank}(\mathbf{M})} \sigma_i^2(\mathbf{M})}$

Max Norm: $\|\mathbf{M}\|_{max} = \max_{ij} |m_{ij}|$

Example: Data from a line

```
1 # generate data on a line
2 M = DataInSubspace(2, 50, 1)
3 print(f'rank(M) = {np.linalg.matrix_rank(M)}')
4 CData(M, 'points on a line - no noise')
5 u, s, vt = np.linalg.svd(M, full_matrices=False)
6 print('Singular values are:')
7 MatPrint(s)
```

rank(M) = 1



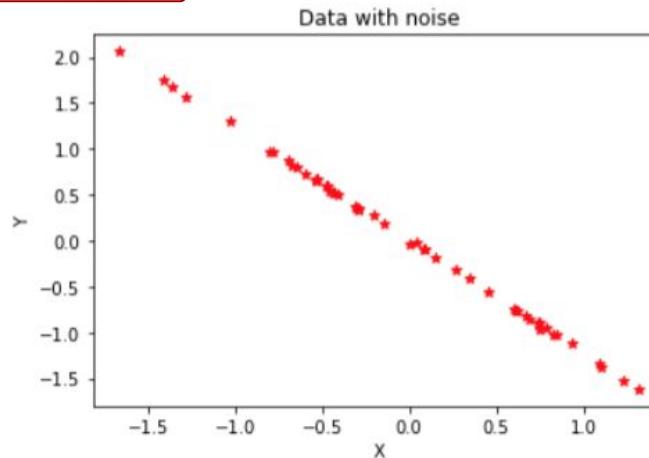
Singular values are:

Matrix:

[8.32756054e+00 4.58017925e-16]

```
1 # add a bit of noise to the data
2 Mn = M + 0.01 * np.random.randn(M.shape[0], M.shape[1])
3 print(f'rank(Mn) = {np.linalg.matrix_rank(Mn)}')
4 CData(Mn, 'Data with noise')
5 un, sn, vtn = np.linalg.svd(Mn, full_matrices=False)
6 print('Singular values are:')
7 MatPrint(sn)
```

rank(Mn) = 2



Singular values are:

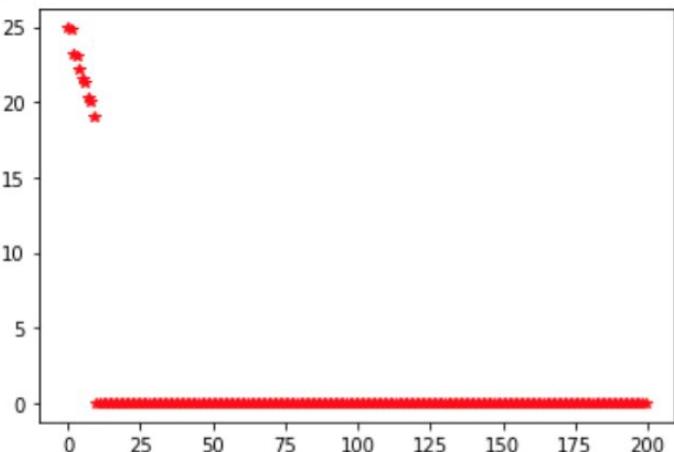
Matrix:

[8.31458315 0.07017064]

Example: Data from a line

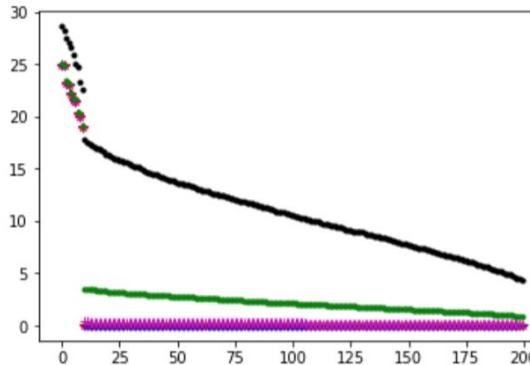
```
1 # generate data on a line
2 M = DataInSubspace(200, 500, 10)
3 print(f'rank(M) = {np.linalg.matrix_rank(M)}')
4 u, s, vt = np.linalg.svd(M, full_matrices=False)
5 plt.plot(s, 'r*')
6
7
```

```
rank(M) = 10
[<matplotlib.lines.Line2D at 0x7ffa4fd47f28>]
```



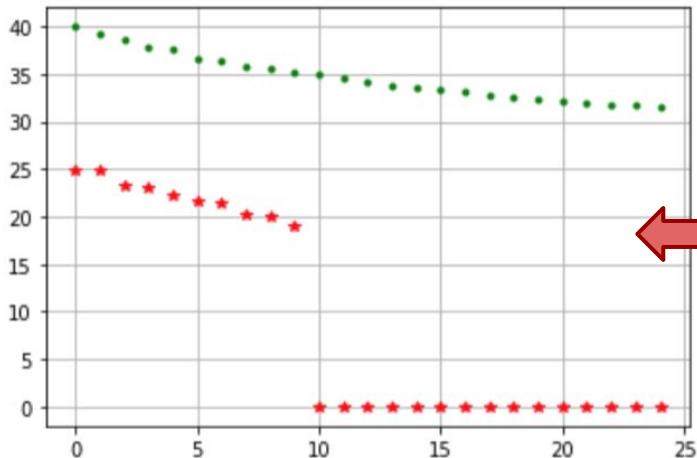
```
1 plt.plot(s, 'r*')
2
3 plotwith = ['b+', 'm+', 'g.', 'k.']
4 noiselevels = [0.00001, 0.01, 0.1, 0.5]
5 # add some tiny amount of noise
6 for i, nLevel in enumerate(noiselevels):
7     Mn = M + nLevel * np.random.randn(M.shape[0], M.shape[1])
8     print(f'rank(Mn{i}) = {np.linalg.matrix_rank(Mn)}')
9     u, s, vt = np.linalg.svd(Mn, full_matrices=False)
10    plt.plot(s, plotwith[i])
11
```

```
rank(Mn1) = 200
rank(Mn1) = 200
rank(Mn1) = 200
rank(Mn1) = 200
```



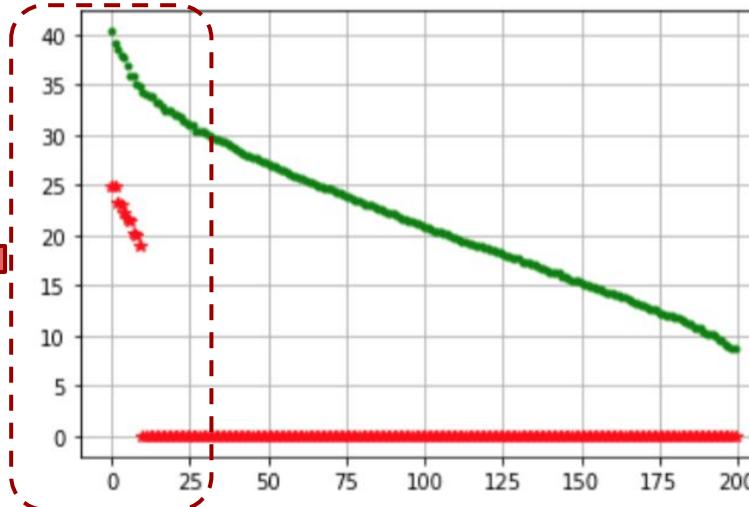
Rank Estimation

By checking the change in singular values can you estimate the rank of matrix **M**?



```
1 plotwith = ['r*', 'g.']
2 noiselevels = [0.0, 1.0]
3 # add some tiny amount of noise
4 for i, nLevel in enumerate(noiselevels):
5     Mn = M + nLevel * np.random.randn(M.shape[0], M.shape[1])
6     print(f'rank(Mn1) = {np.linalg.matrix_rank(Mn)}')
7     u, s, vt = np.linalg.svd(Mn, full_matrices=False)
8     plt.plot(s, plotwith[i])
9     plt.grid(True)
10
```

rank(Mn1) = 10
rank(Mn1) = 200



Condition Number

Depends on the selected norm, but for convenience we will stick with the following:

Given matrix \mathbf{M}

$$\kappa(\mathbf{M}) = \frac{\sigma_{\max}(\mathbf{M})}{\sigma_{\min}(\mathbf{M})}$$

Low condition number → *well-conditioned* case *how low can it get?*

High condition number → *ill-conditioned* case *how high can it get?*

Rank Estimation - *one such* Rule of thumb

A rule of thumb is to keep 90% of the energy in Σ
where total energy is defined as:

$$E(\Sigma) = \sum_{\forall i} \sigma_i^2$$

then energy in the first k singular values are

$$E(\Sigma_k) = \sum_{i=1}^k \sigma_i^2$$

and

$$\frac{E(\Sigma_k)}{E(\Sigma)} \approx 0.9$$

Update 90 to a proper value depending on the problem

Rank Estimation - *another* Approach : $\frac{4}{\sqrt{3}}$

As an exercise:

Check out this paper: <https://arxiv.org/abs/1305.5870>

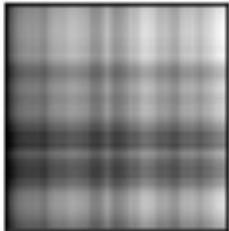
And this video:

<https://youtu.be/epoHE2rex0g?list=PLMrJAkhleNNSVjnsviglFoY2nXildDCcv>

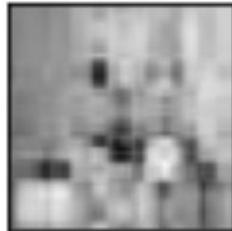
SVD: Not a compression method but...

```
1 def ImageInRankK(img, k):
2     try:
3         U, S, VT = np.linalg.svd(img, full_matrices=False)
4         return np.matmul(U[:, :k], np.matmul(np.diag(S[:k]), VT[:, :k]))
5     except:
6         print('it did not work out, try again later')
7         return img
```

```
1 rankK = [1, 5, 10, 25, 100]
2 nPlots = len(rankK)
3 for i, k in enumerate(rankK):
4     plt.subplot(1, nPlots, i+1)
5     #plt.axis('off')
6     plt.xticks([])
7     plt.yticks([])
8     plt.imshow(ImageInRankK(img, k), cmap=plt.get_cmap("gray"))
9     plt.xlabel(f'k={k}')
```



k=1



k=5



k=10



k=25

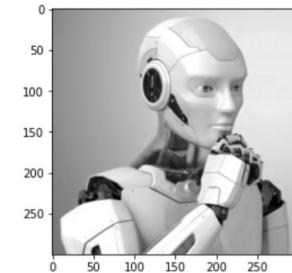


k=100

Play with some images

```
1 # read the image file into a variable
2 img = io.imread('robot.jpg')
3 # let's play with the image's red channel only
4 img = img[:, :, 0]
5 plt.imshow(img, cmap=plt.get_cmap("gray"))
```

<matplotlib.image.AxesImage at 0x7ffa4f055048>



coming up...

More matrices and matrices and matrices and ...

ME 536

— Week 6: Spoiler, Not that inverse, —
SVD, PCA, LDA ...

Spoiler Alert Revisited!!!

You have not and will NOT LEARN directly how to

Design an intelligent machine at the end of the course

Darn it, add drops are over

Many topics deserve a semester on its own

Understanding issues in decision making

Having a start-up background

Intelligent Machines

- Has to be **human-like?**
- **Understand human-intelligence before**
building Intelligent Machines?

Types of Intelligence

- Beats you in chess ?
- Can prove a theorem ?
- Can tie shoelaces ?
- Knows that milk will rot when left out ?
- Resolves conflict between 2 people?
- ...

Goals

Intelligent Machine(s)

Unknown
situation

Predict

Sense → Decide → Act

Where is, if any ?

Thinking

Attention

Other machines

Context

Physical body

Search

Perception

Ethics

Limited time

Self-driven

Self-Awareness
Self-Expression

Conflicts

*Co-operation
Collaboration
Competition*

Consciousness

Focus on ***things*** that are:

- Similar / Related vs *Different / New*
- Changing vs *Not changing*

BUT data is:

- Vast
- high dimensional

Recall:

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

If A is square and non-singular nice and dandy

What if **NOT**

Note that: $A_{m \times n}$

$$Ax = b$$

$$A^T A x = A^T b$$

If A is full rank, then $A^T A$ is **????** hence

$$x = (A^T A)^{-1} A^T b$$

So this is not a generalizable approach :(

Recap-

$$\mathbf{M} = \mathbf{U} \ \boldsymbol{\Sigma} \ \mathbf{V}^T$$

U and V are **Orthogonal**:

→ symmetric

→ transpose is inverse

→ do not scale but just rotate - *check out σ_i of U or V !!!*

Note: Multiplication of 2 orthogonal matrices are orthogonal

Consider: A non-square diagonal matrix

For simplicity, 2×4 matrix is considered w.l.g:

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

Let

$$\Sigma^+ := \begin{bmatrix} \frac{1}{\sigma_1} & 0 \\ 0 & \frac{1}{\sigma_2} \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Then,

$$\Sigma \Sigma^+ = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\Sigma^+ \Sigma = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Finally observe that,

$$\Sigma \Sigma^+ \Sigma = \Sigma \text{ and } \Sigma^+ \Sigma \Sigma^+ = \Sigma^+$$

Consider: A non-square diagonal matrix

This time consider 4x2 matrix:

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

Let

$$\Sigma^+ := \begin{bmatrix} \frac{1}{\sigma_1} & 0 & 0 & 0 \\ 0 & \frac{1}{\sigma_2} & 0 & 0 \end{bmatrix}$$

Then,

$$\Sigma\Sigma^+ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\Sigma^+\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Still observe that,

$$\Sigma\Sigma^+\Sigma = \Sigma \text{ and } \Sigma^+\Sigma\Sigma^+ = \Sigma^+$$

Let's call it: *for the special case of a diagonal matrix*

Σ^+ is the **PSEUDO-INVERSE** of Σ

$$\Sigma \Sigma^+ \Sigma = \Sigma \text{ and } \Sigma^+ \Sigma \Sigma^+ = \Sigma^+$$

Recall:

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

If A is square and non-singular nice and dandy

What if **NOT**

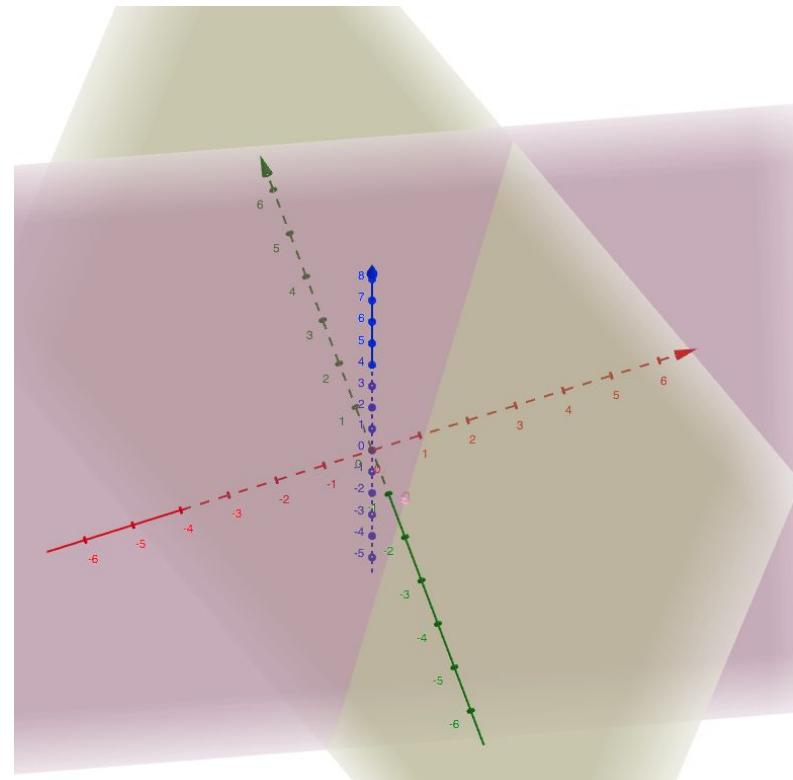
Recall:

$$\mathbf{A}_{m \times n} \mathbf{x}_{n \times 1} = \mathbf{b}_{m \times 1}$$

Underdetermined Case : $n > m$

In general: $\rightarrow \infty$ solutions

$$\left[\begin{array}{c|c|c|c} \text{purple bar} & \text{yellow bar} & \cdots & \text{pink bar} \\ \hline \end{array} \right] \mathbf{x} = \text{red bar}$$



Recall:

$$\mathbf{A}_{mxn} \mathbf{x}_{nx1} = \mathbf{b}_{mx1}$$

Overdetermined Case : $n < m$

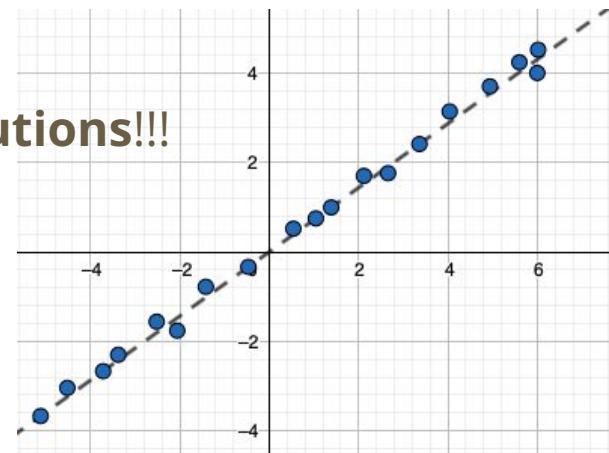
In general: → more equations than unknowns

→ 1 solution if equations are consistent: generally NOT

→ 0 solution otherwise:

Say hi to **approximate solutions!!!**

$$\begin{bmatrix} \text{purple bar} \\ \text{yellow bar} \\ \text{pink bar} \end{bmatrix} \quad \text{=} \quad \text{green bar}$$



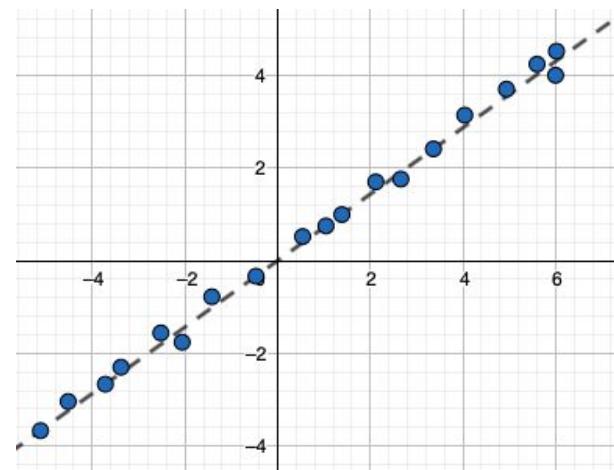
Question:

$$\mathbf{A}_{m \times n} \mathbf{x}_{n \times 1} = \mathbf{b}_{m \times 1}$$

Can we get an approximate solution?

But \mathbf{A} is not square \rightarrow not invertible

$$\begin{bmatrix} \text{purple bar} \\ \text{yellow bar} \\ \text{pink bar} \end{bmatrix} = \text{red bar}$$



the SVD way... $m \neq n$, $\text{rank}(A)$ does not matter

$$A_{m \times n} x_{n \times 1} = b_{m \times 1}$$

SVD think
SVD use
SVD be



SVD it is

Given $\mathbf{A}\mathbf{x} = \mathbf{b}$ and we can use SVD where: $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T$

Hence,

$$\mathbf{U}\Sigma\mathbf{V}^T\mathbf{x} = \mathbf{b}$$

$$\Sigma\mathbf{V}^T\mathbf{x} = \mathbf{U}^T\mathbf{b}$$

$$\mathbf{V}^T\mathbf{x} = \Sigma^+\mathbf{U}^T\mathbf{b}$$

$$\mathbf{x} = \mathbf{V}\Sigma^+\mathbf{U}^T\mathbf{b}$$

Let,

$$\tilde{\mathbf{x}} = \mathbf{A}^+\mathbf{b}$$

where *left* pseudo-inverse of \mathbf{A} is defined as:

$$\mathbf{A}^+ := \mathbf{V}\Sigma^+\mathbf{U}^T$$

and

$\tilde{\mathbf{x}}$ is one of the infinitely many solutions or approximations

Pseudo-inverse: Underdetermined case

For $\mathbf{Ax} = \mathbf{b}$

Let,

$$\tilde{\mathbf{x}} = \mathbf{A}^+ \mathbf{b}$$

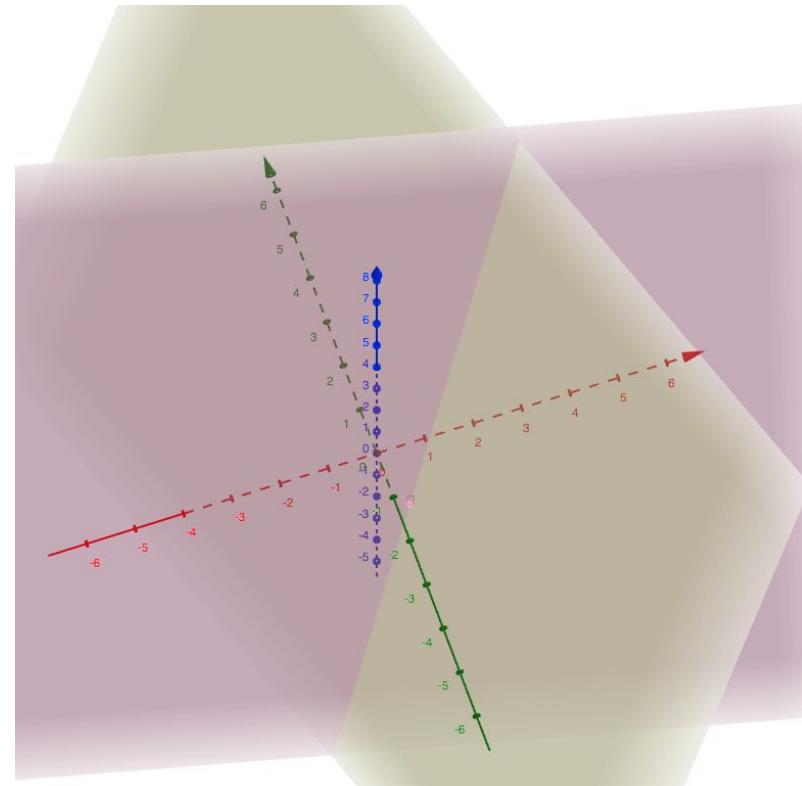
where left pseudo-inverse of \mathbf{A} is defined as:

$$\mathbf{A}^+ := \mathbf{V}\Sigma^+\mathbf{U}^T$$

$\tilde{\mathbf{x}}$ is the solution with minimal l_2 -norm out of the infinitely many solutions.

In other words, $\hat{\mathbf{x}}$ being one of the infinitely many solutions:

$$\|\tilde{\mathbf{x}}\|_2 \leq \|\hat{\mathbf{x}}\|_2, \forall \hat{\mathbf{x}} \text{ s.t. } \mathbf{A}\hat{\mathbf{x}} = \mathbf{b}$$



Pseudo-inverse: Overdetermined case

For $\mathbf{Ax} = \mathbf{b}$

Let,

$$\tilde{\mathbf{x}} = \mathbf{A}^+ \mathbf{b}$$

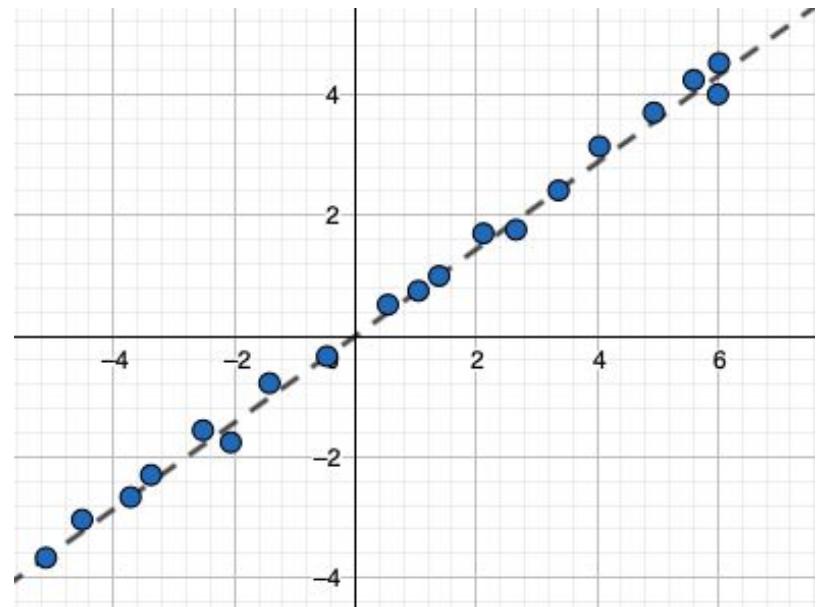
where *left* pseudo-inverse of \mathbf{A} is defined as:

$$\mathbf{A}^+ := \mathbf{V}\Sigma^+\mathbf{U}^T$$

$\tilde{\mathbf{x}}$ is the approximate solution with minimal l_2 -error out of the infinitely many possible approximate solutions.

In other words, $\hat{\mathbf{x}}$ being one of the infinitely many possible approximations of \mathbf{x} :

$$\|\mathbf{A}\tilde{\mathbf{x}} - \mathbf{b}\|_2 \leq \|\mathbf{A}\hat{\mathbf{x}} - \mathbf{b}\|_2, \hat{\mathbf{x}} \in \mathbb{R}^n$$



Good old SVD: \mathbf{U} and \mathbf{C}

$$\mathbf{M} = \mathbf{U} \ \boldsymbol{\Sigma} \ \mathbf{V}^T$$

$$\mathbf{M} = \mathbf{U} \ (\boldsymbol{\Sigma} \ \mathbf{V}^T)$$

$$\mathbf{M} = \mathbf{U} \ \mathbf{C}$$

Good old SVD: \mathbf{U} and \mathbf{C}

Get a rank-k approximation of \mathbf{M}

$$\tilde{\mathbf{M}}_{dxn} = \mathbf{U}_{dxk} \Sigma_{kxk} \mathbf{V}^T_{kxn}$$

$$\tilde{\mathbf{M}} = \mathbf{U}_{dxk} \mathbf{C}_{kxn}$$

SVD → PCA

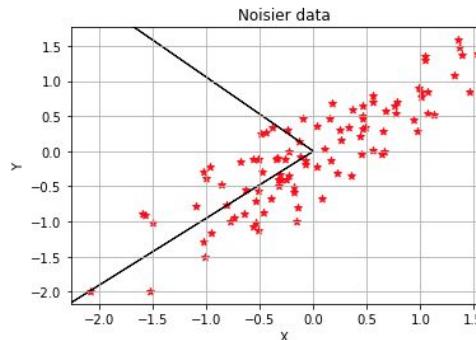
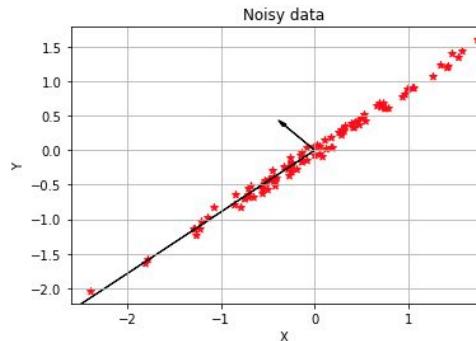
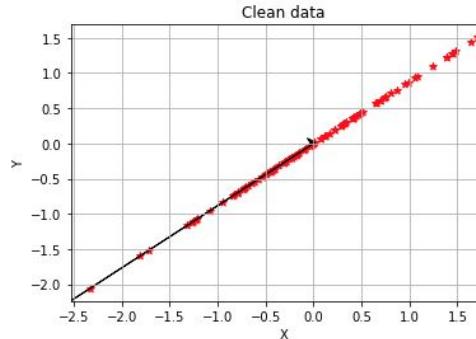
No formal introduction to PCA, but SVD has it all:

- Make sure that your data is zero-mean
- Normalize data if needed
- Columns of \mathbf{U} points along the *Principal Directions*
- Singular values indicate the dominance of *Principal Directions*

Example: SVD → PCA

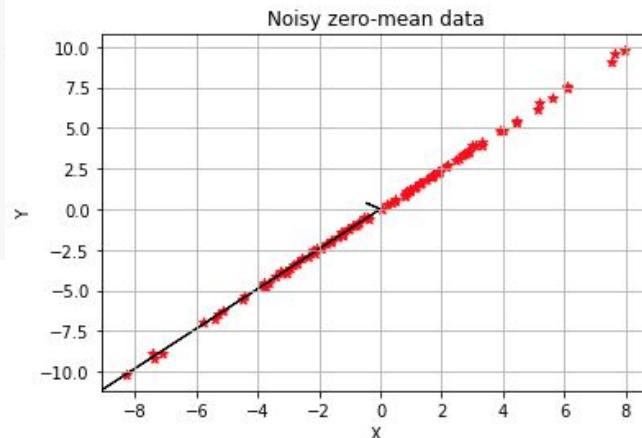
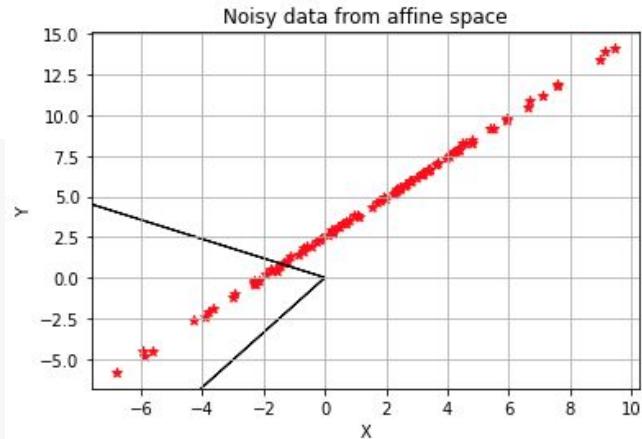
```
1 # let's have datapoints on a line in R2
2 D = DataInSubspace(2, 100, 1)
3 # add a bit of noise
4 Dn = D + np.random.randn(*D.shape)*0.05
5 Dn2 = D + np.random.randn(*D.shape)*0.3
6
7 U,S,Vt = np.linalg.svd(D, full_matrices=False)
8 Un,Sn,Vtn = np.linalg.svd(Dn, full_matrices=False)
9 Un2,Sn2,Vtn2 = np.linalg.svd(Dn2, full_matrices=False)
10
11 # check out the left singular values
12 MatPrint(U @ np.diag(S), 'U of D')
13 MatPrint(Un @ np.diag(Sn), 'U of Dn')
14 MatPrint(Un2 @ np.diag(Sn2), 'U of Dn2')
15
16 CData(D, U @ np.diag(S), title='Clean data')
17 CData(Dn, Un @ np.diag(Sn), title='Noisy data')
18 CData(Dn2, Un2 @ np.diag(Sn2), title='Noisier data')
```

```
U of D
| -7.78   -0.00 |
| -6.88   0.00 |
U of Dn
| -7.82   -0.32 |
| -6.99   0.36 |
U of Dn2
| -7.41   -1.92 |
| -7.05   2.02 |
```



Zero mean it if data comes from an affine space

```
1 # let's have datapoints on a line in R2
2 D = DataInSubspace(2, 100, 1)*5
3 # add a bit of noise
4 Dn = D + np.random.randn(*D.shape)*0.05
5 # translate noisy data
6 TM = np.array( [[2],[5]])
7 Dnt = Dn.copy() + TM
8 # Dnt is assumed to be the real data
9 Un,Sn,Vtn = np.linalg.svd(Dnt, full_matrices=False)
10 CData(Dnt, Un @ np.diag(Sn), title='Noisy data from affine space')
11
12 # make data zero-mean
13 Dz = Dnt - Dnt.mean(axis=1).reshape(2,-1)
14 U,S,Vt = np.linalg.svd(Dz, full_matrices=False)
15 CData(Dz, U @ np.diag(S), title='Noisy zero-mean data')
```



Yet another decomposition method

CUR Decomposition

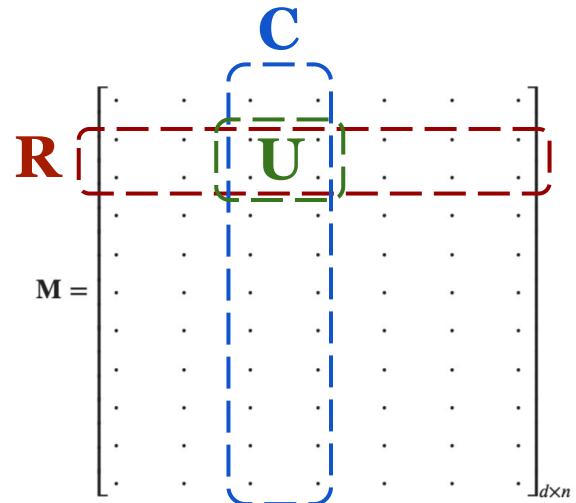
- Compress low rank matrices
- Filter noisy data
- Complete missing data
- Basis from the data itself !!!
- ...

CUR Decomposition: no loss case

- $C = r$ columns
- $R = r$ rows
- $U = \text{Intersection of } C \text{ & } R$

where $r = \text{rank}(M)$

$$M = C U^{-1}R$$



SVD vs CUR

SVD is unique & forms an orthonormal basis for data

CUR is not unique & comes from data itself

Under certain conditions, denoising via CUR will perform better than SVD

CUR Decomposition: no loss case

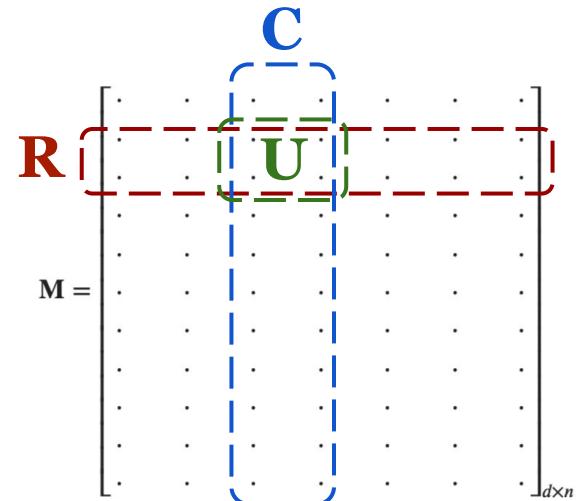
- $C = r$ columns
- $R = r$ rows
- $U = \text{Intersection of } C \text{ & } R$

Where

actual rank(M) = r

But data is noisy so

$\text{rank}(M) = \min(dxn)$



CUR example: $\mathbf{M} = \mathbf{C} \mathbf{U}^{-1} \mathbf{R}$ $\text{rank}(\mathbf{M}) = 2$

| | | | | | | | | | |
|-------|--------|--------|--------|-------|--------|-------|--------|--------|--------|
| 24.00 | -8.00 | -15.00 | 38.00 | -9.00 | 41.00 | -9.00 | -16.00 | -20.00 | 26.00 |
| 7.00 | 33.00 | 42.00 | -11.00 | 4.00 | -30.00 | 4.00 | 13.00 | 3.00 | -41.00 |
| 33.00 | 39.00 | 45.00 | 21.00 | -3.00 | -3.00 | -3.00 | 3.00 | -15.00 | -33.00 |
| 7.00 | -15.00 | -21.00 | 19.00 | -5.00 | 27.00 | -5.00 | -11.00 | -9.00 | 25.00 |
| -9.00 | 1.00 | 3.00 | -13.00 | 3.00 | -13.00 | 3.00 | 5.00 | 7.00 | -7.00 |
| 28.00 | 4.00 | 0.00 | 36.00 | -8.00 | 32.00 | -8.00 | -12.00 | -20.00 | 12.00 |
| 4.00 | -20.00 | -27.00 | 18.00 | -5.00 | 29.00 | -5.00 | -12.00 | -8.00 | 30.00 |
| 20.00 | -4.00 | -9.00 | 30.00 | -7.00 | 31.00 | -7.00 | -12.00 | -16.00 | 18.00 |
| 2.00 | -2.00 | -3.00 | 4.00 | -1.00 | 5.00 | -1.00 | -2.00 | -2.00 | 4.00 |
| 17.00 | 39.00 | 48.00 | -1.00 | 2.00 | -24.00 | 2.00 | 11.00 | -3.00 | -43.00 |

| | | | | | | | | | |
|-------|--------|--------|--------|-------|--------|-------|--------|--------|--------|
| 24.00 | -8.00 | -15.00 | 38.00 | -9.00 | 41.00 | -9.00 | -16.00 | -20.00 | 26.00 |
| 7.00 | 33.00 | 42.00 | -11.00 | 4.00 | -30.00 | 4.00 | 13.00 | 3.00 | -41.00 |
| 33.00 | 39.00 | 45.00 | 21.00 | -3.00 | -3.00 | -3.00 | 3.00 | -15.00 | -33.00 |
| 7.00 | -15.00 | -21.00 | 19.00 | -5.00 | 27.00 | -5.00 | -11.00 | -9.00 | 25.00 |
| -9.00 | 1.00 | 3.00 | -13.00 | 3.00 | -13.00 | 3.00 | 5.00 | 7.00 | -7.00 |
| 28.00 | 4.00 | 0.00 | -36.00 | 8.00 | 32.00 | 8.00 | -12.00 | 20.00 | 12.00 |
| 4.00 | -20.00 | -27.00 | 18.00 | -5.00 | 29.00 | -5.00 | -12.00 | -8.00 | 30.00 |
| 20.00 | -4.00 | -9.00 | 30.00 | -7.00 | 31.00 | -7.00 | -12.00 | -16.00 | 18.00 |
| 2.00 | -2.00 | -3.00 | 4.00 | -1.00 | 5.00 | -1.00 | -2.00 | -2.00 | 4.00 |
| 17.00 | 39.00 | 48.00 | -1.00 | 2.00 | -24.00 | 2.00 | 11.00 | -3.00 | -43.00 |

| | | | | | | | | | |
|-------|--------|--------|--------|-------|--------|-------|--------|--------|--------|
| 24.00 | -8.00 | -15.00 | 38.00 | -9.00 | 41.00 | -9.00 | -16.00 | -20.00 | 26.00 |
| 7.00 | 33.00 | 42.00 | -11.00 | 4.00 | -30.00 | 4.00 | 13.00 | 3.00 | -41.00 |
| 33.00 | 39.00 | 45.00 | 21.00 | -3.00 | -3.00 | -3.00 | 3.00 | -15.00 | -33.00 |
| 7.00 | -15.00 | -21.00 | 19.00 | -5.00 | 27.00 | -5.00 | -11.00 | -9.00 | 25.00 |
| -9.00 | 1.00 | 3.00 | -13.00 | 3.00 | -13.00 | 3.00 | 5.00 | 7.00 | -7.00 |
| 28.00 | 4.00 | 0.00 | 36.00 | -8.00 | 32.00 | -8.00 | -12.00 | -20.00 | 12.00 |
| 4.00 | -20.00 | -27.00 | 18.00 | -5.00 | 29.00 | -5.00 | -12.00 | -8.00 | 30.00 |
| 20.00 | -4.00 | -9.00 | 30.00 | -7.00 | 31.00 | -7.00 | -12.00 | -16.00 | 18.00 |
| 2.00 | -2.00 | -3.00 | 4.00 | -1.00 | 5.00 | -1.00 | -2.00 | -2.00 | 4.00 |
| 17.00 | 39.00 | 48.00 | -1.00 | 2.00 | -24.00 | 2.00 | 11.00 | -3.00 | -43.00 |

What if...

Data is not coming from a subspace?

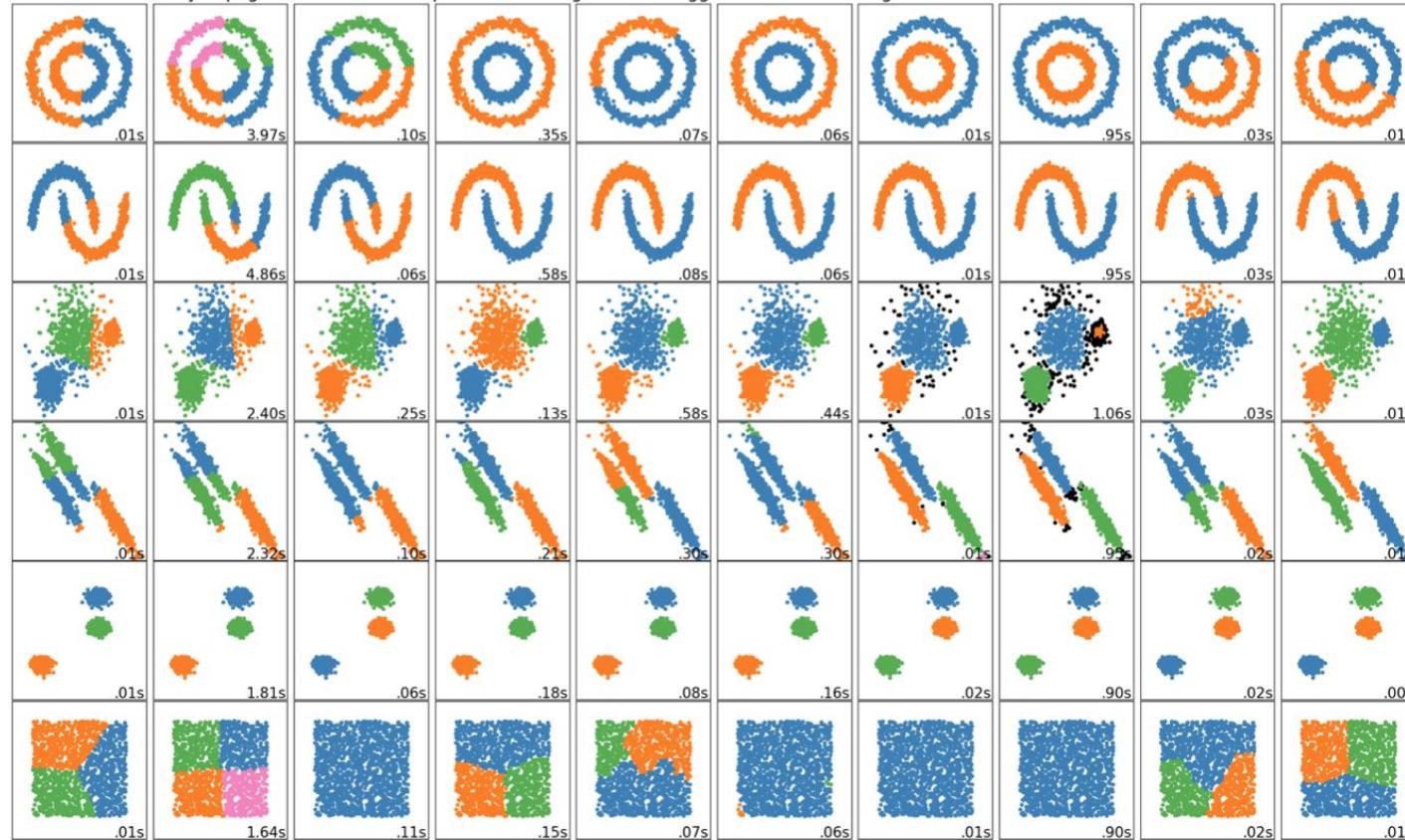
What if multiple subspaces are involved?

What if data is not even coming from subspaces?

... well that's life...

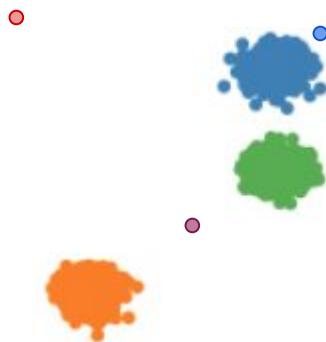
What if data does not come from a single cluster?

And the fun begins...



Multiple Clusters

How do you handle incoming data?



What if there is

- a conflict?
- an outlier?

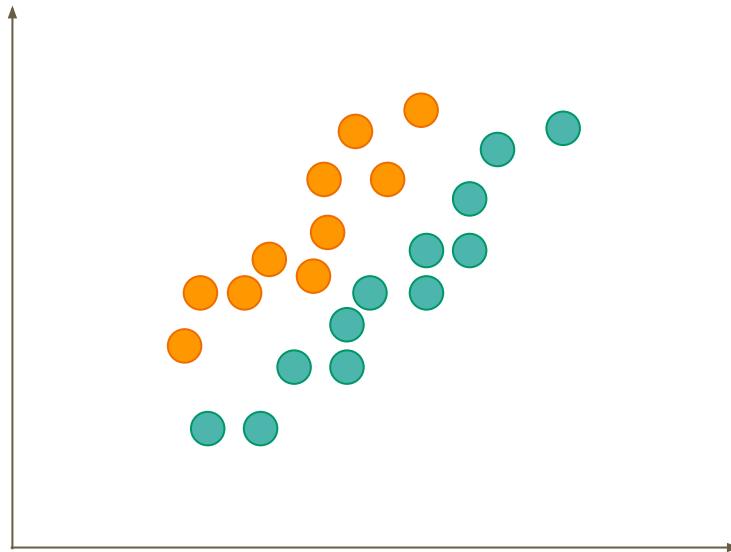
What to do with high dimensional data?

- Cannot visualize it, *or can we?*
- If not *very* low-rank - good luck
- If *very* low rank (3 or less) visualize via dimension reduction

By the way: What is the point in visualization?

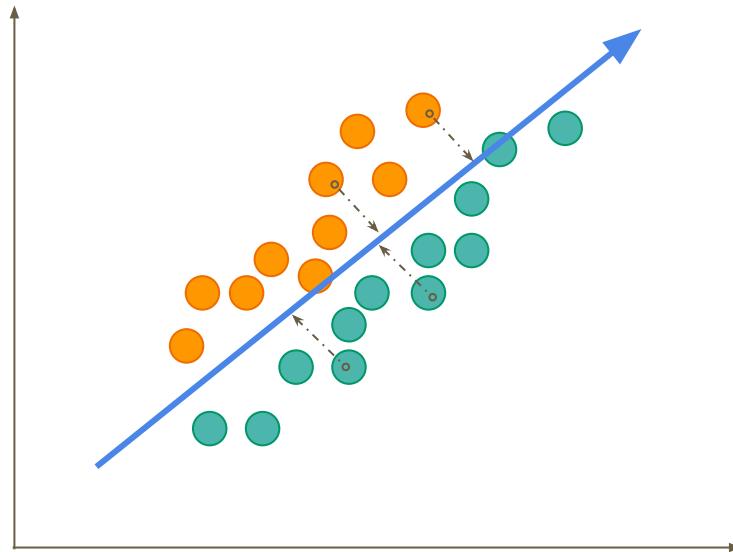
Reducing dimensionality: multiple clusters

What if SVD is used to reduce dimensionality?



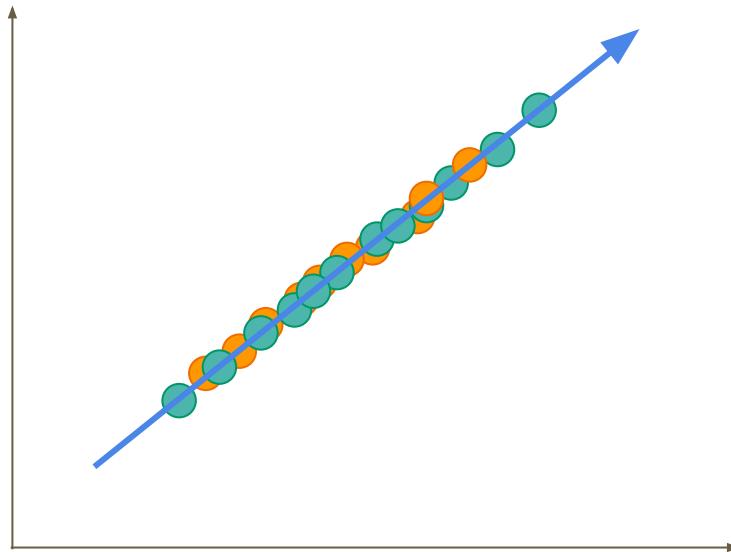
Reducing dimensionality: multiple clusters

SVD way: project data to low dimensions



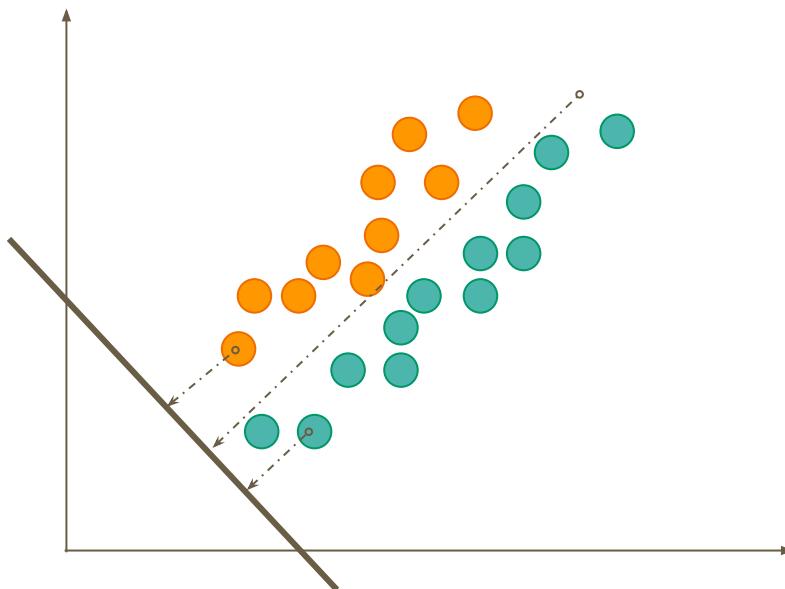
Reducing dimensionality: makes separation harder!

SVD way: project data to low dimensions → did I do something useful?



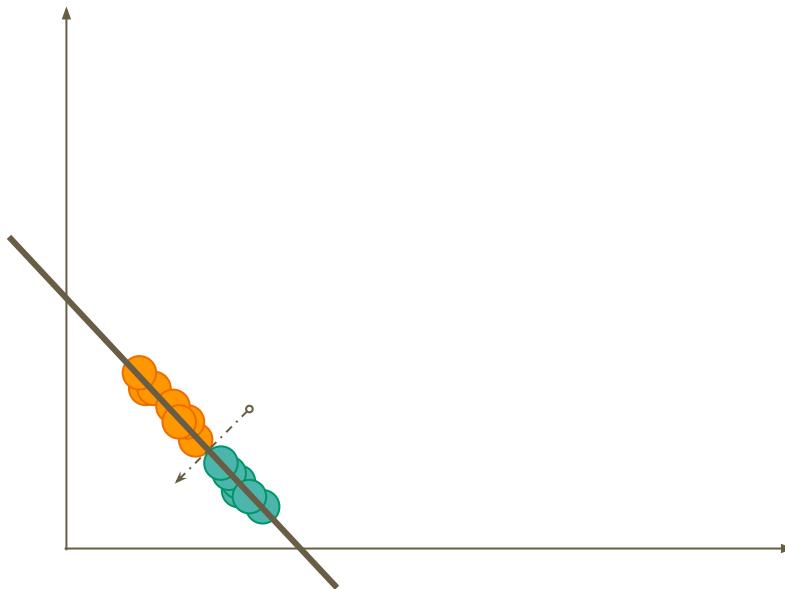
Reducing dimensionality: multiple clusters

LDA: Linear Discriminant Analysis



Reducing dimensionality: multiple clusters

LDA: Linear Discriminant Analysis



PCA vs LDA

- Both reduce dimension
- PCA provides principal directions
- PCA is unsupervised
- LDA provides best directions for discriminating data
- LDA is supervised
- PCA can be used within clusters once clusters are set

Similarity Matrix

Assume that data is in the columns of matrix $\mathbf{M}_{d \times n}$, where m_i are the columns corresponding to data points.

An *ideal* similarity matrix for \mathbf{M} is an $n \times n$ matrix \mathbf{S}_M which is (in most cases) symmetric. If m_i and m_j belong to the same cluster / group / subspace etc., $s_{ij} = 1$ and 0 otherwise.

Example: Let's start with a small matrix

Complete the IDEAL Similarity Matrix

$S_M =$

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

Complete the IDEAL Similarity Matrix

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

Complete the IDEAL Similarity Matrix: *with a graph on the side*

$$S_N =$$

| | | | | | |
|---|--|--|---|---|---|
| | | | | | |
| | | | 1 | | |
| 0 | | | | | 1 |
| | | | | 1 | |
| | | | | | 1 |

Similarity Matrix

- Good for analyzing high dimensional data
 - Can be displayed like an image
 - In general, data can be ordered w.l.g.
 - A block diagonal similarity matrix is expected

$$\mathbf{M} = \begin{bmatrix} & & & & & & & \\ \textcolor{purple}{\blacksquare} & \textcolor{purple}{\blacksquare} & \textcolor{purple}{\blacksquare} & \textcolor{yellow}{\blacksquare} & \textcolor{yellow}{\blacksquare} & \textcolor{yellow}{\blacksquare} & \textcolor{yellow}{\blacksquare} & \textcolor{purple}{\blacksquare} & \textcolor{purple}{\blacksquare} \\ & & & & & & & \end{bmatrix}$$

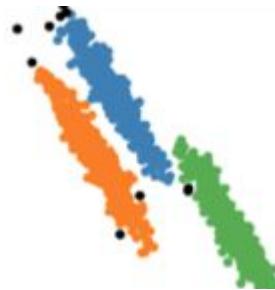
Heuristics in play:

If data is coming from clusters C_p where $p = 1, \dots, r$

$$f_s(\mathbf{c}_i^M, \mathbf{c}_j^M) = s_{ij} = 1 \text{ if } \mathbf{c}_i^M, \mathbf{c}_j^M \in C_p$$

$$f_s(\mathbf{c}_i^M, \mathbf{c}_j^M) = s_{ij} = 0 \text{ if } \mathbf{c}_i^M \in C_p, \mathbf{c}_j^M \in C_q, p \neq q$$

How do you find f_s given a problem?



Heuristics in play:

If data is coming from clusters C_p where $p = 1, \dots, r$

$$f_s(\mathbf{c}_i^M, \mathbf{c}_j^M) = s_{ij} = 1 \text{ if } \mathbf{c}_i^M, \mathbf{c}_j^M \in C_p$$

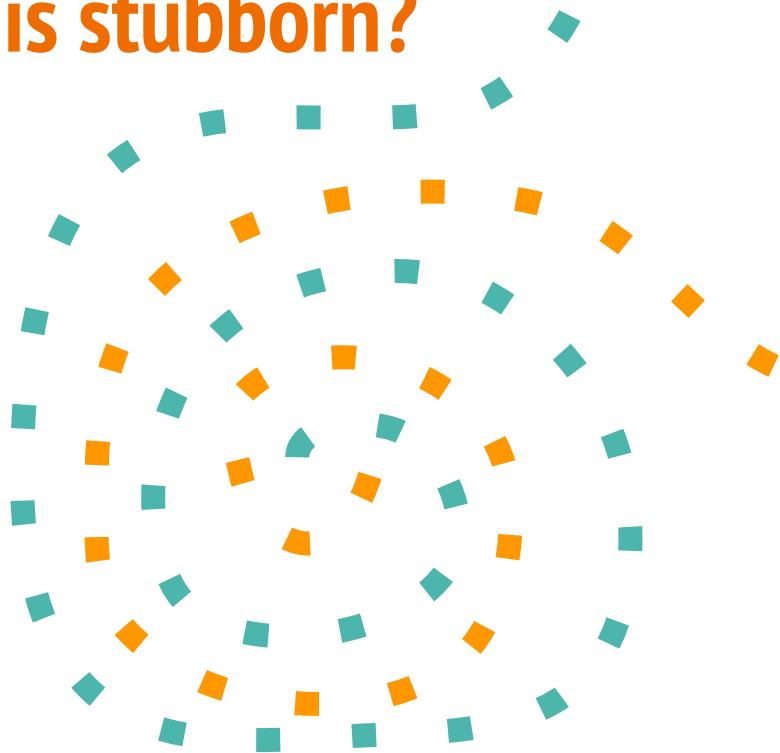
$$f_s(\mathbf{c}_i^M, \mathbf{c}_j^M) = s_{ij} = 0 \text{ if } \mathbf{c}_i^M \in C_p, \mathbf{c}_j^M \in C_q, p \neq q$$

What if you do NOT:

- have any clue about the structure of the data
- know the number of clusters
- Cannot visualize data for cues

Can you still find some f_s given data ?

What if data is stubborn?



See you in another universe brother...

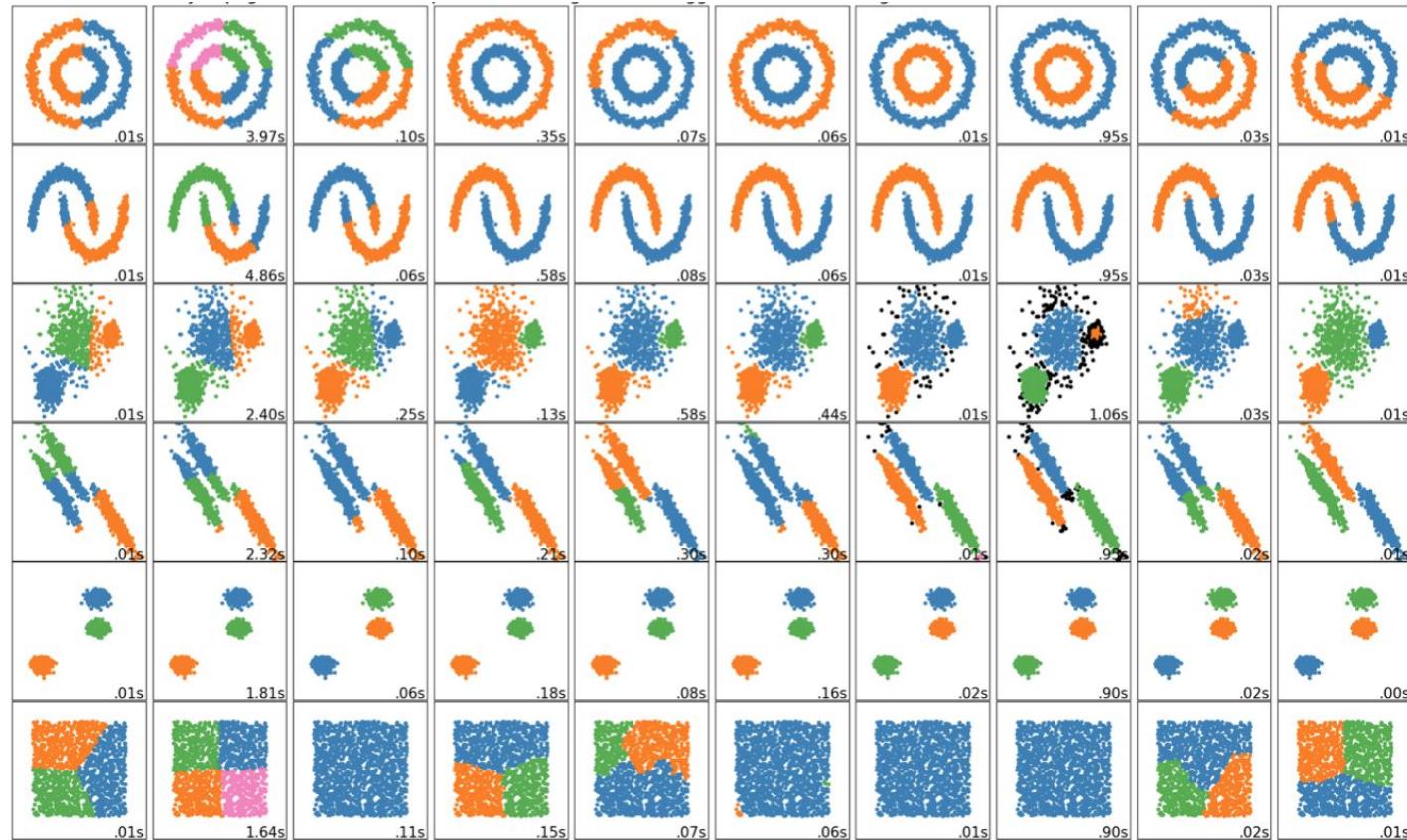
to be continued...

With clustering methods

ME 536

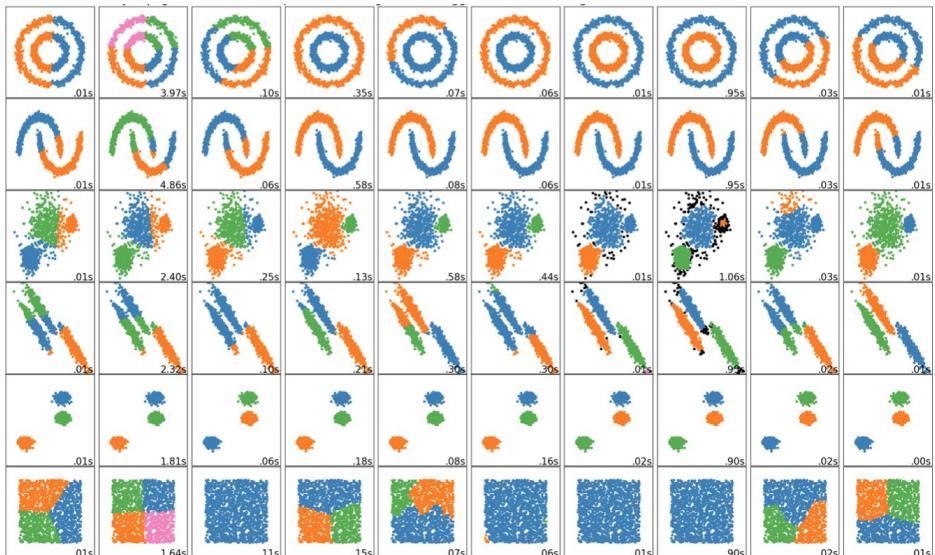
— Week 7-8: Clustering —

Problems in different forms



Cluster 'em all

- Independent of the problem?
- Supervised unsupervised?
- If some information is available?
 - Number of clusters
 - Number of data in clusters



Simple yet basic: k -means - what does k mean?

Algorithm:

Given k

1. Randomly assign k points as cluster center (CM) points
2. Assign points closest to each CM to that cluster
3. Update CMs
4. If update changed any CM goto step 2 else STOP



Questions:

- Why random start ?
- Works best when? i.e. limitations ?
- Using k -means, can we guess k ? Chicken-Egg ?

Clustering: k -means k no more no less

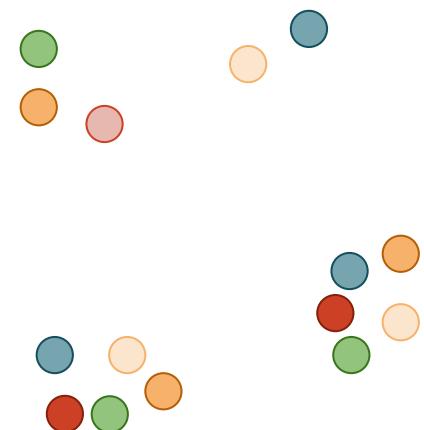
What do you expect for $k = 1, 2, 3$

Can you guarantee that solutions is repeatable? i.e. unique?

Algorithm:

Given k

1. Randomly assign k points as cluster center (CM) points
2. Assign points closest to each CM to that cluster
3. Update CMs
4. If update changed any CM goto step 2 else STOP



Hands on with: *k*-Means

Check out the following link and play with it for various cases:

<http://user.ceng.metu.edu.tr/~akifakkus/courses/ceng574/k-means/>

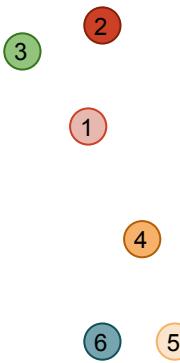
Clustering: k -means

Algorithm:

Given k

1. Randomly assign k points as cluster center (CM) points
2. Assign points closest to each CM to that cluster
3. Update CMs
4. If update changed any CM goto step 2 else STOP

Questions: pros and cons



- Random start
 - Random points in space vs random data points
 - Uniformly distributed
- Better guess than random?
 - Preprocess data ?

Clustering: k -means

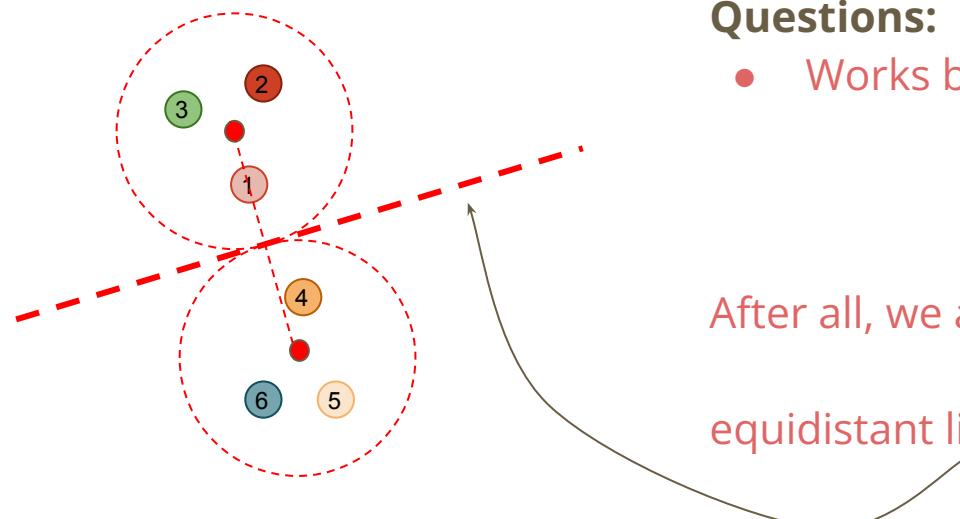
Algorithm:

Given k

1. Randomly assign k points as cluster center (CM) points
2. Assign points closest to each CM to that cluster
3. Update CMs
4. If update changed any CM goto step 2 else STOP

Questions:

- Works best when? i.e. limitations ?



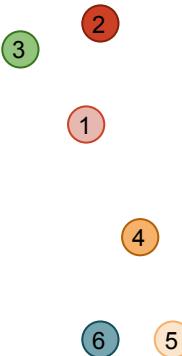
After all, we are finding $\binom{k}{2}$ many
equidistant lines for pairs of CMs: i.e. *Decision boundary*

Clustering: k -means

Algorithm:

Given k

1. Randomly assign k points as cluster center (CM) points
2. Assign points closest to each CM to that cluster
3. Update CMs
4. If update changed any CM goto step 2 else STOP



Questions:

- Just using k -means

figure out best k ?



Improving: k -means

Minibatch k -means:

- Subsets of the input data, randomly sampled in each training iteration.
- Speed on the expense of quality

k -means with constraints:

- Balance cluster size
 - Check out the literature

Best out of several runs

Many other improvements in general exist in the literature

Thinking time: ~5 min

How to improve k-means initial guesses

Without using a search engine or the Internet at all, think about how you can further improve k-means initial guess

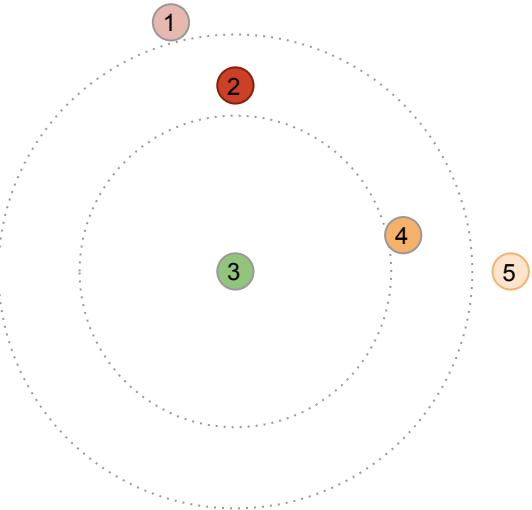
Mechanical Engineer instincts might be helpful :)

Hierarchy matters: even in data :(

Hierarchical Clustering

Algorithm:

- Assign each and every single datapoint in its own cluster
- Join *closest* two clusters
- Stop when there is 1 cluster left



Questions:

- Stop when there is 1 cluster left ???
Genius why not put them into a single cluster to start with?
- What does *closest* mean?

Hierarchical Clustering

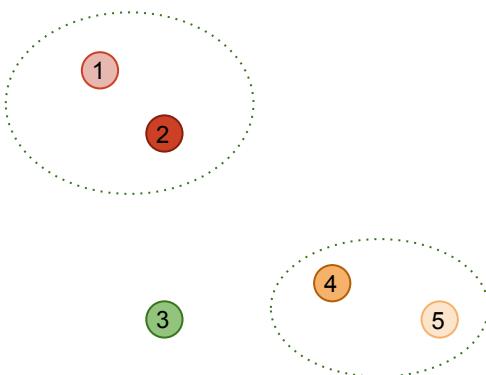
Algorithm:

- Assign each datapoint in a cluster
- **Join closest two clusters**
- Stop when there is 1 cluster left

Neighbour := Any two elements / data points in the same cluster

Questions:

- What does *closest* mean?
 - Single linkage → **shortest** distance between any non-neighbours
 - Complete linkage → **longest** distance between any non-neighbours
 - Average linkage → **average** of all distances between non-neighbours
 - Ward → minimize sum of squared distances between non-neighbours



Note that all possible distances between non-neighbors forms a matrix → **matrix norms** can be used to define closeness as well

Hierarchical Clustering

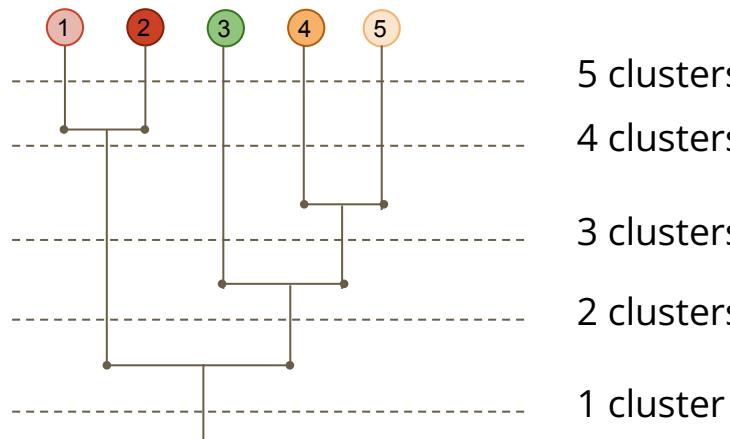
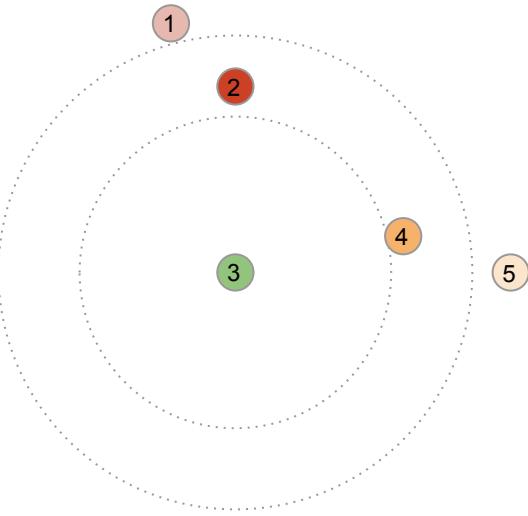
Algorithm:

- Assign each datapoint in a cluster
- Join *closest* two clusters
- Stop when there is 1 cluster left

Questions:

- Genius why not **put them into a single cluster to start with?**

→ let's get the DENDROGRAM : *simpler than it sounds*



Clustered using
single linkage

Try to form the
dendrogram using
complete linkage

Hierarchical Clustering



dendrogram

/dəndrə(ʊ)grəm/

noun

plural noun: **dendograms**

a tree diagram, especially one showing taxonomic relationships.

Definitions from Oxford Languages

- Dendograms
 - visually inspectable - high dimensional data → no problem
 - a means of choosing best cluster size

Distance between cities

Self study: Plan your business

If you were to put **two logistic centers**
to *deliver goods* sent to these cities:

- 1) List of cities that belong to each logistic center?
- 2) City in which the center to be established?
- 3) 'closeness' measure you have chosen

Also elaborate on what other parameters might be considered to make a better decision?
Focus on the ones that can be quantified for automated decision making.

If these additional parameters would change your decisions that is solely based on physical distance, try to explain if you would update the 'distance metric' or what else?

Use dendogram, all other relevant work is better be on a piece of paper: [ellemeden bellenmez](#)

NOTE: This table is given in course website under files

| City | ANKARA | BALIKESİR | BİLECİK | BOLU | BURSA | ÇANKIRI | KİRŞEHİR | KONYA | AKSARAY |
|-----------|--------|-----------|---------|------|-------|---------|----------|-------|---------|
| ANKARA | 0 | 536 | 316 | 191 | 385 | 130 | 184 | 258 | 225 |
| BALIKESİR | 536 | 0 | 245 | 422 | 151 | 655 | 706 | 551 | 693 |
| BİLECİK | 316 | 245 | 0 | 213 | 94 | 446 | 486 | 421 | 526 |
| BOLU | 191 | 422 | 213 | 0 | 271 | 233 | 378 | 456 | 423 |
| BURSA | 385 | 151 | 94 | 271 | 0 | 504 | 555 | 490 | 595 |
| ÇANKIRI | 130 | 655 | 446 | 233 | 504 | 0 | 213 | 353 | 310 |
| KİRŞEHİR | 184 | 706 | 486 | 378 | 555 | 213 | 0 | 258 | 110 |
| KONYA | 258 | 551 | 421 | 456 | 490 | 353 | 258 | 0 | 148 |
| AKSARAY | 225 | 693 | 526 | 423 | 595 | 310 | 110 | 148 | 0 |

Cluster into 2: using k -Means

How will you represent a city as a data point?

What will be elements (or features) of the vector that correspond to a data point?

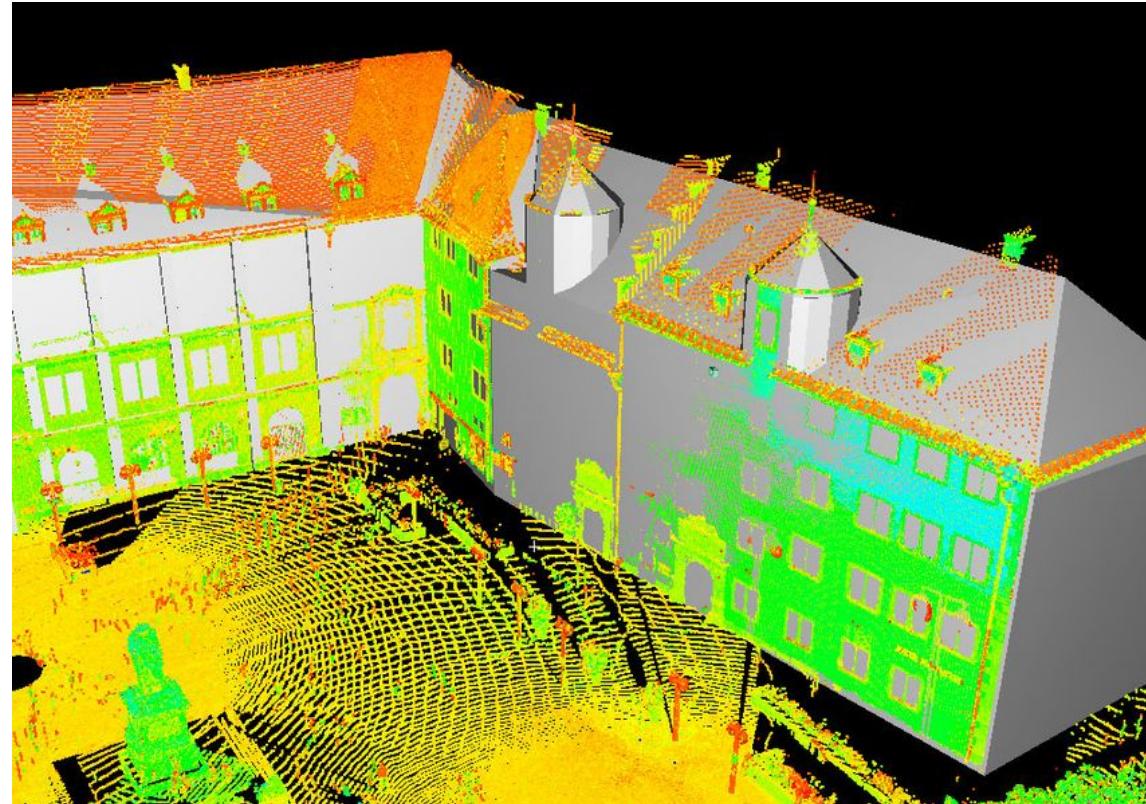
| City | Distance between cities | | | | | | | | |
|-----------|-------------------------|-----------|---------|------|-------|---------|----------|-------|---------|
| | ANKARA | BALIKESİR | BİLECİK | BOLU | BURSA | ÇANKIRI | KİRŞEHİR | KONYA | AKSARAY |
| ANKARA | 0 | 536 | 316 | 191 | 385 | 130 | 184 | 258 | 225 |
| BALIKESİR | 536 | 0 | 245 | 422 | 151 | 655 | 706 | 551 | 693 |
| BİLECİK | 316 | 245 | 0 | 213 | 94 | 446 | 486 | 421 | 526 |
| BOLU | 191 | 422 | 213 | 0 | 271 | 233 | 378 | 456 | 423 |
| BURSA | 385 | 151 | 94 | 271 | 0 | 504 | 555 | 490 | 595 |
| ÇANKIRI | 130 | 655 | 446 | 233 | 504 | 0 | 213 | 353 | 310 |
| KİRŞEHİR | 184 | 706 | 486 | 378 | 555 | 213 | 0 | 258 | 110 |
| KONYA | 258 | 551 | 421 | 456 | 490 | 353 | 258 | 0 | 148 |
| AKSARAY | 225 | 693 | 526 | 423 | 595 | 310 | 110 | 148 | 0 |

Physical Distance makes sense: to an extent

Treat **point coordinates** as data and find:

Walls, Ceiling, ground

→ Given Point Cloud



Physical Distance makes sense: to an extent

Feature: a measurable / quantifiable aspect

→ location, weight, color, number of corners, has wings etc.

Feature vector: a vector of features

Clustering over feature vectors is common but tricky

Larger range for a feature might dominate

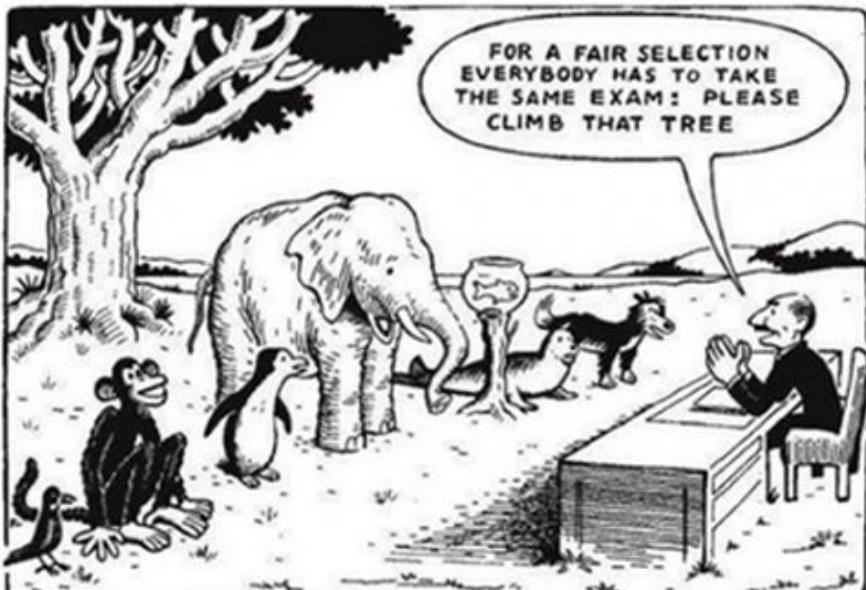
Consider:

Where features have significantly different ranges

| | | | | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| 26.00 | 71.00 | 28.00 | 30.00 | 78.00 | 15.00 | 43.00 | 77.00 | 92.00 | 26.00 |
| 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.30 | 0.01 | 0.02 | 0.33 | 0.00 | 0.01 | 0.39 | 0.01 | 0.02 |

Fair advantage for all features: Feature Scaling

a.k.a Data Normalization



Objective:

- Rescale the range
- Reshape the distribution

Scale all values of individual features

Different methods can be adopted
for different features

Fair advantage for all features: Feature Scaling

a.k.a Data Normalization

min-max-scaling:

Scale all **values** in [0-1]

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

z-scaling:

a.k.a. z-score normalization

all values have
zero-mean and unit
standard deviation

$$x' = \frac{x - x_{mean}}{\sigma}$$

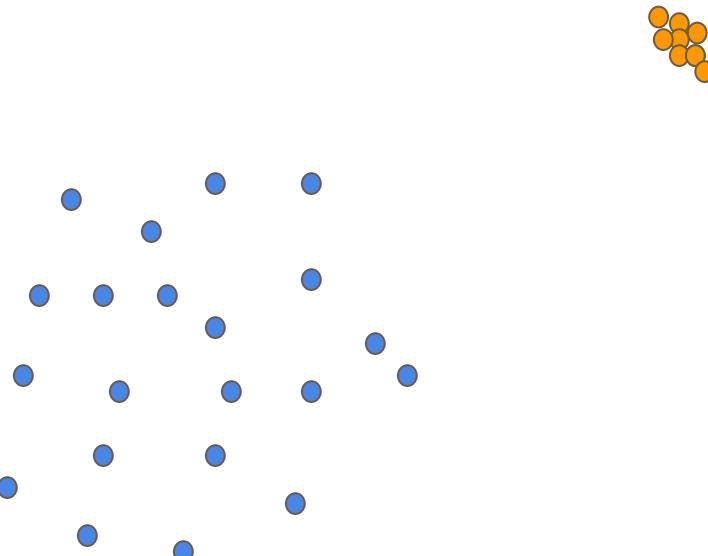
just normalization

all values have
zero-mean scaled over
the range of values

$$x' = \frac{x - x_{mean}}{x_{max} - x_{min}}$$

Example - clusters live separately in their own ways?

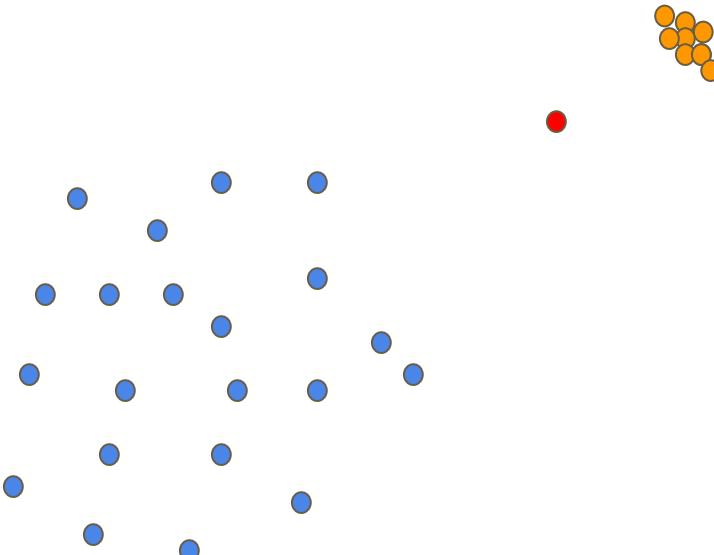
Moving beyond ℓ_1 , ℓ_2 etc



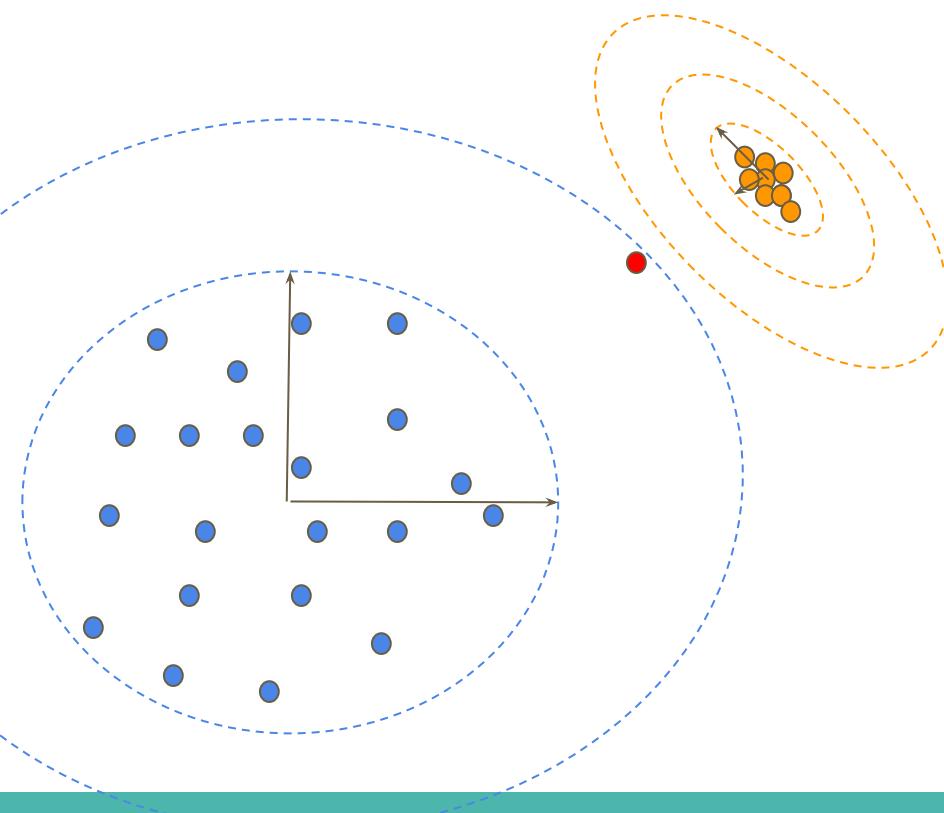
How many clusters do you see?

Mahalanobis distance - say who

Where does the new point belong to?



such as - Mahalanobis distance - say who



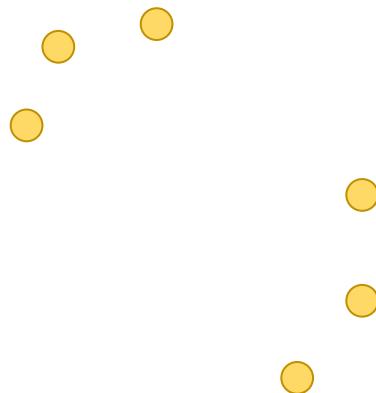
What if distances are **normalized** along **principal axis**?

Or based on **standard deviation** **within the cluster**?

Graph Interpretation of Data

Data points are nodes, edges connect nodes (i,j) , where edge value = $f_s(m_i, m_j)$

Example: Form the graph, propose an f_s and similarity matrix

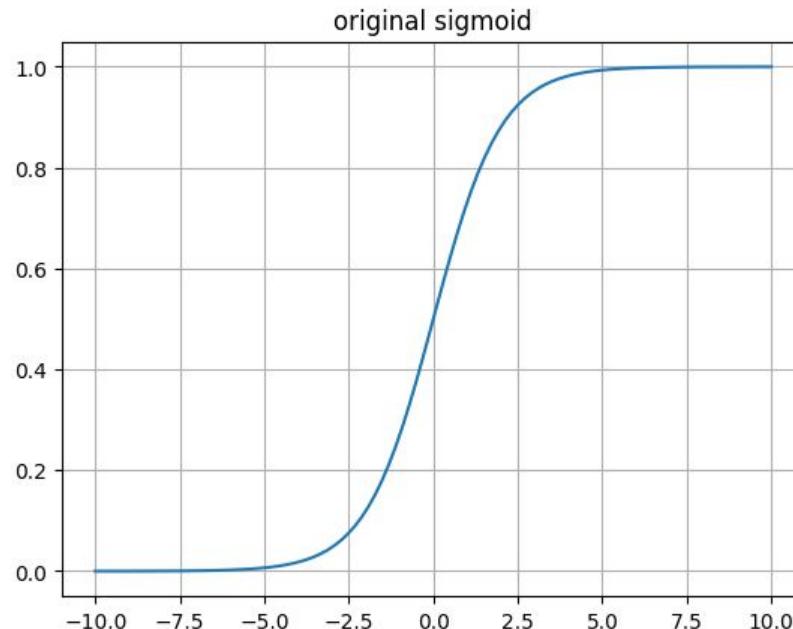


How do I write a function
that switch between
0 and 1?

A useful function: Sigmoid

$$\frac{1}{1+e^{-x}}$$

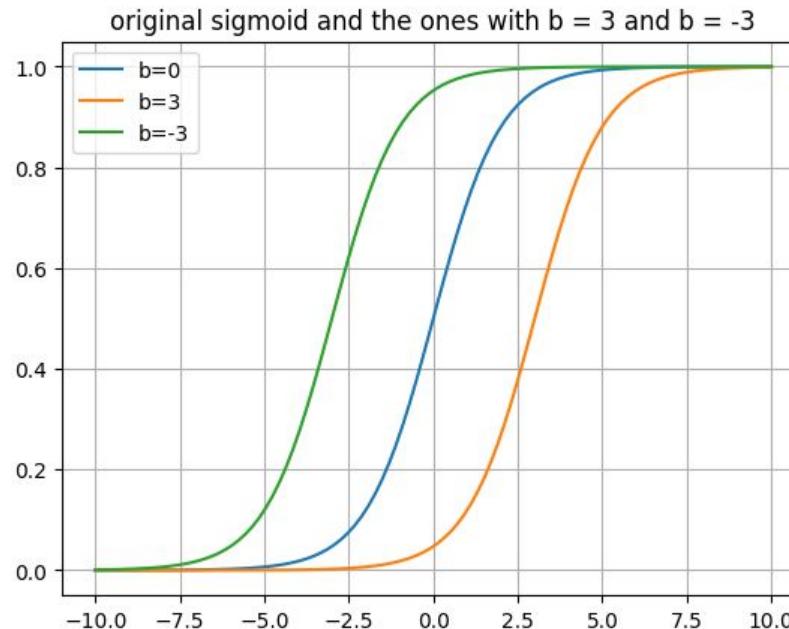
In its original form it is a switching function, similar to a step function, but smooth



A useful function: Sigmoid

$$\frac{1}{1+e^{b-x}}$$

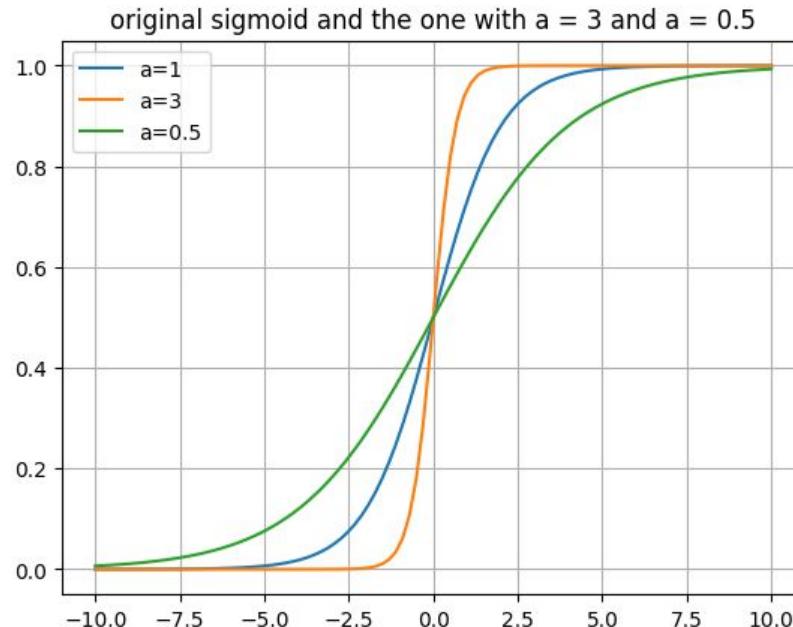
What if the transition should be somewhere else?



A useful function: Sigmoid

$$\frac{1}{1+e^{-ax}}$$

What if the transition should be faster?



A useful function: Sigmoid

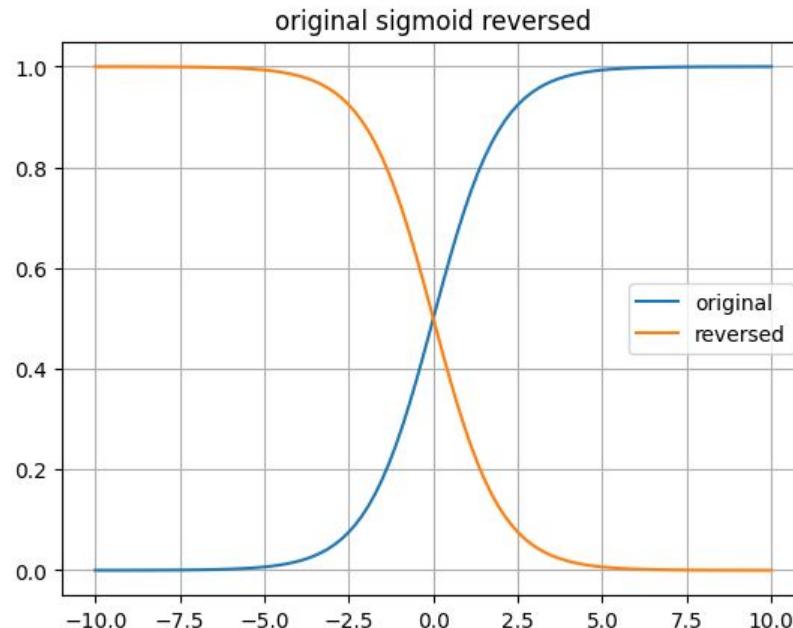
All together

$$\frac{1}{1+e^{b-ax}}$$

A useful function: Sigmoid

$$1 - \frac{1}{1+e^{b-ax}}$$

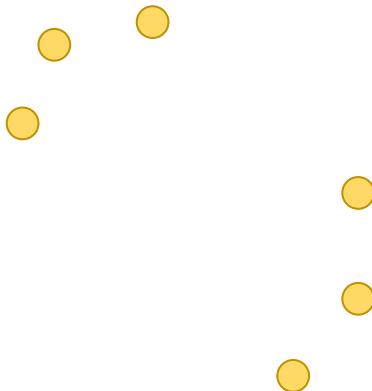
What if the switching logic should be reversed?



Graph Interpretation of Data

Data points are nodes, edges connect nodes (i,j) , where edge value = $f_s(m_i, m_j)$

Example: Form the graph, propose an f_s and similarity matrix



Graph theory: because

Maria Chudnovsky: *a mathematician working on graph theory*

"In mathematics, a graph is an abstraction that represents a set of similar things and the connections between them -- e.g., cities and their roads connecting them, networks of friendship among people, websites and their links to other sites."

My Translation 4 U:

You can model and analyze many real life problems with graph theory → you can make money off of graph theory even if you are not a mathematician working on it.

Formal Definitions

Symbolism is Not universally unique

Formal Definitions

Data points → nodes → **vertex** : a vertex is *incident* to the edges it connects

Connects 2 *adjacent* vertices → **edge**

Connects a vertex to itself → self-loop

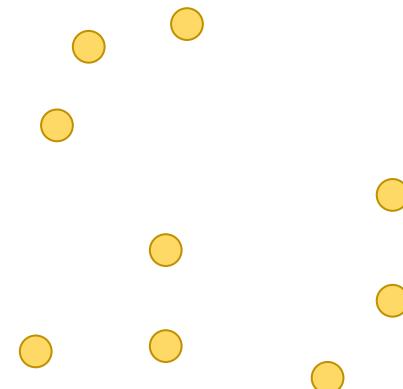
No edge → no relationship

Edges might be directed → directed graph

Otherwise → **undirected graph**

Edges might have (*commonly positive*) values, if not, edge values are assumed to be equal (1)

Multi-graph has one or more parallel edges (i.e. multiple edges between 2 vertices)



Formal Definitions

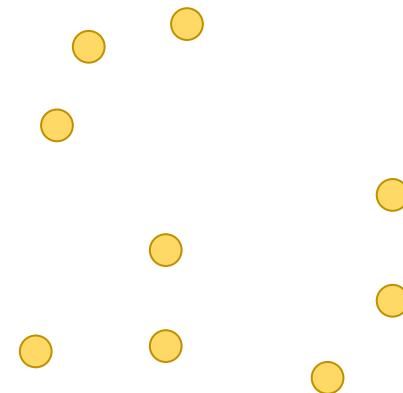
Simple graph:

- No self-loops
- Undirected
- No multiple / parallel edges

$G=\{V,E\}$ defines a graph with vertices $V=\{v_1, v_2, \dots, v_n\}$ and edges $E=\{e_{12}, e_{23}, \dots, e_{mk}\}$.

Let $|V|$ =Number of vertices, $|E|$ =Number of edges for a simple graph:

$$2|E| \leq |V|^2 - |V|$$



Formal Definitions

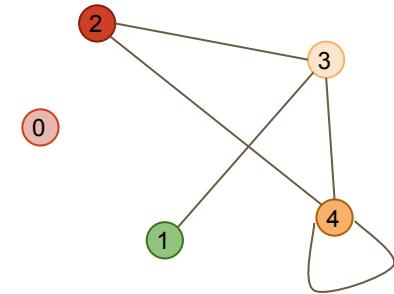
$G = \{V, E\}$ defines a graph with vertices $V = \{v_1, v_2, \dots, v_n\}$ and edges $E = \{e_{12}, e_{23}, \dots, e_{mk}\}$.

e_{12} is an edge from v_1 to v_2

Degree of a vertex: Number of edges (in or out) connected to a vertex.

A *Degree Zero vertex* (a.k.a. *isolated vertex*) is *disconnected* from the rest of the graph, but possible.

Self-loops *naturally* add 2 to the degree!



Formal Definitions

$G = \{V, E\}$ is a graph with vertices $V = \{v_1, v_2, \dots, v_n\}$ and edges $E = \{e_{12}, e_{23}, \dots, e_{mk}\}$.

Adjacency Matrix of G : *similar in nature to a similarity matrix*

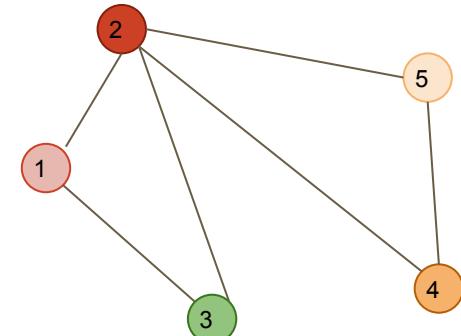
$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Degree Matrix of G : diagonally encodes degrees of vertices

$$D = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

Laplacian Matrix of G →

$$L := D - A$$

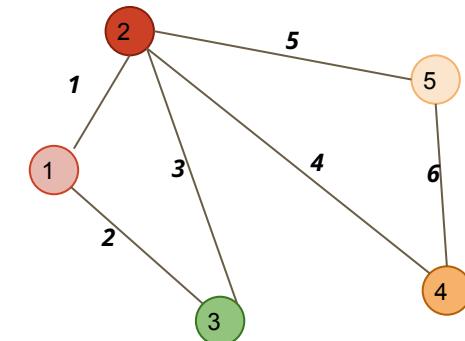


Formal Definitions

$G = \{V, E\}$ is a graph with vertices $V = \{v_1, v_2, \dots, v_n\}$ and edges $E = \{e_{12}, e_{23}, \dots, e_{mk}\}$.

Incidence Matrix of G : Rows correspond to edges, columns to vertices

$$J_{6 \times 5} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$



Laplacian Matrix of G →

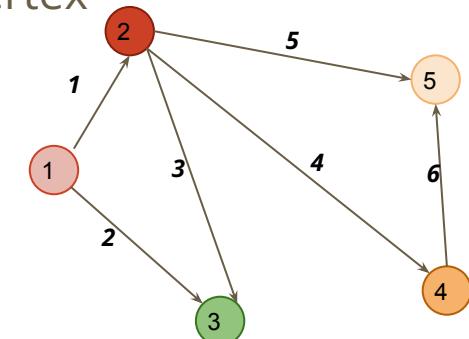
$$L := J^T J$$

Formal Definitions

$G = \{V, E\}$ is a graph with vertices $V = \{v_1, v_2, \dots, v_n\}$ and edges $E = \{e_{12}, e_{23}, \dots, e_{mk}\}$.

Incidence Matrix of G : Rows correspond to edges, columns to vertices
-1 for the left vertex, 1 for the arrived vertex

$$J_{6 \times 5} = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$



Laplacian Matrix of $G \rightarrow$

$$L := J^T J$$

Formal Definitions

$G = \{V, E\}$ is a graph with vertices $V = \{v_1, v_2, \dots, v_n\}$ and edges $E = \{e_{12}, e_{23}, \dots, e_{mk}\}$.

Walk: Any sequence of vertices connected with edges

ex: $W = \{e_{32}, e_{24}, e_{45}, e_{52}\}$ is a walk, $W = \{e_{32}, e_{23}, e_{31}, e_{13}\}$ is a **closed-walk**

Trail: A sequence of vertices connected with distinct edges. (*self-crossing walk possible*)

ex: $T = \{e_{32}, e_{24}, e_{45}, e_{52}\}$ is a trail, $T = \{e_{12}, e_{25}, e_{54}, e_{42}, e_{23}, e_{31}\}$ is a **closed-trail** → **circuit**

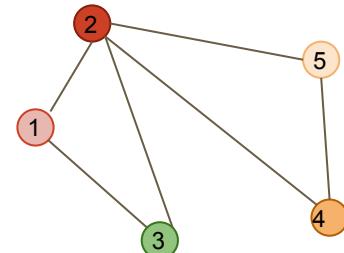
Path: A sequence of distinct vertices and connected with distinct edges
(*hence no self-intersection*).

ex: $P = \{e_{32}, e_{24}\}$ is a path between v_3 and v_4 over v_2

$P = \{e_{12}, e_{23}, e_{31}\}$ is a **closed-path** → **cycle**

Path length: Sum of edge values.

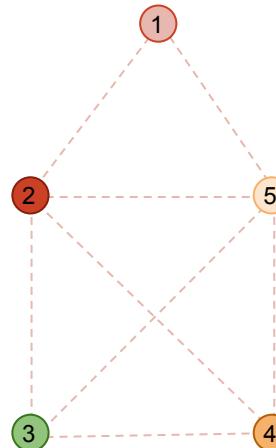
In case all edges have the same value, *number of edges on the path*.



Puzzle from childhood years

Can you connect all vertices with a single:

- Path = ?
- Trail = ?
- Walk = ?

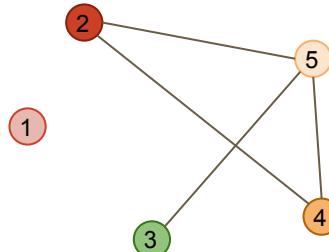
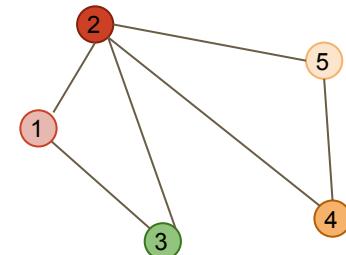


Formal Definitions

$G = \{V, E\}$ defines a graph with vertices $V = \{v_1, v_2, \dots, v_n\}$ and edges $E = \{e_{12}, e_{23}, \dots, e_{mk}\}$.

G is a **connected graph**, if for **any two vertices** $\{v_i, v_j\}$ there **there is a path**.

Disconnected otherwise



Why Graph Laplacian

$G = \{V, E\}$ is a graph with vertices $V = \{v_1, v_2, \dots, v_n\}$ and edges $E = \{e_{12}, e_{23}, \dots, e_{mk}\}$.

Laplacian Matrix of $G \rightarrow L := J^T J$ or $L := D - A$

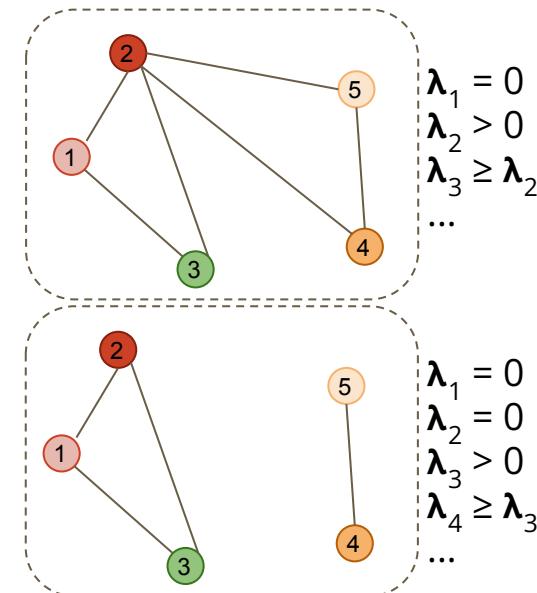
L is beautiful:

Eigenvalues of L are first sorted in ascending order

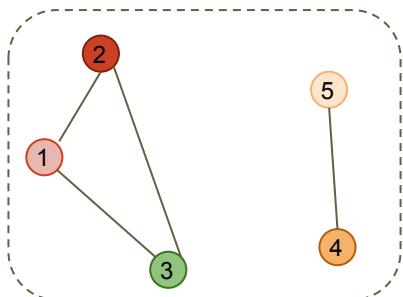
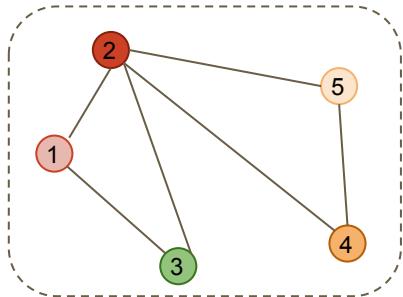
Real eigenvalues with orthogonal eigenvectors

Number of eigenvalues that are 0
→ number of connected sub-graphs

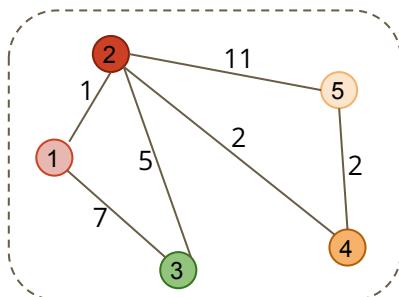
NOTE: Definition of Laplacian Matrix L is not unique



Self Study: Find the graph Laplacian & eigenvalues



Self Study: If you were to cut this in to 2 clusters



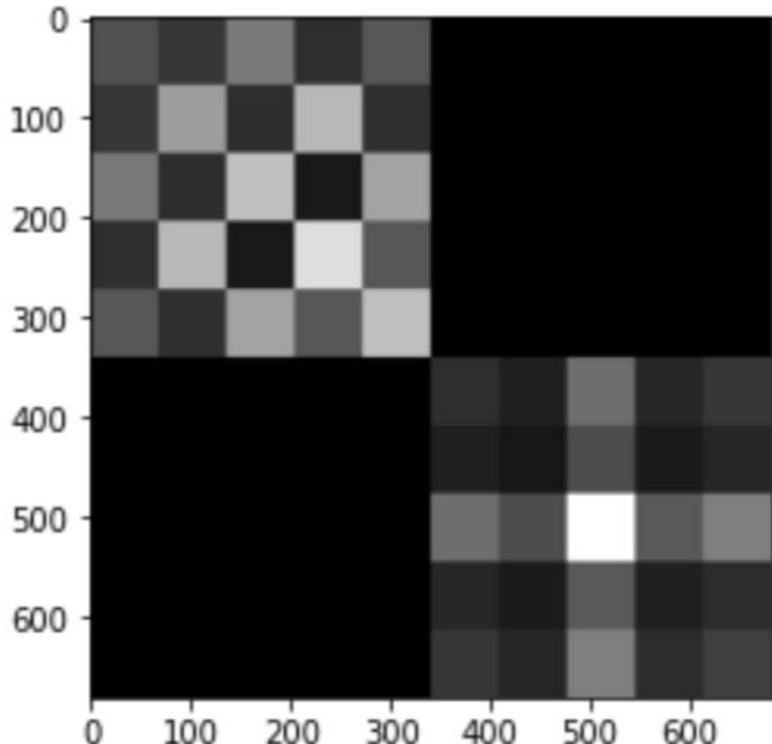
Which edges would you eliminate?

Try to come up with a heuristic function that generates similarity values for this graph and generate graph Laplacian

Recall: Similarity Matrix

- Similarity matrix is similar to adjacency matrix from a graph
- Example:
 - 2 clusters
 - First 350 come from the first clusters
 - Second 350 from the second

HOW CAN YOU FIND such a matrix given the data matrix **M** ?



Shape Interaction Matrix: SIM

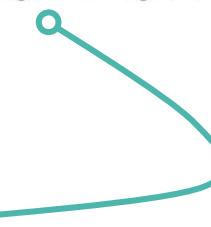
Given a data matrix $\mathbf{M}_{d \times n}$ the similarity matrix $\mathbf{S}_\mathbf{M}$ is calculated using the skinny SVD of \mathbf{M} as follows:

$$\text{let } r = \text{rank}(\mathbf{M})$$

$$\mathbf{M}_{d \times n} = \mathbf{U}_{d \times r} \boldsymbol{\Sigma}_{r \times r} \mathbf{V}_{r \times n}^T$$

$$\mathbf{S}_\mathbf{M} = \mathbf{V}_{n \times r} \mathbf{V}_{r \times n}^T$$

Note that $\mathbf{S}_\mathbf{M}$ is $n \times n$.



One use of SIM:

Assume that data (i.e. columns of $\mathbf{M}_{d \times n}$) are coming from union of r many subspaces: $\mathbf{c}_i^{\mathbf{M}} \in \bigcup \mathbb{S}_j$ where, $i = 1, \dots, n$ and $j = 1, \dots, r$.

Further assume that, data is coming from independent subspaces (i.e. they are *not disjoint*).

For example, in \mathbb{R}^2 **two 1-dimensional subspaces** (i.e. 2 lines) $\mathbb{S}_1, \mathbb{S}_2$ can co-exist independently.

In \mathbb{R}^3 alternatives are more:

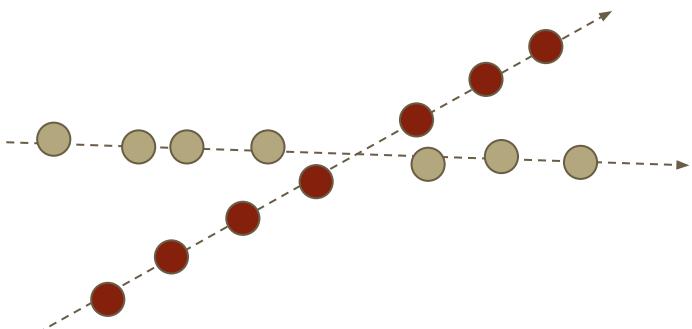
- 2 subspaces: 1 line & 1 plane
- 2 subspaces: 2 lines
- 3 subspaces: 3 lines

and so on...

If data is clean, i.e. free of noise, can you tell which data points $\mathbf{c}_i^{\mathbf{M}}$ belong to the same subspace \mathbb{S}_j ?

EX: Can you put points on their line? Octave for a change

Given \mathbf{M} : points in \mathbf{M} come from 2 *lines* (i.e. subspaces)



```
octave:61> V1 = rand(3,1) * rand(1,4);
octave:62> V2 = rand(3,1) * rand(1,3);
octave:63> M = [V1 V2];
octave:64> [U S V] = svd(M);
octave:65> rank(M)
ans = 2
octave:66> V2 = V(:,1:2);
octave:67> SIM = V2 * V2'
SIM =
```

| | | | | | | |
|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| 4.6556e-02 | 1.2106e-01 | 1.2424e-01 | 1.1957e-01 | -8.7600e-17 | -3.6306e-17 | -1.4280e-16 |
| 1.2106e-01 | 3.1481e-01 | 3.2307e-01 | 3.1092e-01 | 7.8905e-18 | 7.7383e-18 | -7.8028e-19 |
| 1.2424e-01 | 3.2307e-01 | 3.3156e-01 | 3.1908e-01 | -5.7803e-17 | -1.2327e-17 | -9.3168e-17 |
| 1.1957e-01 | 3.1092e-01 | 3.1908e-01 | 3.0708e-01 | -4.4055e-18 | 5.3514e-18 | -2.0896e-17 |
| -8.7600e-17 | 7.8905e-18 | -5.7803e-17 | -4.4055e-18 | 2.8729e-01 | 1.2378e-01 | 4.3524e-01 |
| -3.6306e-17 | 7.7383e-18 | -1.2327e-17 | 5.3514e-18 | 1.2378e-01 | 5.3327e-02 | 1.8752e-01 |
| -1.4280e-16 | -7.8028e-19 | -9.3168e-17 | -2.0896e-17 | 4.3524e-01 | 1.8752e-01 | 6.5938e-01 |

```
octave:68> SIM > 5*eps
ans =
```

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 |

let $r = \text{rank}(\mathbf{M})$

$\mathbf{M}_{dxn} = \mathbf{U}_{dxr} \boldsymbol{\Sigma}_{rxr} \mathbf{V}_{rxn}^T$

$\mathbf{S}_\mathbf{M} = \mathbf{V}_{nxr} \mathbf{V}_{rxn}^T$

Why does SIM work: Some linear algebra

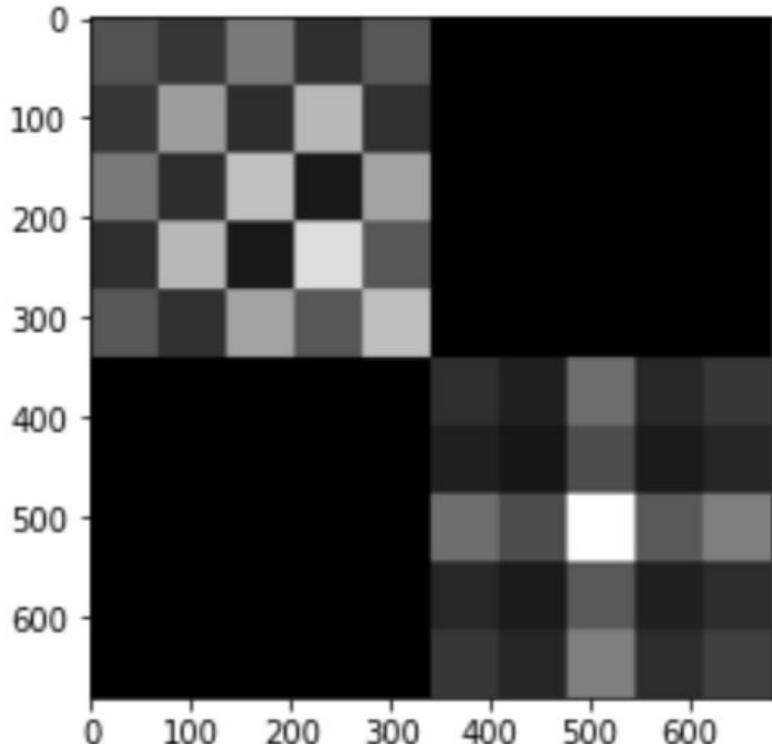
Theorem 3

Let $\mathbf{W} = [w_1 \cdots w_N] \in \mathbb{R}^{D \times N}$ be a matrix whose columns are drawn from a union of subspaces \mathcal{U} as in Assumptions 1. Let the skinny SVD of \mathbf{W} be given by $\mathbf{W} = U\Sigma V^T$, and define $Q = \text{abs}(VV^T)$. Then, $\Xi_{\mathbf{W}} = Q^{d_{max}}$ is a similarity matrix for \mathbf{W} , where $d_{max} = \max \{d_i\}_{i=1}^M$.

Theorem 3 is from [this paper](#) with the [assumption](#).

Revisit: Similarity Matrix

- What if you treat the Similarity matrix as data?
- Where each column in the similarity matrix corresponds to the original data
- Then you cluster columns of the similarity matrix?
- What about the basis for distinct blocks?



Spectral Clustering: It's a kind of magic Or is it?

- Start with the data matrix \mathbf{M}_{dxn} and form a similarity matrix \mathbf{S}_{nxn} anyway you find proper for the problem.
- Calculate the Laplacian matrix \mathbf{L}_{nxn} for \mathbf{S}_{nxn}
- Find the eigenvalues λ_i and corresponding eigenvectors \mathbf{x}_i .
- Sort eigenvalues in ascending order so that $0 = \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \dots \lambda_n$.
- If you want to divide the data into k clusters, select the first $k - 1$ eigenvectors that correspond to the smallest $k - 1$ eigenvectors and form a new matrix \mathbf{R} so that, eigenvectors \mathbf{x}_i^T form the rows of \mathbf{R} .

$$\mathbf{R}_{(k-1)xn} = \begin{bmatrix} - & \mathbf{x}_2^T & - \\ - & \dots & - \\ - & \mathbf{x}_k & - \end{bmatrix}$$

- Cluster columns of \mathbf{R} using any convenient method such as k -means.
- At the end you will have a vector with values that are same for data points that belong to the same cluster.

Trial and error: more iterations

There is noise

No noise models advanced filters etc in play

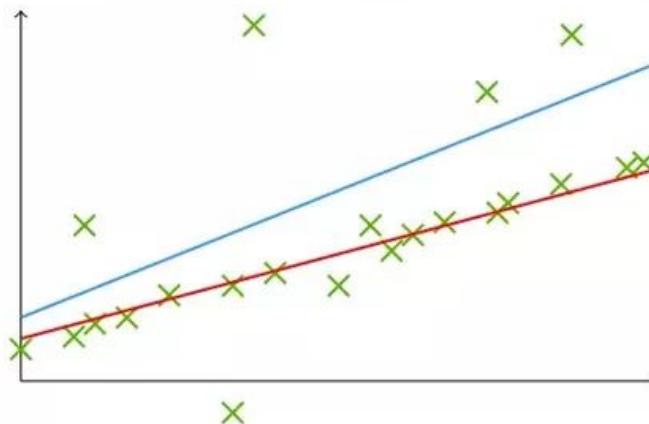
I need a quick, practical approach to find structures in data?

Outliers: Recall ℓ_1 vs ℓ_2

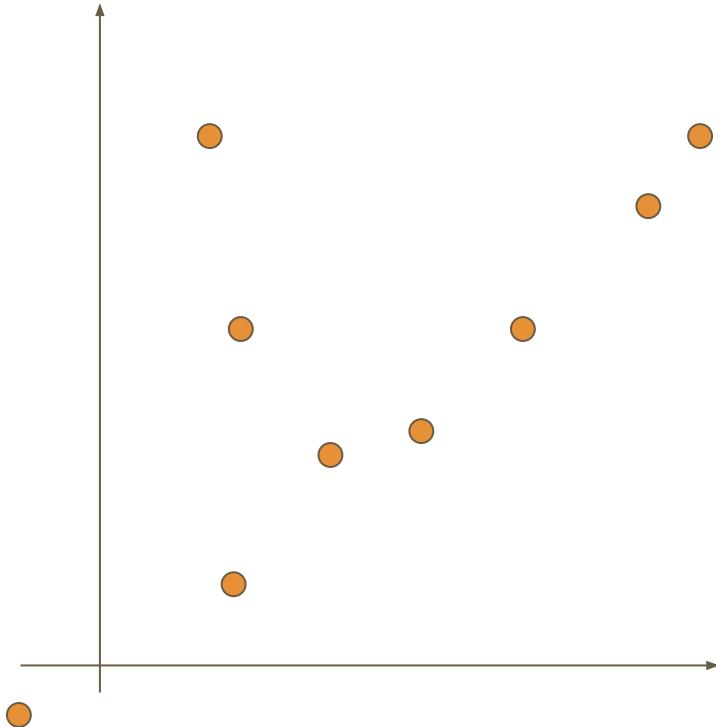
Given : A set of points in 2-dimension

Goal : Find a line to fit those points

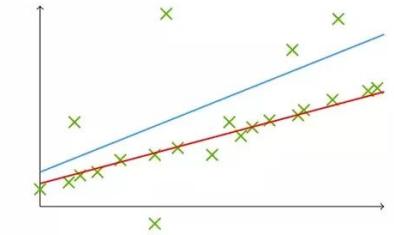
Output : ℓ_2 minimizer line, ℓ_1 minimizer line



Outliers: An iterative perspective

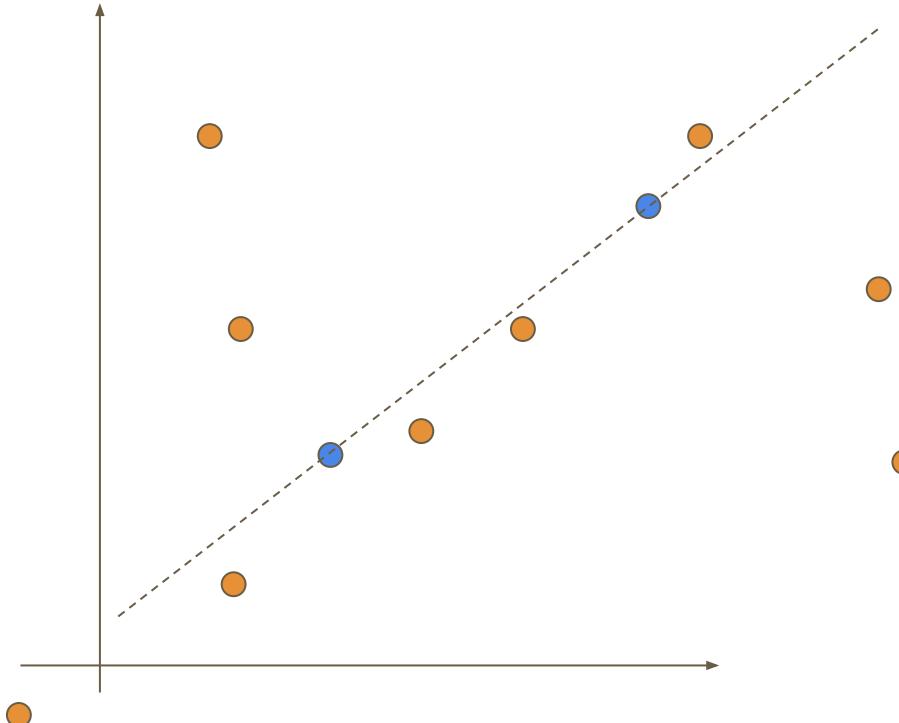


Given : A set of **points** in 2-dimension
Goal : Find a line to fit those points
Output : ℓ_2 minimizer **line**, ℓ_1 minimizer **line**



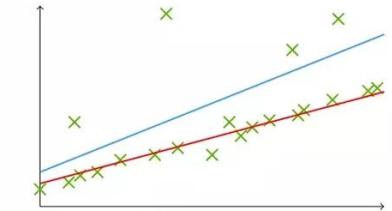
1. Sample 2 points randomly

Outliers: An iterative perspective

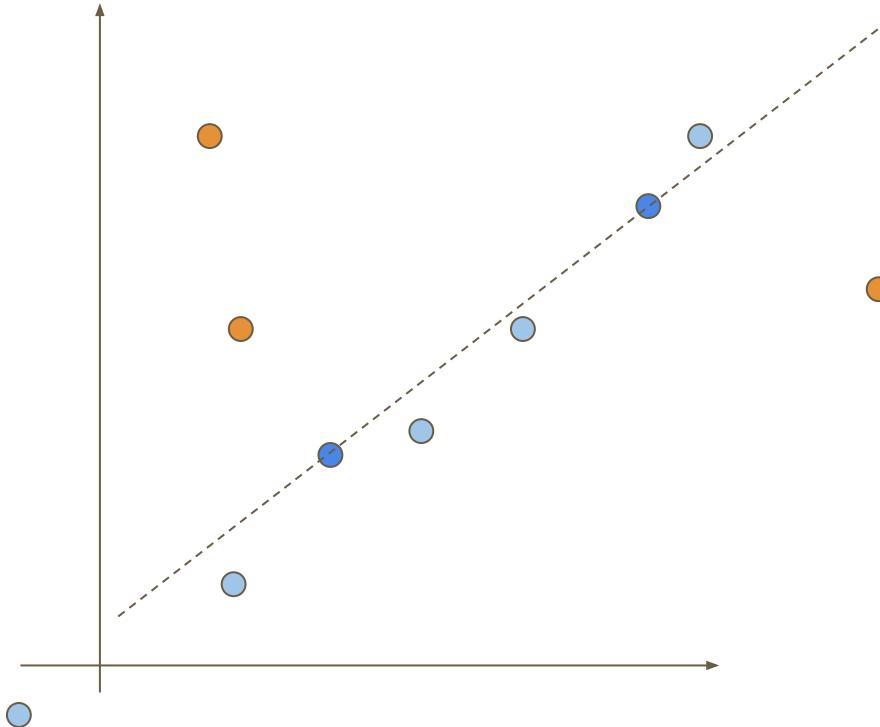


1. Sample 2 points randomly
2. Find the line

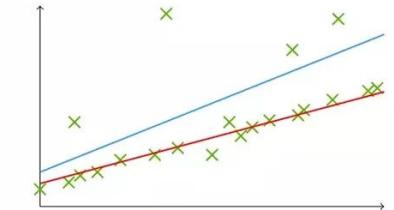
Given : A set of **points** in 2-dimension
Goal : Find a line to fit those points
Output : ℓ_2 minimizer **line**, ℓ_1 minimizer **line**



Outliers: An iterative perspective



Given : A set of **points** in 2-dimension
Goal : Find a line to fit those points
Output : ℓ_2 minimizer **line**, ℓ_1 minimizer **line**



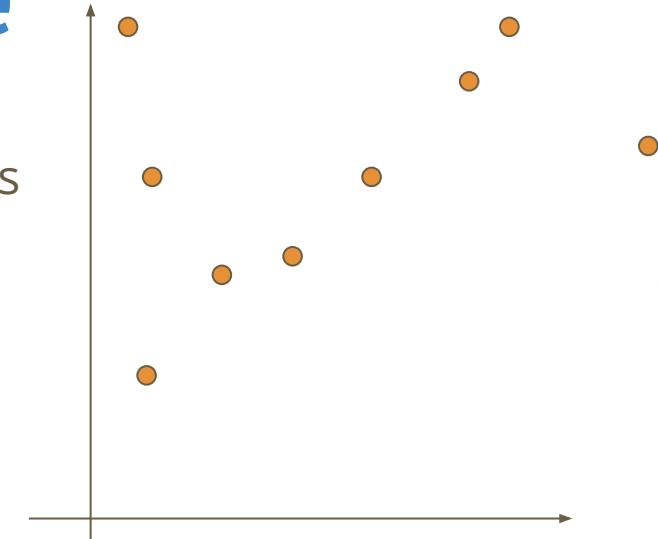
1. Sample 2 points randomly
2. Find the line
3. Find the **good** points

Repeat N times or until?

RANSAC: An iterative perspective

Select a model and determine number of data points
needed for the model: s

1. **Sample:** Randomly select s data points
2. **Fit:** Find a model using s data points
3. **Test:** all data points to find the **good** ones → *inliers*

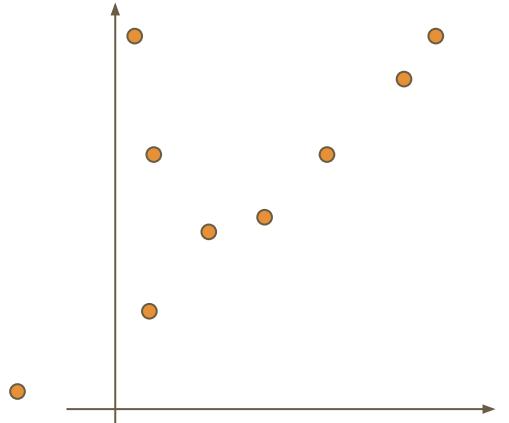


Repeat N times at the end choose the **best** model

RANSAC: An iterative perspective

$$N = \frac{\log(1-p)}{\log(1-(1-e)^s)}$$

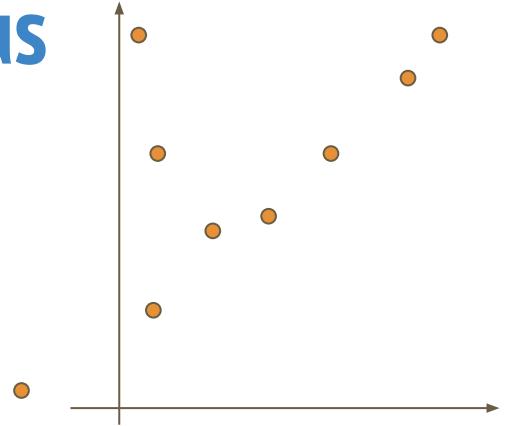
1. N : Number of trials / samplings
2. s : Number of data points needed to build a candidate model
3. p : desired probability of a good model
 - o 99.9% chance that I will end up with the best model: $p = 0.999$
4. e : probability that a point is an outlier
 - o $e = 0.15$: 15% of my data is contaminated with outliers
 - o What if you over- or under-estimate e ?



RANSAC: RANdom SAmple Consensus

Pros:

- Robust to outliers
- Easy to implement
- Works for a wide range of models,
i.e. manageable for $s = 1 \rightarrow 10$
- e to be under %50!
- A simple yet clear [intro to RANSAC](#)

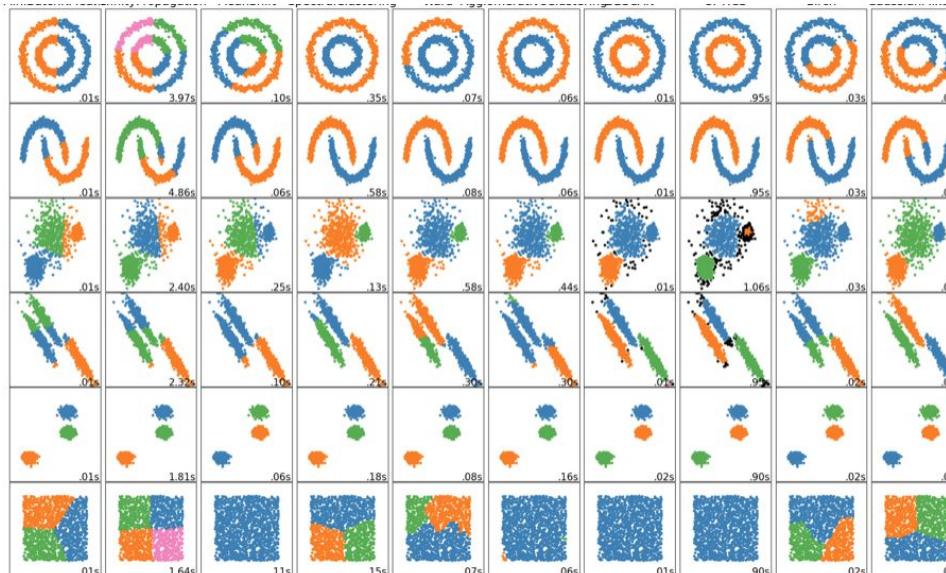


Cons:

- N explodes with s
- Not designed to work with
multiple fits / hypothesis

Scikit learn: Clustering tools

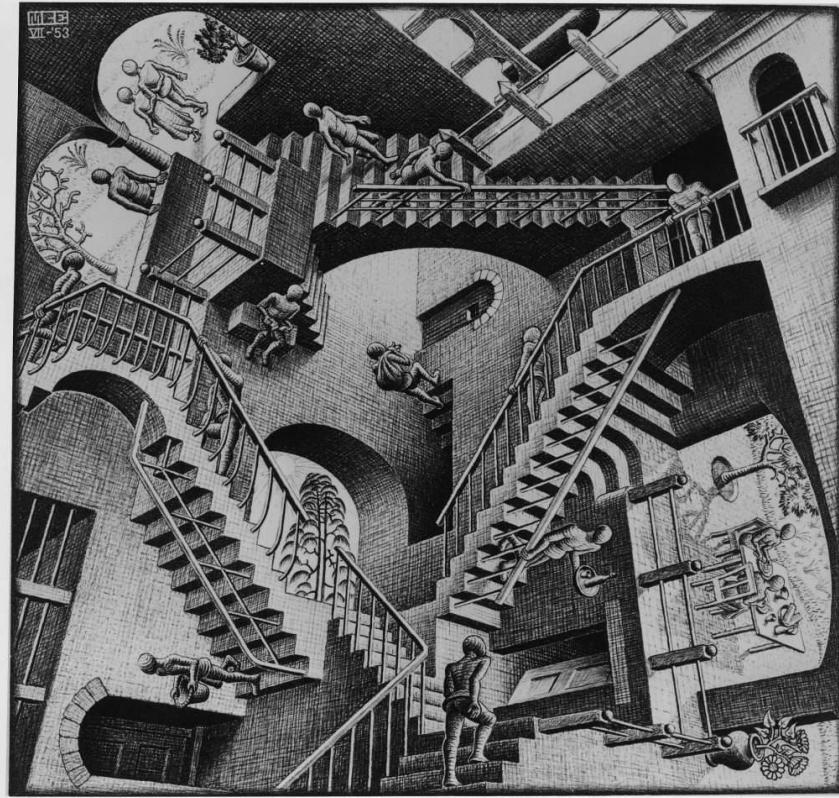
Check out: <https://scikit-learn.org/stable/modules/clustering.html>



ME 536

Week 9-10: Images as Tensors

Pixels: oh pixels

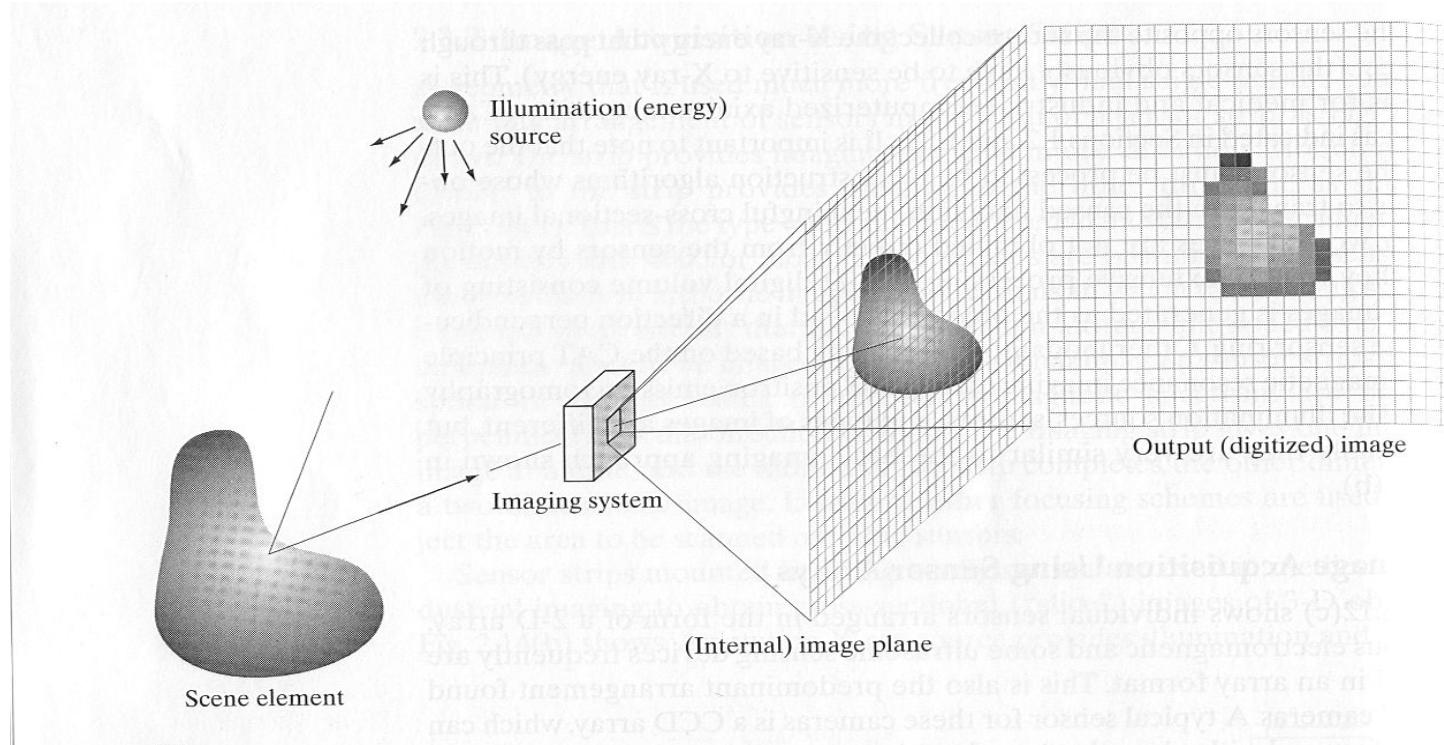


Neighbor pixels are mostly related

Far away pixels are sometimes / somehow related

- Physically related: a *path*
- Contextually related: common *meaning*

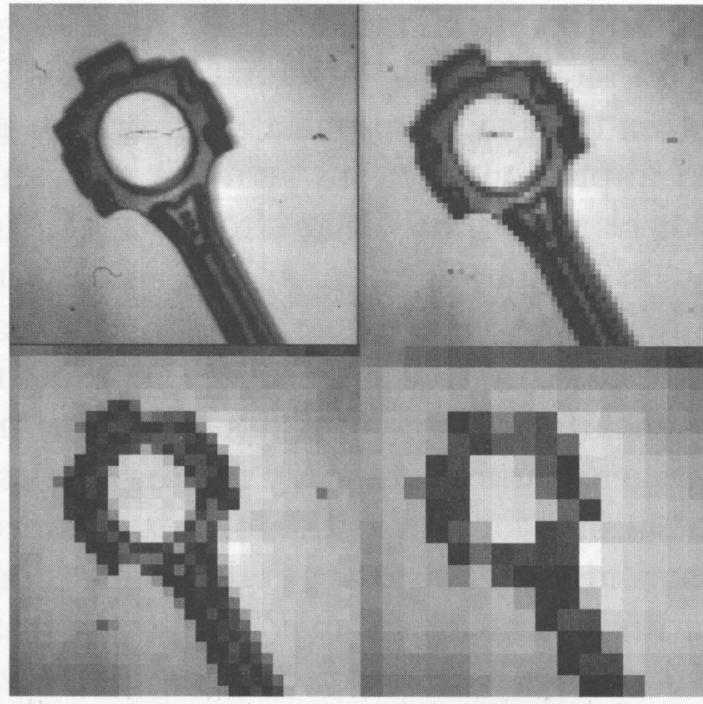
Digital Image Acquisition Process



Quantization
Sampling →

Gray-Level Resolution
Spatial Resolution

Example: Quantization vs Sampling



Quantization → from 32 to 4 gray levels all with 256x256 sampling
Sampling → 256x256 to 16x16 all with 128 gray levels

Pixels in Images: RGB implied...

Pixels in tensor I:

$$I [i, j, k]$$

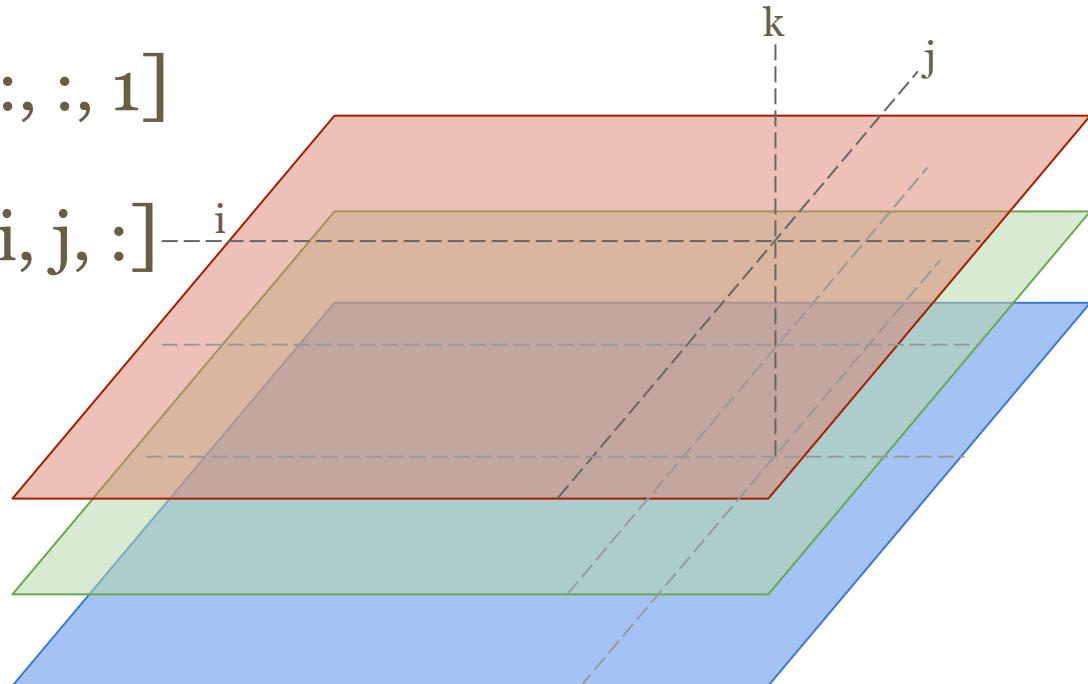
All pixels on Green plane:

$$I [:, :, 1]$$

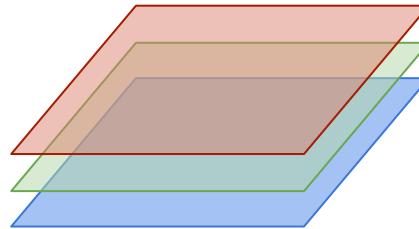
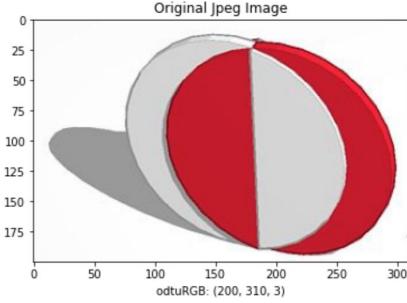
RGV values of pixel (i,j):

$$I [i, j, :]$$

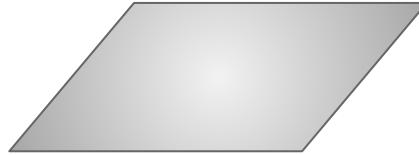
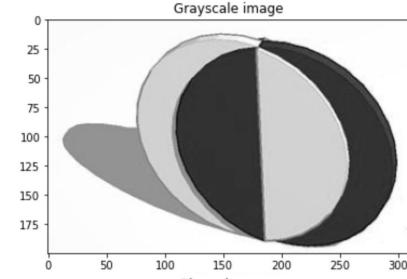
Any plane -> grayscale



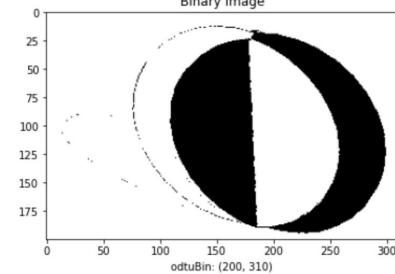
Color Depth in Images: Color, Gray, Binary



- Color images:
 - has layers such as RGB
 - Typical values: [0-255] or [0.0 - 1.0]



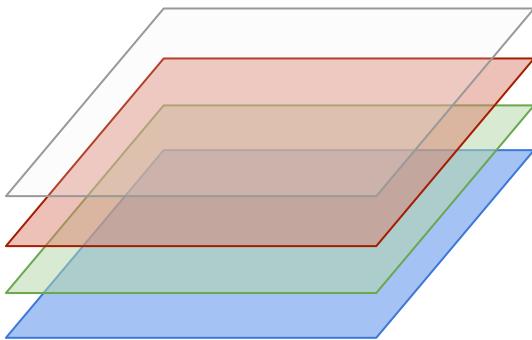
- Gray-scale images:
 - has 1 layer
 - Typical values: [0-255] or [0.0 - 1.0]



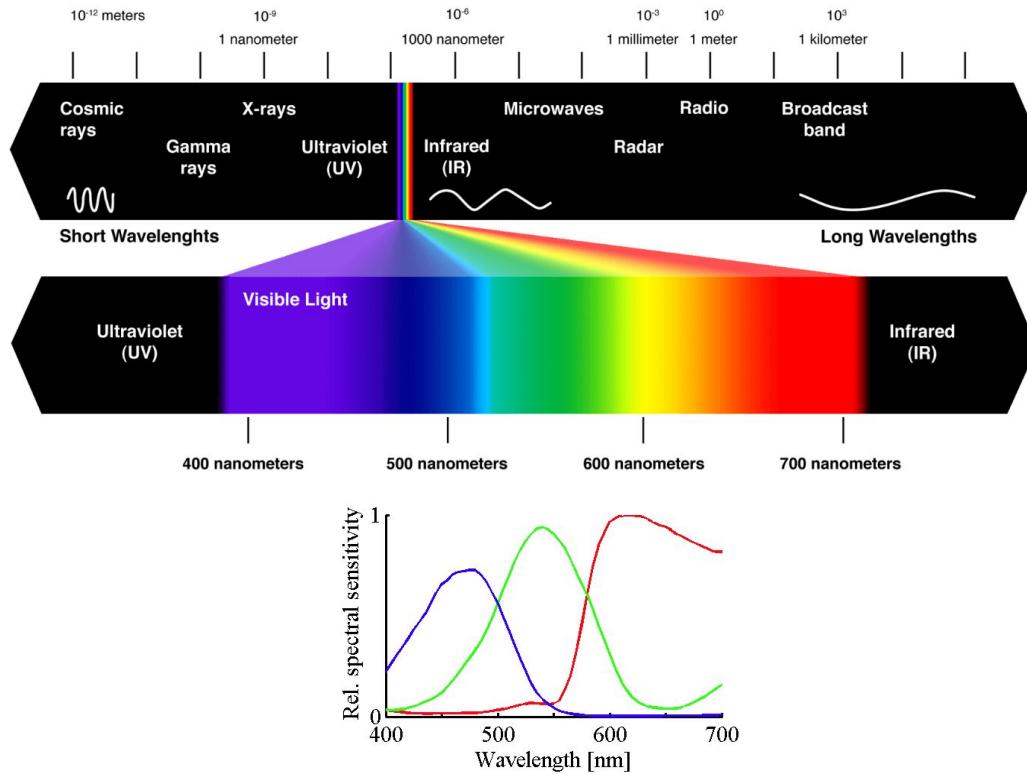
- Binary images
 - just like Gray scale has 1 layer
 - Values are 0 or 1

Color Depth in Images: Transparency, Hyperspectral

- Color images with transparency:
 - Typically has an additional layer

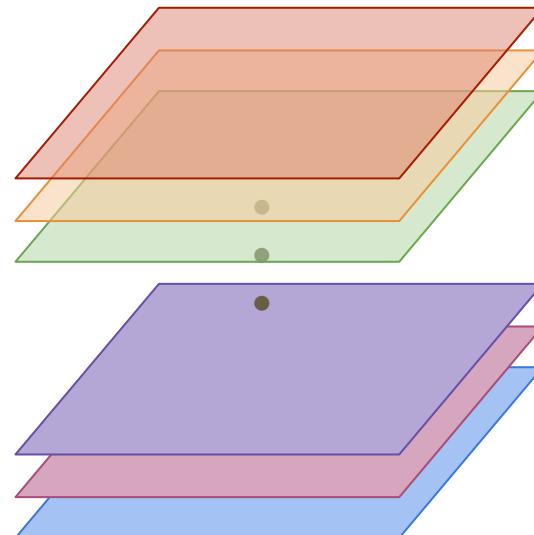
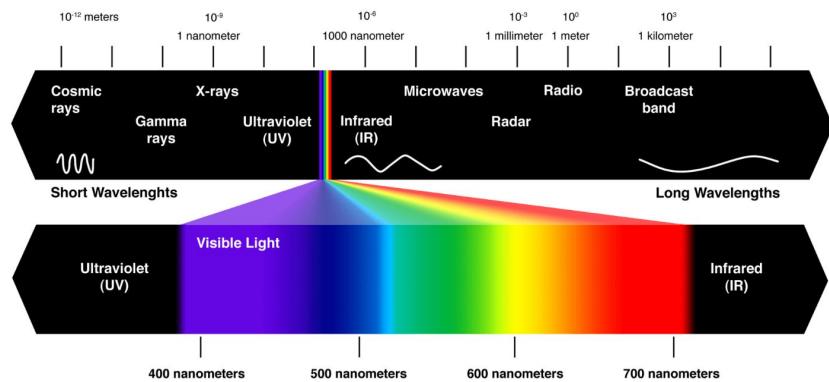


RGB response: of some/typical daylight camera*



Color Depth in Images: Transparency, Hyperspectral

- Multispectral / Hyperspectral images:
 - has several layer corresponding to different wavelengths
 - Near IR (~700nm - 900nm)
 - Short-Wave IR (SWIR: ~1.7um-2.5um)
 - Mid-Wave IR (MWIR: ~2.7um-5.3um)
 - Long-Wave IR (LWIR: ~ 8um-12um)

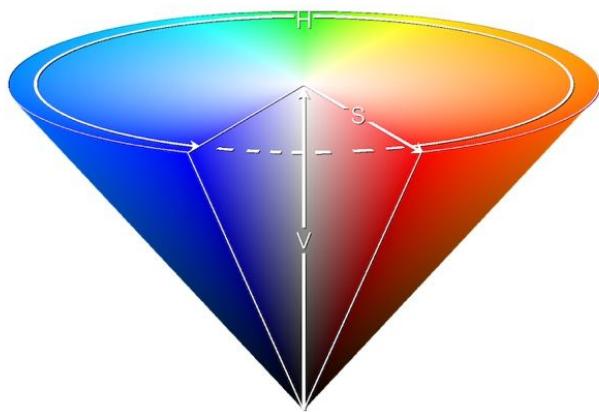


Color Spaces: RGB → Most famous but ?

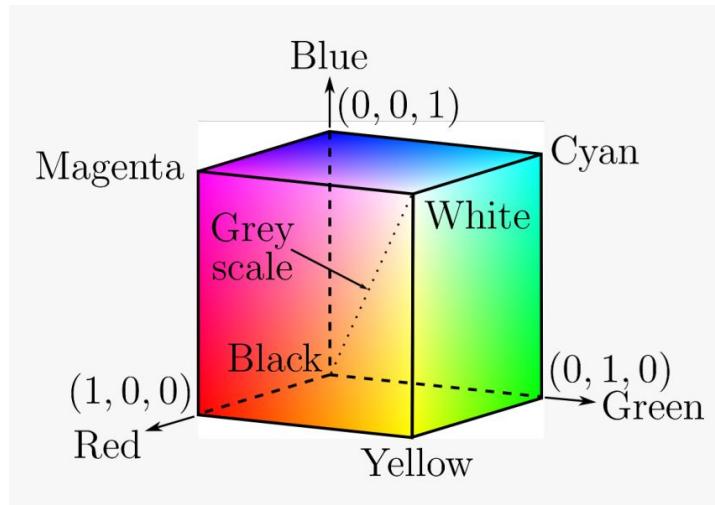
- a.k.a. Color Formats
- RGB is one of many: CMYK, CIE, HSV, ...
- Different uses, different formats
- HSV / (a.k.a. *HSB*) is more ?
- How about you come up with one? Other do so*

Color Spaces: RGB vs HSV

When lighting conditions change less play on HSV



HSB Cone. Image courtesy of Wikimedia.



Python libraries available

Scikit-Image

OpenCV

PIL

...

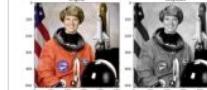
Scikit-Image

Check out:

Examples page

for more...

Manipulating exposure and color channels



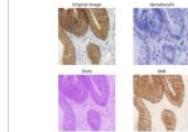
RGB to grayscale



RGB to HSV



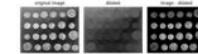
Histogram matching



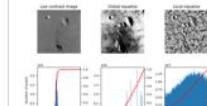
Immunohistochemical staining colors separation



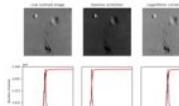
Adapting gray-scale filters to RGB images



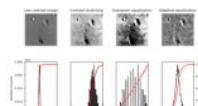
Filtering regional maxima



Local Histogram Equalization



Gamma and log contrast adjustment



Histogram Equalization

Format change:

- RGB → HSV
 - Linear transformation
 - not unique different transformation in literature
- RGB → Gray
 - Generally a linear transformation
 - not unique different transformation in literature
 - $\text{GrayValue} = 0.2125 \text{ R} + 0.7154 \text{ G} + 0.0721 \text{ B}$ ¹
 - Used by CRT phosphors as they better represent human perception of red, green and blue than equal weights
- Gray → Binary
 - Tricky
 - Depends on the objective
 - Manual inspection is useful
 - Automatic thresholding methods exist such as OTSU² and many more³

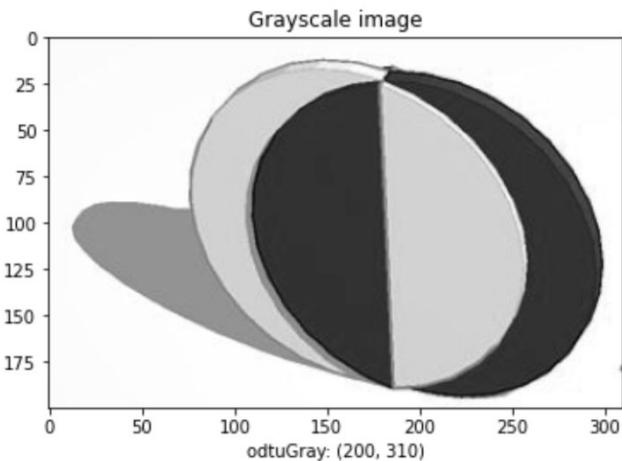
1- https://scikit-image.org/docs/stable/auto_examples/color_exposure/plot_rgb_to_gray.html#id2

2- https://en.wikipedia.org/wiki/Otsu%27s_method

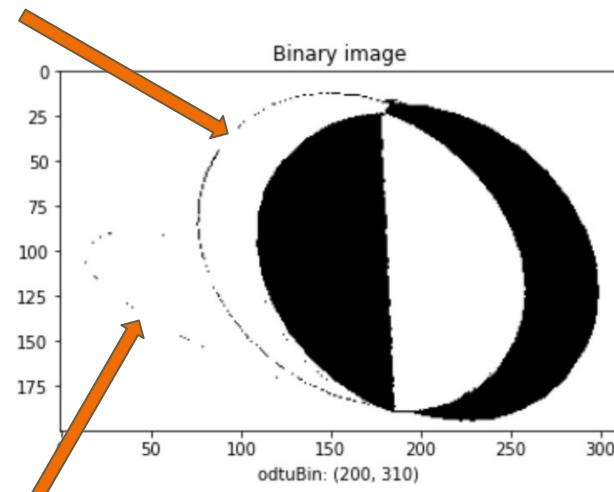
3- https://scikit-image.org/docs/dev/api/skimage.filters.html#skimage.filters.threshold_isodata

Binary images: Thresholding

- Proper thresholding improves detection performance

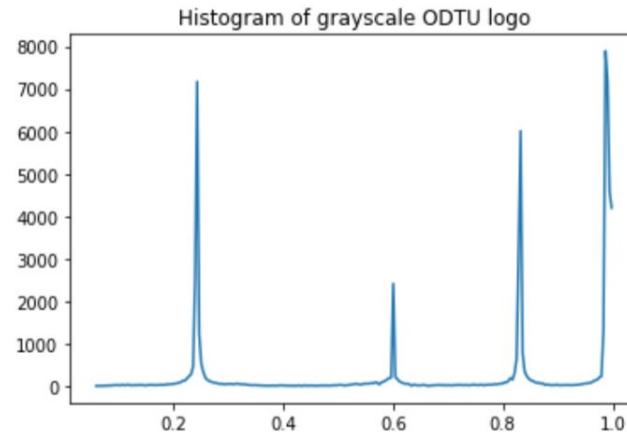
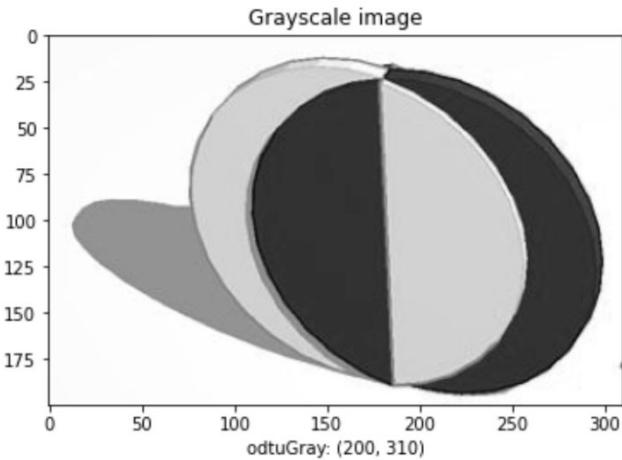


?
→

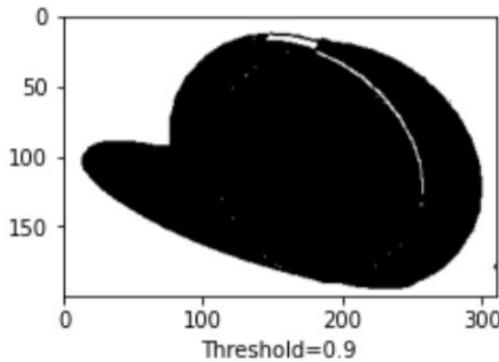
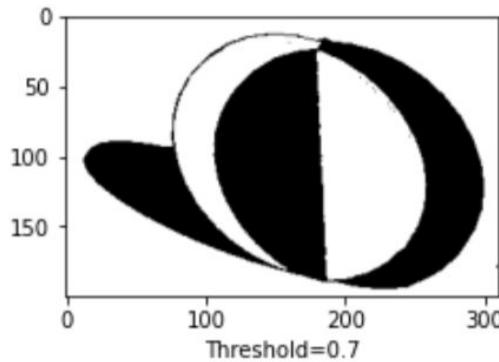
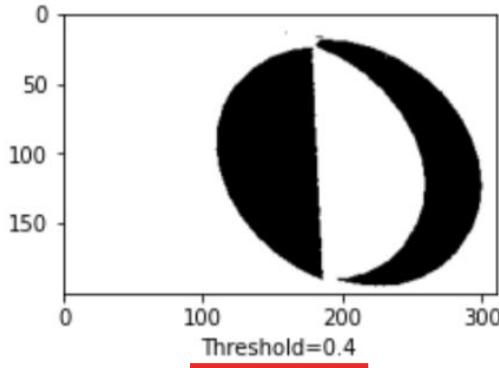
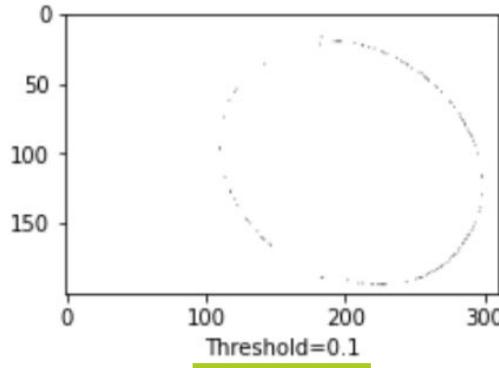
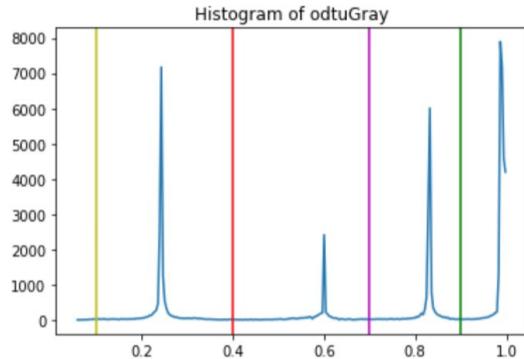
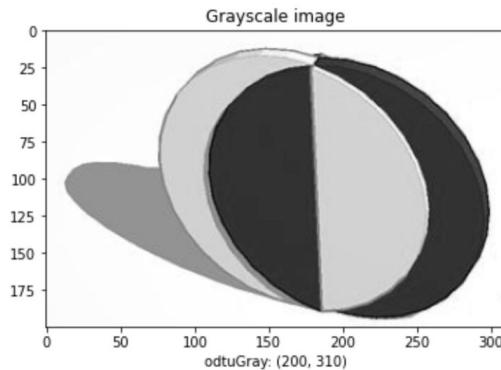


Histograms:

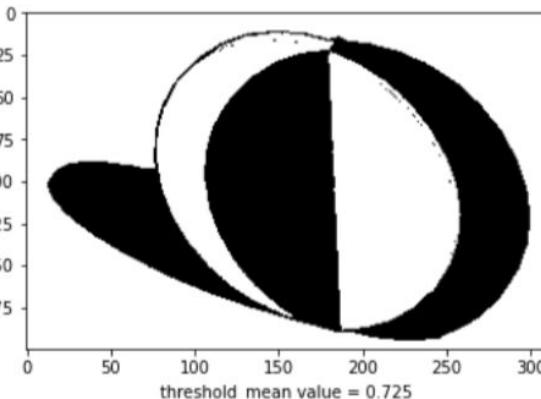
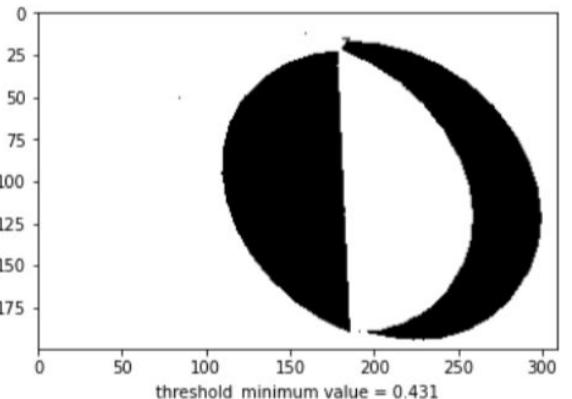
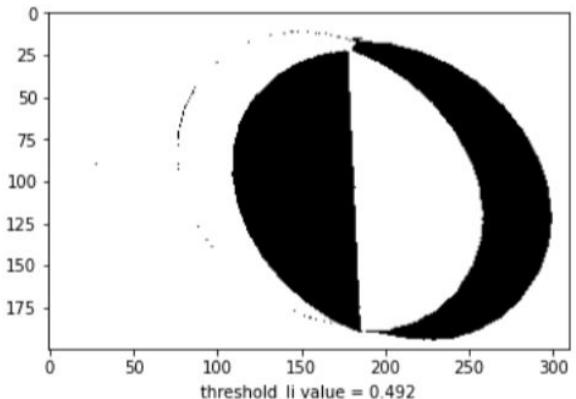
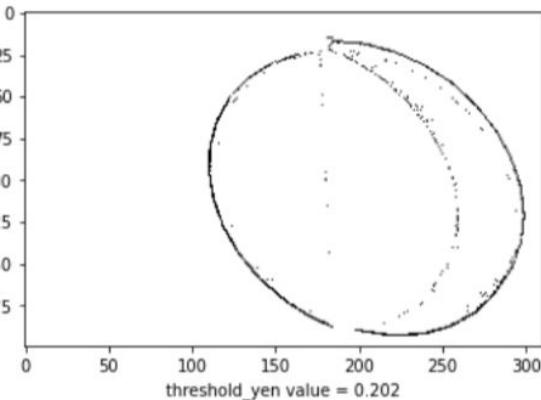
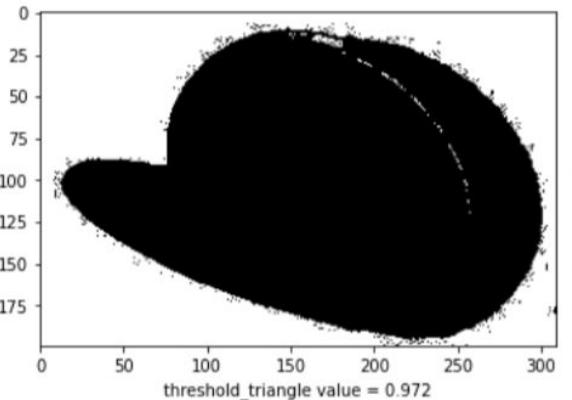
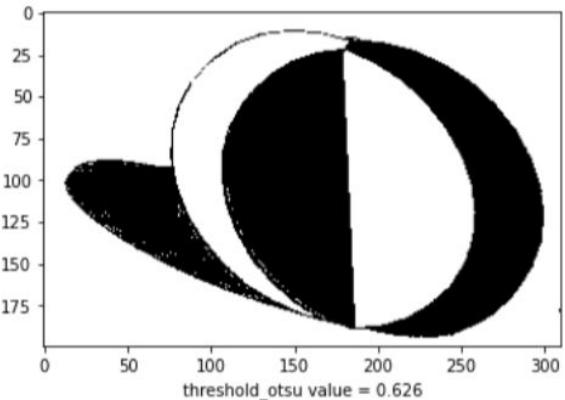
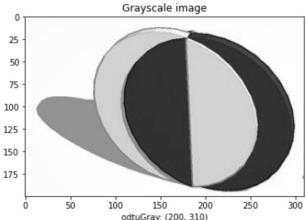
A way of analyzing distribution of data



Histograms: Manual Thresholding a delicate balance



Histograms: Automatic Thresholding



Histograms: Proper Choice

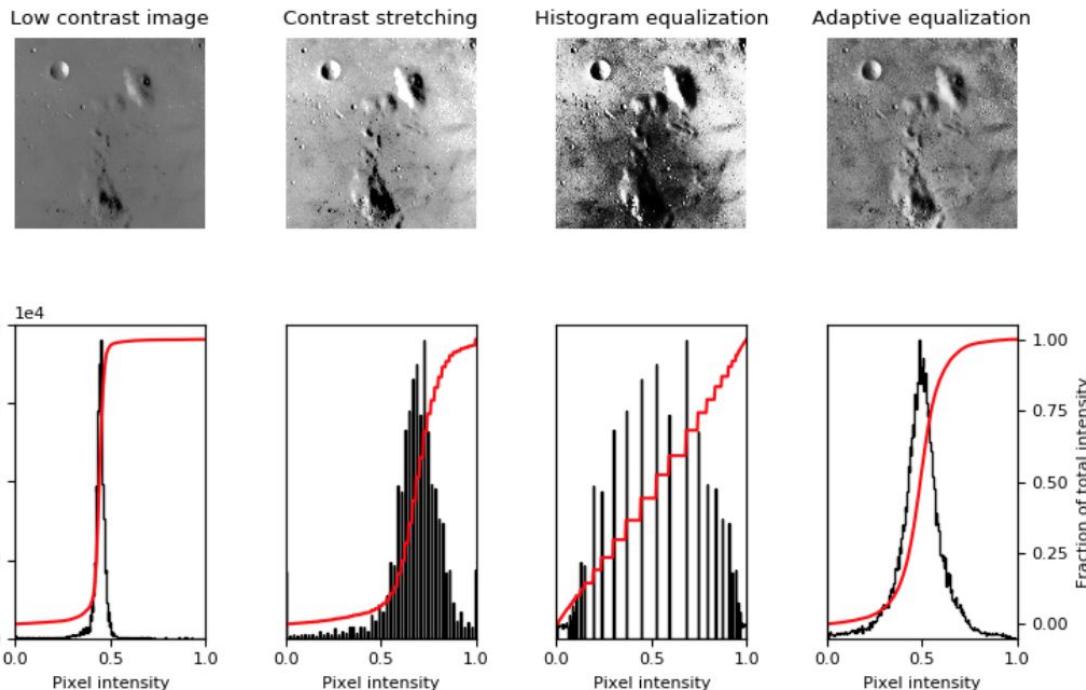
Doesn't it sound like a clustering problem in a way? Where there are 2 clusters...



How about a **thresholding method** that **understands what's in the image?**

Histogram: Beyond Thresholding

Histogram enhancement



Histogram: Beyond Thresholding

Histogram matching:

- when groups of images are analyzed
- extends to histogram equalization

Source



Reference



Matched

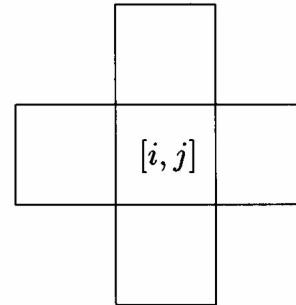


Basic Definitions: used in dealing with images

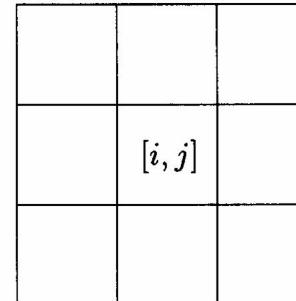
- Neighbors
- Path
- Foreground, Background
- Connectivity
- Connected Components
- Boundary, Interior, Surrounds

Neighbors: 4 or 8

4-neighbors $[i + 1, j]$, $[i - 1, j]$, $[i, j + 1]$, $[i, j - 1]$



8-neighbors $[i + 1, j + 1]$, $[i + 1, j - 1]$, $[i - 1, j + 1]$, $[i - 1, j - 1]$ plus all of the 4-neighbors



Path:

- A path from the pixel P_o to pixel P_n is a sequence of pixel indices $(i_o, j_o), (i_1, j_1), (i_2, j_2), \dots, (i_n, j_n)$ such that the pixel at P_k is a neighbor of the pixel at P_{k+1} for all k with $0 \leq k \leq n-1$
- If a 4-connection is used the path is a 4-path, and it is an 8-path in case 8-connection is used

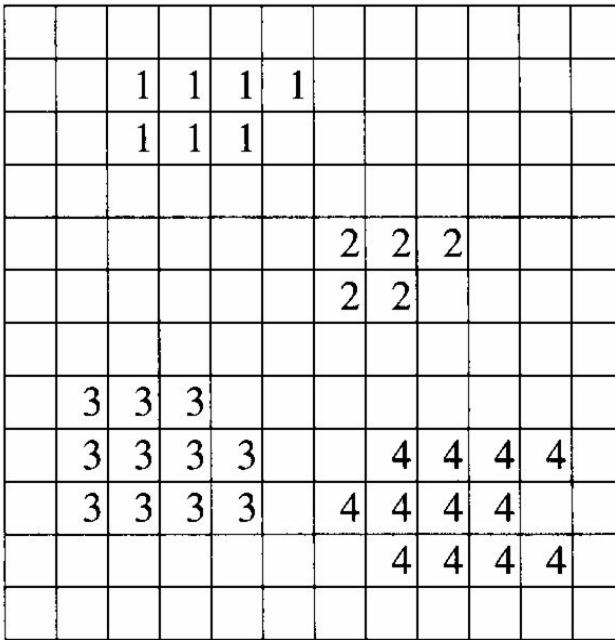
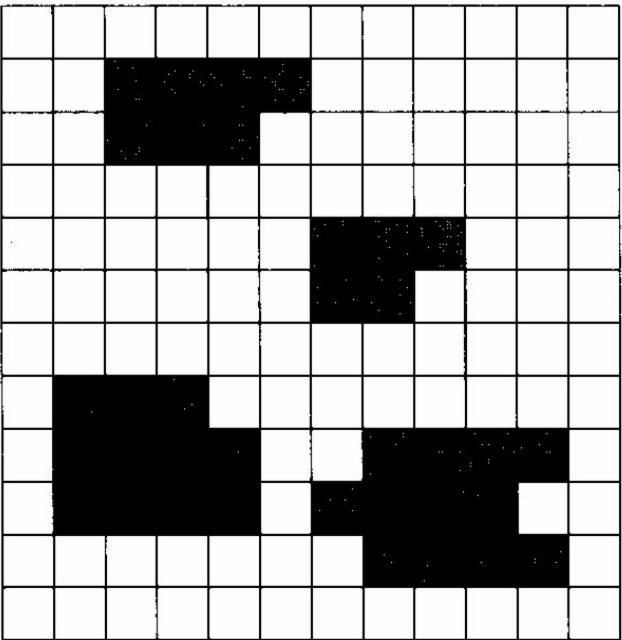
Foreground vs Background:

- Set of all 1 pixels in an image is called the foreground and denoted by S .
- Set of all 0 pixels in an image is called the *background* and denoted by \bar{S} .

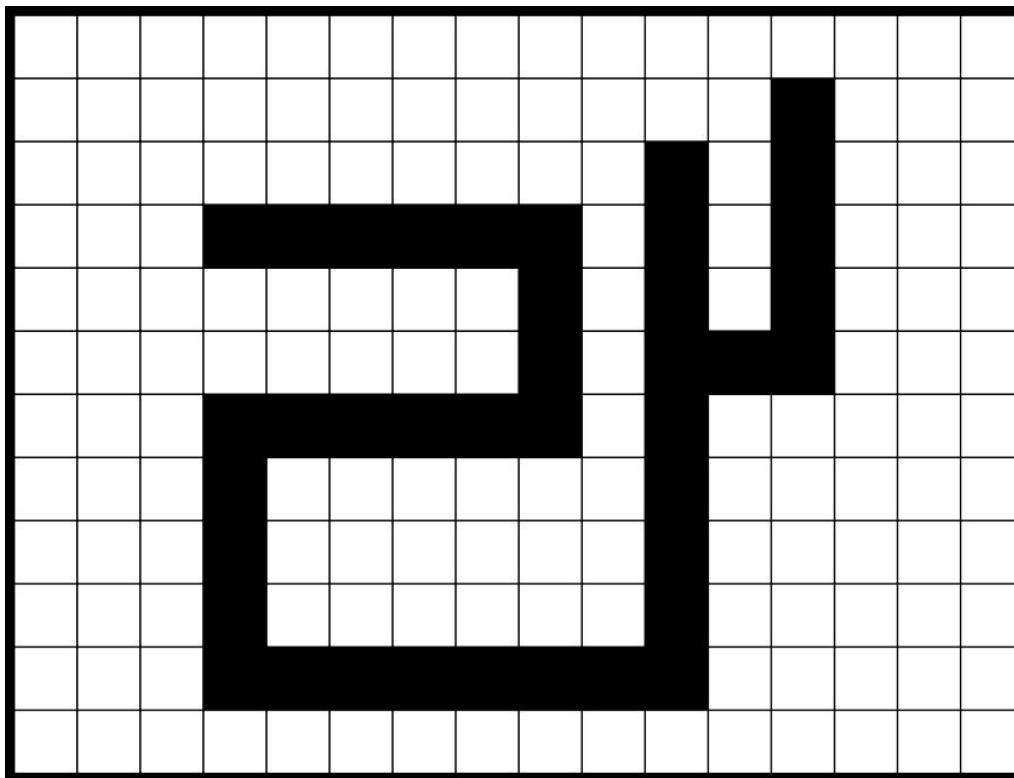
Connectivity:

- A pixel p in S is said to be connected to pixel q in S if there is a path from p to q .
- Connectivity is:
 - Reflexive: p is connected to p
 - Commutative: if p is connected to q , then q is connected to p
 - Transitive: If p is connected to q and q is connected to r , then p is connected to r

Connected Component Labeling

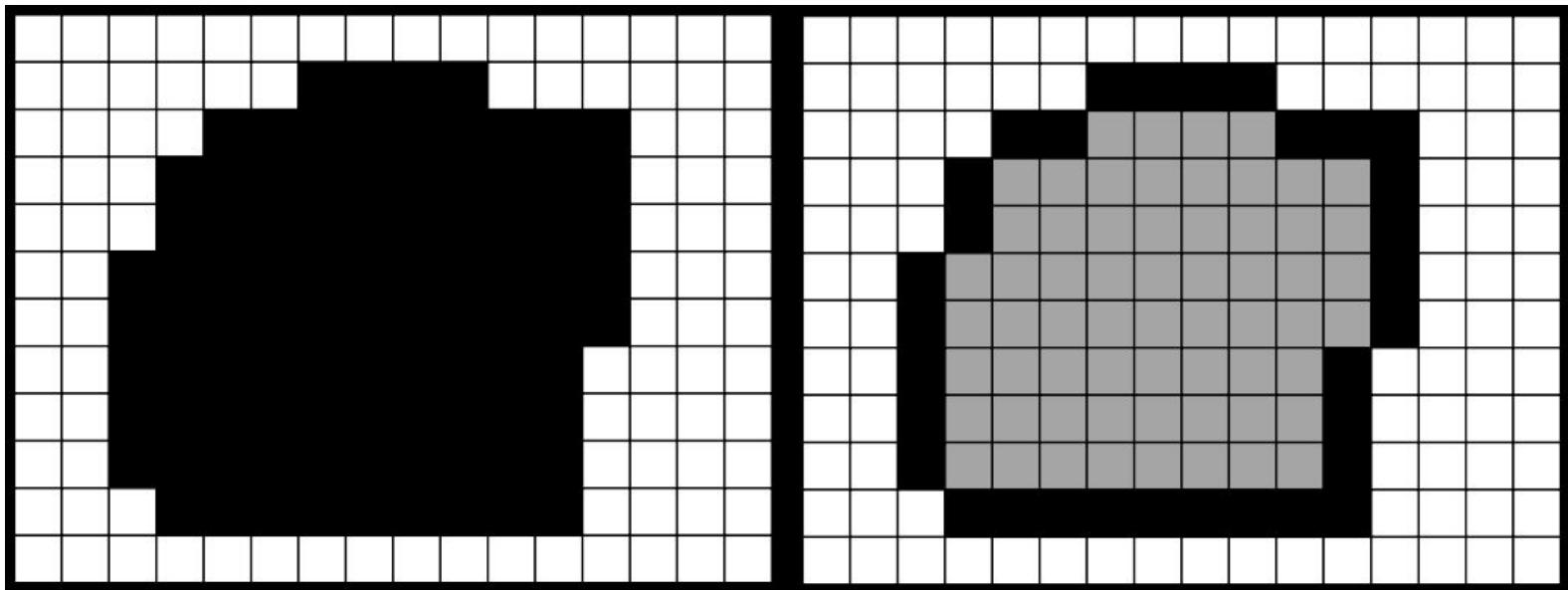


Connected Component Labeling



Boundary, Interior, Surrounds

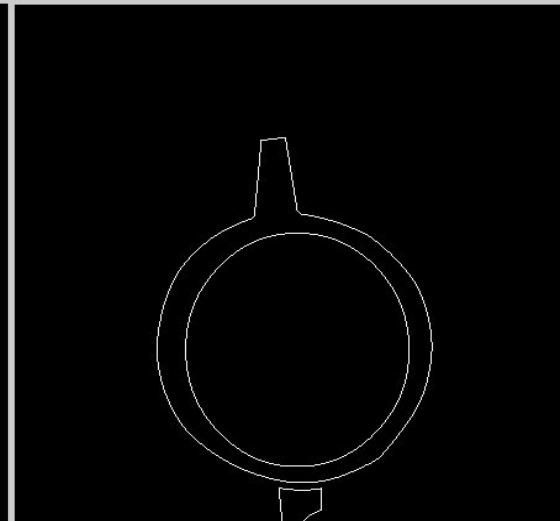
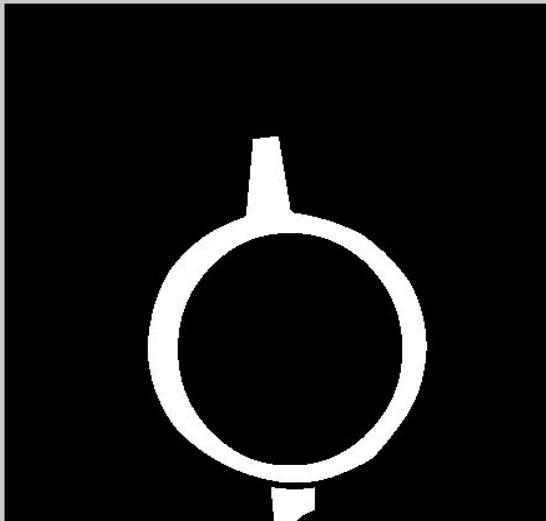
Neighborhood?



Properties

- Size → Area, perimeter
- Position
- Orientation
- Compactness
- Euler number
- Distance measures
- Medial Axis

Size: Area, Perimeter,...

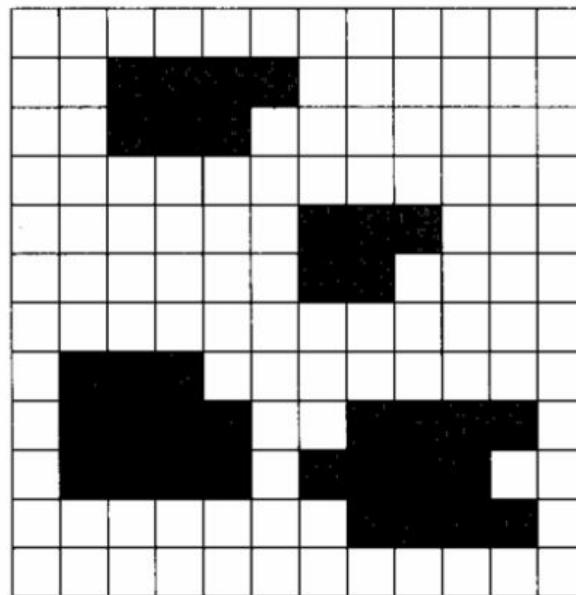


Area = 15132

Perimeter = 1414

Position: Center of mass of each component

Orientation: Axis that minimized second moment

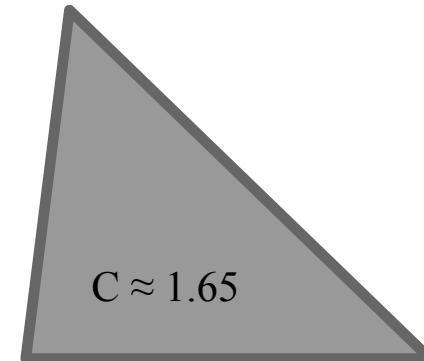
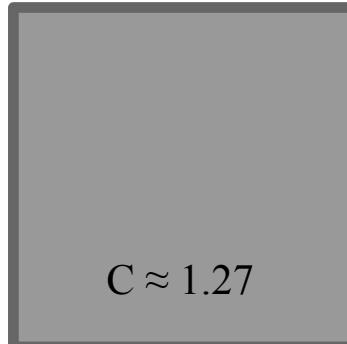
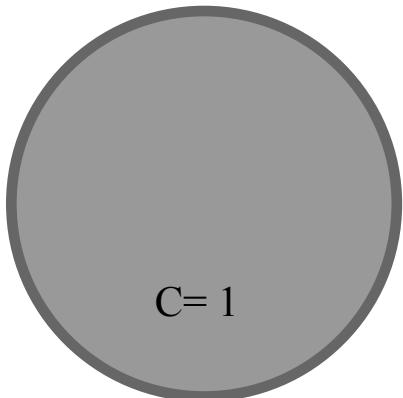


Compactness:

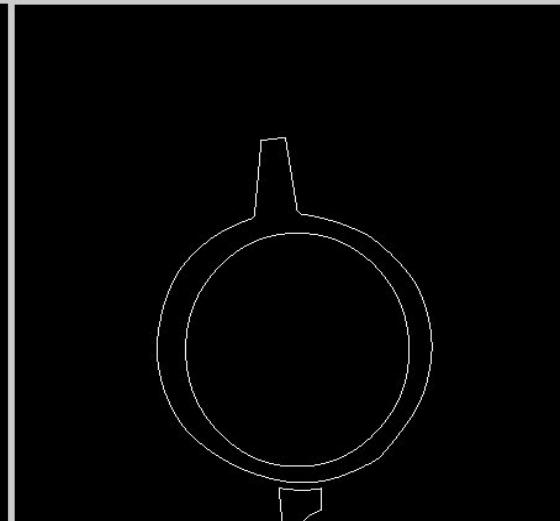
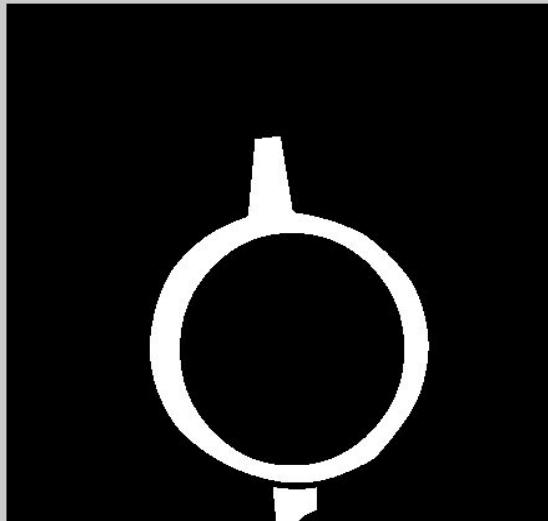
- Compactness of a continuous geometric figure is measured by the isoperimetric inequality:

$$C = (P^2/A) / 4\pi \geq 1$$

where P is the perimeter and A is the area of the object



Compactness: $C \approx 132$



Area = 15132

Perimeter = 1414

Euler Number:

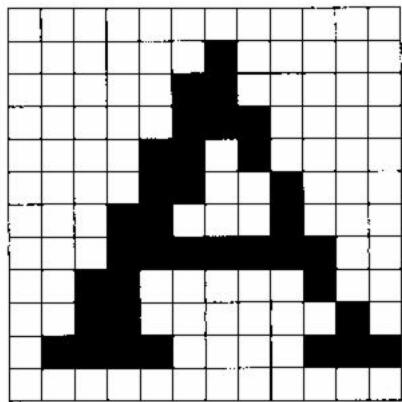
- Euler number is the number of components minus the number of holes on the image:

$$E = C - H$$

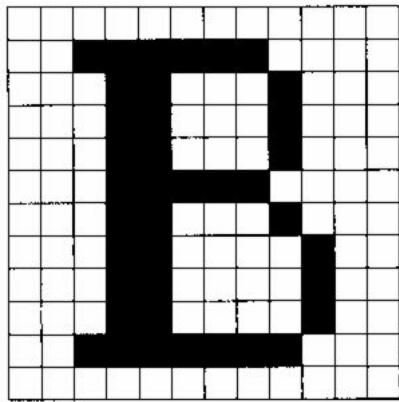
- Neighborhood definition is important for both the background and the foreground
- Invariant to translation, rotation, and scaling!

Euler Number: $E = C - H$

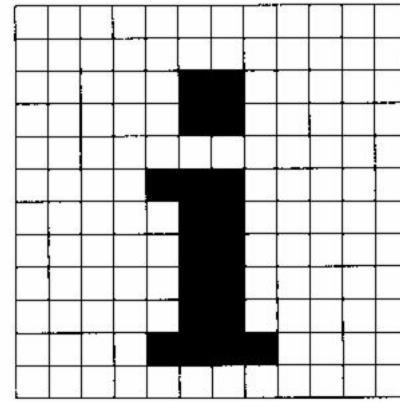
Background N4, foreground N8



$$\begin{aligned}C &=? \\H &=?\end{aligned}$$



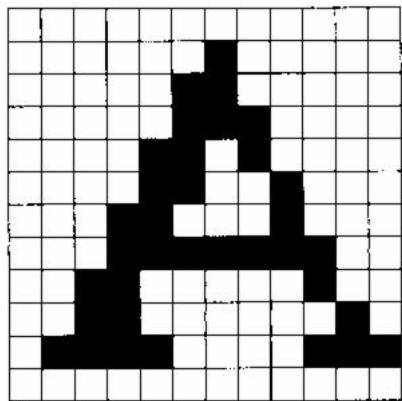
$$\begin{aligned}C &=? \\H &=?\end{aligned}$$



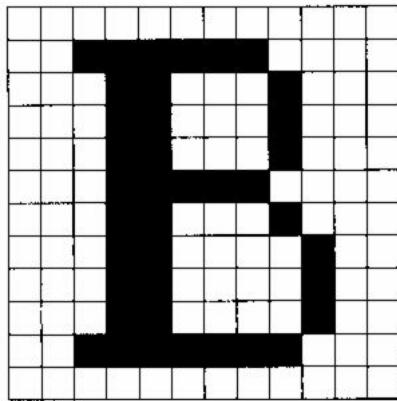
$$\begin{aligned}C &=? \\H &=?\end{aligned}$$

Euler Number: $E = C - H$

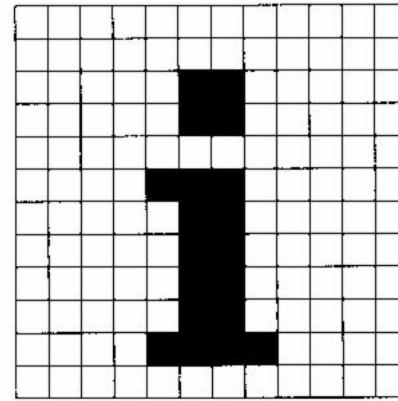
Background N4, foreground N8



$$\begin{aligned}C &= 1 \\H &= 1\end{aligned}$$



$$\begin{aligned}C &= 1 \\H &= 2\end{aligned}$$



$$\begin{aligned}C &= 2 \\H &= 0\end{aligned}$$

Distance:

Euclidean

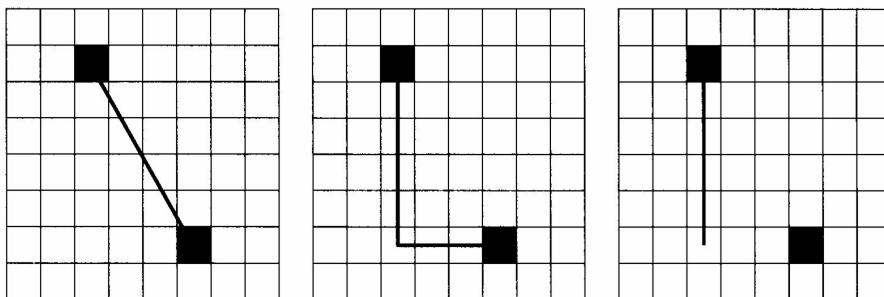
$$d_{\text{Euclidean}}([i_1, j_1], [i_2, j_2]) = \sqrt{(i_1 - i_2)^2 + (j_1 - j_2)^2}$$

City-block

$$d_{\text{city}} = |i_1 - i_2| + |j_1 - j_2|$$

Chessboard

$$d_{\text{chess}} = \max(|i_1 - i_2|, |j_1 - j_2|)$$



Euclidean distance:

| | | | | |
|---|------------|------------|---|------------|
| | | 3 | | |
| | $\sqrt{8}$ | $\sqrt{5}$ | 2 | $\sqrt{5}$ |
| | $\sqrt{5}$ | $\sqrt{2}$ | 1 | $\sqrt{2}$ |
| 3 | 2 | 1 | 0 | 1 |
| | $\sqrt{5}$ | $\sqrt{2}$ | 1 | $\sqrt{2}$ |
| | $\sqrt{8}$ | $\sqrt{5}$ | 2 | $\sqrt{5}$ |
| | | | 3 | |

City-block distance:

$$\begin{array}{ccccc}
 & & 3 & & \\
 & 3 & 2 & 3 & \\
 3 & 2 & 1 & 2 & 3 \\
 3 & 2 & 1 & 2 & 3 \\
 3 & 2 & 3 & & \\
 & & 3 & &
 \end{array}$$

Chessboard distance:

| | | | | | | |
|---|---|---|---|---|---|---|
| 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| 3 | 2 | 2 | 2 | 2 | 2 | 3 |
| 3 | 2 | 1 | 1 | 1 | 2 | 3 |
| 3 | 2 | 1 | 0 | 1 | 2 | 3 |
| 3 | 2 | 1 | 1 | 1 | 2 | 3 |
| 3 | 2 | 2 | 2 | 2 | 2 | 3 |
| 3 | 3 | 3 | 3 | 3 | 3 | 3 |

Medial Axis

- If the distance $d((i,j), \bar{S})$ for the pixel (i,j) in S to \bar{S} is locally maximum i.e.

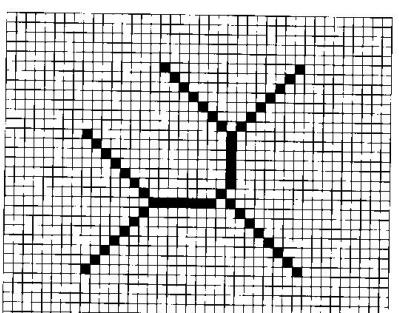
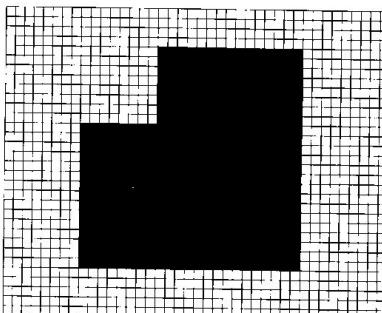
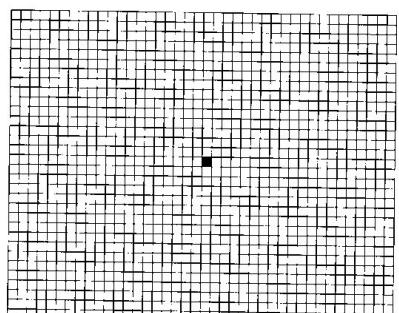
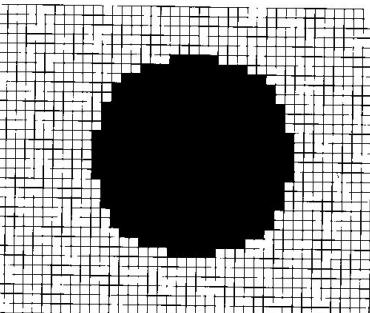
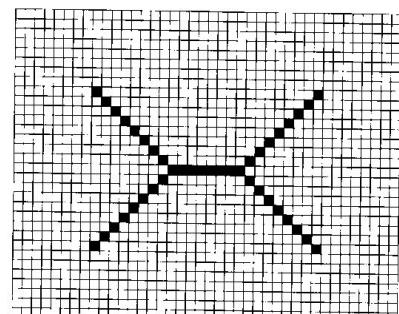
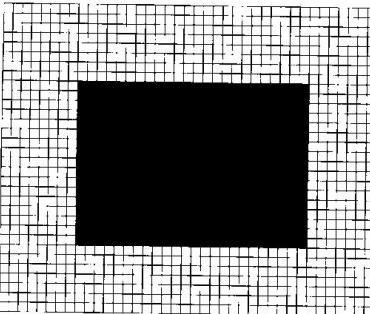
$$d((i,j), \bar{S}) \geq d((u,v), \bar{S})$$

for all pixels (u,v) in the neighborhood of (i,j) , then pixel (i,j) is on the medial axis.

(Recall that: \bar{S} is the background and S is the foreground)

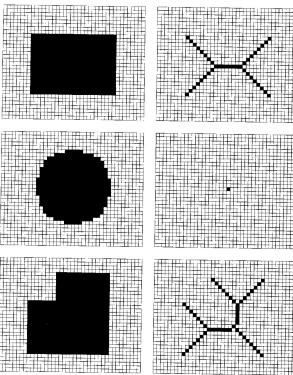
- Medial axis has been used for compact representation of objects.
- Also check out **scikit-image** : [skeletonize](#)

Medial Axis: Examples



Recalls or extends to:

Voronoi graphs

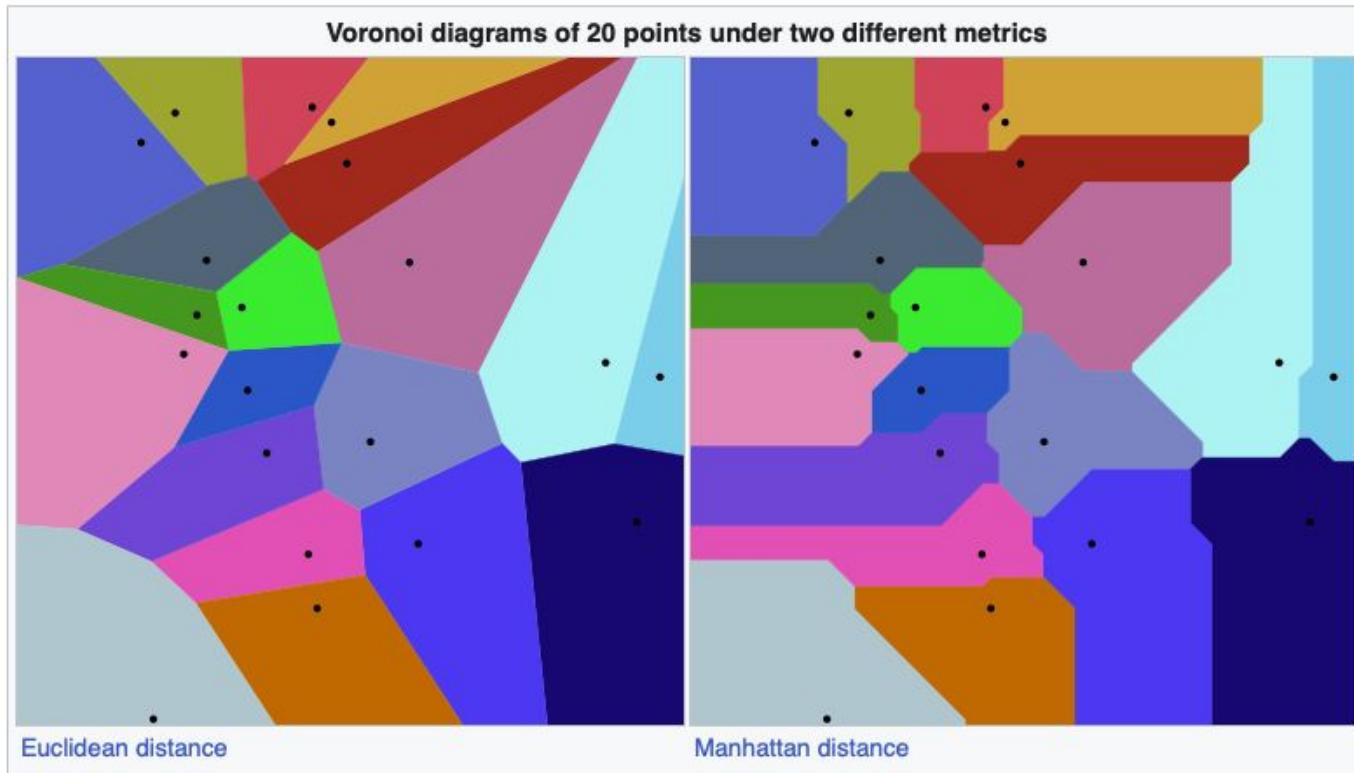


Observe that:

Voronoi graphs
has a **k-means**
flavor



Voronoi: Distance Metrics



Dilation & Erosion

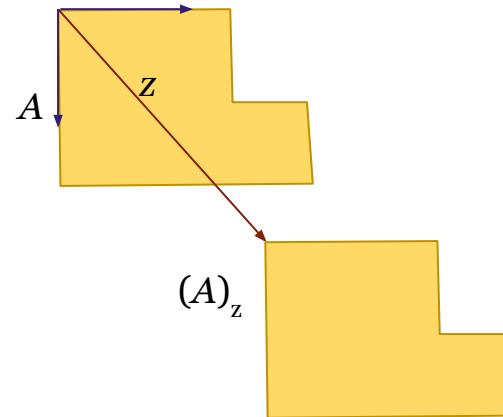
- Fundamental to morphological processing of binary images
- Many other algorithms are based on these two primitives

Dilation & Erosion: Translation and Reflection

On image \mathbf{I} , A & B are foreground objects:

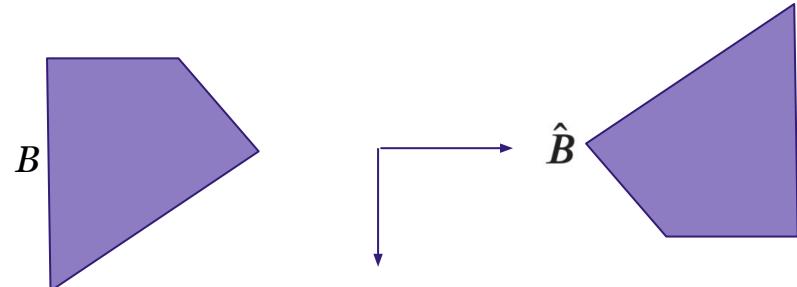
- Translation

$$(A)_z = \{c \mid c = a + z, \text{ for } a \in A\}$$



- Reflection

$$\hat{B} = \{w \mid w = -b, \text{ for } b \in B\}$$



erosion noun

 Save Word

ero·sion | \ i-'rō-zhən \

Definition of *erosion*

- 1 **a** : the action or process of eroding
- b** : the state of being eroded
- 2 : an instance or product of erosion

Erosion:

$$A \ominus B = \{z \mid (B)_z \subseteq A\}$$

A: foreground on the image to be eroded

B: structuring element (i.e. `selem` in `skimage`)

Erosion → contains all points where the translated `selem` is fully contained by the image foreground (i.e. $(B)_z$ is completely inside A)

Analogy → using an eraser going over the perimeter of the foreground components

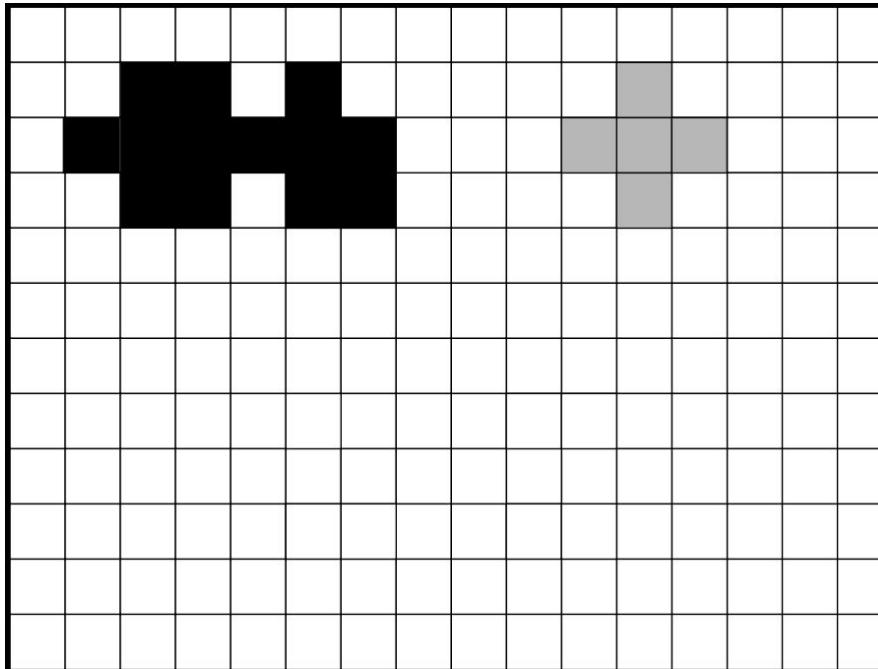
Erosion: the process in practice

$$A \ominus B = \{z \mid (B)_z \subseteq A\}$$

1. Hold `selem` at its center
2. Put it on the top left of the image
3. If `selem` is *fully* contained by the foreground (1s of `selem` match the 1s that of the image), mark the pixel that coincides with the center of the `selem` to belong to the eroded image
4. Systematically move `selem` pixel by pixel all over the *foreground*

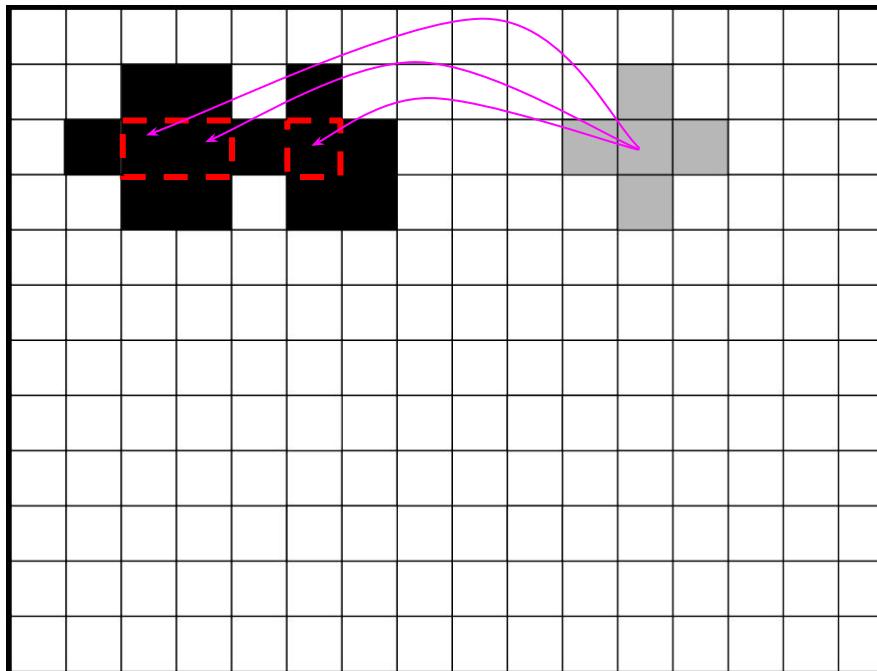
Erosion: the process in practice

$$A \ominus B = \{z \mid (B)_z \subseteq A\}$$



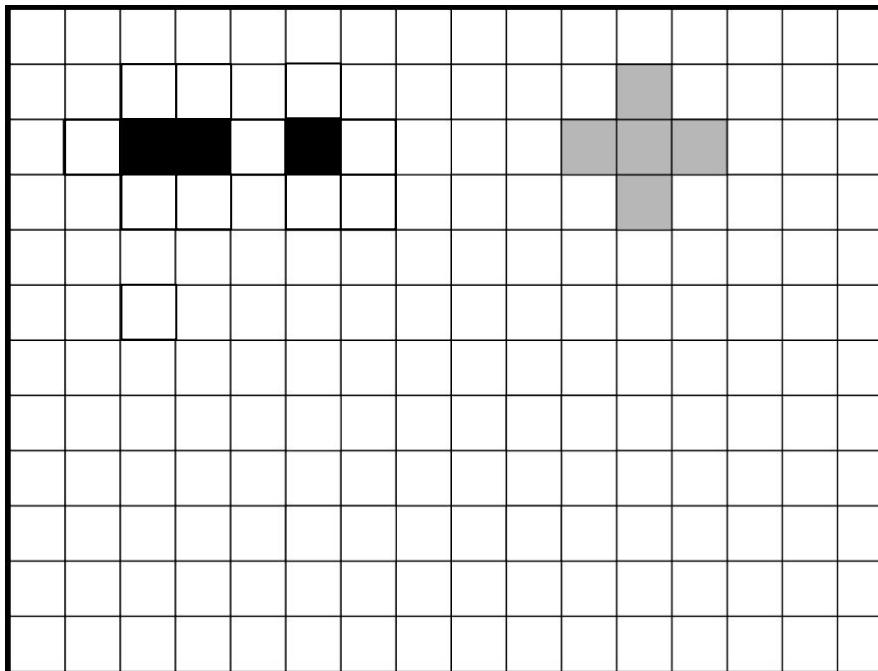
Erosion: the process in practice

$$A \ominus B = \{z \mid (B)_z \subseteq A\}$$



Erosion: the process in practice

$$A \ominus B = \{z \mid (B)_z \subseteq A\}$$



Dilation:

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\}$$

A: foreground on the image to be dilated

B: structuring element

Dilation → contains all points where the reflection of the translated *selem* has any intersection (even 1 pixel) with the image foreground

Analogy → painting over the perimeter of the foreground components with a *Brush* (i.e. the structuring element *B*)

dilation noun

 Save Word

di·la·tion | \ dī-'lā-shən \

Definition of *dilation*

- 1 : the act or action of *dilating* : the state of being *dilated* : EXPANSION, DILATATION
- 2 : the action of stretching or enlarging an organ or part of the body

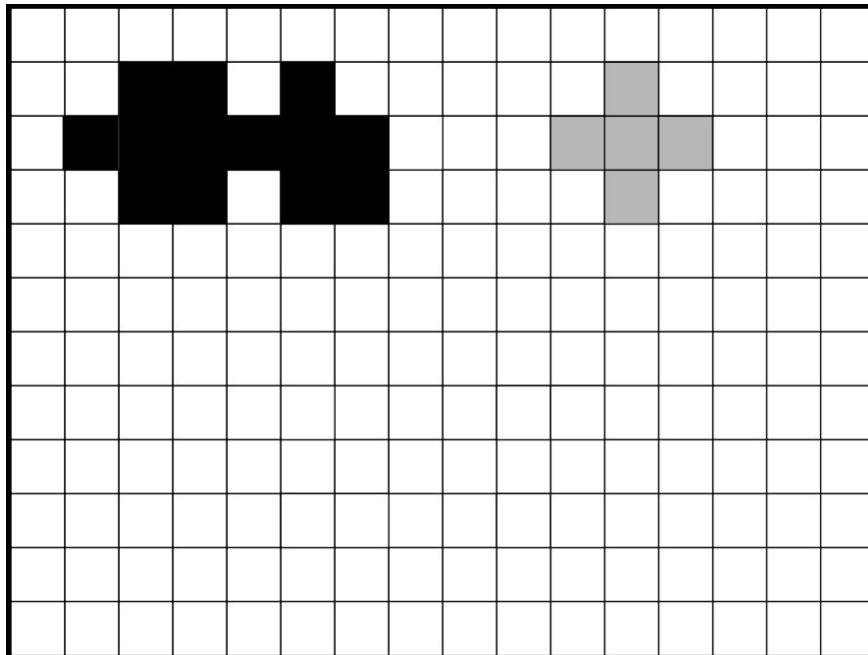
Dilation: the process in practice

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\}$$

1. Hold the reflection of `selem` at its center
2. Put it on the top left of the image
3. If `selem` has any overlap with the foreground, mark the pixel that coincides with the center of the `selem` to belong to the dilated image
4. Systematically move `selem` pixel by pixel all over the image

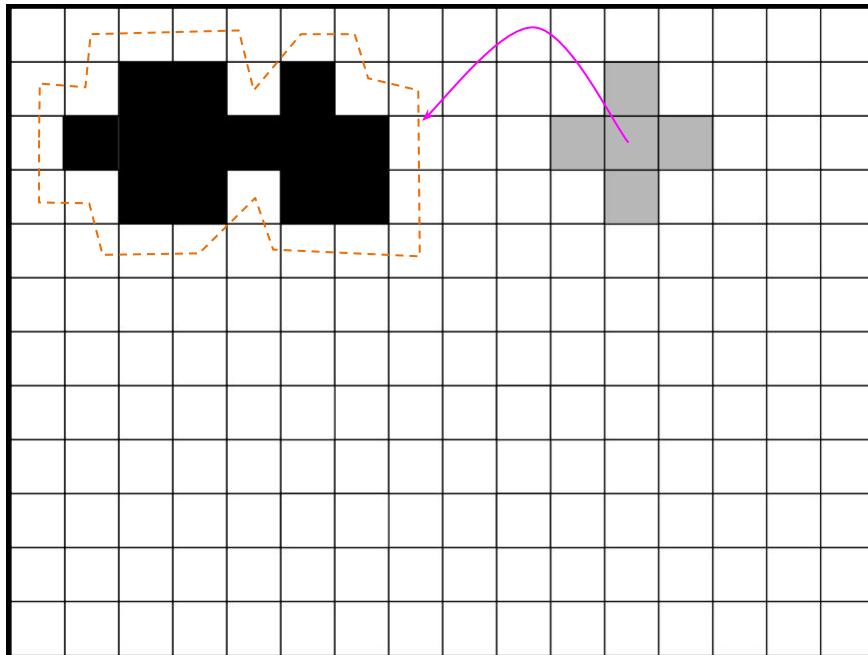
Dilation: the process in practice

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\}$$



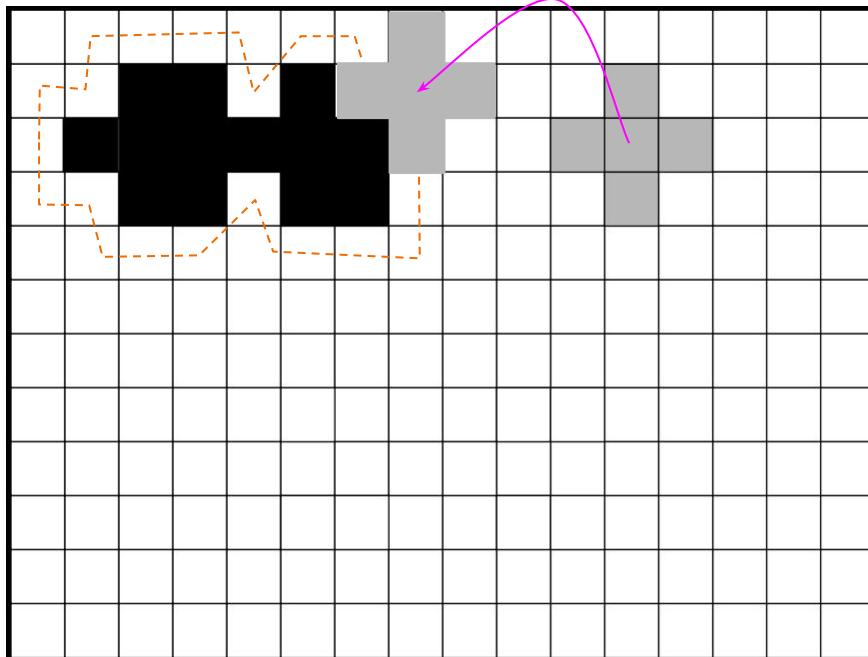
Dilation: the process in practice

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\}$$



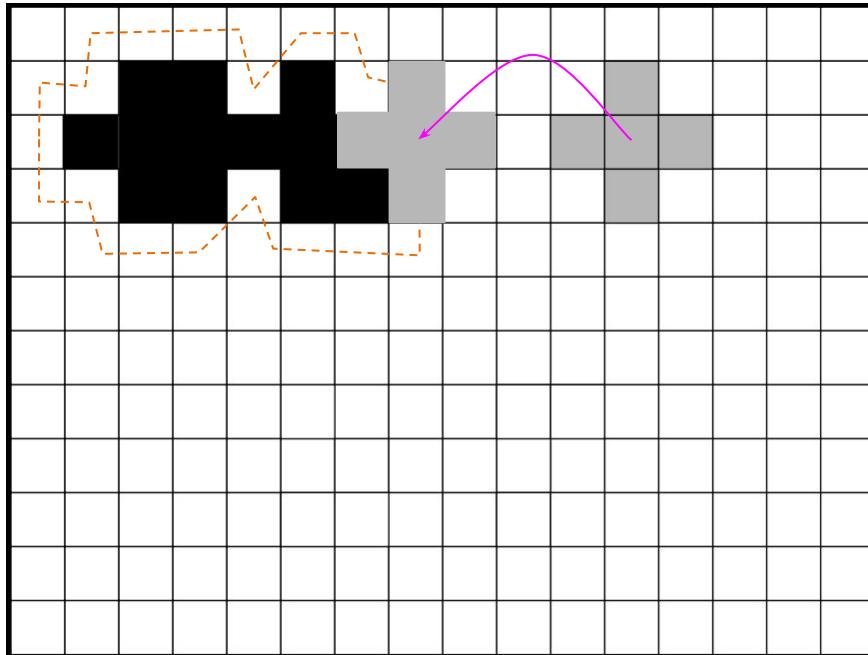
Dilation: the process in practice

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\}$$



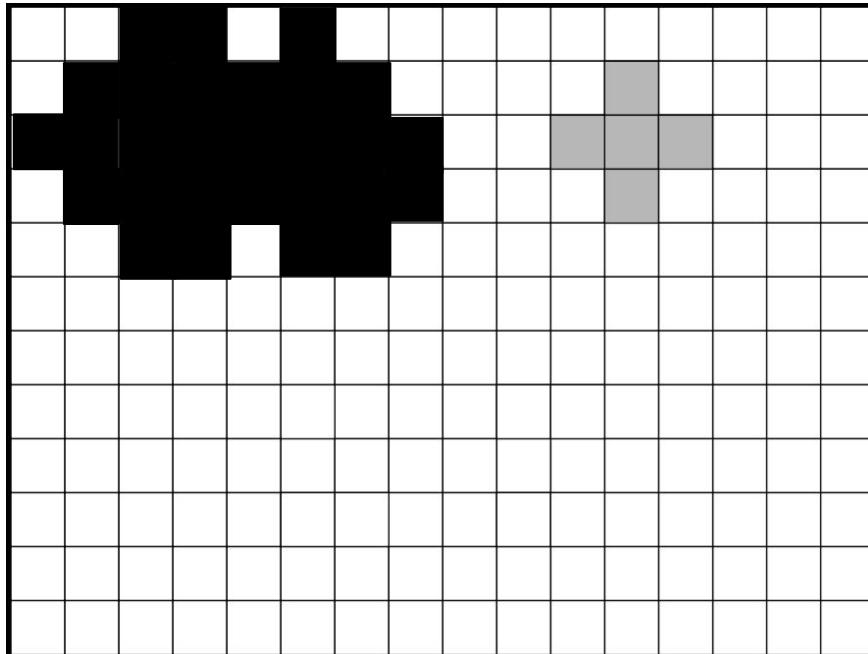
Dilation: the process in practice

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\}$$



Dilation: the process in practice

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\}$$



Erosion & Dilation

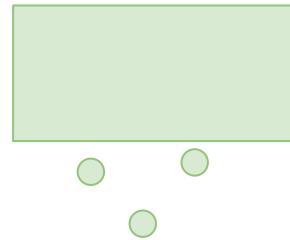
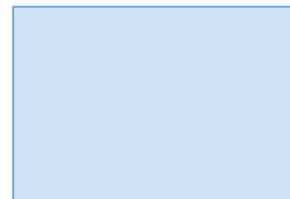
Erosion



Original Image



Dilation



Opening & Closing an Image

Opening A : $A \circ B = (A \ominus B) \oplus B$

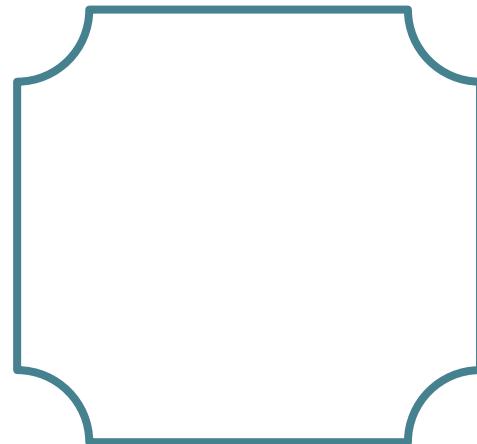
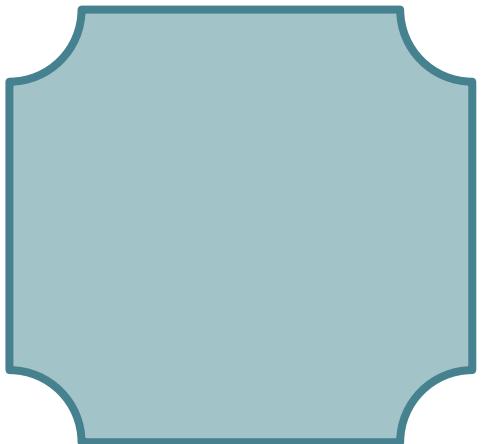
- Erode first, then dilate with B
- Get rid of tiny foreground components
- Separate weakly connected components

Closing A : $A \bullet B = (A \oplus B) \ominus B$

- Dilate first, then erode with B
- Get rid of holes and small irregularities on the perimeter
- close gaps between components

Boundary Extraction:

How to extract the boundary?



Images multiplied: say what?

Hadamard (a.k.a Schur) product: $\mathbf{A} \circ \mathbf{B}$

Yes even the simplest things/ ideas have names as usual

Note that this is the default multiplication for numpy, pytorch etc for matrices of equal size

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

Images multiplied: come again Hadamard product

A

| | | |
|---|---|----|
| 1 | 5 | 10 |
| 1 | 2 | 0 |
| 3 | 0 | 5 |

B

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

$A \circ B$

| | | |
|---|---|----|
| 1 | 5 | 10 |
| 1 | 2 | 0 |
| 3 | 0 | 5 |

$\Sigma(A \circ B)$

27

What is the meaning of $\Sigma(A \circ B)$?

Images multiplied: sum of Hadamard product

A

| | | |
|---|---|----|
| 1 | 5 | 10 |
| 1 | 2 | 0 |
| 3 | 0 | 5 |

B

| | | |
|-----|-----|-----|
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |

$A \circ B$

| | | |
|-----|-----|------|
| 1/9 | 5/9 | 10/9 |
| 1/9 | 2/9 | 0/9 |
| 3/9 | 0/9 | 5/9 |

$\sum(A \circ B)$

3

What is the meaning of $\sum(A \circ B)$?

Why are all neighbors equally important?

Images multiplied: come again Hadamard product

A

| | | |
|---|---|----|
| 1 | 5 | 10 |
| 1 | 2 | 0 |
| 3 | 0 | 5 |

B

| | | |
|-----|------|-----|
| 1/9 | 5/9 | 1/9 |
| 5/9 | 10/9 | 5/9 |
| 1/9 | 5/9 | 1/9 |

$A \circ B$

| | | |
|-----|------|------|
| 1/9 | 25/9 | 10/9 |
| 5/9 | 20/9 | 0/9 |
| 3/9 | 0/9 | 5/9 |

$\sum(A \circ B)$

7.66

What is the meaning of $\sum(A \circ B)$?

Note that $\rightarrow \sum B = 34 / 9$

Images multiplied: come again Hadamard product

A

| | | |
|---|---|----|
| 1 | 5 | 10 |
| 1 | 2 | 0 |
| 3 | 0 | 5 |

B

| | | |
|------|-------|------|
| 1/34 | 5/34 | 1/34 |
| 5/34 | 10/34 | 5/34 |
| 1/34 | 5/34 | 1/34 |

$A \circ B$

| | | |
|------|-------|-------|
| 1/34 | 25/34 | 10/34 |
| 5/34 | 20/34 | 0/34 |
| 3/34 | 0/34 | 5/34 |

$\sum(A \circ B)$

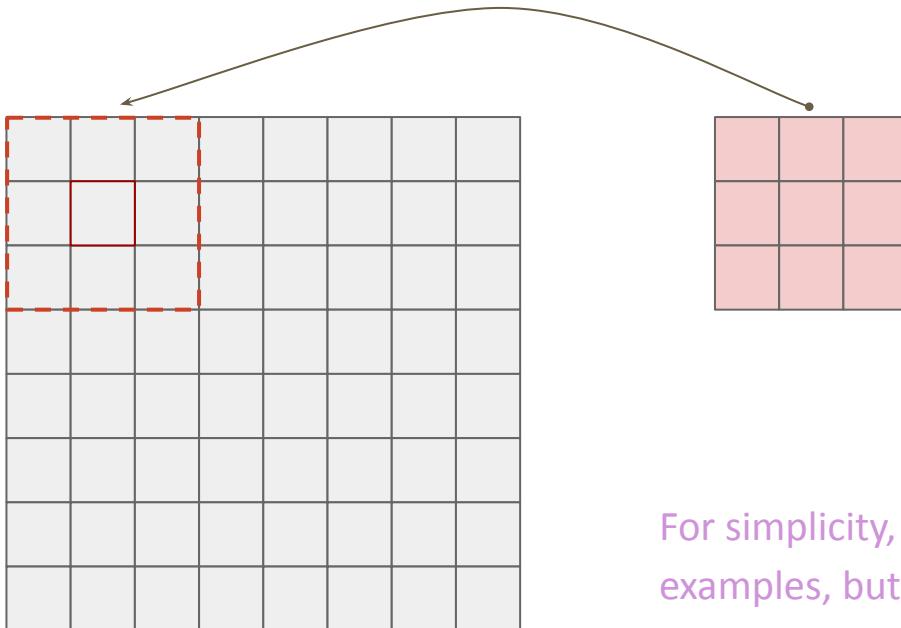
2.03

What is the meaning of $\sum(A \circ B)$?

$$\sum B = 1$$

Convolution: very similar to cross correlation

Recall: reflection from morphological operators $\hat{B} = \{w \mid w = -b, \text{ for } b \in B\}$



I : image

B : kernel

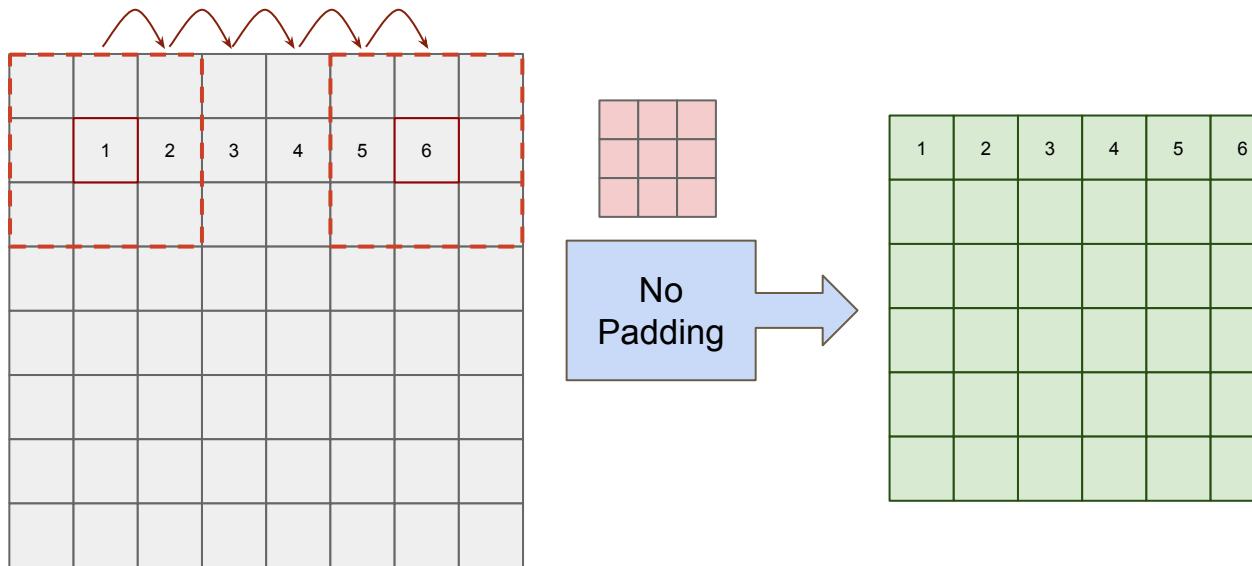
Convolution: Reflection of the kernel

B travels around the image I

For simplicity, we will **not** incorporate **reflection** in the following examples, but sample codes demonstrate the actual usage

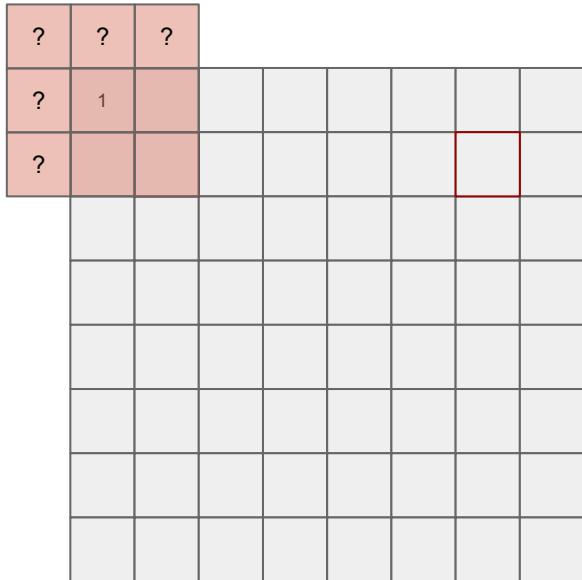
Convolution dilemma: pad or shrink

What happens when center pixel of B is on the edge of the I ?

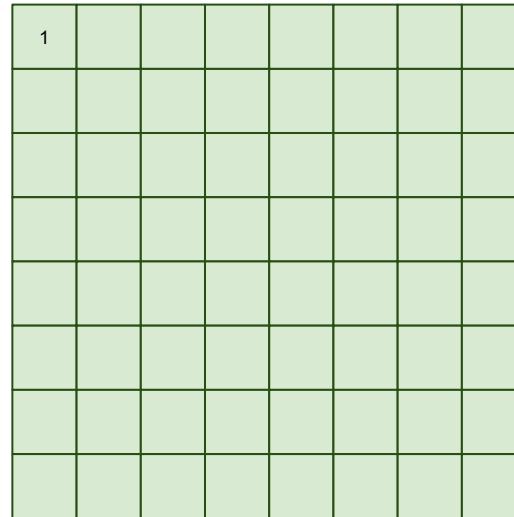


Convolution dilemma: pad or shrink

What happens when center pixel of B is on the edge of the I ?



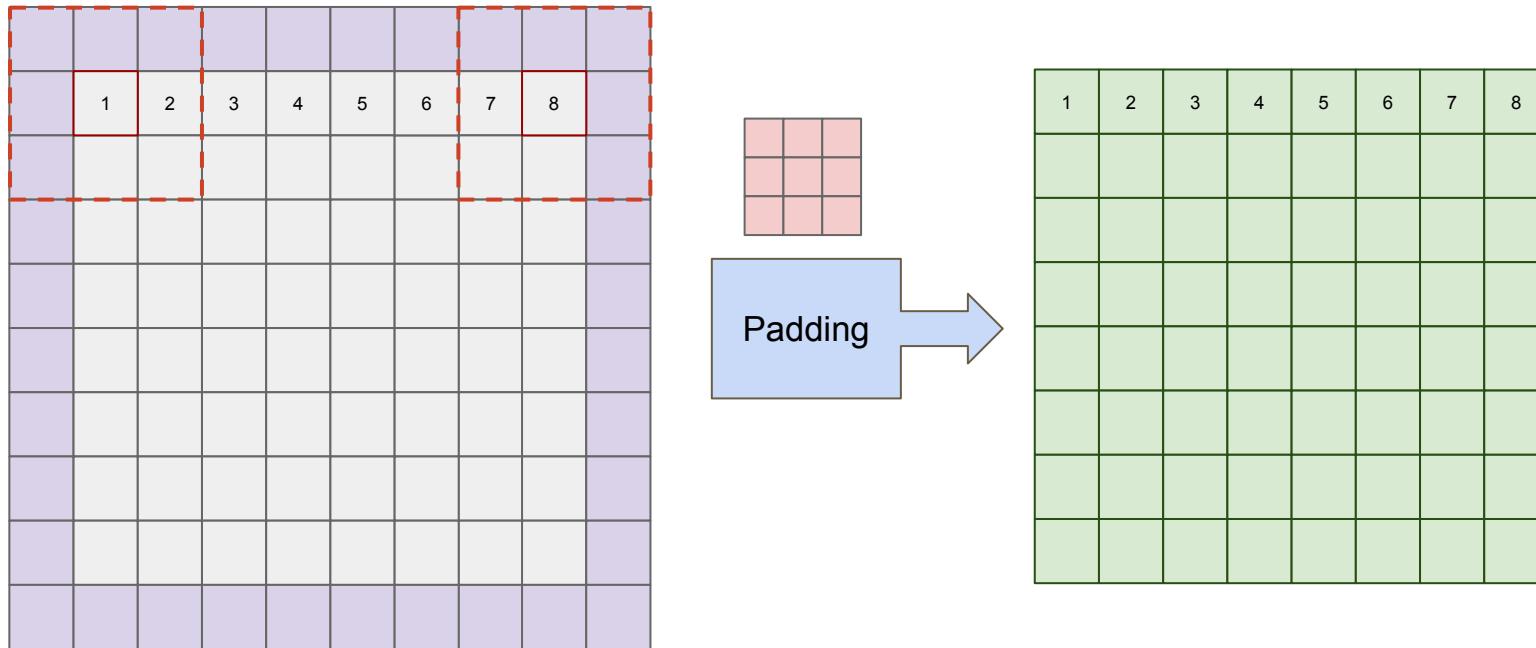
Padding
but how?



Convolution dilemma: pad not to shrink

Note that padding depends on the size of the kernel

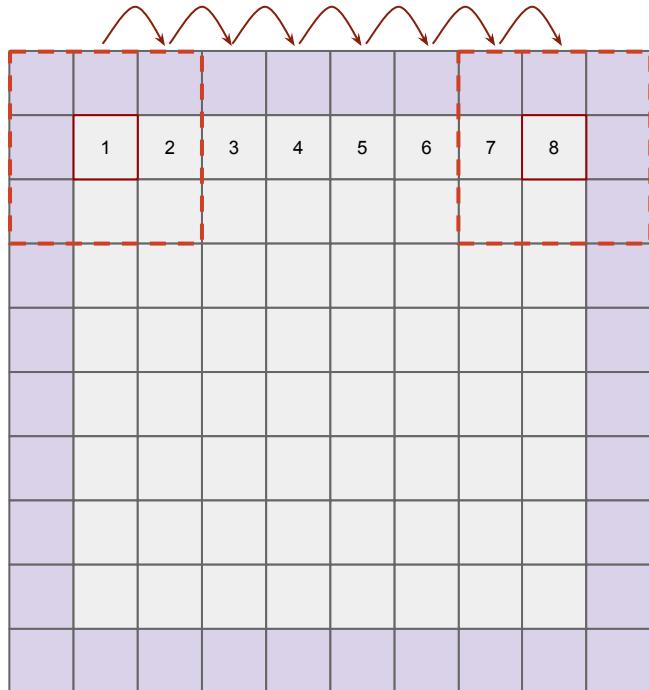
Kernel size: in general a center pixel (i.e. kernel fully contained by the image) can be found **without ambiguity**



Convolution: pad but how?

Note that **number** of padded pixels **depends** on the **size of the kernel**

Kernel size: in general a center pixel can be found without ambiguity



[scipy.ndimage.convolve\(\)](#)

Constant: zero, one padding commonly used

Reflect: $d \ c \ b \ a \ | \ a \ b \ c \ d \ | \ d \ c \ b \ a$

Nearest: $a \ a \ a \ a \ | \ a \ b \ c \ d \ | \ d \ d \ d \ d$

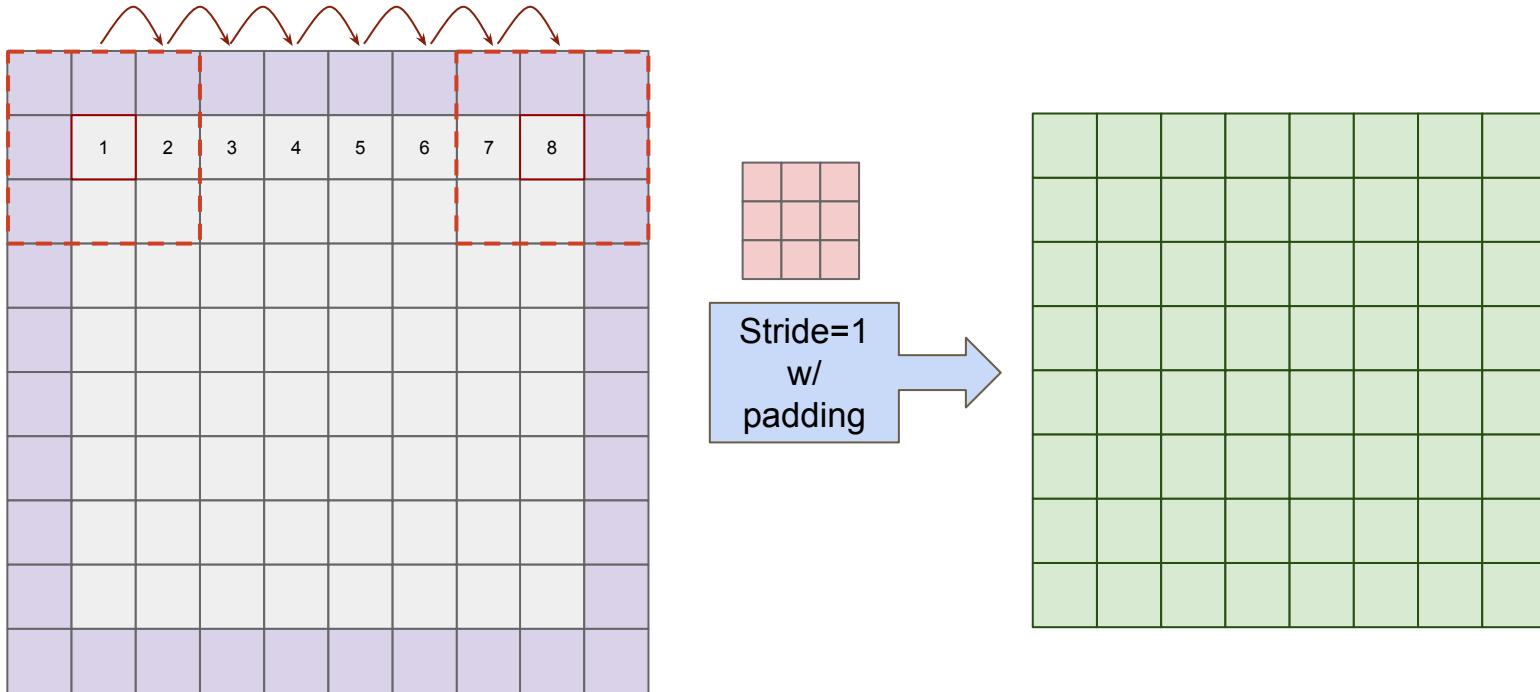
Mirror: $d \ c \ b \ | \ a \ b \ c \ d \ | \ c \ b \ a$

Wrap: $a \ b \ c \ d \ | \ a \ b \ c \ d \ | \ a \ b \ c \ d$

and more...

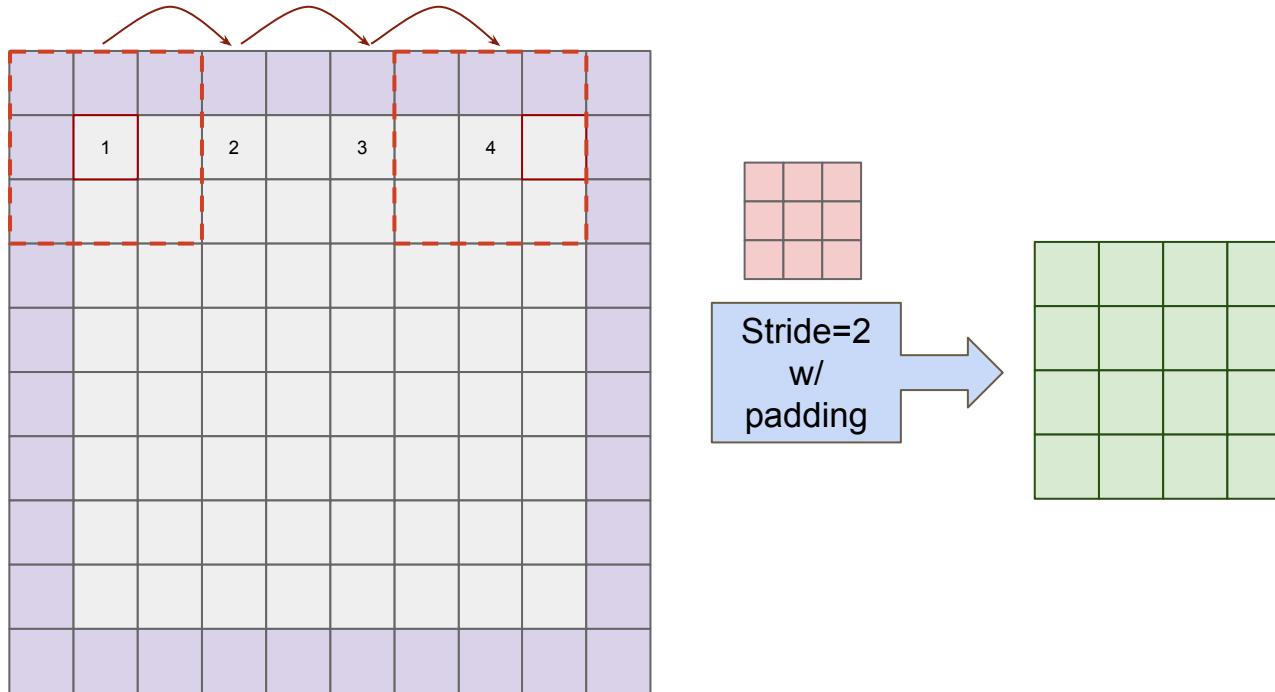
Convolution: stride - 1

Does the kernel have to move 1 step at a time?



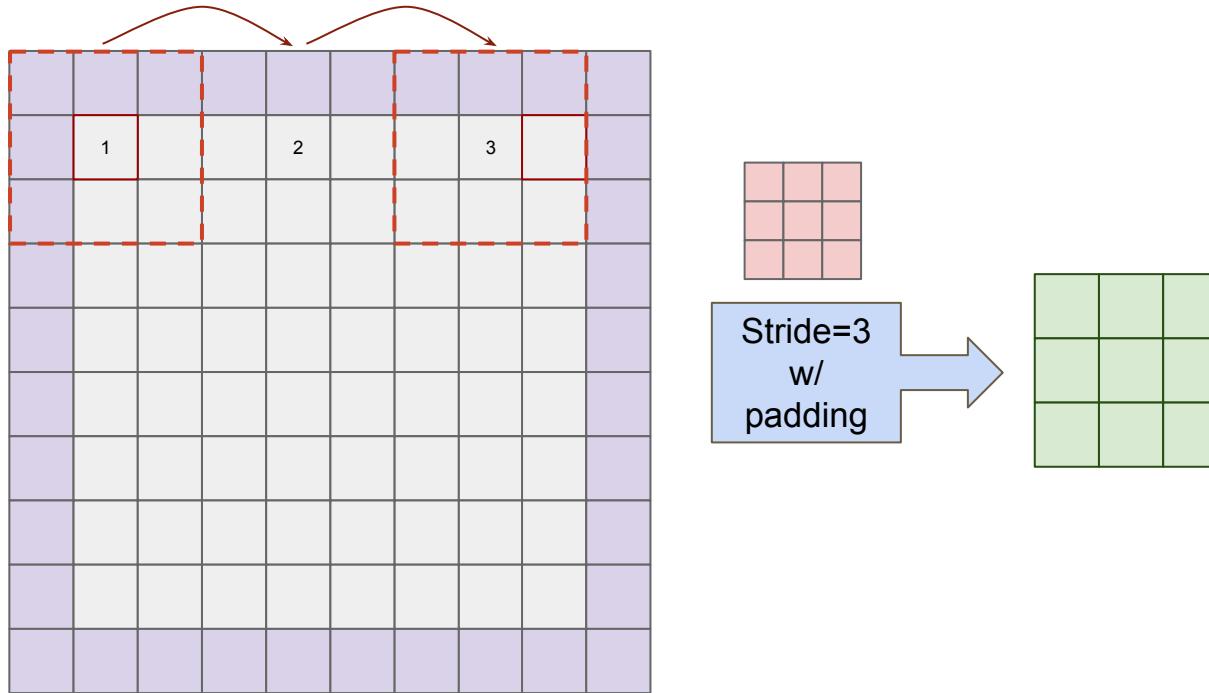
Convolution: stride - 2

Does the kernel have to move 1 step at a time?



Convolution: stride - 3

With a proper kernel, result is like a summary of the bigger picture



Convolution: popular kernels

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

Edge Detection:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$
$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Sharpen:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Blur:

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Gaussian Blur 3x3:

$$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Gaussian Blur 5x5:

$$\frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Different Kernels in action

| Operation | Kernel ω | Image result $g(x,y)$ |
|--------------------------|---|--|
| Identity | $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ |  |
| Ridge or edge detection | $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$ |  |
| | $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$ |  |
| Sharpen | $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$ |  |
| Box blur (normalized) | $\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ |  |

| Operation | Kernel ω | Image result $g(x,y)$ |
|--|---|---|
| Gaussian blur 3×3 (approximation) | $\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$ |  |
| Gaussian blur 5×5 (approximation) | $\frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$ |  |
| Unsharp masking 5×5 Based on Gaussian blur with amount as 1 and threshold as 0 (with no image mask) | $\frac{-1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & -476 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$ |  |

Kernels: how to choose the best?

Choice depends on the need

Tailored based on *experience or trial & error*

trial & error → neural network training ?

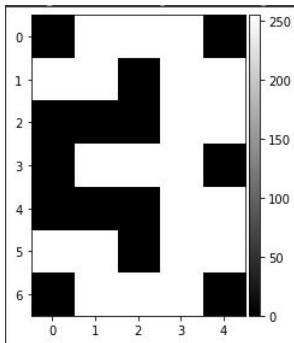
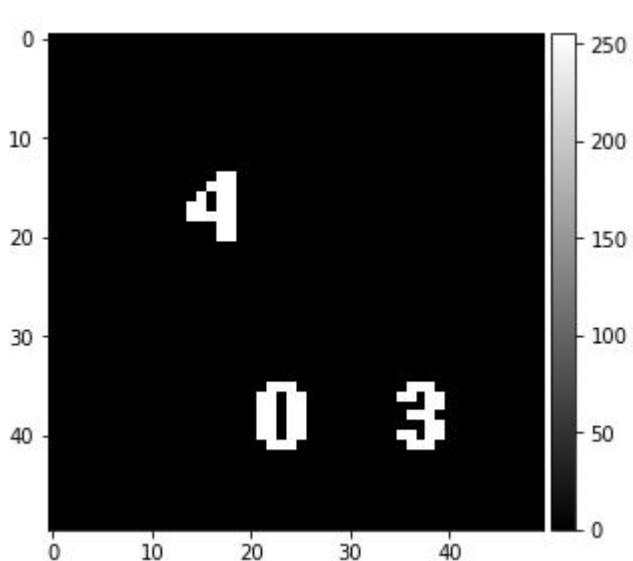
Using a set of specialized kernels to detect objects

Hands on with [this notebook](#)

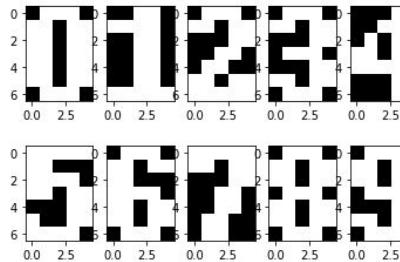
This is where convolutional networks shine!!!

Talking about detection:

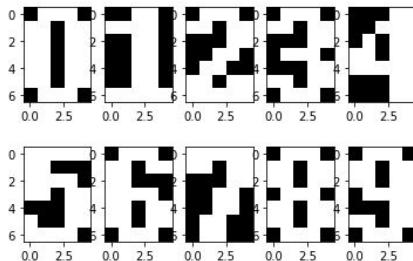
Check [this page](#) out for matching patterns using the simple convolution idea



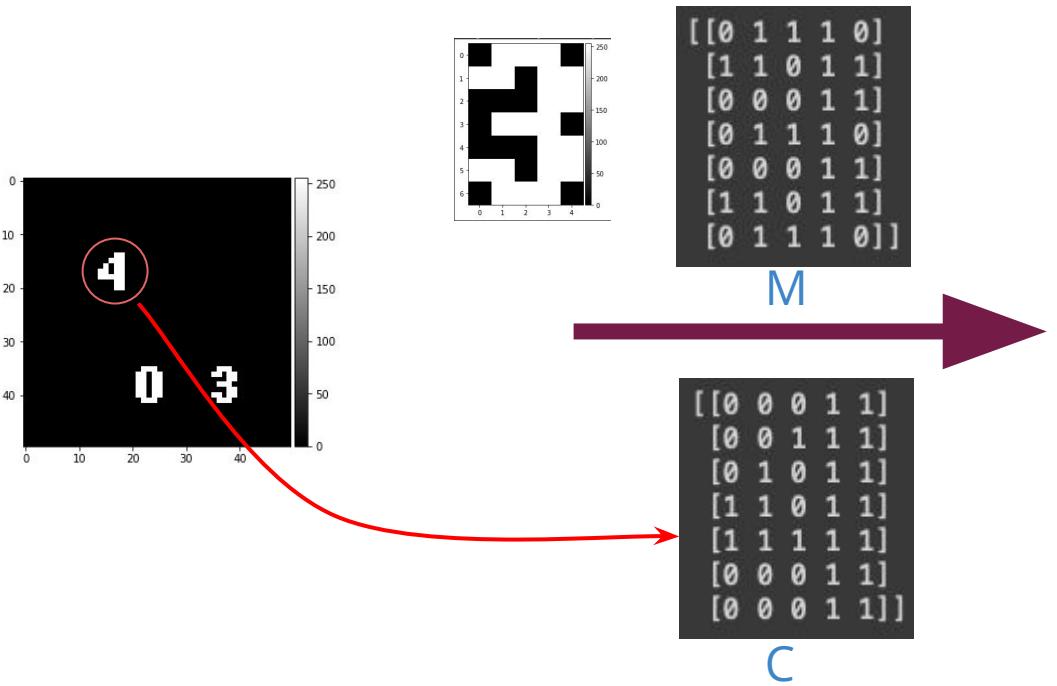
???



Talking about detection: Method 1



Match same foreground pixels in the mask (kernel, template etc)



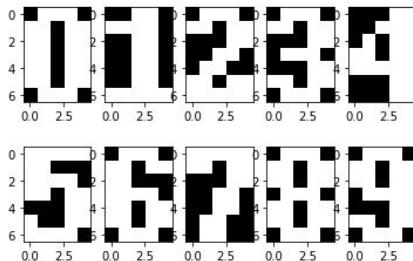
`np.sum(M*C)`

Result: number of matching 1s,
i.e. foreground pixels

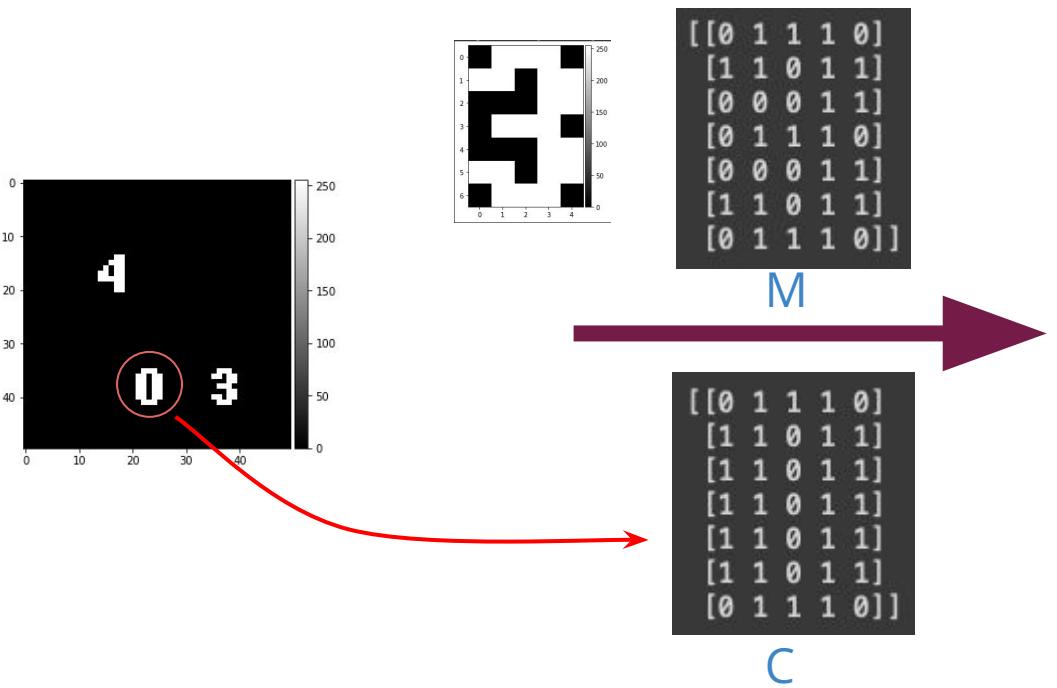
A perfect match → 21

Current Score → 12

Talking about detection: Method 1



Match same foreground pixels in the mask (kernel, template etc)



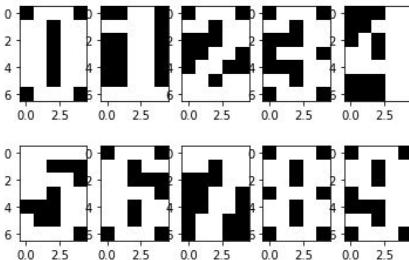
np.sum(M*C)

Result: number of matching 1s,
i.e. foreground pixels

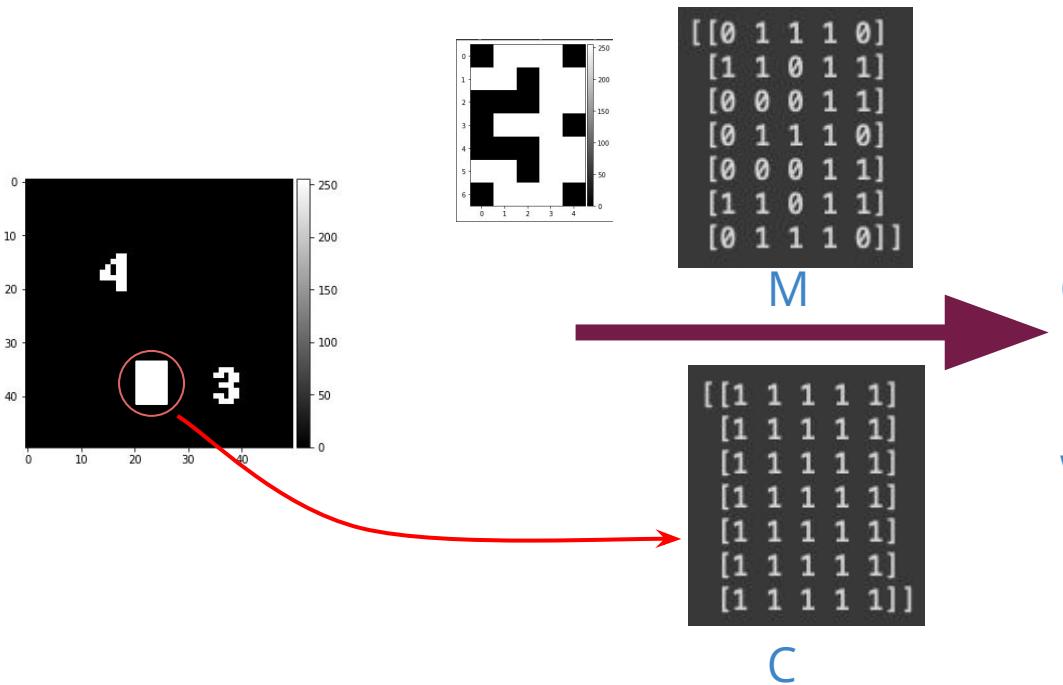
A perfect match → 21

Current Score → 20

Talking about detection: Method 1



Match same foreground pixels in the mask (kernel, template etc)

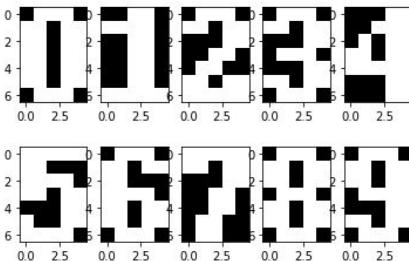


`np.sum(M*C)`

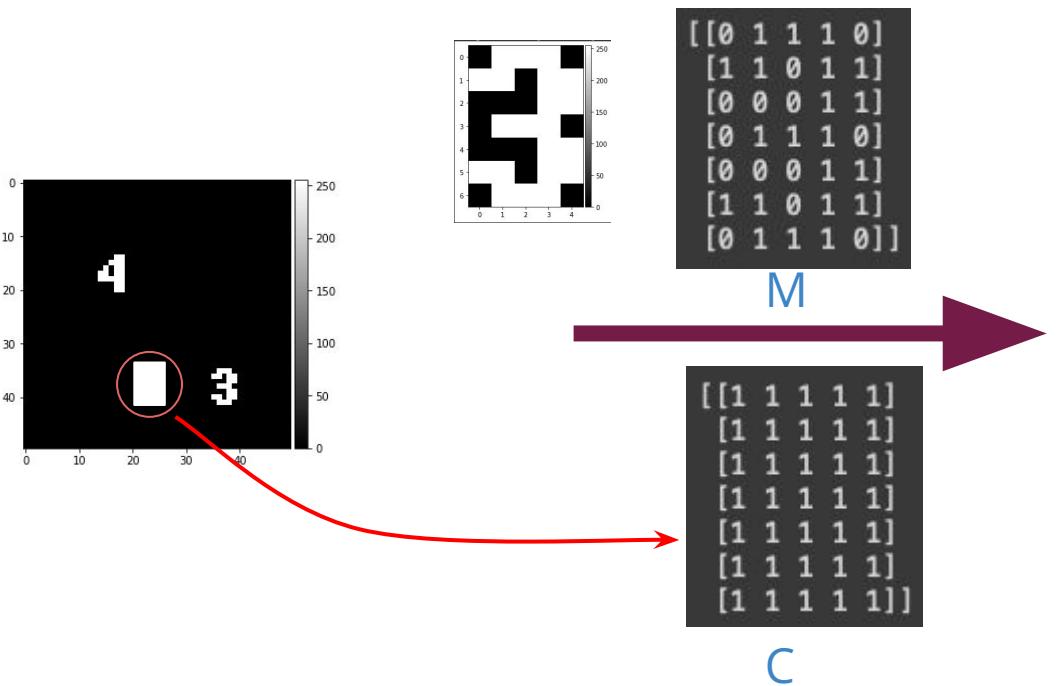
Oooops

Note that a perfectly white region will result in perfect match!!!

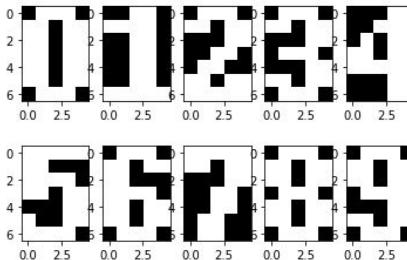
Talking about detection: Method 1



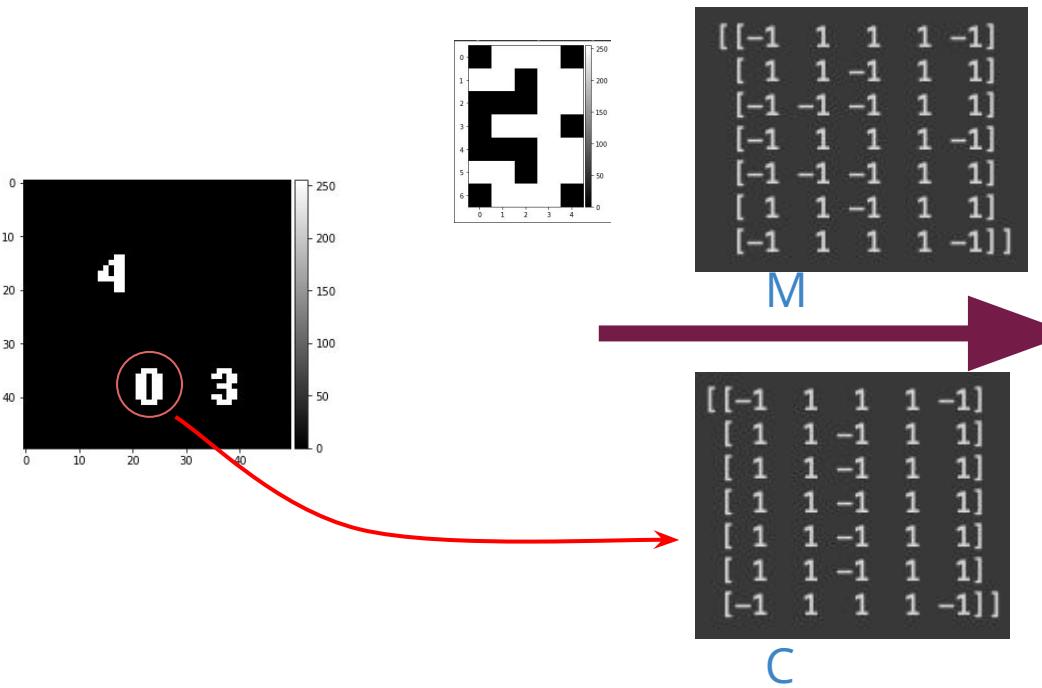
Match same foreground pixels in the mask (kernel, template etc)



Talking about detection: Method 2



Match same foreground and background pixels and punish for mismatches



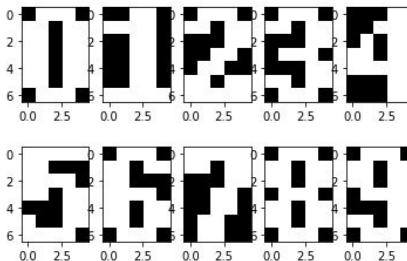
`np.sum(M*C)`

Result: number of all matching pixels minus all non-matching ones

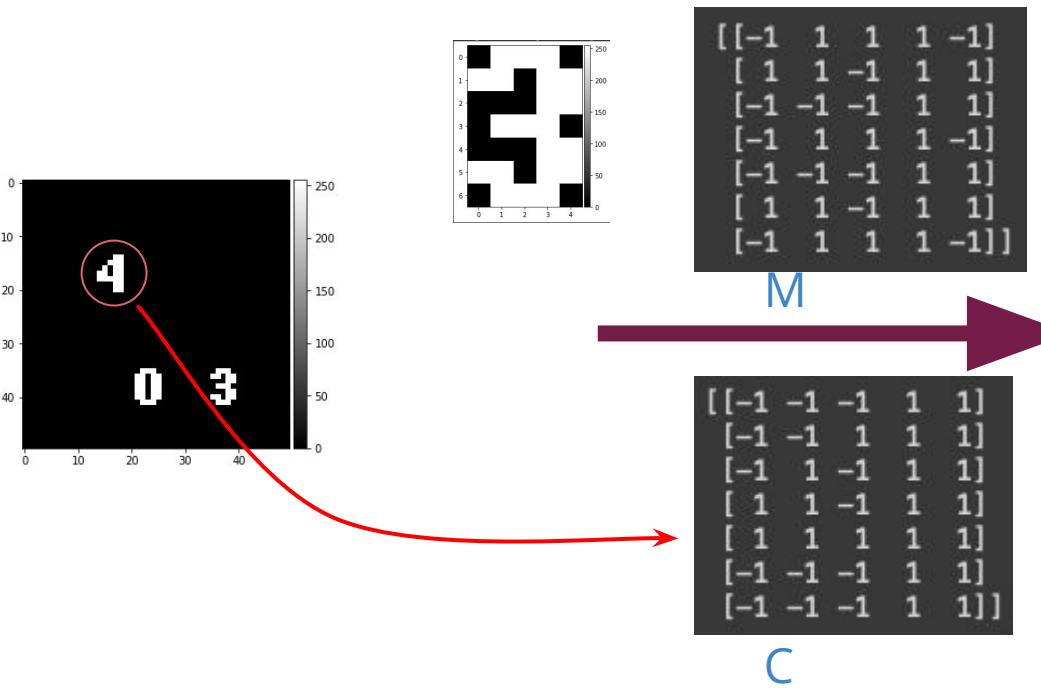
A perfect match → 35

Current Score → 21

Talking about detection: Method 2



Match same foreground and background pixels and punish for mismatches



`np.sum(M*C)`

Result: number of all matching pixels minus all non-matching ones

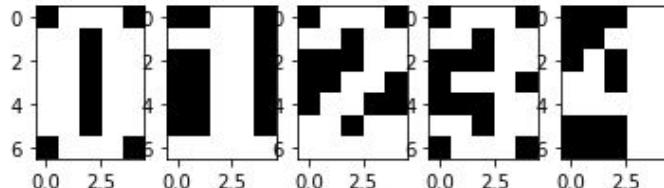
A perfect match → 35

Current Score → -1

Matching results for all the digits

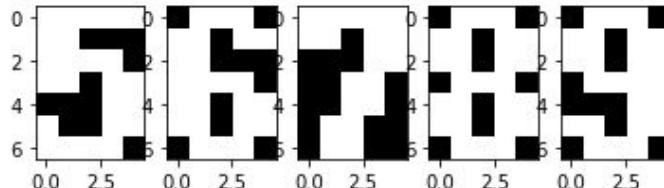
Method 1

| | | | | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| 26.00 | 12.00 | 18.00 | 20.00 | 17.00 | 20.00 | 23.00 | 14.00 | 24.00 | 23.00 |
| 12.00 | 19.00 | 15.00 | 13.00 | 10.00 | 13.00 | 12.00 | 13.00 | 13.00 | 13.00 |
| 18.00 | 15.00 | 22.00 | 18.00 | 12.00 | 14.00 | 17.00 | 15.00 | 19.00 | 18.00 |
| 20.00 | 13.00 | 18.00 | 21.00 | 12.00 | 16.00 | 19.00 | 15.00 | 21.00 | 21.00 |
| 17.00 | 10.00 | 12.00 | 12.00 | 21.00 | 13.00 | 14.00 | 9.00 | 15.00 | 14.00 |
| 20.00 | 13.00 | 14.00 | 16.00 | 13.00 | 24.00 | 18.00 | 12.00 | 18.00 | 19.00 |
| 23.00 | 12.00 | 17.00 | 19.00 | 14.00 | 18.00 | 24.00 | 13.00 | 23.00 | 21.00 |
| 14.00 | 13.00 | 15.00 | 15.00 | 9.00 | 12.00 | 13.00 | 19.00 | 15.00 | 15.00 |
| 24.00 | 13.00 | 19.00 | 21.00 | 15.00 | 18.00 | 23.00 | 15.00 | 25.00 | 23.00 |
| 23.00 | 13.00 | 18.00 | 21.00 | 14.00 | 19.00 | 21.00 | 15.00 | 23.00 | 24.00 |

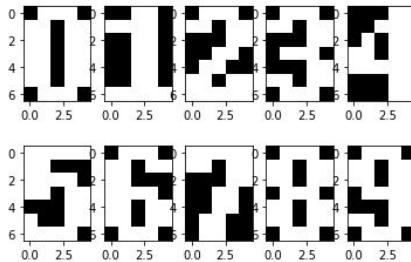


Method 2

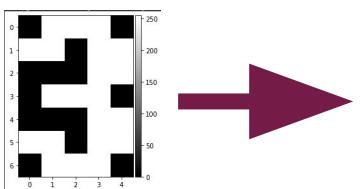
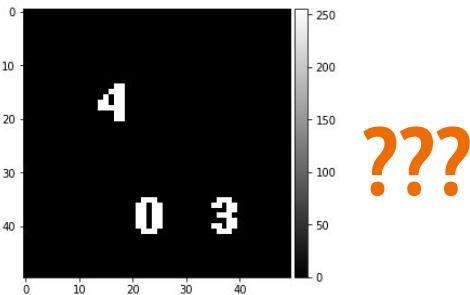
| | | | | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| 35.00 | -7.00 | 11.00 | 21.00 | 9.00 | 15.00 | 27.00 | 1.00 | 29.00 | 27.00 |
| -7.00 | 35.00 | 13.00 | 7.00 | -5.00 | 1.00 | -3.00 | 11.00 | -1.00 | 1.00 |
| 11.00 | 13.00 | 35.00 | 21.00 | -3.00 | -1.00 | 11.00 | 13.00 | 17.00 | 15.00 |
| 21.00 | 7.00 | 21.00 | 35.00 | -1.00 | 9.00 | 21.00 | 15.00 | 27.00 | 29.00 |
| 9.00 | -5.00 | -3.00 | -1.00 | 35.00 | -3.00 | 1.00 | -9.00 | 3.00 | 1.00 |
| 15.00 | 1.00 | -1.00 | 9.00 | -3.00 | 35.00 | 11.00 | -3.00 | 9.00 | 15.00 |
| 27.00 | -3.00 | 11.00 | 21.00 | 1.00 | 11.00 | 35.00 | 1.00 | 29.00 | 23.00 |
| 1.00 | 11.00 | 13.00 | 15.00 | -9.00 | -3.00 | 1.00 | 35.00 | 7.00 | 9.00 |
| 29.00 | -1.00 | 17.00 | 27.00 | 3.00 | 9.00 | 29.00 | 7.00 | 35.00 | 29.00 |
| 27.00 | 1.00 | 15.00 | 29.00 | 1.00 | 15.00 | 23.00 | 9.00 | 29.00 | 35.00 |



My way or highway:



You can come up with your own method...



What if:

- Digits are scaled?
- Digits are rotated?
- There is noise in the image?

ME 536

— Week 11: Excuse me,
how can I ... ? —

How can I:

solve this math problem ?

make an optimal decision ?

beat that guy in chess ?

go there ?

prove that ?

...

Find the croc :

Find → Search

1- You should know
when you find it

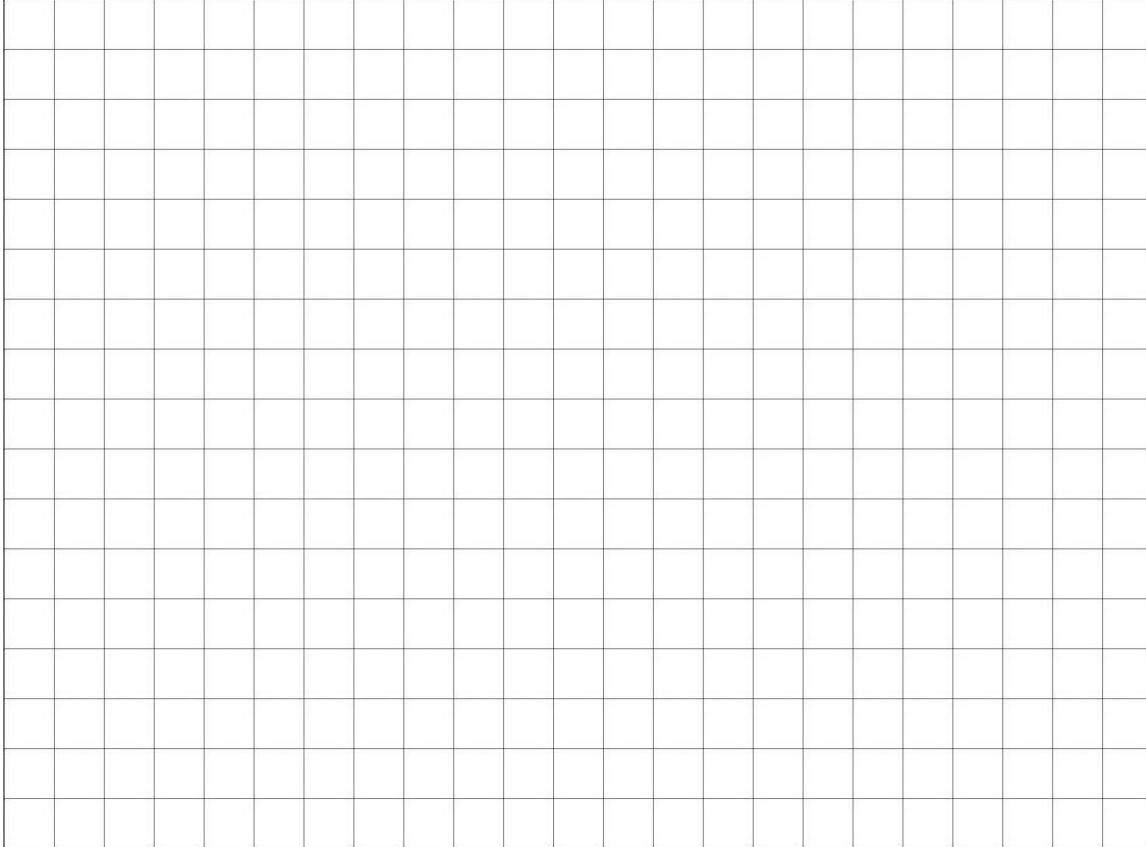
2- Scan

Randomly

Systematically



How can I: get there ?



Get where?

From where?

What are the alternatives?

Let's go: from S to G

A lot to think about:

Cost of moving?

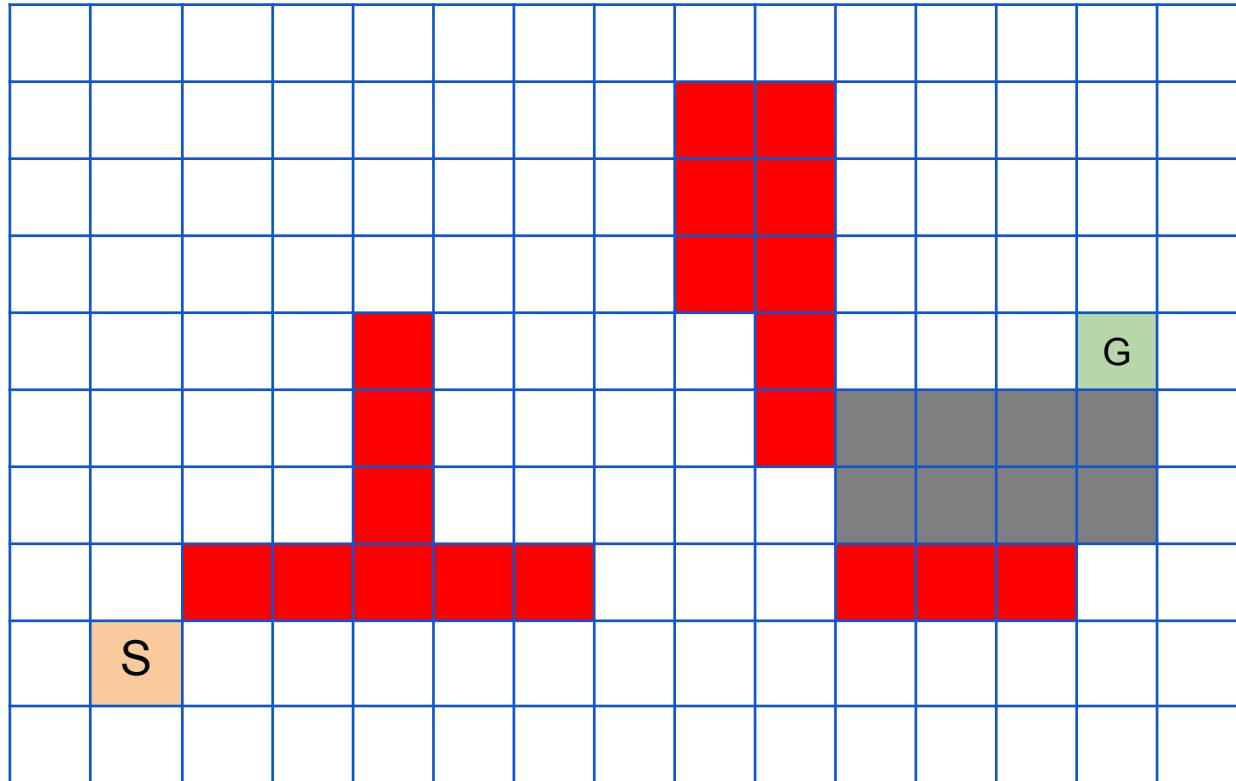
Directions to move?

Direction to choose?

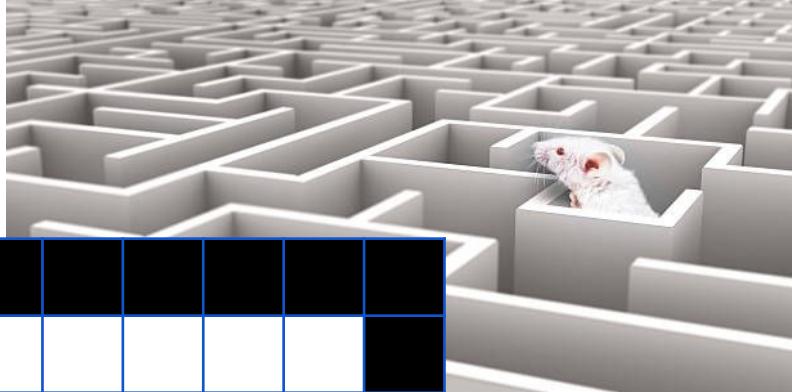
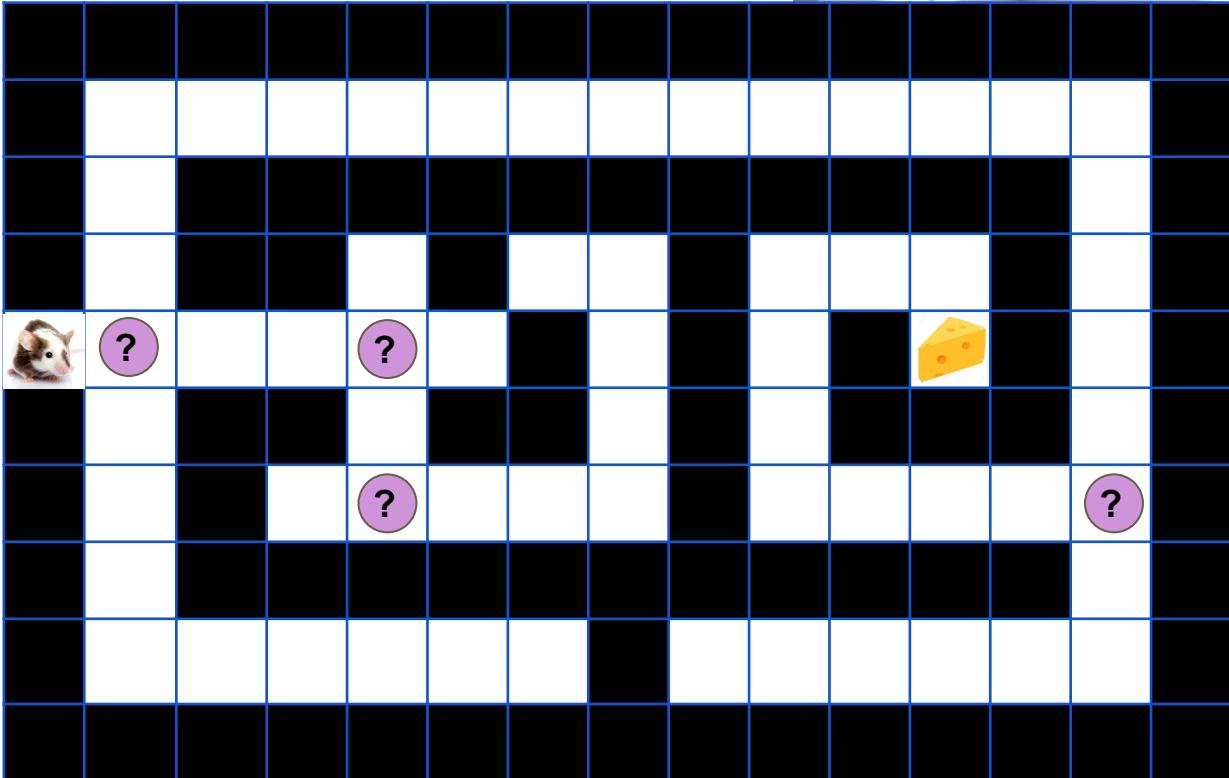
Completely blind

Partially blind

Not blind at all



Let's go: run for the cheese



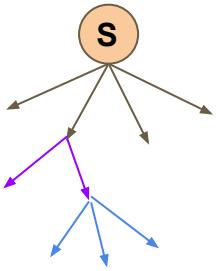
Terminology:

- The **initial state** is the entry point to the **search space**.
- An **operator (successor function $S()$)** returns the set of **reachable states** from state x by a single action given x .
 - $S()$ should avoid repeated and unreachable states
- A **path** is a sequence of actions leading *from one state to another*.
- A **solution** is a **path** between the *initial and goal states*.

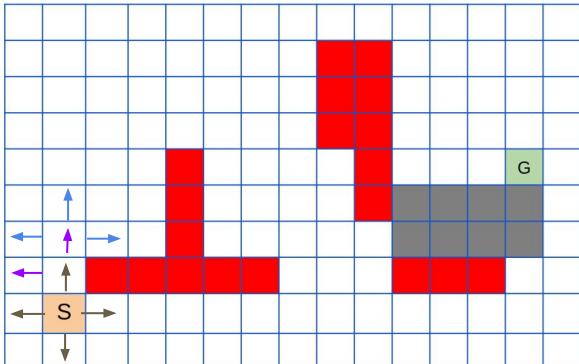
Terminology:

- A **path cost** (*online cost*) function is a function that assigns a cost to a path
- **Search Cost** (*offline cost*) is defined with time and memory required to find a solution / to make a goal reaching plan.
- **Total cost** of the search is the sum of the path cost and search costs.

Search Strategy



- Determines *which states* should be **expanded** and **checked *next***.
- Generally ends up building a so called:
search tree.
- The root of the tree is at the *initial state*.



Search Strategy: How good is it?

Can be evaluated based on:

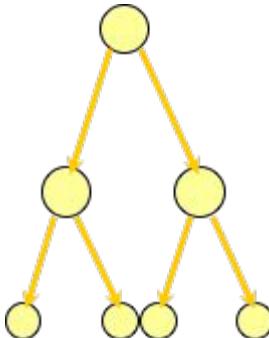
- **Completeness:** is the strategy guaranteed to find a solution when there is one?
- **Time complexity:** how long does it take to find a solution?
- **Space complexity:** how much memory does it need to perform the search?
- **Optimality:** does the strategy find the highest quality solution when there are several alternative solutions?

Search Where:

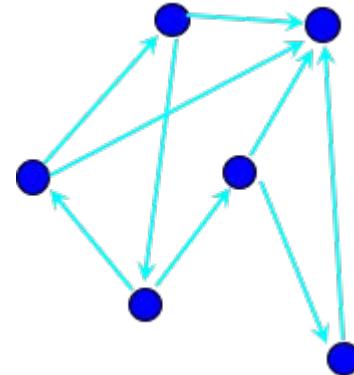
List

11
54
32
22
27
99
19
...

Tree



Graph



- Branching factor b
- $b=2$ is a binary tree

- Is the list sorted?

Informed vs Uninformed Search: what is the catch?

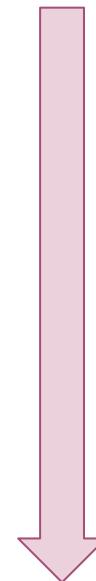
So cliche: *number guessing game*

Guess the number I have in my mind that is between 0-100

*Uninformed or
a.k.a **BLIND** search*



1
NO
2
NO
3
NO
4
NO
...



50
Higher
75
Lower
62
Higher
68
Lower
...

Informed search



Uninformed Search Strategies

- Only binary evaluation is possible after every action
- Distinguished by the order in which nodes are expanded:
 - Breadth-first
 - Depth-first
 - Depth-limited
 - Uniform cost
 - Bidirectional

Breadth-First Search: Fish in the shallows first

Initial State



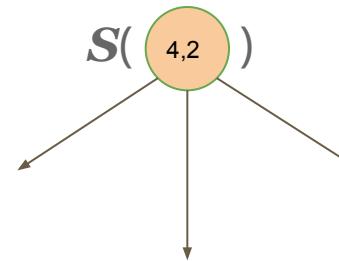
Goal State



Unpassable



| | | | | |
|-----|-----|-----|-----|-----|
| 1,1 | 1,2 | 1,3 | 1,4 | 1,5 |
| 2,1 | 2,2 | 2,3 | 2,4 | 2,5 |
| 3,1 | 3,2 | 3,3 | 3,4 | 3,5 |
| 4,1 | 4,2 | 4,3 | 4,4 | 4,5 |
| 5,1 | 5,2 | 5,3 | 5,4 | 5,5 |



Assume:

- movement in **4-neighbors**
- CCW starting from 6 o'clock

Breadth-First Search: Fish in the shallows first

Fill in the search tree

Initial State



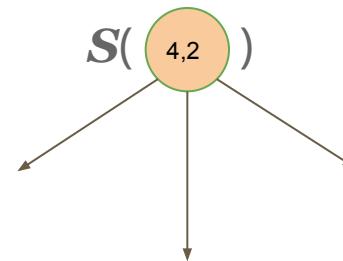
Goal State



Unpassable



| | | | | |
|-----|-----|-----|-----|-----|
| 1,1 | 1,2 | 1,3 | 1,4 | 1,5 |
| 2,1 | 2,2 | 2,3 | 2,4 | 2,5 |
| 3,1 | 3,2 | 3,3 | 3,4 | 3,5 |
| 4,1 | 4,2 | 4,3 | 4,4 | 4,5 |
| 5,1 | 5,2 | 5,3 | 5,4 | 5,5 |



Assume:

- movement in **4-neighbors**
- CCW starting from 6 o'clock

Breadth-First Search: Fish in the shallows first

Fill in the search tree

Initial State



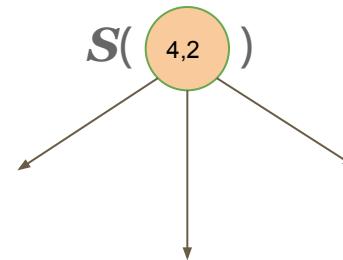
Goal State



Unpassable



| | | | | |
|-----|-----|-----|-----|-----|
| 1,1 | 1,2 | 1,3 | 1,4 | 1,5 |
| 2,1 | 2,2 | 2,3 | 2,4 | 2,5 |
| 3,1 | 3,2 | 3,3 | 3,4 | 3,5 |
| 4,1 | 4,2 | 4,3 | 4,4 | 4,5 |
| 5,1 | 5,2 | 5,3 | 5,4 | 5,5 |



Assume:

- movement in **4-neighbors**
- CCW starting from 6 o'clock

Breadth-First Search: Fish in the shallows first

Fill in the search tree

Initial State



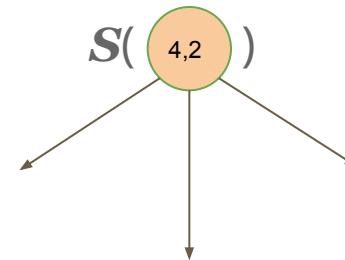
Goal State



Unpassable



| | | | | |
|-----|-----|-----|-----|-----|
| 1,1 | 1,2 | 1,3 | 1,4 | 1,5 |
| 2,1 | 2,2 | 2,3 | 2,4 | 2,5 |
| 3,1 | 3,2 | 3,3 | 3,4 | 3,5 |
| 4,1 | 4,2 | 4,3 | 4,4 | 4,5 |
| 5,1 | 5,2 | 5,3 | 5,4 | 5,5 |



Assume:

- movement in **4-neighbors**
- CCW starting from 6 o'clock

Breadth-First Search: Fish in the shallows first

- If there is a solution, *breadth-first search* is **guaranteed to find** it.
- If there are *several solutions*, it will **find the shallowest one**.
- Space and time complexities are the similar since *all leaf nodes must be maintained in the memory*.
- The time complexity is $O(b^d)$
- The space complexity is $O(b^d)$

Where \mathbf{d} is the depth of the search tree

Problem: Cost

Depth-First Search: Check for fish until...

Initial State



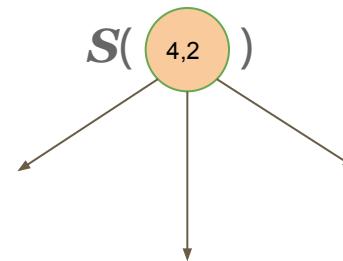
Goal State



Unpassable



| | | | | |
|-----|-----|-----|-----|-----|
| 1,1 | 1,2 | 1,3 | 1,4 | 1,5 |
| 2,1 | 2,2 | 2,3 | 2,4 | 2,5 |
| 3,1 | 3,2 | 3,3 | 3,4 | 3,5 |
| 4,1 | 4,2 | 4,3 | 4,4 | 4,5 |
| 5,1 | 5,2 | 5,3 | 5,4 | 5,5 |



Assume:

- movement in **4-neighbors**
- CCW starting from 6 o'clock

Depth-First Search: Check for fish until...

Initial State



Goal State

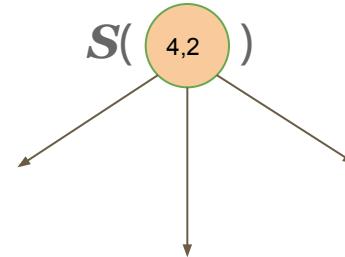


Unpassable



| | | | | |
|-----|-----|-----|-----|-----|
| 1,1 | 1,2 | 1,3 | 1,4 | 1,5 |
| 2,1 | 2,2 | 2,3 | 2,4 | 2,5 |
| 3,1 | 3,2 | 3,3 | 3,4 | 3,5 |
| 4,1 | 4,2 | 4,3 | 4,4 | 4,5 |
| 5,1 | 5,2 | 5,3 | 5,4 | 5,5 |

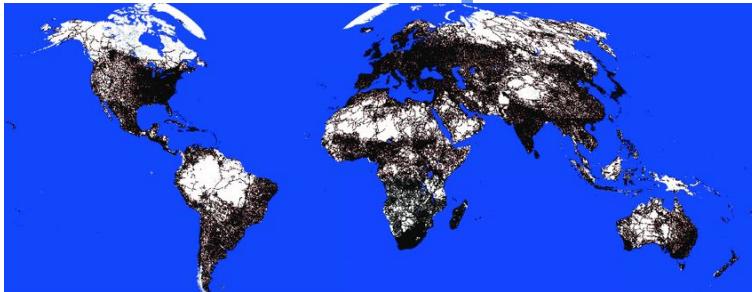
Fill in the search tree



Assume:

- movement in **4-neighbors**
- **CCW starting from 6 o'clock**
 - Consider different alternatives only to see ?

Depth-First Search: Fish in the depths first



- **Major drawback:** some problems have *very deep* or even *infinite-depth* search trees.
- Space complexity is also superior to *breadth-first search* since only the current branch has to be kept in memory.
- The time complexity is $O(b^d)$
- The space complexity is $O(bd)$

Where d is the depth of the search tree

Depth-Limited Search: Quitters may win

- Improve *depth-first search* by *limiting the depth to search*
 - *Pre-determine a max-depth*
 - *Assume bottom when max-depth reached*
 - *Increase max-depth if no solution found*

Uniform Cost Search: Not all states are created equal

- Uniform cost search modifies the breadth-first search strategy by always expanding the **lowest-cost node on the fringe** rather than the lowest-depth.
- Cost of each node is measured by a function:
 - $g(n)$: the path cost of reaching state n from the initial state.
- Path cost is generally assumed to be non-negative: $g(n) \geq 0$
- If costs are constant, $g(n)$ gives the path length between n and the initial state.

Uniform Cost Search: Not all states are created equal

Initial State



Goal State



Unpassable



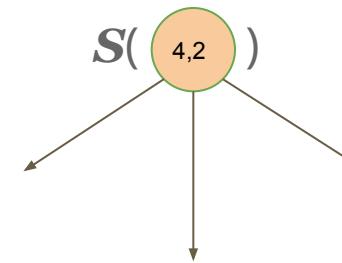
Cost of 1



Cost of 2



| | | | | |
|-----|-----|-----|-----|-----|
| 1,1 | 1,2 | 1,3 | 1,4 | 1,5 |
| 2,1 | 2,2 | 2,3 | 2,4 | 2,5 |
| 3,1 | 3,2 | 3,3 | 3,4 | 3,5 |
| 4,1 | 4,2 | 4,3 | 4,4 | 4,5 |
| 5,1 | 5,2 | 5,3 | 5,4 | 5,5 |



Assume:

- movement in **4-neighbors**
- ~~CCW starting from 6 o'clock~~

Uniform Cost Search - Exercise: Draw the search tree

Initial State



Goal State



Unpassable



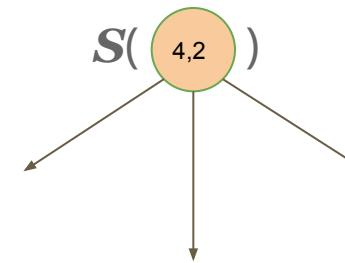
Cost of 1



Cost of 2



| | | | | |
|-----|-----|-----|-----|-----|
| 1,1 | 1,2 | 1,3 | 1,4 | 1,5 |
| 2,1 | 2,2 | 2,3 | 2,4 | 2,5 |
| 3,1 | 3,2 | 3,3 | 3,4 | 3,5 |
| 4,1 | 4,2 | 4,3 | 4,4 | 4,5 |
| 5,1 | 5,2 | 5,3 | 5,4 | 5,5 |



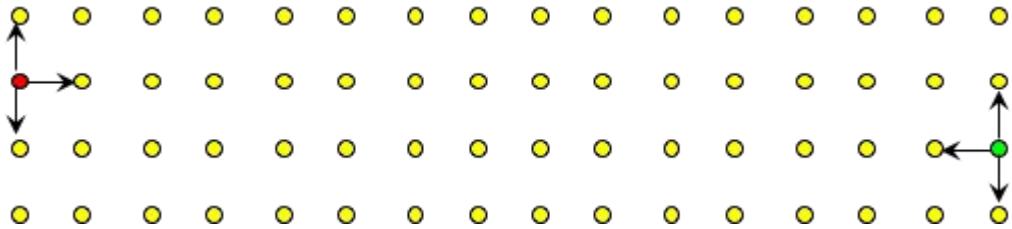
Assume:

- movement in **4-neighbors**
- Let cost of 2 be 3 or 4

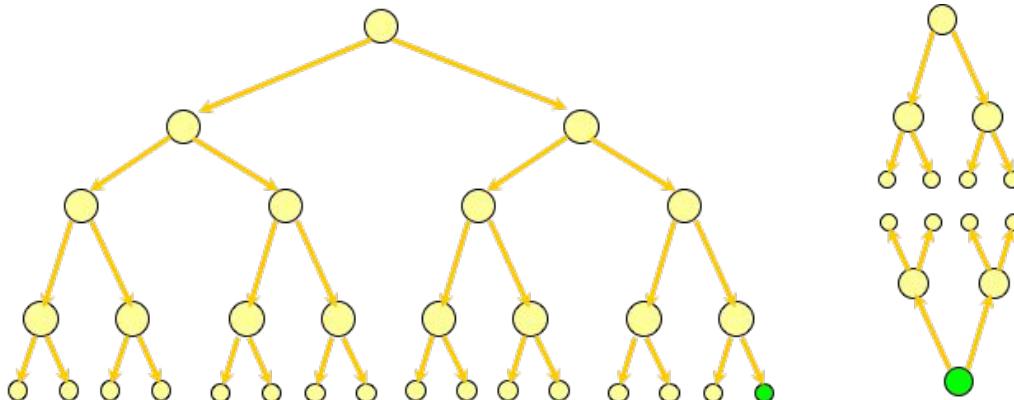
Uniform Cost Search: Not all states are created equal

- Finds the **lowest-cost or cheapest** solution if cost does not decrease:
 - $g(\mathbf{S}(\mathbf{n})) \geq g(\mathbf{n})$
- Finds the **cheapest** solution without exploiting the whole search space / tree

Bi-directional Search:

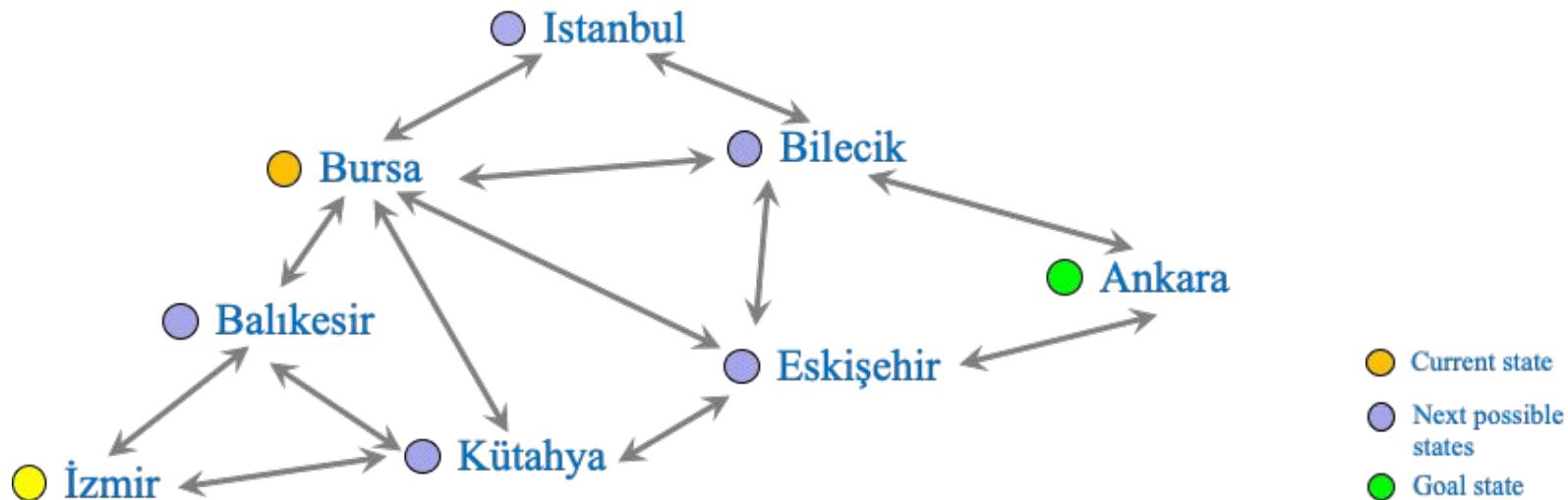


- A way to reduce space complexity
- Solution is found when 2 concurrent search trees meet at one node
- Note that how you can use bi-directional search with *different strategies*



Informed Search: Informed about what ?

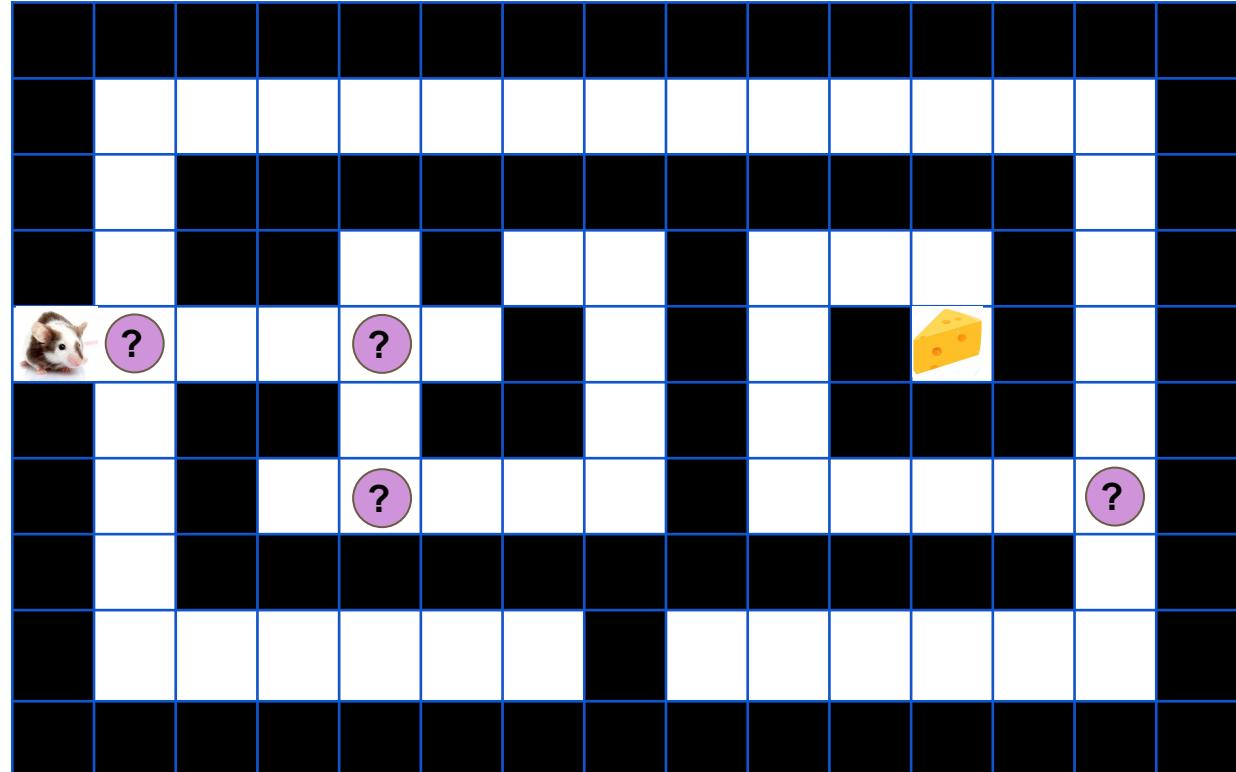
- If **information** or even a *hunch* about **cost to goal** is available
 - *Uninformed search* methods become *inefficient*



Informed Search: Not all *goals* are created equal

- Assume cost is:
 - direct distance to goal
 - smell of the goal

If you **do NOT** know the **whole world**, can you have a very accurate cost estimate?



Heuristic Function: Gut feeling, a hunch,...

For many problems cost of reaching a goal can be **estimated but cannot be determined** (otherwise the problem is already solved :)

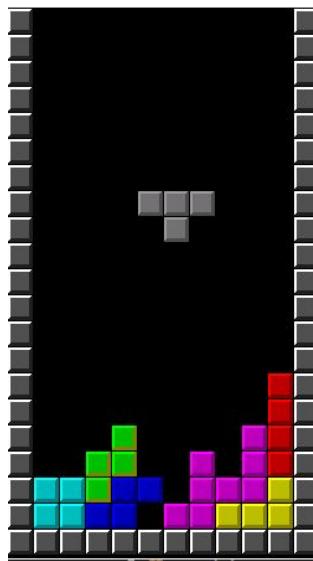
A **heuristic function** estimates the cost of reaching the goal given the current state **n** :

$$h(n) = f(n, \text{goal_state})$$

Writing a **heuristic function** is not necessarily easy

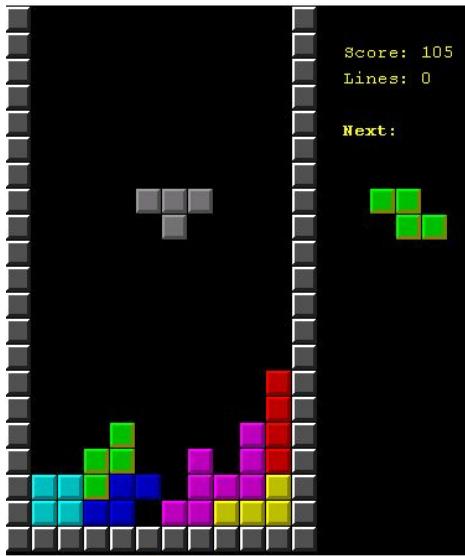
Informed Search: Not all *goals* are created equal

- What is my goal anyway? Short- & long-term?



Informed Search: Not all *goals* are created equal

- What is my goal anyway? Short- & long-term?
- *I am informed* about the next piece then what?



Greedy Search: Best First Search

Best in the sense of your *heuristic function*.

- Expand the *most opportunistic* node next: i.e. the node that is evaluated best based on the *heuristic function*.
- Reminds uniform cost search (most opportunistic vs cheapest)
- Incomplete and not optimal
- very deep or infinite depth search trees are problematic
- good choice of $h(n)$ might substantially improve performance
- bad choice of $h(n)$ might trigger a death-roll

Greedy: with *caution* → *Educated Greed*

Best of Uniform-cost & Greedy Search:

- Uniform-cost search:
Optimal and complete but not goal oriented
 $g(\mathbf{n}) = f(\mathbf{n}, \text{initial_state})$
- Greedy Search:
Neither optimal nor complete but goal oriented
 $h(\mathbf{n}) = f(\mathbf{n}, \text{goal_state})$

A* search : Best of two worlds

Cost function:

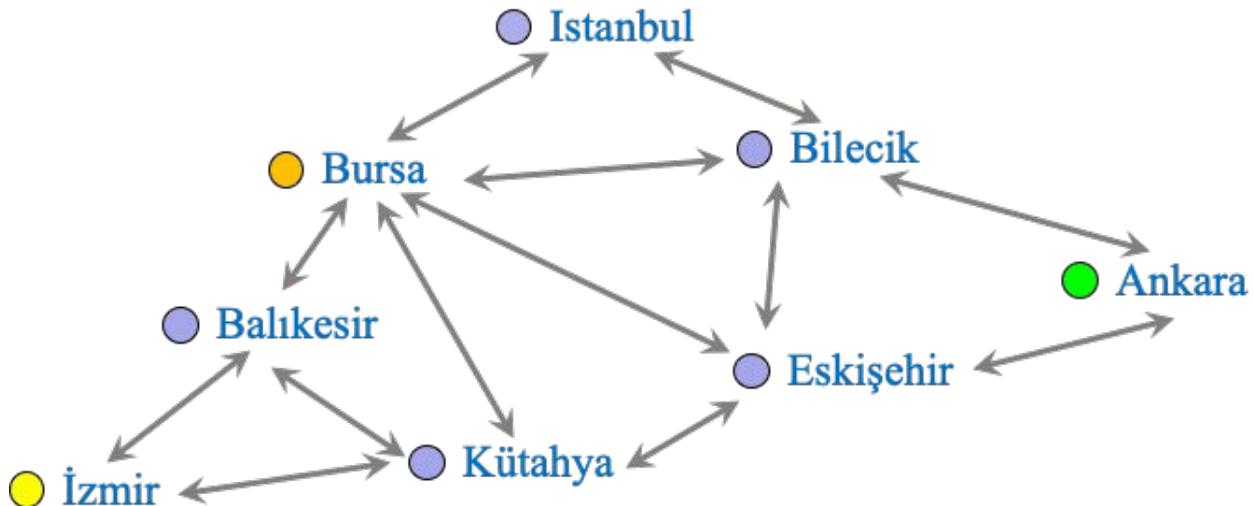
$$f(n) = g(n) + h(n)$$

For an admissible heuristic $h(n)$ solution is optimal

$h(n)$ is admissible if it never over-estimates the cost to goal

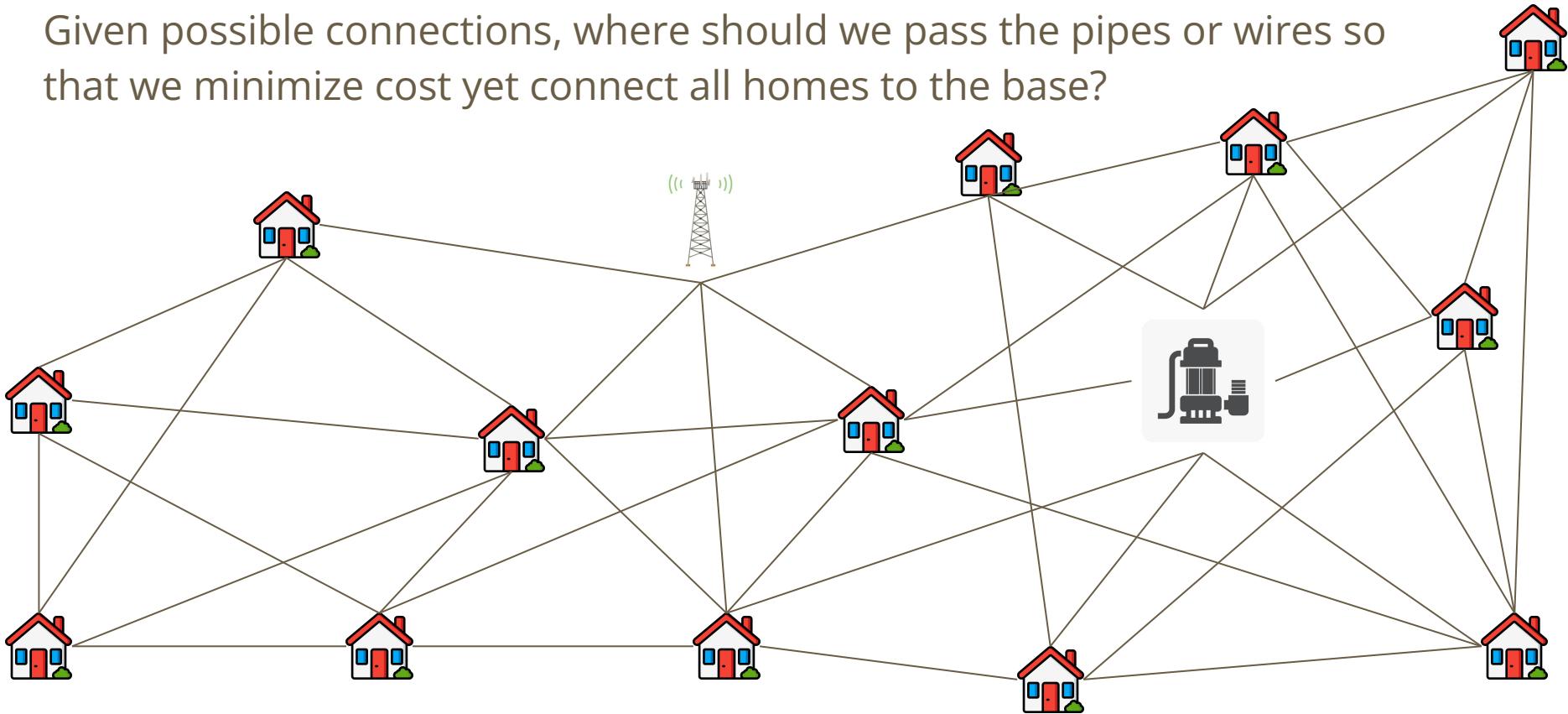
Recall: Not always have a tree!

- How to search over a graph?
- Bursa to Ankara? Which way?
- Trees emerge!



Let's pump it or wire it up? Directed vs Undirected?

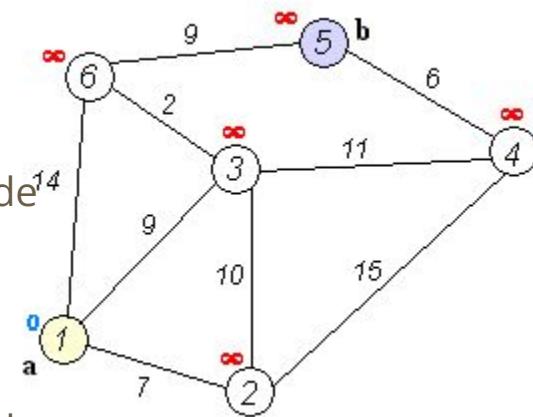
Given possible connections, where should we pass the pipes or wires so that we minimize cost yet connect all homes to the base?



Famous *di*-Graph Search Methods: $G=\{V,E,w\}$

All work on weighted *directed*-graphs

- Dijkstra's - time complexity $O(|V| \cdot \log |V| + |E|)$
 - Find shortest path to all other nodes from the starting node
 - No negative edge costs are allowed
 - Very much like uniform-cost **search on a tree**
- Bellman-Ford - time complexity $O(|V| \cdot |E|)$
 - Find shortest path to all other nodes from the starting node
 - Negative edge costs are welcomed!
 - Negative cycles are NOT!
- Floyd Warshall - time complexity $O(V^3)$
 - Find shortest path between any two nodes
 - Negative edge costs are welcomed!



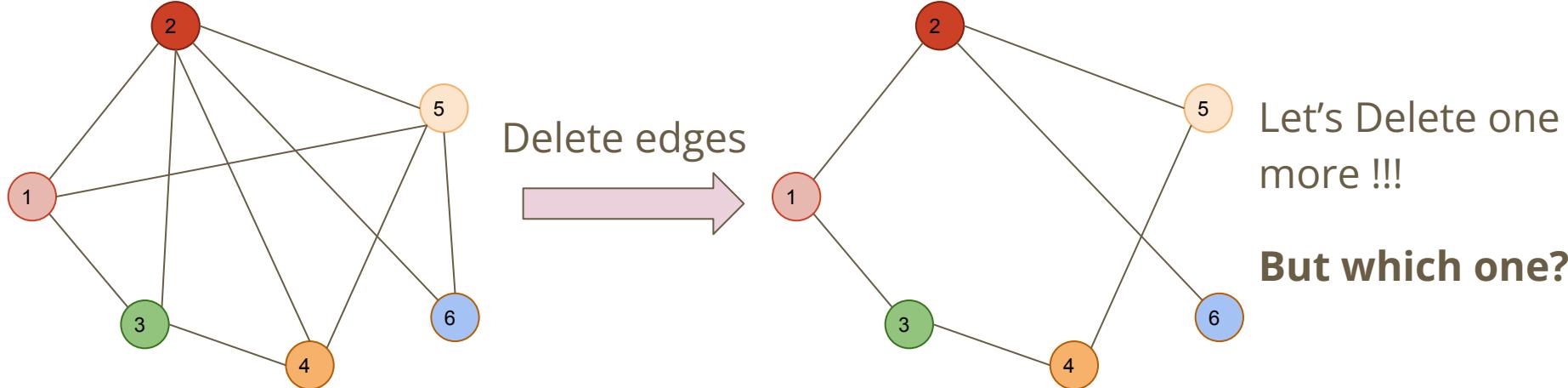
Details left for self study...

Tree from a Graph

Can you extract a tree from a graph $G=\{V,E\}$?

A spanning tree is a sub-graph containing all nodes without cycles

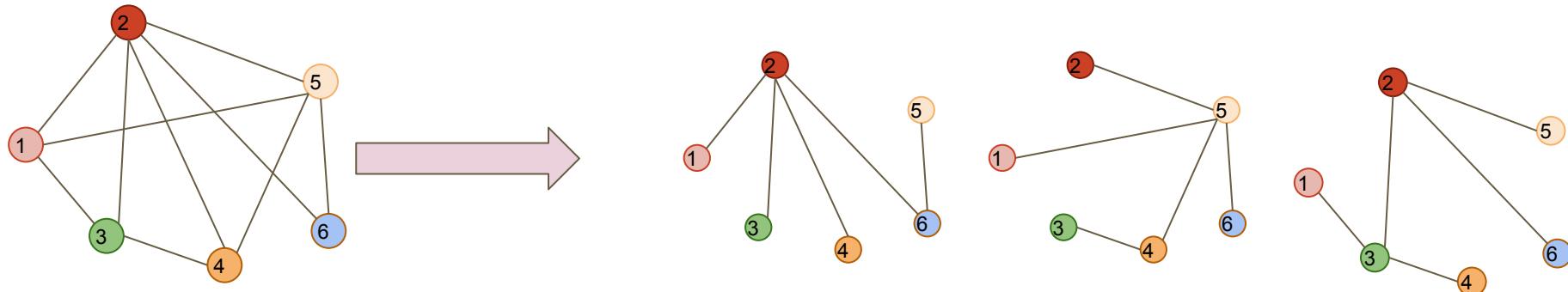
Number of edges in the spanning tree = $|V|-1$



Forest from a Graph

Can you extract a tree from a graph $G=\{V,E\}$?

Many spanning trees are possible!

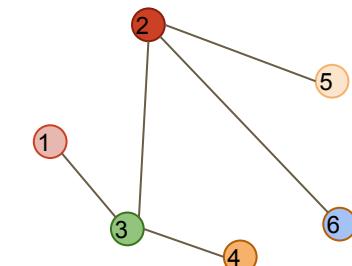
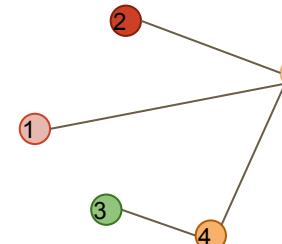
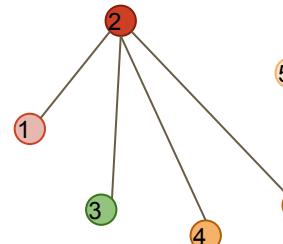
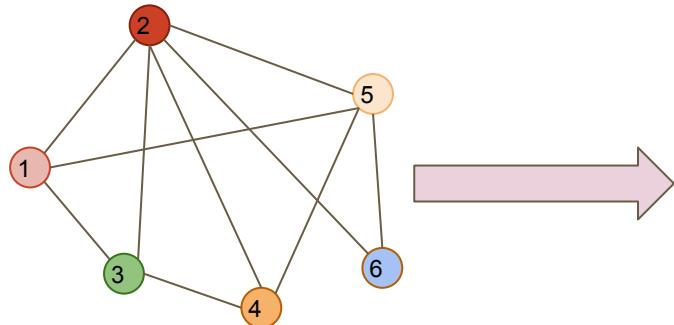


Forest from a Graph : Which one has minimum cost?

Can you extract a tree from a graph $G=\{V,E,w\}$?

Generate all trees and select one?

Or better?



Forest from a Graph: cheapest ones in $G=\{V,E,w\}$

A minimum (minimal) spanning tree (MST) is a spanning tree with minimal total edge cost

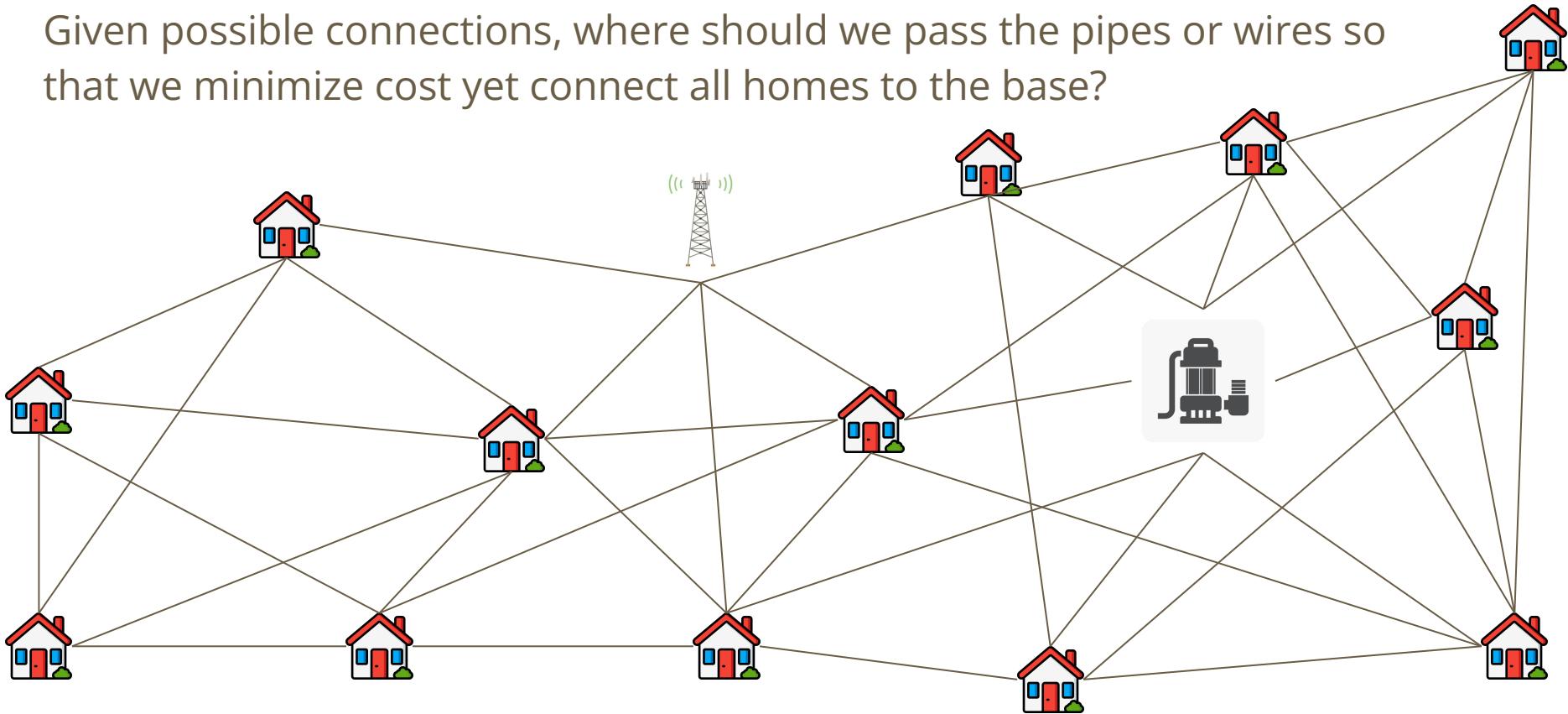
MST is not necessarily unique

- Undirected
 - Prim's → better on dense graphs - feels like uniform cost search
 - Kruskal's → better on sparse graphs - feels like greedy search
- Directed
 - Edmonds'

Details left for self study...

Can there be other trees? Other than MST?

Given possible connections, where should we pass the pipes or wires so that we minimize cost yet connect all homes to the base?



How about you are not the only actor?

- Multiplayer games
- Actions are probabilistic
- Closed-World assumption totally fails
- ...

Minimax: Minimum damage control

At any point in the game it is A's turn

A can take 3 actions

In return B can take 4 actions

In the Payoff matrix: + is gain, - is loss

Which action should A choose to minimize potential damage / loss?

| | B ₁ | B ₂ | B ₃ | B ₄ |
|----------------|----------------|----------------|----------------|----------------|
| A ₁ | 5 | -2 | -1 | 0 |
| A ₂ | -3 | 4 | 1 | -2 |
| A ₃ | 9 | -5 | 4 | -3 |

Payoff Matrix

What if B is not that clever?

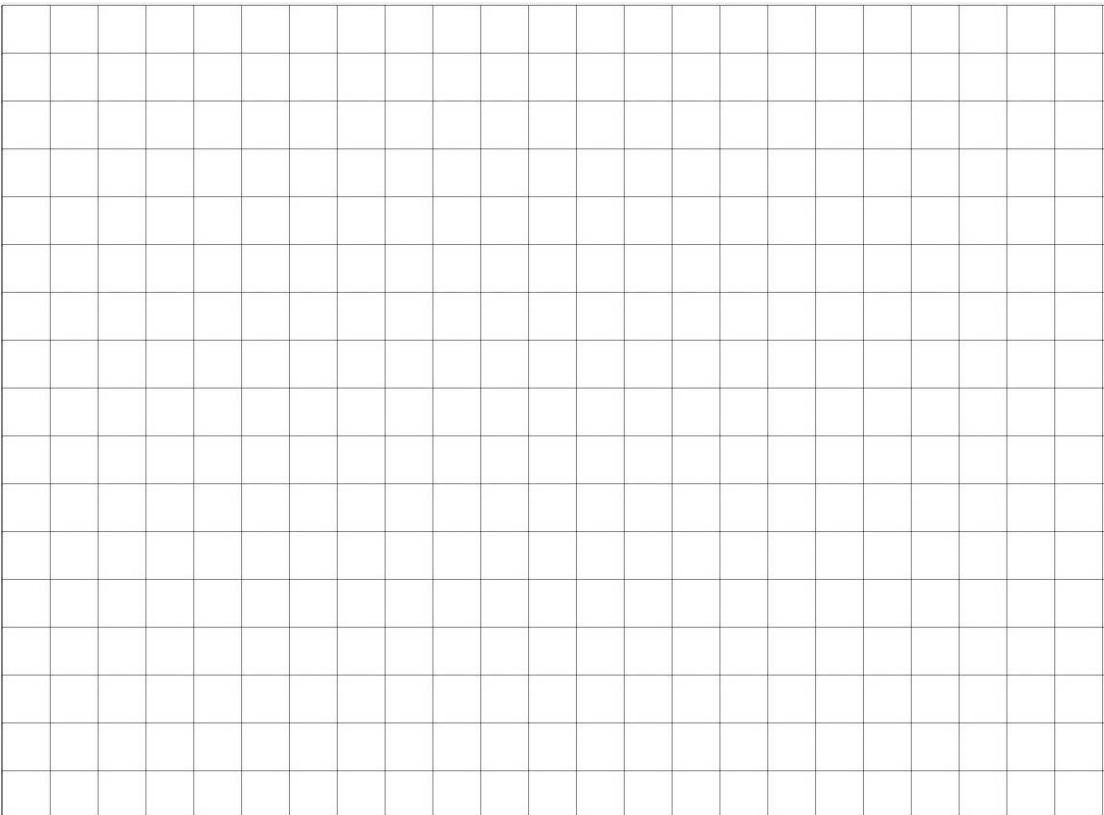
Perhaps: Assign probabilities to each possible action by B?

Design & Search: Search to Design

Search in a high dimensional space

States are not easily quantifiable

Tools of search might be useful in design



to be continued...

ME 536

— Week 12: How deep is your love^{*} *to learn ?* —

* old times were very funny... what do you think today will look like in 45 years?

Learning:

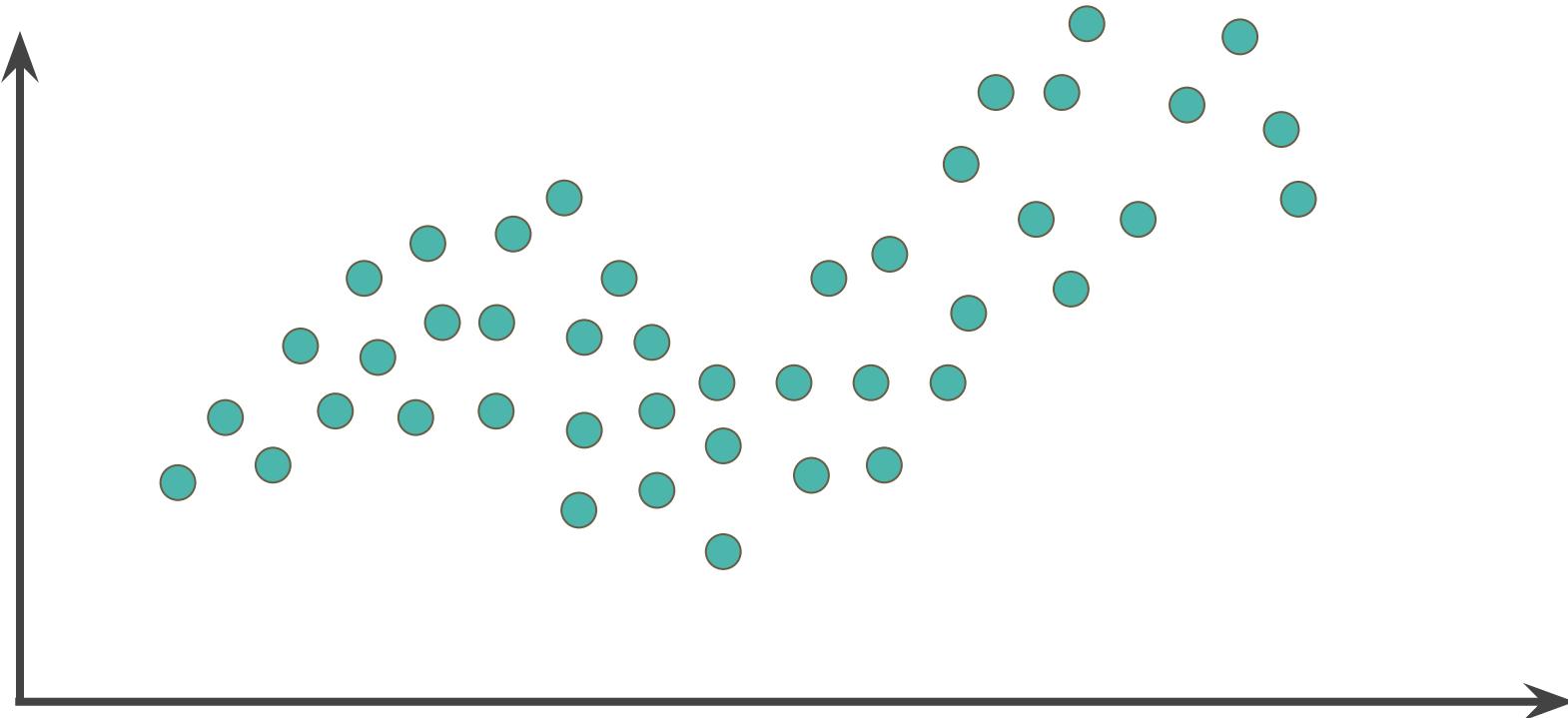
What to learn ?

How to learn ?

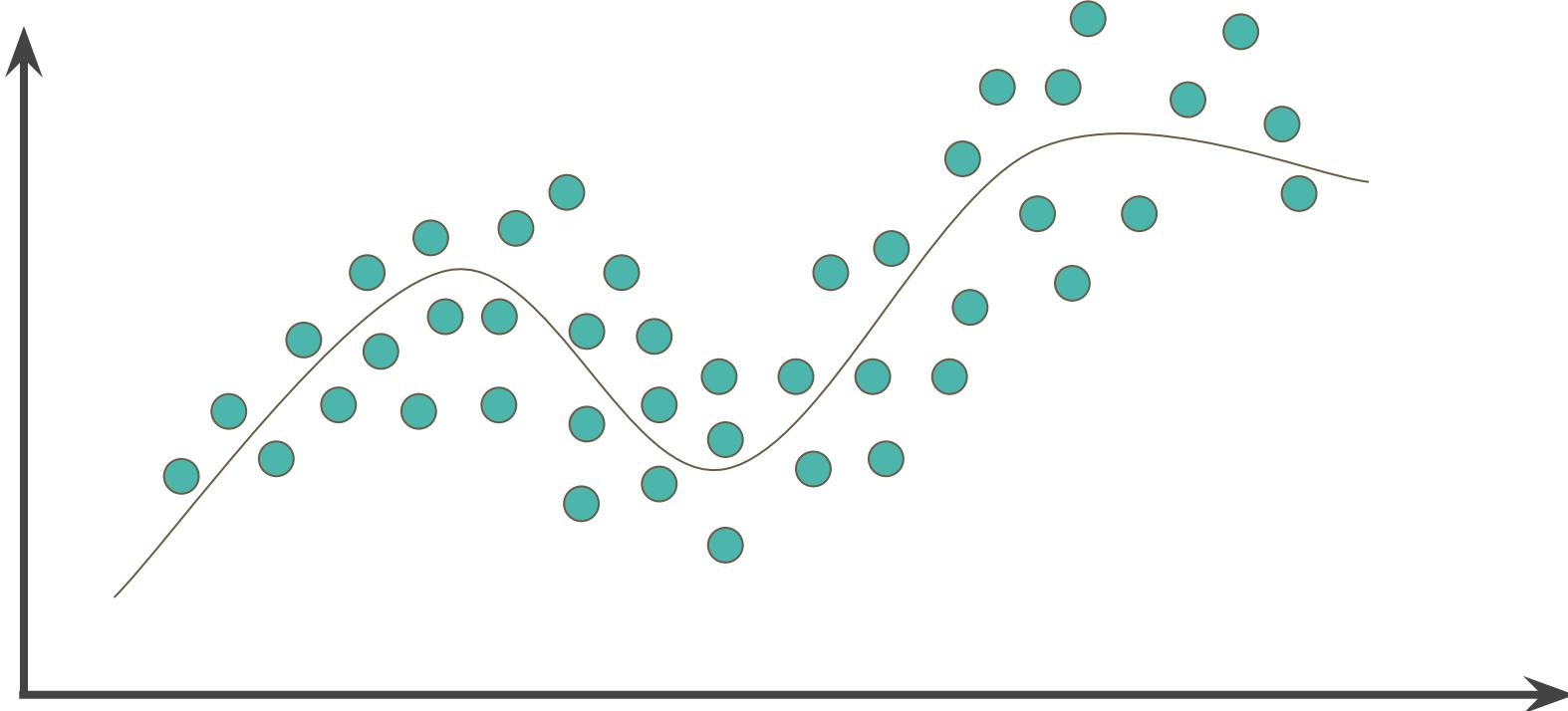
Learn once or life-long learning ?

...

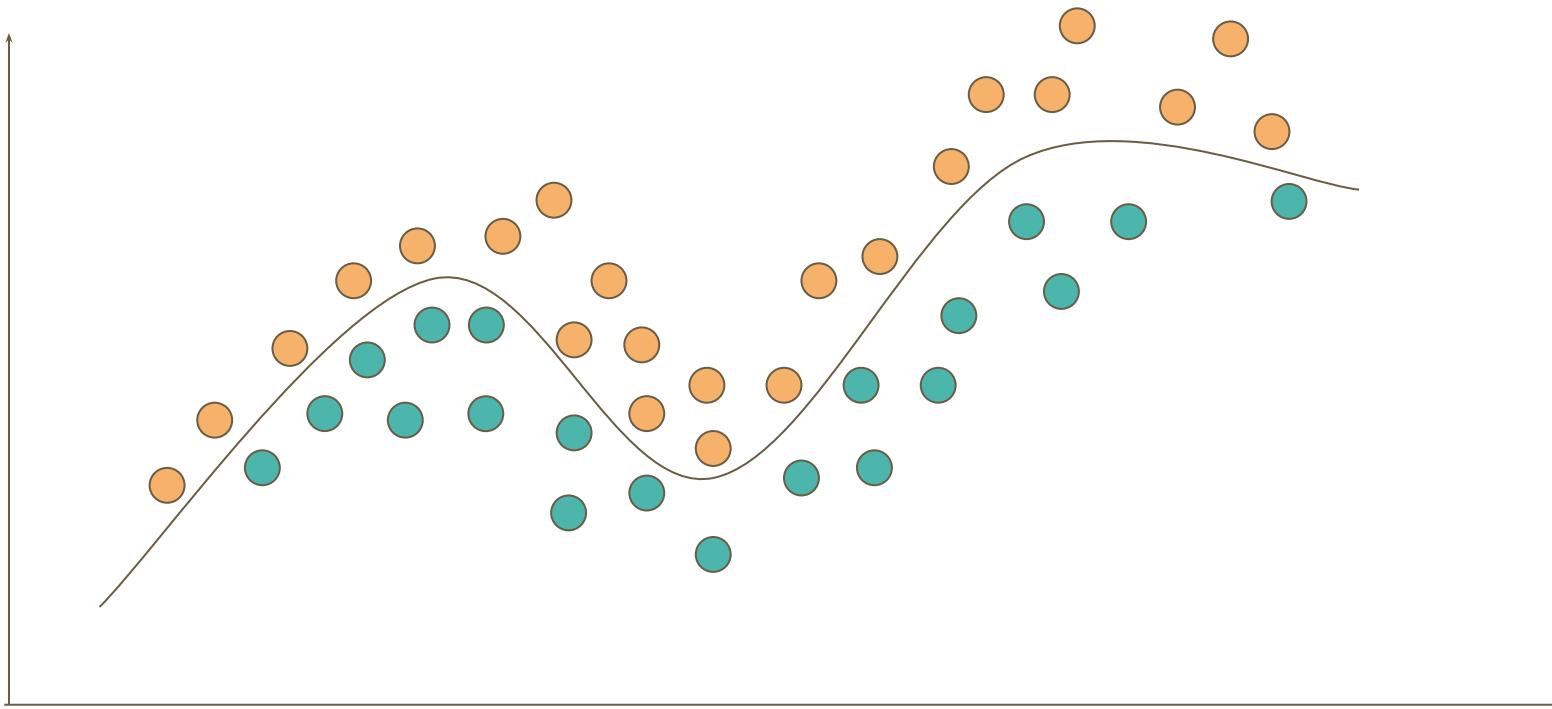
DATA



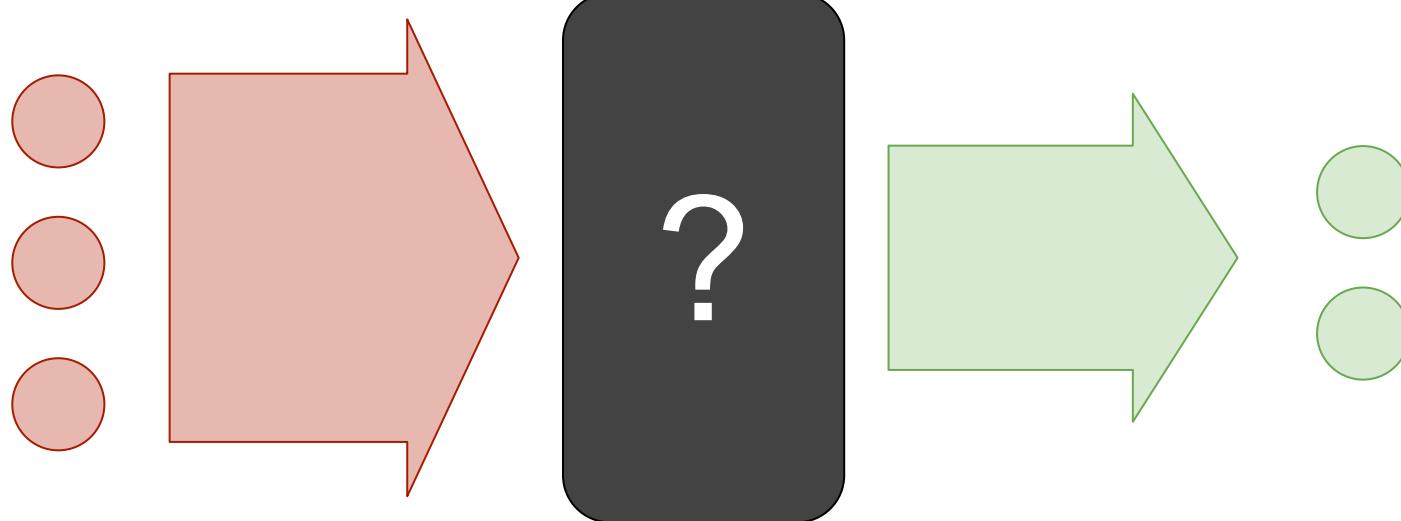
Regression



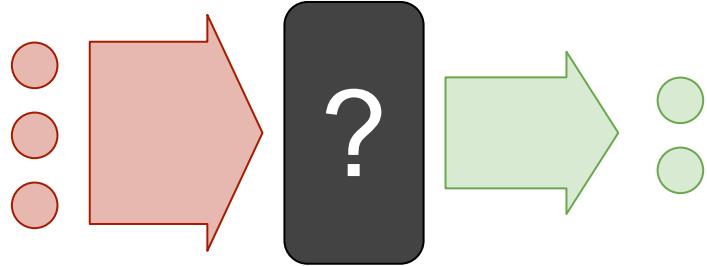
or Classification?



A black box: In general

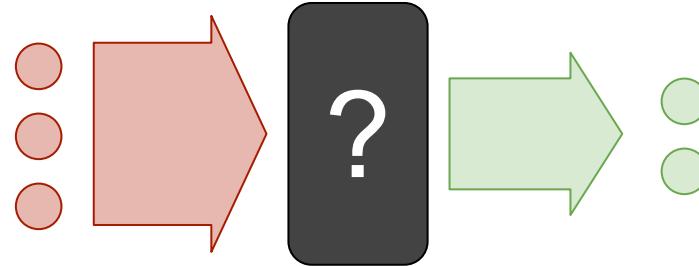


A black box: In general



- Data needed
- Type of model available ?
 - What if model is not known at all?
 - What if the model is hard to derive?
- What if all inputs do not propagate at the same speed to output?
- What if data is available?
 - What if whole input spectrum is not covered in data?
- ...
- Does nature help?

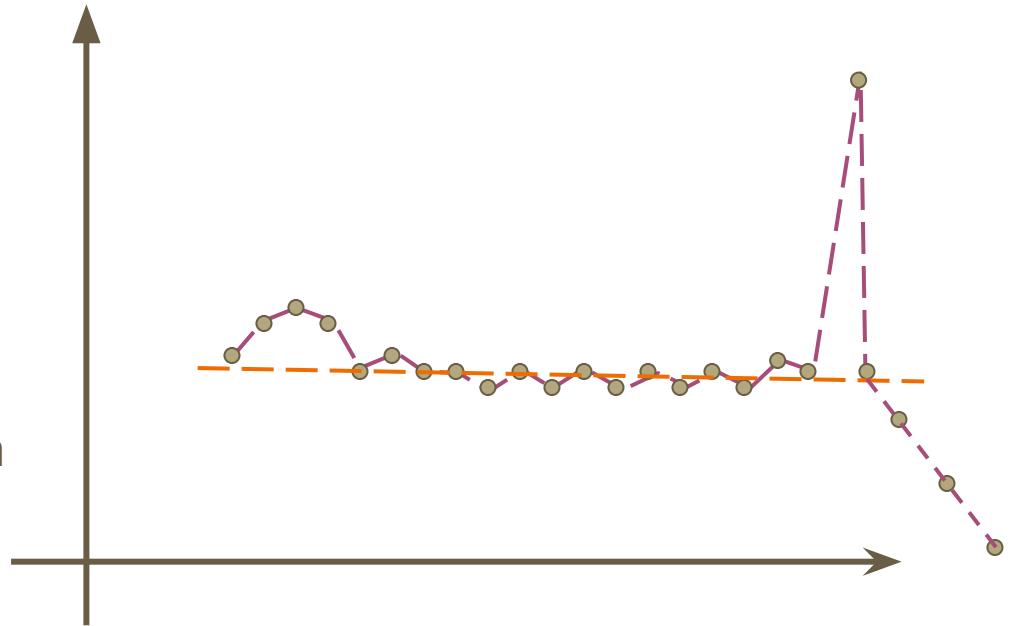
A black box: In general



While blackbox is identified:

- Can it generalize?
- Over- / under-fitting ... /learning

If this is from the step response of a second order system? Why use an ANN?



Rules of thumb



- Do not expect **miracles** from ANN and do not **blindly** use them.
- Direct use of ANNs without prior analysis might be **more costly** than expected.
- If you have **a good understanding of the model** why not identify the parameters and use the model?
- Analysis of why **ANNs misbehave** is tricky
- **Best ANN topology** to start with is **not necessarily known** given the problem.
- If you have partial prior **understanding of your model**, try to inject into the ANN if possible.

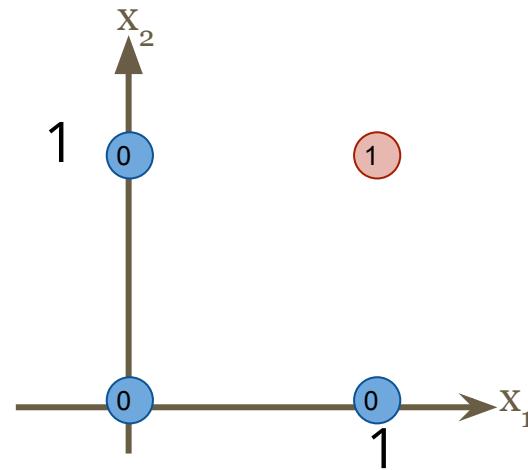
A simple case: logic AND

$\&$

| | | x_1 |
|-------|---|-------|
| | | 0 |
| x_2 | 0 | 0 |
| | 1 | 0 |

Just a line is enough...

Isn't it?



A simple case: perceptron w/o bias

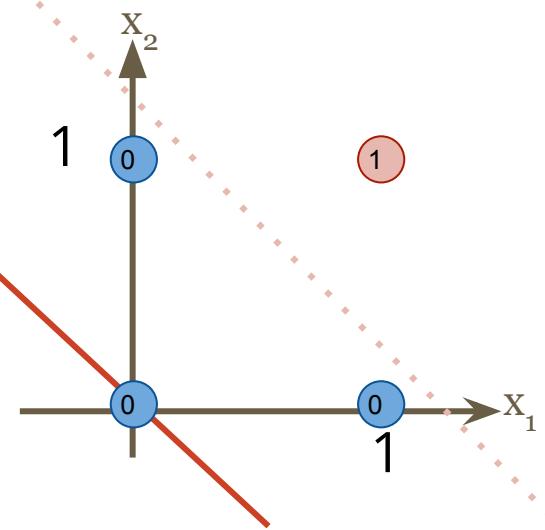
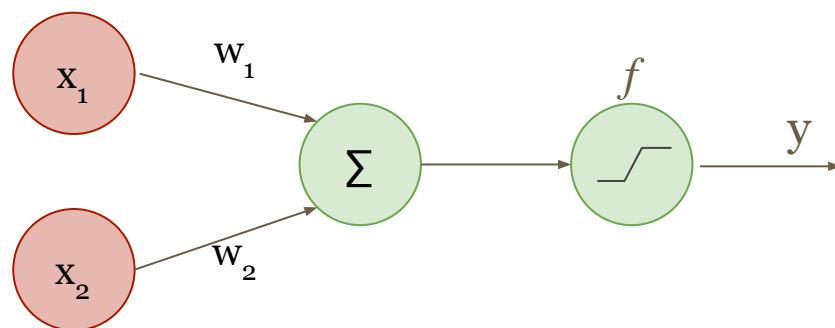
&

| | | x_1 |
|-------|---|----------|
| | | 0 1 |
| x_2 | 0 | 0 0 |
| | 1 | 0 1 |

Just a line is enough...

Isn't it? or how about a scalar field?

$$y = f(x_1 w_1 + x_2 w_2)$$



A simple case: perceptron

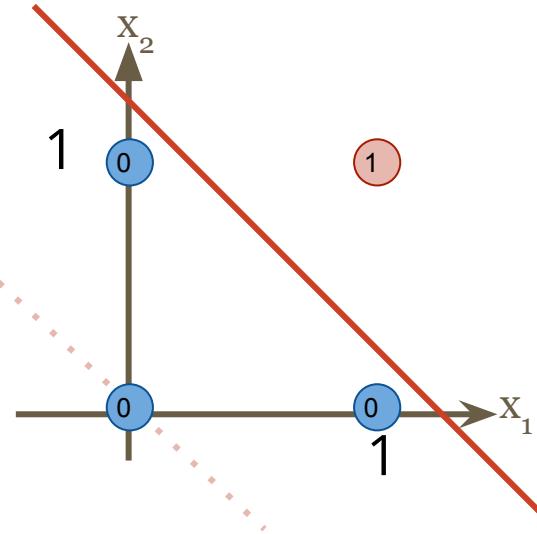
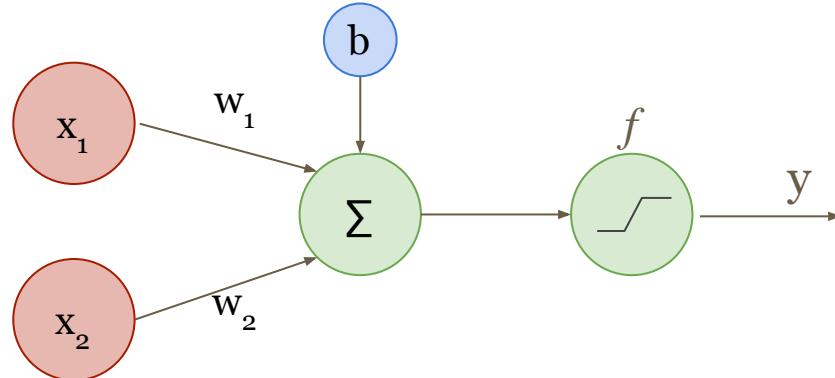
&

| | | x_1 | |
|-------|---|-------|---|
| | | 0 | 1 |
| x_2 | 0 | 0 | 0 |
| | 1 | 0 | 1 |

Just a line is enough...

Isn't it? or how about a scalar field?

$$y = f(x_1 w_1 + x_2 w_2 + b) \quad \text{where is the } y \text{ axis?}$$



A simple case: More compact form - dimension free

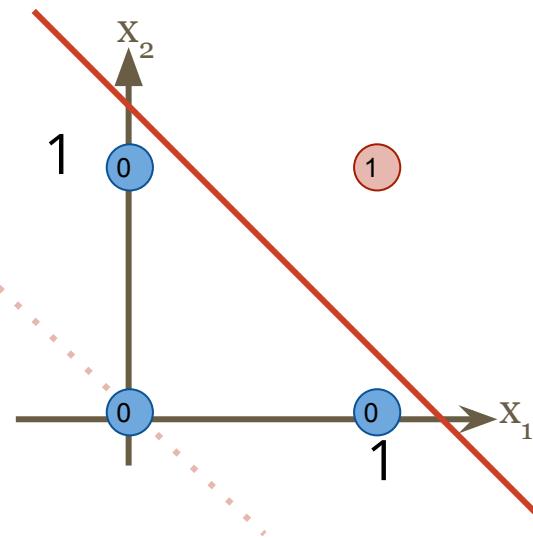
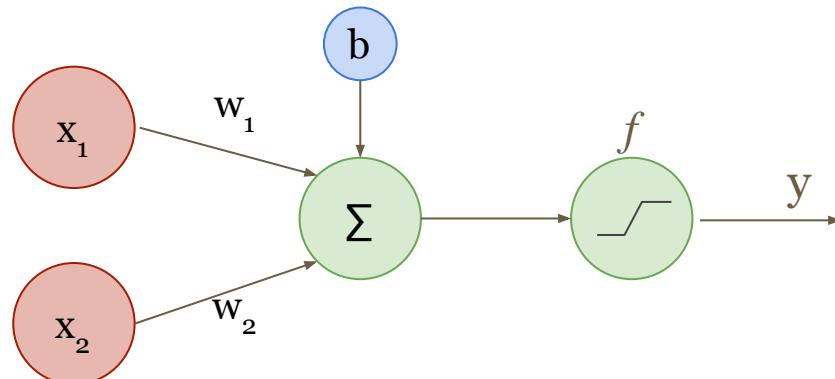
&

| | | x_1 |
|-------|---|-------|
| | | 0 |
| x_2 | 0 | 0 |
| | 1 | 1 |

$$y = f(x_1 w_1 + x_2 w_2 + b)$$

Let $\mathbf{x}^T = [x_1 \ x_2]$, $\mathbf{w}^T = [w_1 \ w_2]$ - dimension of \mathbf{x} , \mathbf{w} does not matter

$$y = f(\mathbf{x}^T \mathbf{w} + b)$$



A simple case: Alternative form

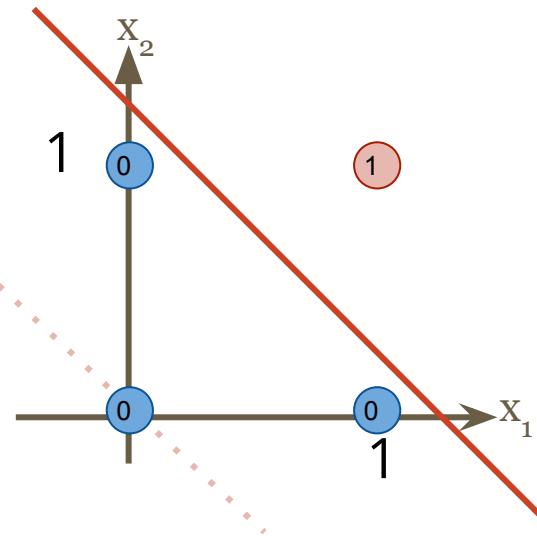
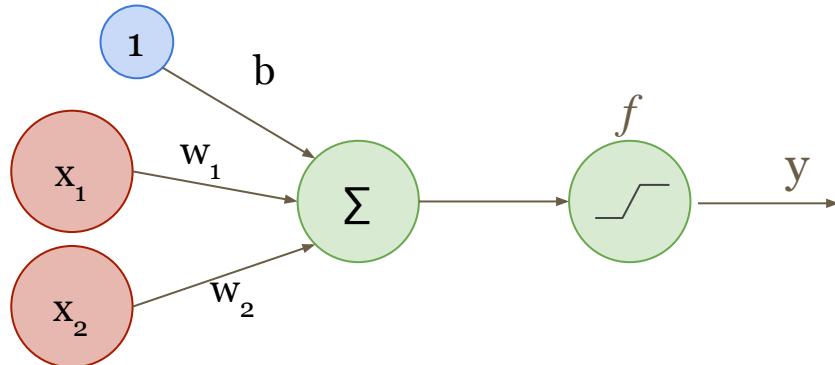
&

| | | x_1 |
|-------|---|-------|
| | | 0 |
| x_2 | 0 | 0 |
| | 1 | 1 |

$$y = f(x_1 w_1 + x_2 w_2 + b)$$

Let $\mathbf{x}^T = [1 \ x_1 \ x_2]$, $\mathbf{w}^T = [b \ w_1 \ w_2]$

$$y = f(\mathbf{x}^T \mathbf{w})$$



A simple case: How to initialize w_i ?

| | | |
|-------|---|-------|
| | | x_1 |
| | 0 | 1 |
| x_2 | 0 | 0 |
| 1 | 0 | 1 |

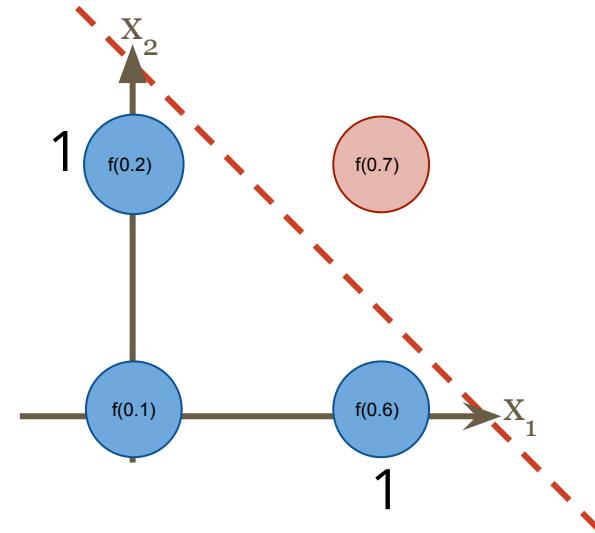
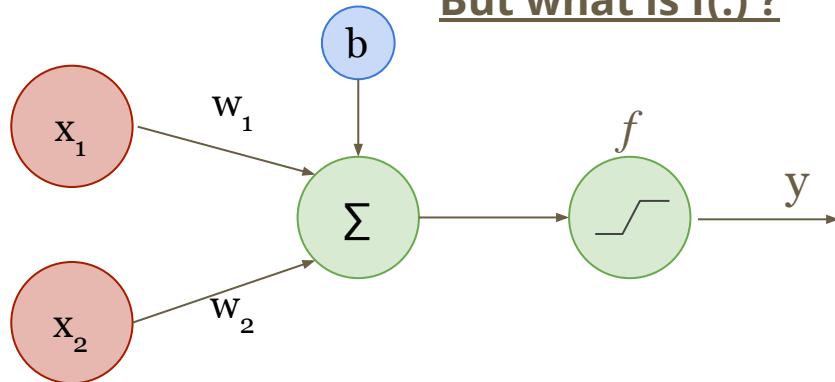
Random sounds good in general,

so let: $\mathbf{w}^T = [0.5 \ 0.1]$, $b = 0.1$

$$y = f(\mathbf{x}^T \mathbf{w} + b) = f(0.5x_1 + 0.1x_2 + 0.1)$$

What should $f(\cdot)$ return?

But what is $f(\cdot)$?



A simple case: How lucky can we get?

Random sounds good in general, so let: $\mathbf{w}^T = [0.5 \ 0.1]$, $b = 0.1$

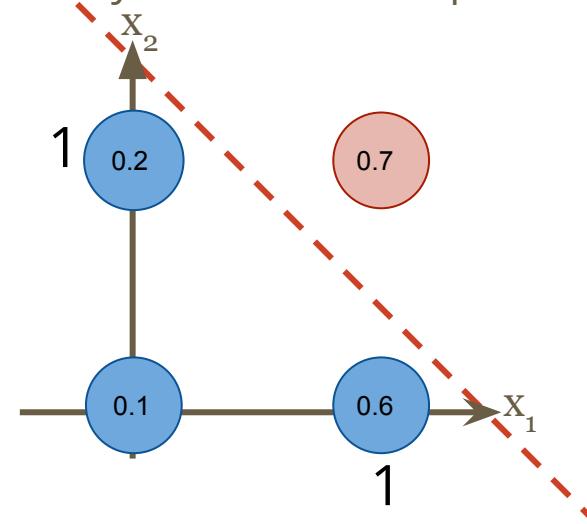
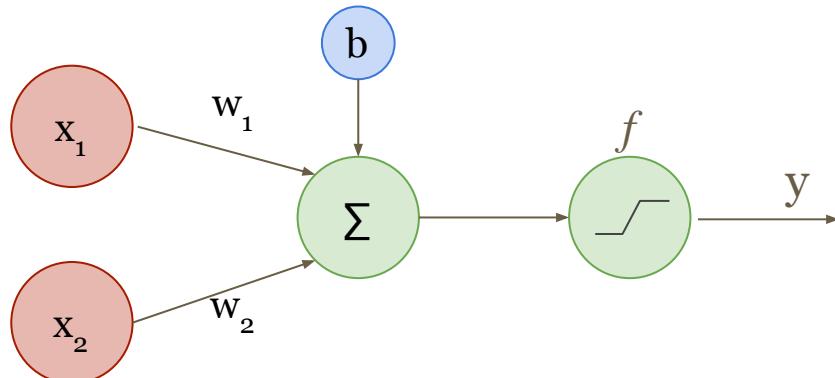
| | |
|-------|-------------------------|
| | x_1 |
| & | $\overbrace{0 \quad 1}$ |
| x_2 | { 0 0 0 1 } |
| | 0 1 |

$$y = f(\mathbf{x}^T \mathbf{w} + b) = f(0.5x_1 + 0.1x_2 + 0.1)$$

What if $f(x) = x$

$$y = 0.5x_1 + 0.1x_2 + 0.1$$

Will simple case work here? May be with a bit of post-work?



A simple case: Which f ?

Let: $\mathbf{w}^T = [0.5 \ 0.1]$, $b = 0.1$

&

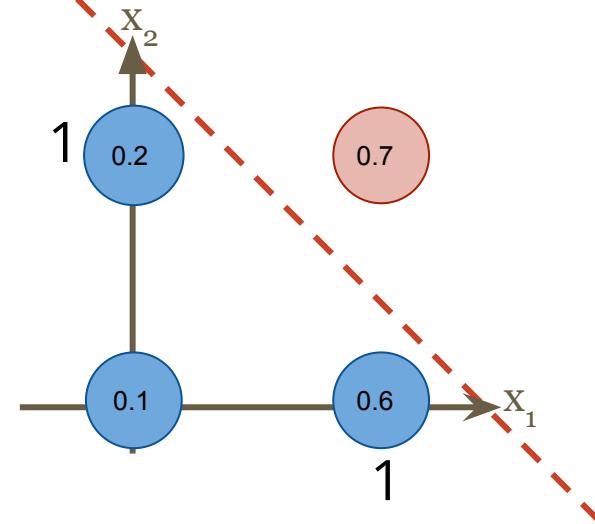
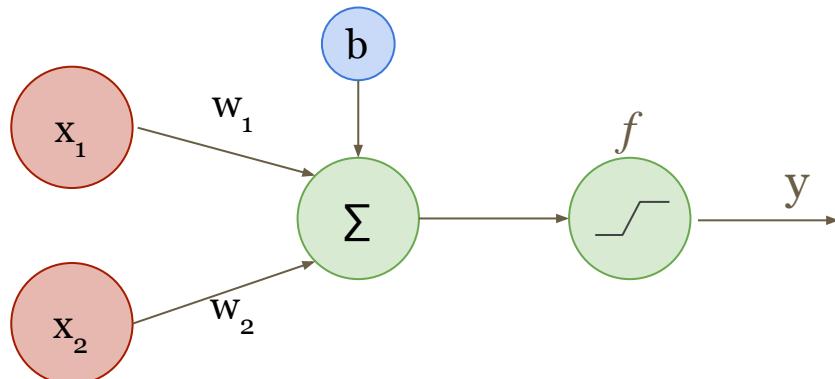
| | | x_1 |
|-------|---|----------|
| | | 0 1 |
| x_2 | 0 | 0 0 |
| | 1 | 0 1 |

$$y = f(\mathbf{x}^T \mathbf{w} + b) = f(0.5x_1 + 0.1x_2 + 0.1),$$

simple case $f(x) = x$,

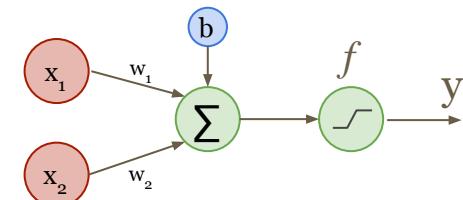
$$y = 0.5x_1 + 0.1x_2 + 0.1$$

Check out <https://www.geogebra.org/3d/dbqykxwg>



In general: Which f ?

Check out: <https://www.geogebra.org/calculator/kzexwpwz>



Sigmoid

$$f(x) = \frac{1}{1+e^{-x}}$$

$$f'(x) = f(x)(1 - f(x))$$

Tangent Hyperbolic

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

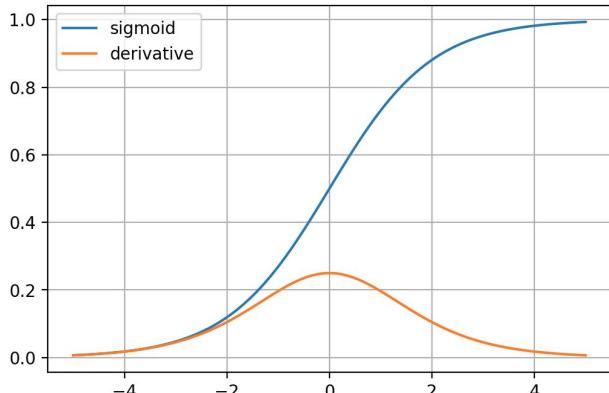
$$f'(x) = 1 - f(x)^2$$

Rectified Linear Unit: a.k.a ReLU

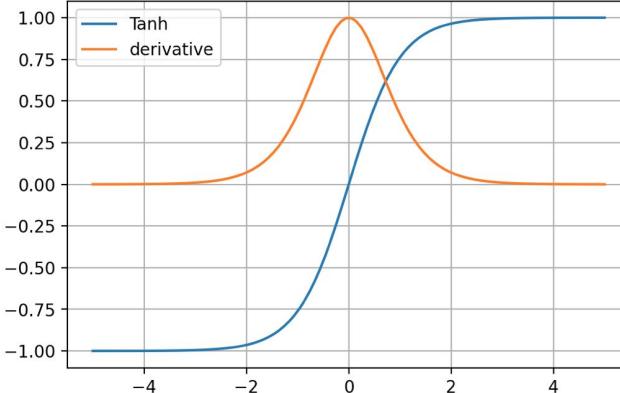
$$f(x) = \max(0, x)$$

$$f'(x) = \begin{cases} 1, & \text{for } x > 0 \\ 0, & \text{otherwise} \end{cases}$$

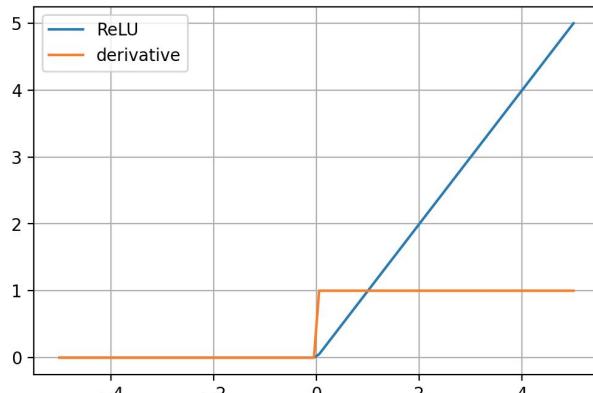
Sigmoid and its derivative



Tanh and its derivative



ReLU and its derivative



A simple case: Which f is ?

Let: $\mathbf{w}^T = [0.5 \ 0.1]$, $b = 0.1$

&

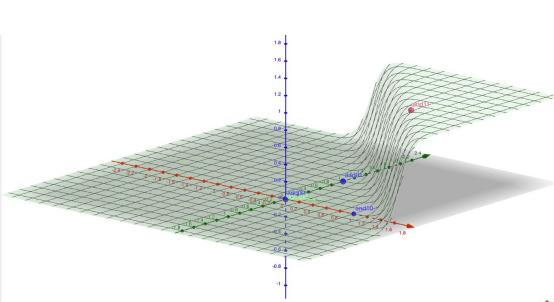
| | |
|-------|-------|
| | x_1 |
| x_2 | { |
| 0 | 0 |
| 1 | 0 |
| | 1 |

$$y = f(\mathbf{x}^T \mathbf{w} + b) = f(0.5x_1 + 0.1x_2 + 0.1),$$

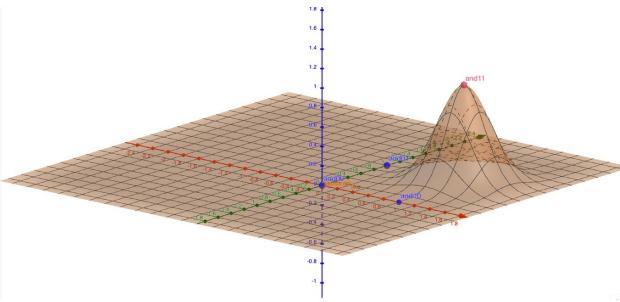
sigmoid case $f(x) = (1+e^{-x})^{-1}$,

Check out <https://www.geogebra.org/3d/dbqykxwg>

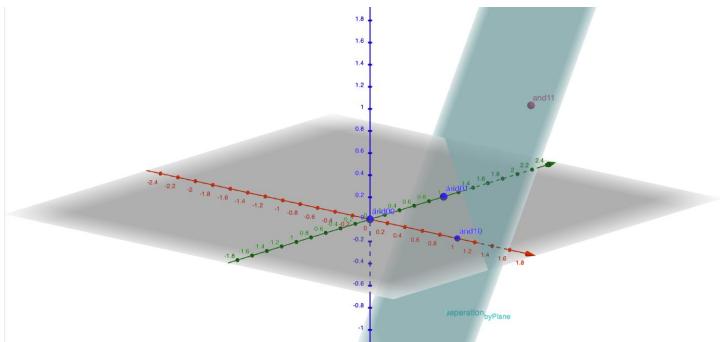
Sigmoid



Gaussian



Plane



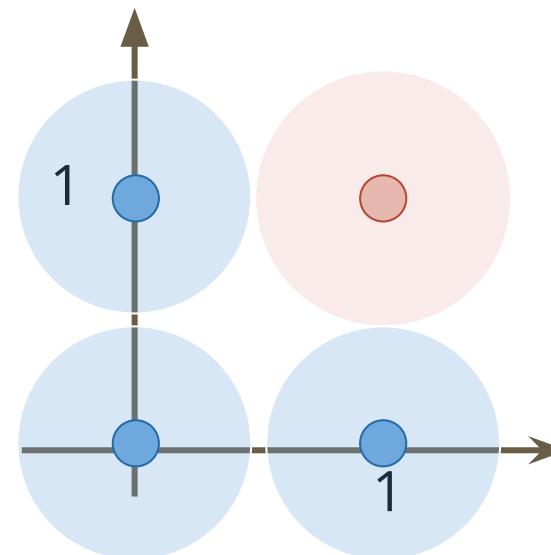
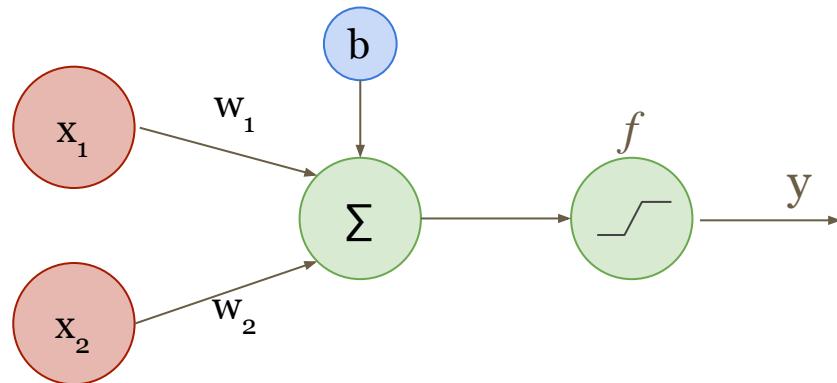
A simple case: How to generate training data?

| | | |
|---|---|-------|
| | | x_1 |
| | 0 | 1 |
| 0 | 0 | 0 |
| 1 | 0 | 1 |

$$y = f(x_1 w_1 + x_2 w_2 + b)$$

$$\text{Let } \mathbf{x}^T = [x_1 \ x_2], \mathbf{w}^T = [w_1 \ w_2]$$

$$y = f(\mathbf{x}^T \mathbf{w} + b)$$



| x_1 | x_2 | y_t |
|-------|-------|-------|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |
| ? | ? | ? |
| ... | ... | ... |

A simple case: How to generate training data?

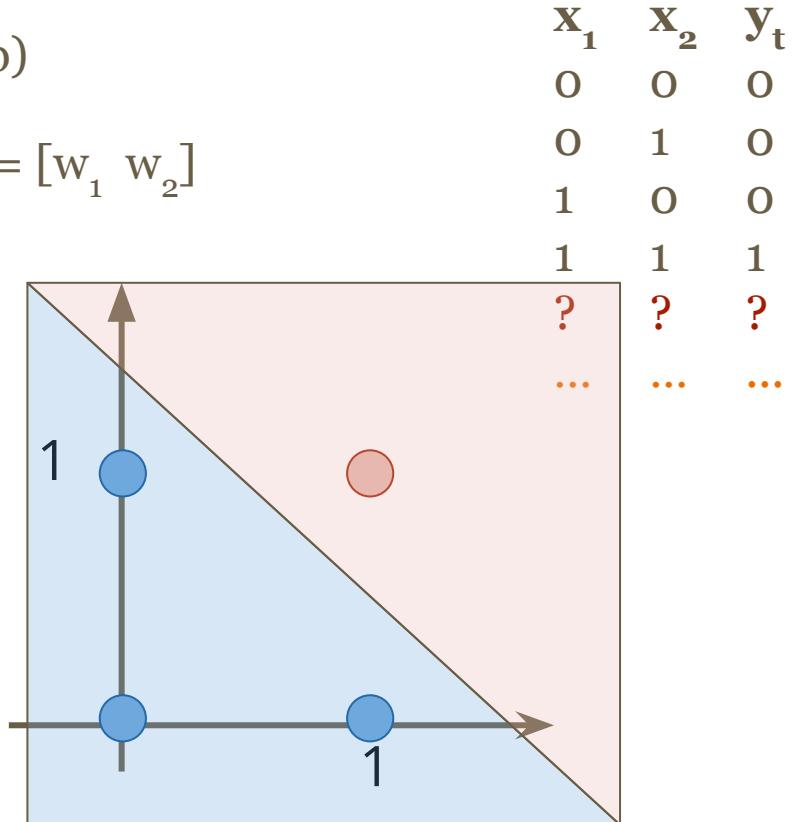
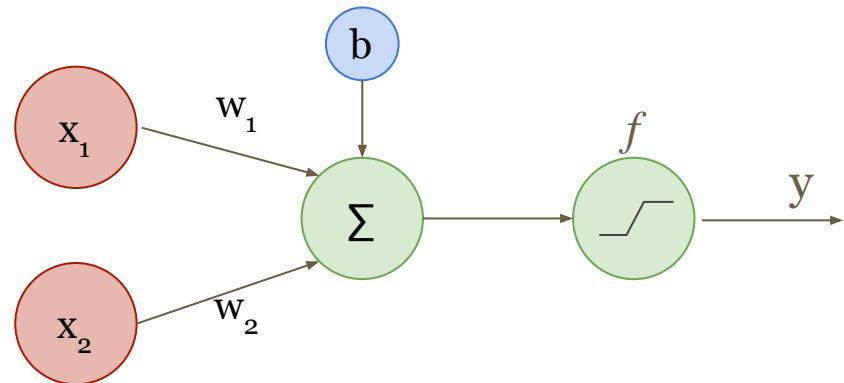
&

| | | x_1 |
|-------|---|-------|
| | | 0 |
| x_2 | 0 | 0 |
| | 1 | 1 |

$$y = f(x_1 w_1 + x_2 w_2 + b)$$

Let $\mathbf{x}^T = [x_1 \ x_2]$, $\mathbf{w}^T = [w_1 \ w_2]$

$$y = f(\mathbf{x}^T \mathbf{w} + b)$$



A *not so simple* case: when one line ain't enough

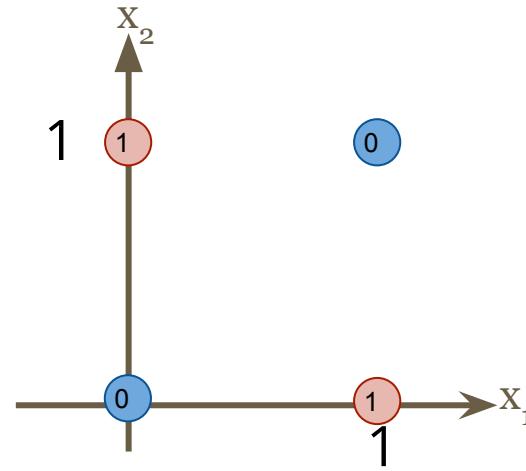
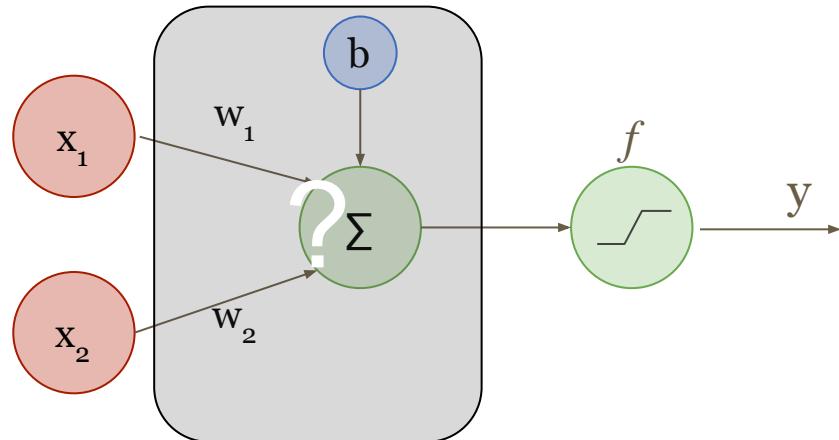
XOR

| | | x_1 |
|-------|---|----------|
| | | 0 1 |
| x_2 | 0 | 0 1 |
| | 1 | 1 0 |

$$y = f(x_1 w_1 + x_2 w_2 + b)$$

Let $\mathbf{x}^T = [x_1 \ x_2]$, $\mathbf{w}^T = [w_1 \ w_2]$

$$y = f(\mathbf{x}^T \mathbf{w} + b)$$



A *not so simple* case: when one line ain't enough

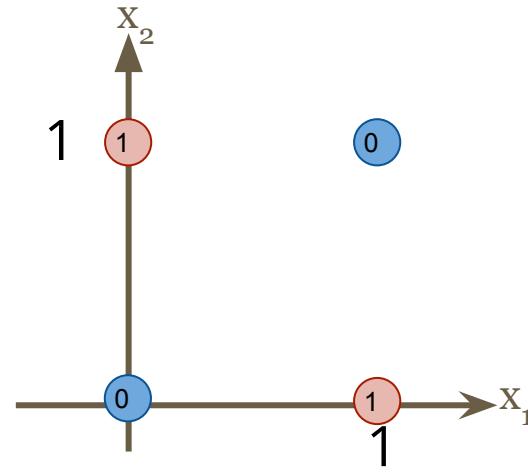
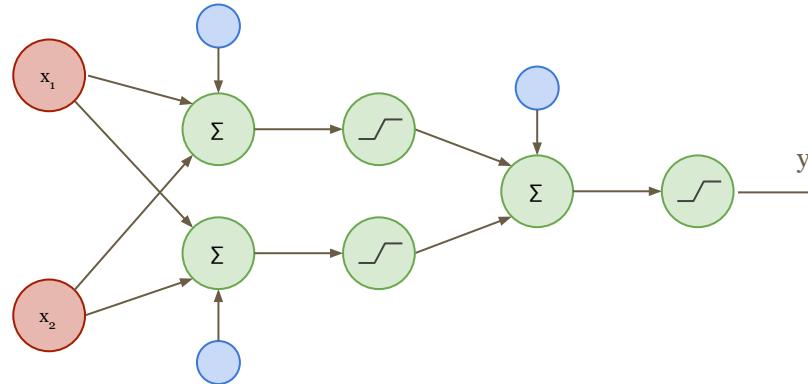
XOR

| | | x_1 |
|-------|---|----------|
| | | 0 1 |
| x_2 | 0 | 0 1 |
| | 1 | 1 0 |

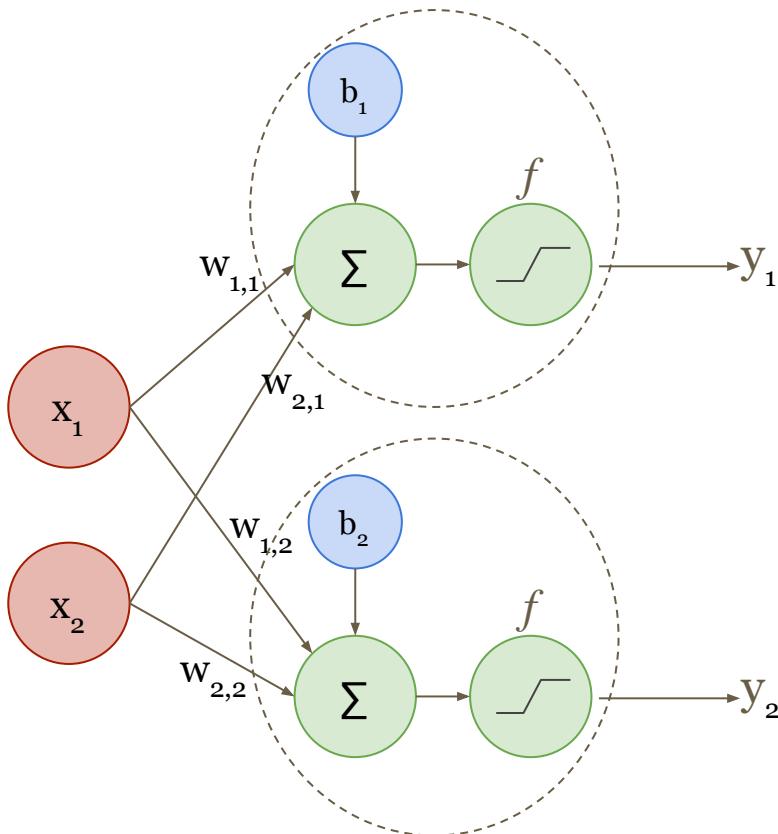
$$y = f(x_1 w_1 + x_2 w_2 + b)$$

Let $\mathbf{x}^T = [x_1 \ x_2]$, $\mathbf{w}^T = [w_1 \ w_2]$

$$y = f(\mathbf{x}^T \mathbf{w} + b)$$



A more general case: MIMO - 2x2



Let:

$$\mathbf{x}^T = [x_1 \ x_2]$$

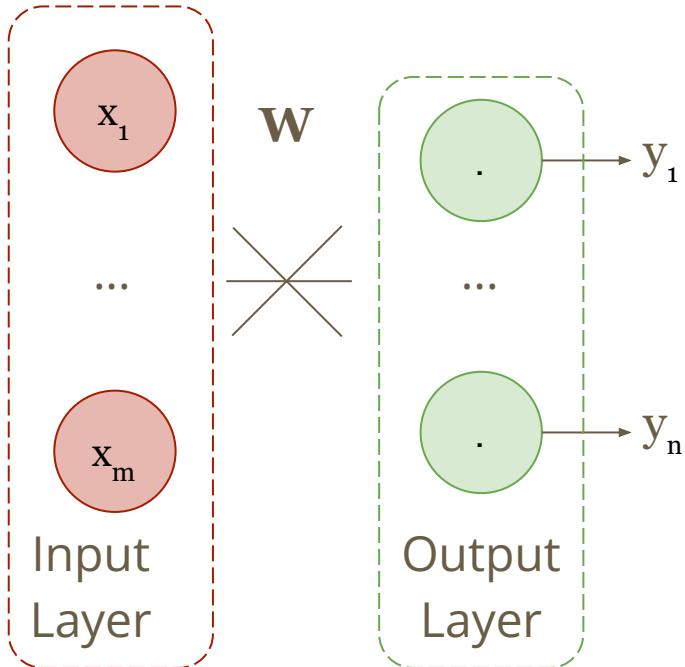
$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix}$$

$$\mathbf{b} = [b_1 \ b_2]$$

$$\mathbf{y} = [y_1 \ y_2]$$

$$\mathbf{y} = f(\mathbf{x}^T \mathbf{W} + \mathbf{b})$$

A more general case: Input & Output Layers - MIMO



Let:

$$\mathbf{x}^T = [x_1 \dots x_m]$$

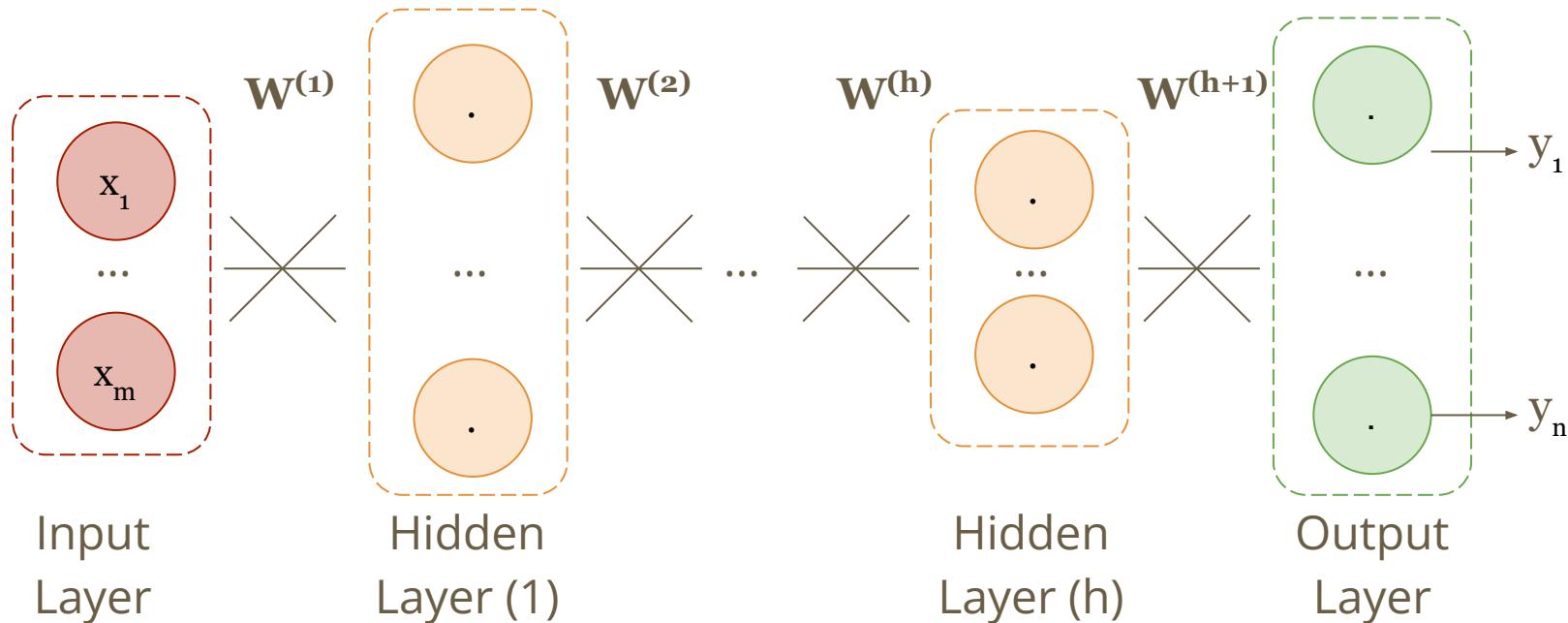
$$\mathbf{W} = \begin{bmatrix} w_{1,1} & \dots & w_{1,n} \\ \vdots & \ddots & \vdots \\ w_{m,1} & \dots & w_{m,n} \end{bmatrix}$$

$$\mathbf{b} = [b_1 \dots b_n]$$

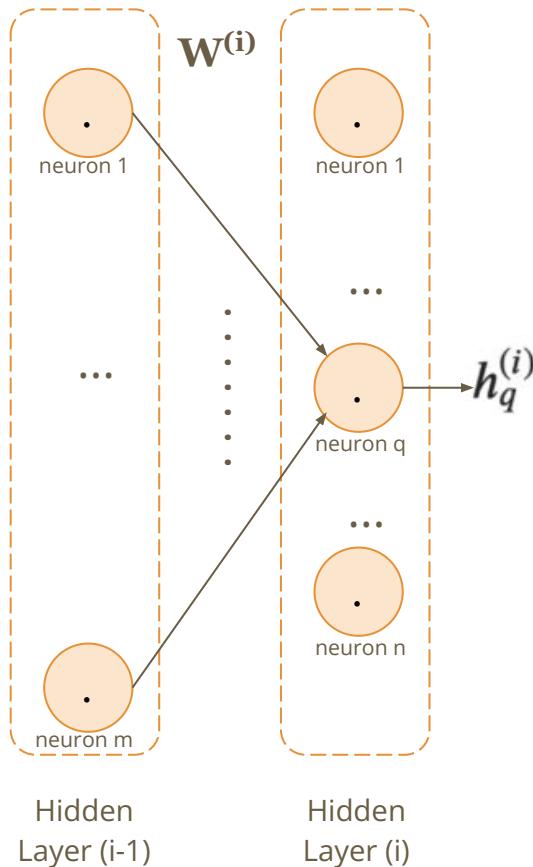
$$\mathbf{y} = [y_1 \dots y_n]$$

$$\mathbf{y} = f(\mathbf{x}^T \mathbf{W} + \mathbf{b})$$

Most general case: Shallow & Deep & Deeper



Most general case: Output of any neuron



Let hidden layers $(i - 1)$, i have m, n neurons respectively.

$h_q^{(i)}$ is the output of the q^{th} neuron at the i^{th} hidden layer.

Weight $w_{p,q}^{(i)}$ is between the q^{th} neuron at hidden layer i and p^{th} neuron at the previous layer.

Bias for neuron q at hidden layer i is $b_q^{(i)}$.

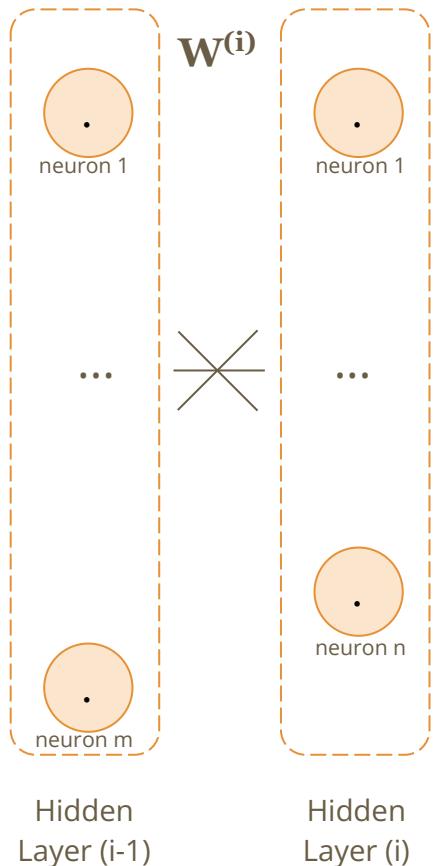
Then $h_q^{(i)}$ can be written as:

$$h_q^{(i)} = f\left(\sum_{j=1}^m [h_j^{(i-1)} w_{j,q}^{(i)} + b_q^{(i)}]\right)$$

where,

$f(\cdot)$ is the activation function.

Most general case: Output of any layer



Let $\mathbf{h}^{(i)}$ be the output of layer i , then:

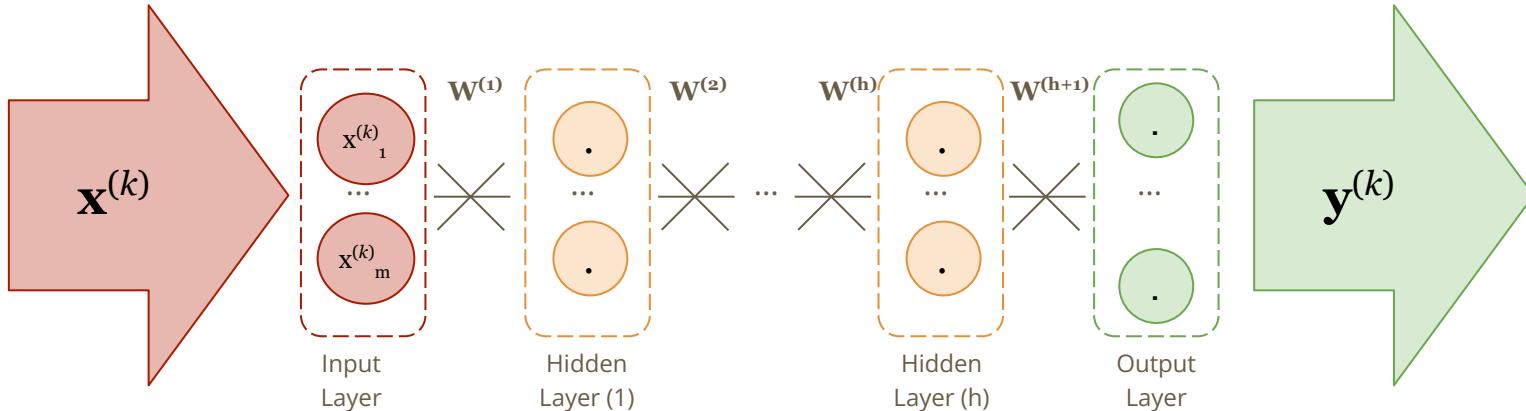
$$\mathbf{h}^{(i)} = f(\mathbf{h}^{(i-1)T} \mathbf{W}^{(i)} + \mathbf{b}^{(i)})$$

where,

$$\mathbf{b}^{(i)} = [b_1^{(i)}, \dots, b_n^{(i)}] \text{ and,}$$

$\mathbf{W}_{m \times n}^{(i)}$ is the weight matrix between layers $(i - 1)$ and i .

Most general case: Output of the network



Note that, in general there will be several inputs, so let there be d many data points $\mathbf{x}^{(k)}$ as input and $\mathbf{y}^{(k)}$ is the corresponding network *prediction/output*, where $k = 1, \dots, d$.

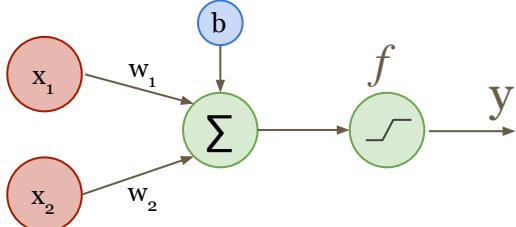
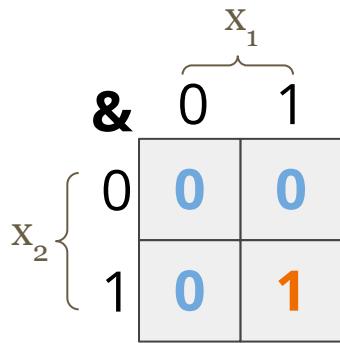
$$\mathbf{y}^{(k)} = f\left(f(\dots f(f(\mathbf{x}^{(k)T} \mathbf{W}^{(1)} + \mathbf{b}^{(1)})^T \mathbf{W}^{(2)} + \mathbf{b}^{(2)})^T \dots)^T \mathbf{W}^{(h+1)} + \mathbf{b}^{(h+1)}\right)$$

In more general terms,

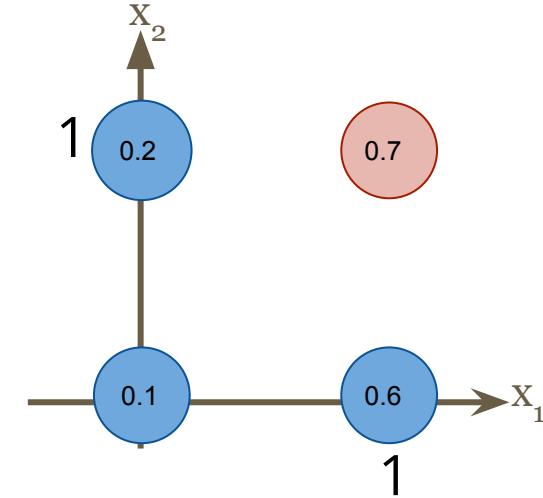
$$\mathbf{y}^{(k)} = F(\mathbf{x}^{(k)}, \mathbf{W}),$$

where $\mathbf{W} = \{\mathbf{W}^{(1)}, \dots, \mathbf{W}^{(h+1)}\}$ i.e. it represents the set of all $\mathbf{W}^{(i)}$'s.

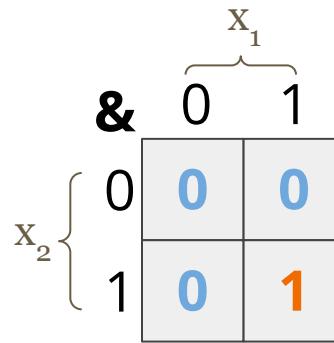
A sample case: How to train? When to train?



| x ₁ | x ₂ | y _t | y | error |
|----------------|----------------|----------------|-----|-------|
| 0 | 0 | 0 | 0.1 | 0.1 |
| 0 | 1 | 0 | 0.2 | 0.2 |
| 1 | 0 | 0 | 0.6 | 0.6 |
| 1 | 1 | 1 | 0.7 | 0.3 |



A sample case: What to minimize at the end? J ?



Initialize: $\mathbf{w}^T = [0.5 \ 0.1]$, $b = 0.1$

$y = f(\mathbf{x}^T \mathbf{w} + b) = f(0.5x_1 + 0.1x_2 + 0.1)$, simple case $f(x) = x$

$$y = 0.5x_1 + 0.1x_2 + 0.1$$

| x_1 | x_2 | y_t | y | err |
|-------|-------|-------|-----|-----|
| 0 | 0 | 0 | 0.1 | 0.1 |
| 0 | 1 | 0 | 0.2 | 0.2 |
| 1 | 0 | 0 | 0.6 | 0.6 |
| 1 | 1 | 1 | 0.7 | 0.3 |

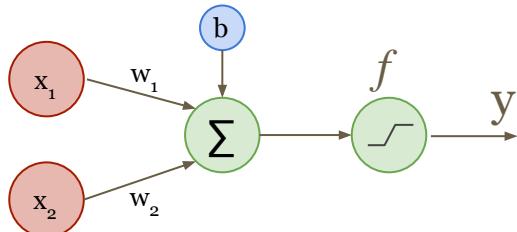
Loss function: $L(y_t, y)$

$L_i(y_t, y)$ Loss for input i

Total loss in this case:

$$J = L_1 + \dots + L_4$$

Where J is the *cost function*



where y_t , y are the vectors representing the **ground truth** and the **network output**, i.e. *network prediction* respectively.

Tensor Flow: Loss functions...

```
class BinaryCrossentropy : Computes the cross-entropy loss between true labels and predicted labels.  
class CategoricalCrossentropy : Computes the crossentropy loss between the labels and predictions.  
class CategoricalHinge : Computes the categorical hinge loss between y_true and y_pred .  
class CosineSimilarity : Computes the cosine similarity between labels and predictions.  
class Hinge : Computes the hinge loss between y_true and y_pred .  
class Huber : Computes the Huber loss between y_true and y_pred .  
class KLDivergence : Computes Kullback-Leibler divergence loss between y_true and y_pred .  
class LogCosh : Computes the logarithm of the hyperbolic cosine of the prediction error.  
class Loss : Loss base class.  
class MeanAbsoluteError : Computes the mean of absolute difference between labels and predictions.  
class MeanAbsolutePercentageError : Computes the mean absolute percentage error between y_true and y_pred .  
class MeanSquaredError : Computes the mean of squares of errors between labels and predictions.  
class MeanSquaredLogarithmicError : Computes the mean squared logarithmic error between y_true and y_pred .  
class Poisson : Computes the Poisson loss between y_true and y_pred .  
class Reduction : Types of loss reduction.  
class SparseCategoricalCrossentropy : Computes the crossentropy loss between the labels and predictions.  
class SquaredHinge : Computes the squared hinge loss between y_true and y_pred .
```

pyTorch Flow: Loss functions...

`nn.L1Loss`

Creates a criterion that measures the mean absolute error (MAE) between each element in the input x and target y .

`nn.MSELoss`

Creates a criterion that measures the mean squared error (squared L2 norm) between each element in the input x and target y .

`nn.CrossEntropyLoss`

This criterion computes the cross entropy loss between input and target.

`nn.CTCLoss`

The Connectionist Temporal Classification loss.

`nn.NLLLoss`

The negative log likelihood loss.

`nn.PoissonNLLLoss`

Negative log likelihood loss with Poisson distribution of target.

`nn.GaussianNLLoss`

Gaussian negative log likelihood loss.

`nn.KLDivLoss`

The Kullback-Leibler divergence loss measure

`nn.BCELoss`

Creates a criterion that measures the Binary Cross Entropy between the target and the input probabilities:

`nn.BCEWithLogitsLoss`

This loss combines a Sigmoid layer and the BCELoss in one single class.

`nn.MarginRankingLoss`

Creates a criterion that measures the loss given inputs x_1 , x_2 , two 1D mini-batch Tensors, and a label 1D mini-batch tensor y (containing 1 or -1).

`nn.HingeEmbeddingLoss`

Measures the loss given an input tensor x and a labels tensor y (containing 1 or -1).

`nn.MultiLabelMarginLoss`

Creates a criterion that optimizes a multi-class multi-classification hinge loss (margin-based loss) between input x (a 2D mini-batch Tensor) and output y (which is a 2D Tensor of target class indices).

`nn.HuberLoss`

Creates a criterion that uses a squared term if the absolute element-wise error falls below delta and a delta-scaled L1 term otherwise.

`nn.SmoothL1Loss`

Creates a criterion that uses a squared term if the absolute element-wise error falls below beta and an L1 term otherwise.

`nn.SoftMarginLoss`

Creates a criterion that optimizes a two-class classification logistic loss between input tensor x and target tensor y (containing 1 or -1).

`nn.MultiLabelSoftMarginLoss`

Creates a criterion that optimizes a multi-label one-versus-all loss based on max-entropy, between input x and target y of size (N, C) .

`nn.CosineEmbeddingLoss`

Creates a criterion that measures the loss given input tensors x_1, x_2 and a Tensor label y with values 1 or -1.

`nn.MultiMarginLoss`

Creates a criterion that optimizes a multi-class classification hinge loss (margin-based loss) between input x (a 2D mini-batch Tensor) and output y (which is a 1D tensor of target class indices, $0 \leq y \leq x.size(1) - 1$):

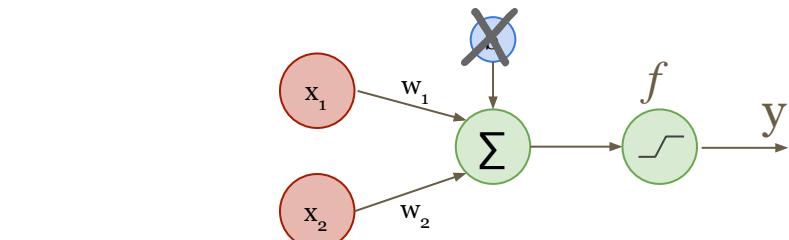
`nn.TripletMarginLoss`

Creates a criterion that measures the triplet loss given an input tensors x_1, x_2, x_3 and a margin with a value greater than 0.

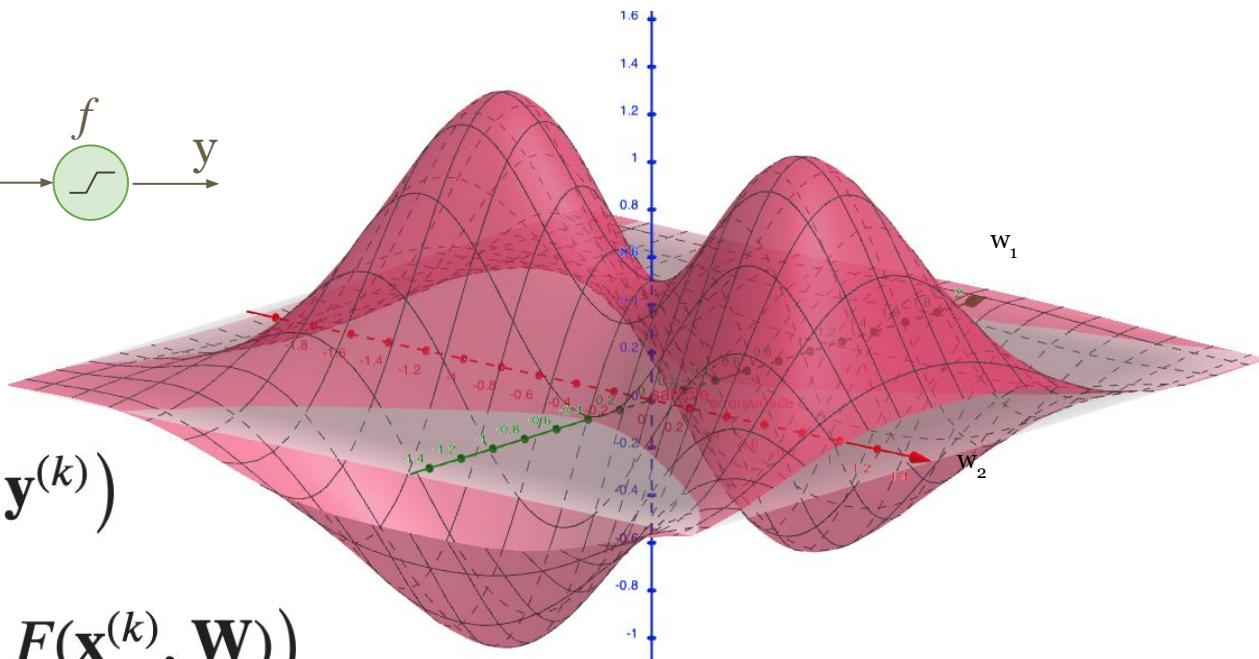
`nn.TripletMarginWithDistanceLoss`

Creates a criterion that measures the triplet loss given input tensors a, p, n (representing anchor, positive, and negative examples, respectively), and a nonnegative, real-valued function ("distance function") used to compute the relationship between the anchor and positive example ("positive distance") and the anchor and negative example ("negative distance").

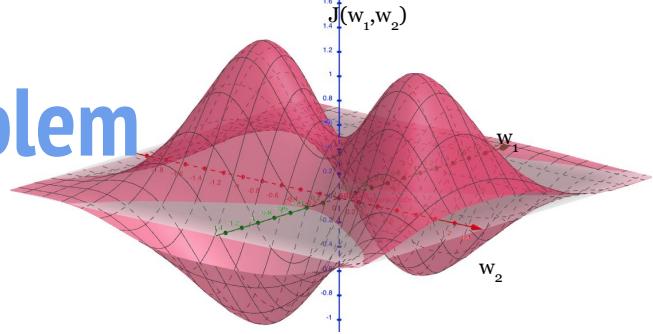
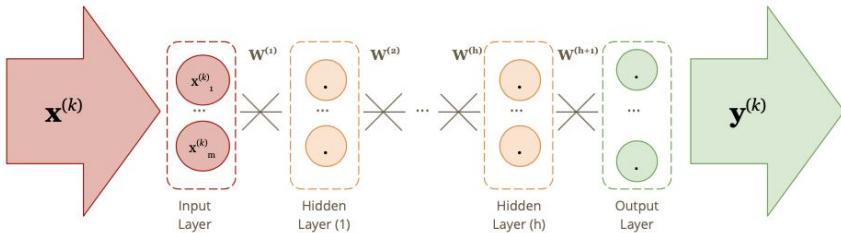
simple Loss Surface & Cost Function: A hypothetical case



$$\begin{aligned} J(\mathbf{W}) &= \frac{1}{d} \sum_{k=1}^d L\left(\mathbf{y}_t^{(k)}, \mathbf{y}^{(k)}\right) \\ &= \frac{1}{d} \sum_{k=1}^d L\left(\mathbf{y}_t^{(k)}, F(\mathbf{x}^{(k)}, \mathbf{W})\right) \end{aligned}$$



Loss Surface Minima: A search problem



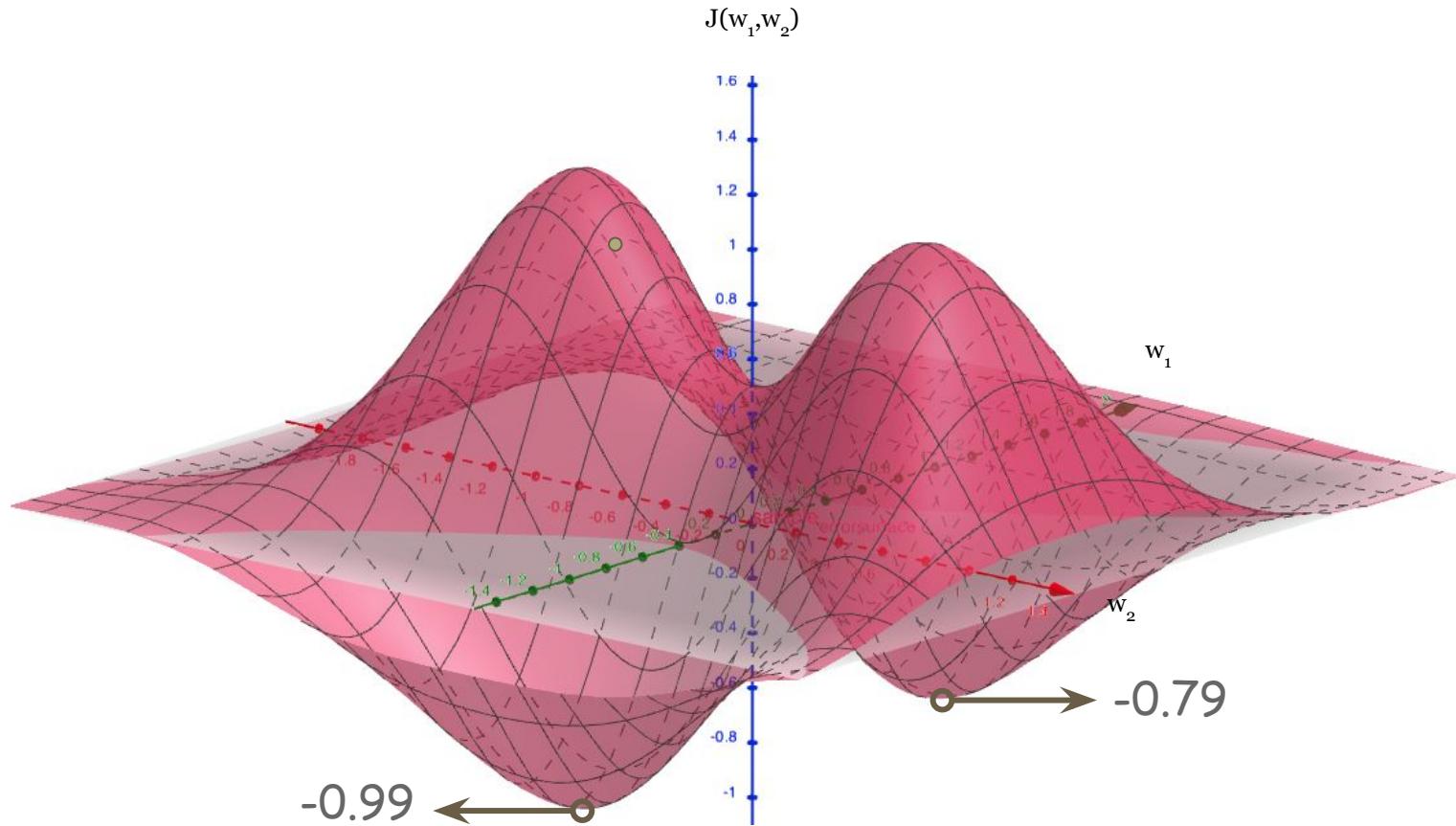
For d data points, cost function can be written as:

$$J(\mathbf{W}) = \frac{1}{d} \sum_{k=1}^d L\left(\mathbf{y}_t^{(k)}, \mathbf{y}^{(k)}\right) = \frac{1}{d} \sum_{k=1}^d L\left(\mathbf{y}_t^{(k)}, F(\mathbf{x}^{(k)}, \mathbf{W})\right)$$

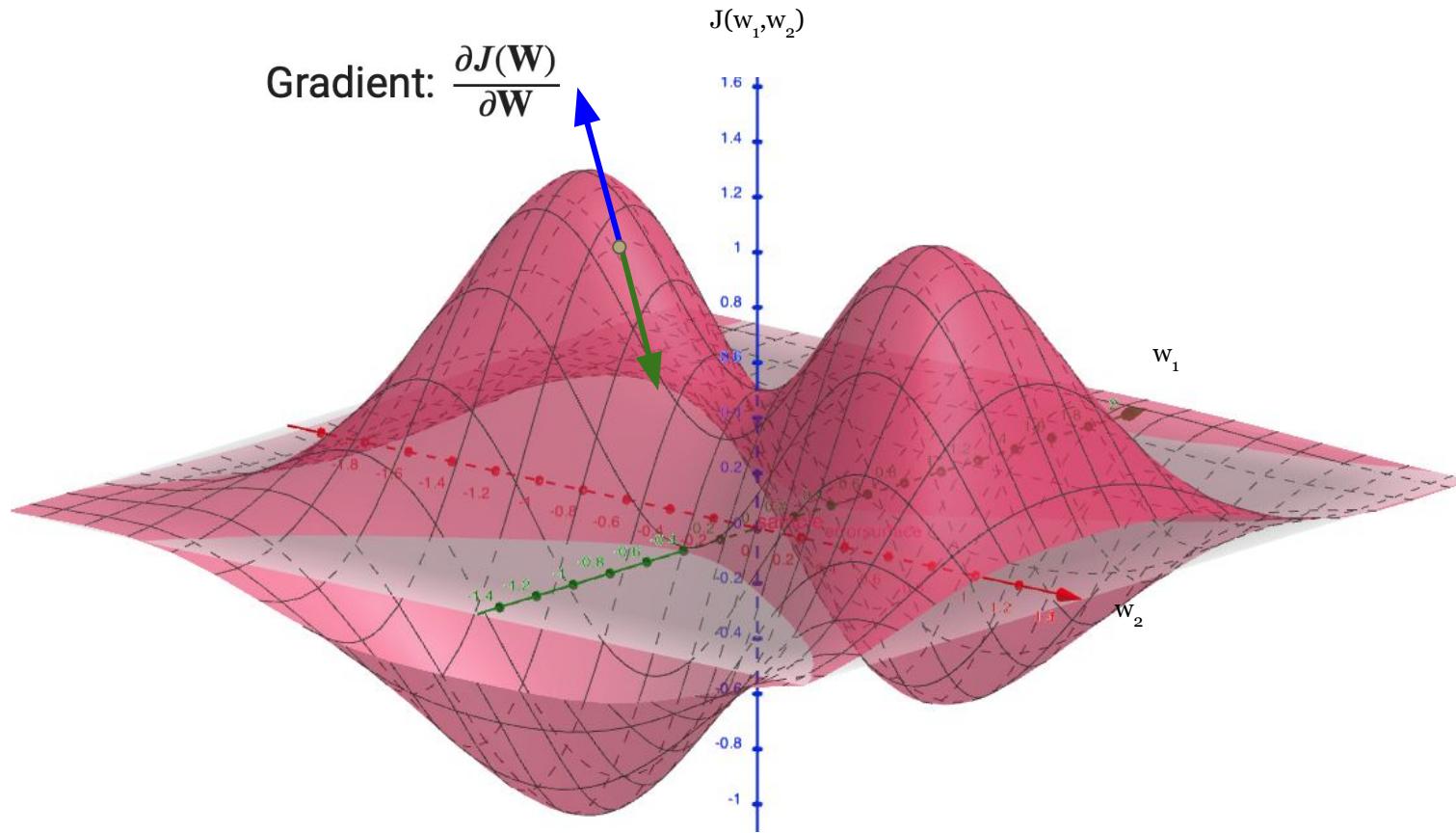
Best set of weight matrices given the selected cost function:

$$\mathbf{W}^* = \arg \min_{\mathbf{W}} J(\mathbf{W})$$

Loss Surface Gradient: Slide to Minima but which?



Loss Surface Gradient: Steepest Descent to Minima



Loss Surface Gradient: Steepest Descent to Minima

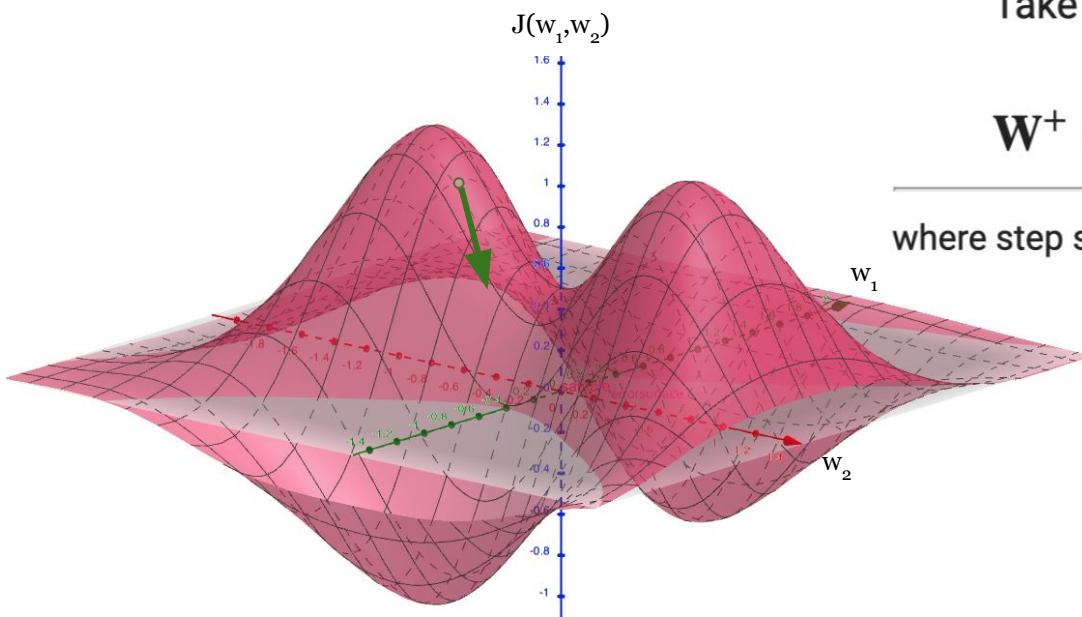
Gradient Descent Algorithm:

- Initialize network: \mathbf{W} , random is a good choice
- Loop until not worth it:
Take a step in $-\text{gradient direction}$ to update \mathbf{W} as:

$$\mathbf{W}^+ = \mathbf{W} - \eta \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$$

where step size η is referred to as the learning rate.

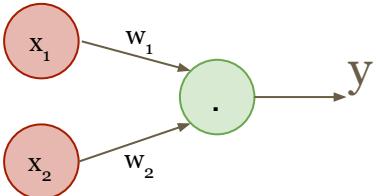
Alternative methods result
in different optimizers



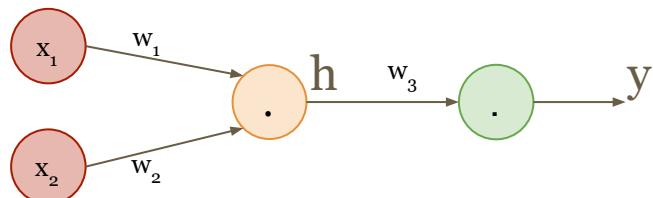
$$J(\mathbf{W}) = \frac{1}{d} \sum_{k=1}^d L(\mathbf{y}_t^{(k)}, \mathbf{y}^{(k)}) \\ = \frac{1}{d} \sum_{k=1}^d L(\mathbf{y}_t^{(k)}, F(\mathbf{x}^{(k)}, \mathbf{W}))$$

How to update w_i : Backpropagation

What is the effect of w_1 on J :



$$\frac{\partial J(\mathbf{W})}{\partial w_1} = \frac{\partial J(\mathbf{W})}{\partial y} \cdot \frac{\partial y}{\partial w_1}$$



$$\frac{\partial J(\mathbf{W})}{\partial w_1} = \frac{\partial J(\mathbf{W})}{\partial y} \cdot \frac{\partial y}{\partial h} \cdot \frac{\partial h}{\partial w_1}$$

Tensor Flow: Optimizers... Gradient Descent

`class Adadelta`: Optimizer that implements the Adadelta algorithm.

`class Adagrad`: Optimizer that implements the Adagrad algorithm.

`class Adam`: Optimizer that implements the Adam algorithm.

`class Adamax`: Optimizer that implements the Adamax algorithm.

`class Ftrl`: Optimizer that implements the FTRL algorithm.

`class Nadam`: Optimizer that implements the NAdam algorithm.

`class Optimizer`: Base class for Keras optimizers.

`class RMSprop`: Optimizer that implements the RMSprop algorithm.

`class SGD`: Gradient descent (with momentum) optimizer.

pyTorch Flow: Optimizers... Gradient Descent

Adadelta

Implements Adadelta algorithm.

Adagrad

Implements Adagrad algorithm.

Adam

Implements Adam algorithm.

AdamW

Implements AdamW algorithm.

SparseAdam

Implements lazy version of Adam algorithm suitable for sparse tensors.

Adamax

Implements Adamax algorithm (a variant of Adam based on infinity norm).

ASGD

Implements Averaged Stochastic Gradient Descent.

LBFGS

Implements L-BFGS algorithm, heavily inspired by `minFunc`.

NAdam

Implements NAdam algorithm.

RAdam

Implements RAdam algorithm.

RMSprop

Implements RMSprop algorithm.

Rprop

Implements the resilient backpropagation algorithm.

SGD

Implements stochastic gradient descent (optionally with momentum).

Big Picture: Getting started - Labelled Data

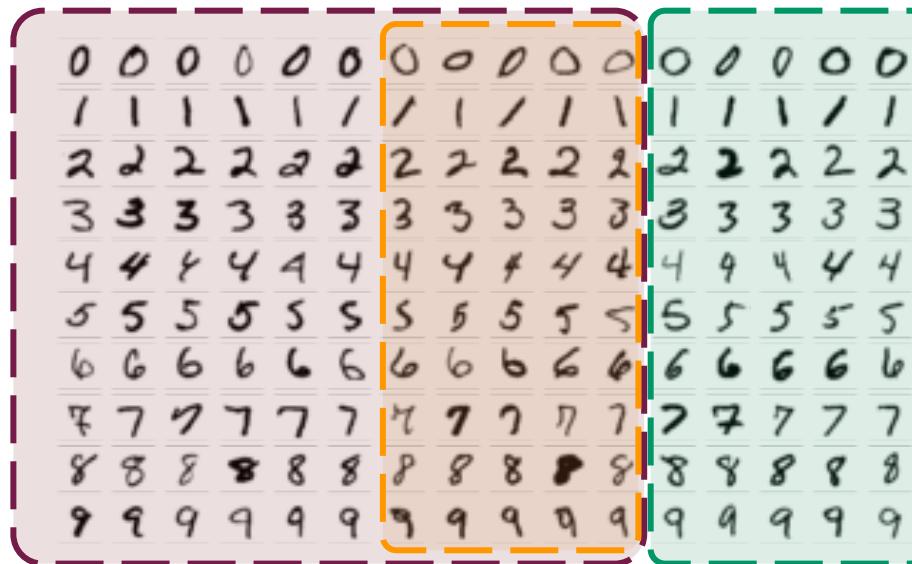
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9 9 9 9 9 9 9



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

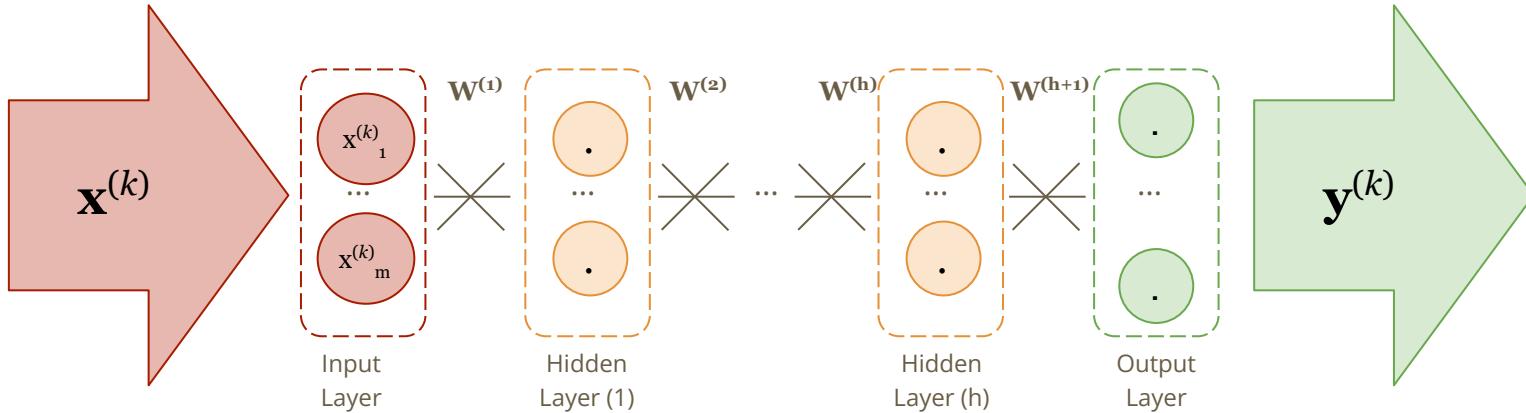
Labelled data: If not available you get the honor to label 70K of them! Enjoy...

Big Picture: Getting started - Divide Data



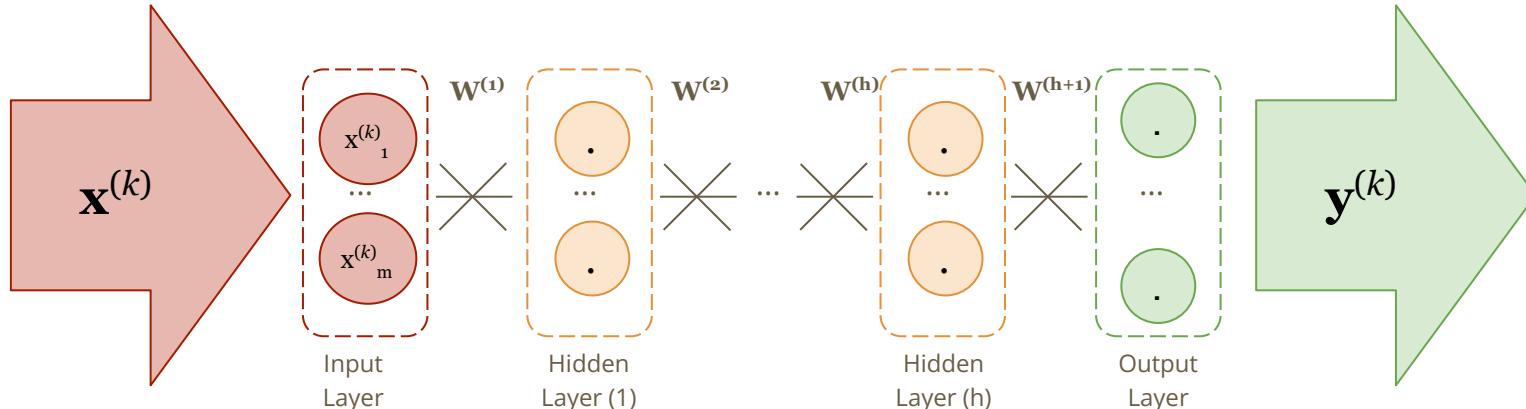
Data → **Training** data, **Validation** data, **Test** data

Big Picture: Getting started - Train where?



Hyperparameters: network topology, number of layers, number of neurons etc, *regularization* (dropout, early stopping, data augmentation, etc), optimizer, activation function, ...

Big Picture: Regularization ...



Fight against: complex solutions lead to **overfitting** / overlearning / memorizing data

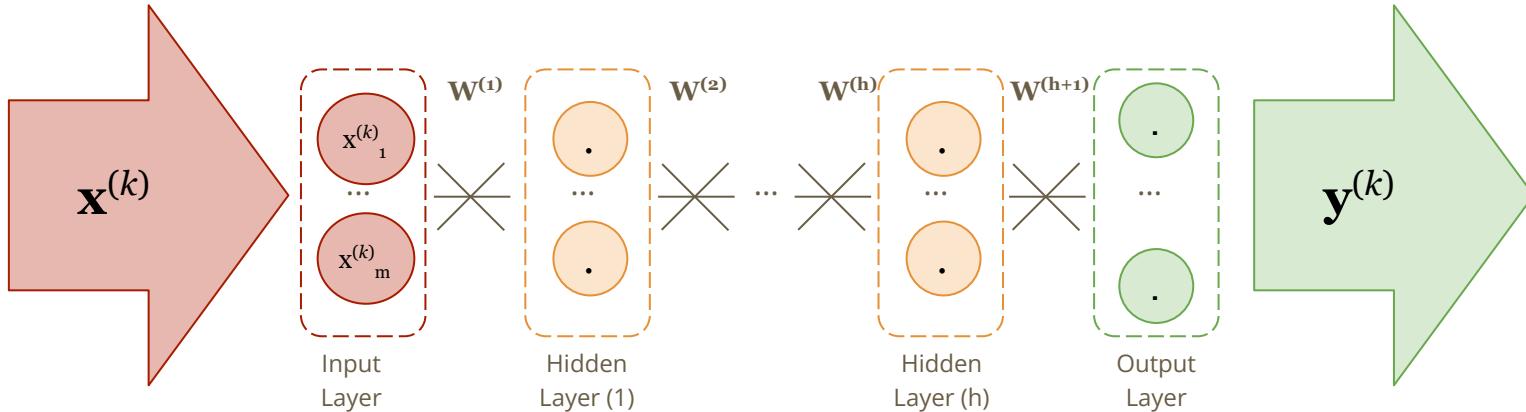
Dropout: Probabilistically pop some of the neurons in each iteration

Data Augmentation: shift, scale, rotate, add noise, etc. to generate variations

Regularization term: add a term to the cost function

Early Stopping: Stop when validation error stops improving

Big Picture: When to update?

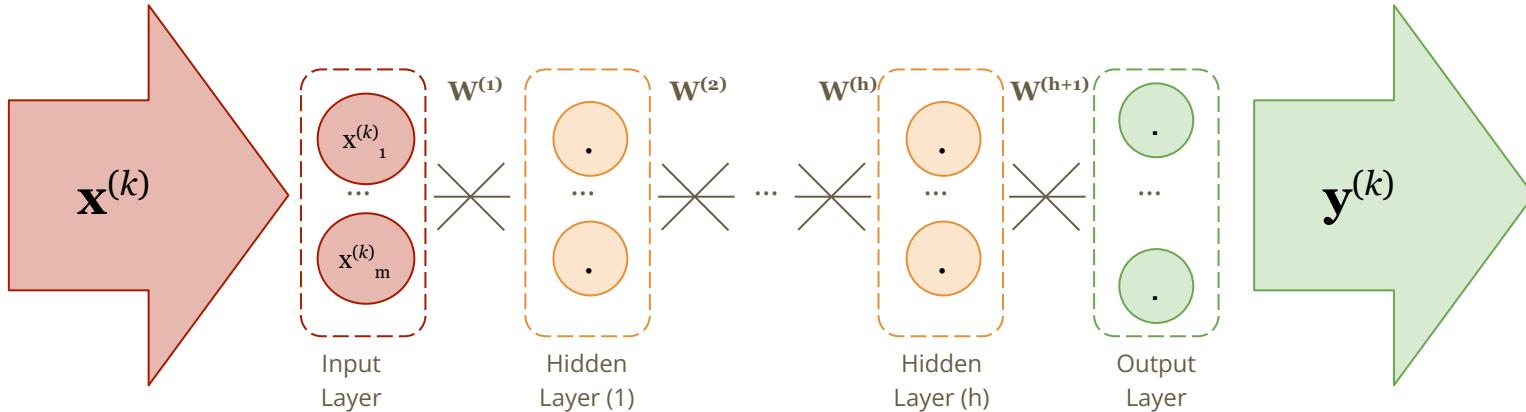


Epoch / Batch: Pass all the data through the network, calculate loss, update weights.

After every data point: Randomly select one - [SGD - check this video out](#)

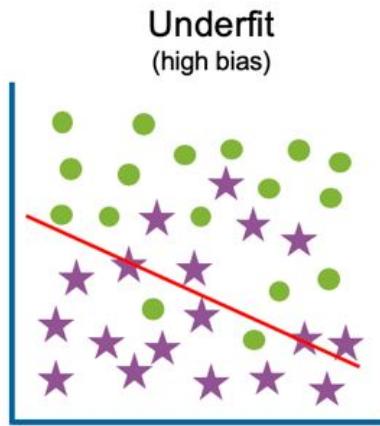
Mini-Batch: Data passed in subsets and weights updated after each batch

Big Picture: Loop until not worth it?

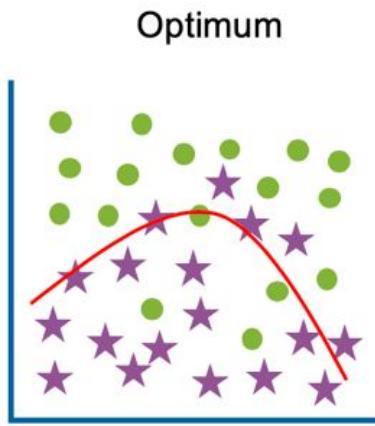


- Error is low enough - *i.e. error is below a preset threshold*
- Got bored - *i.e. maximum epoch limit is reached*
- Validation error started to increase while training decreases - *how is this even possible?*
- ???

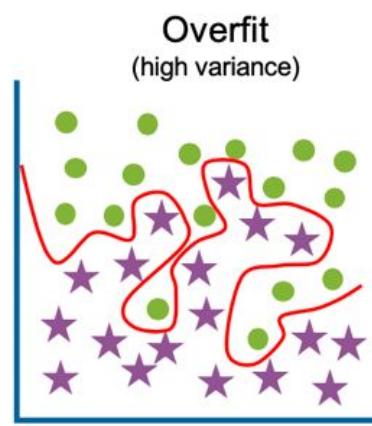
Big Picture: Try not to over- or under-fit



High training error
High test error

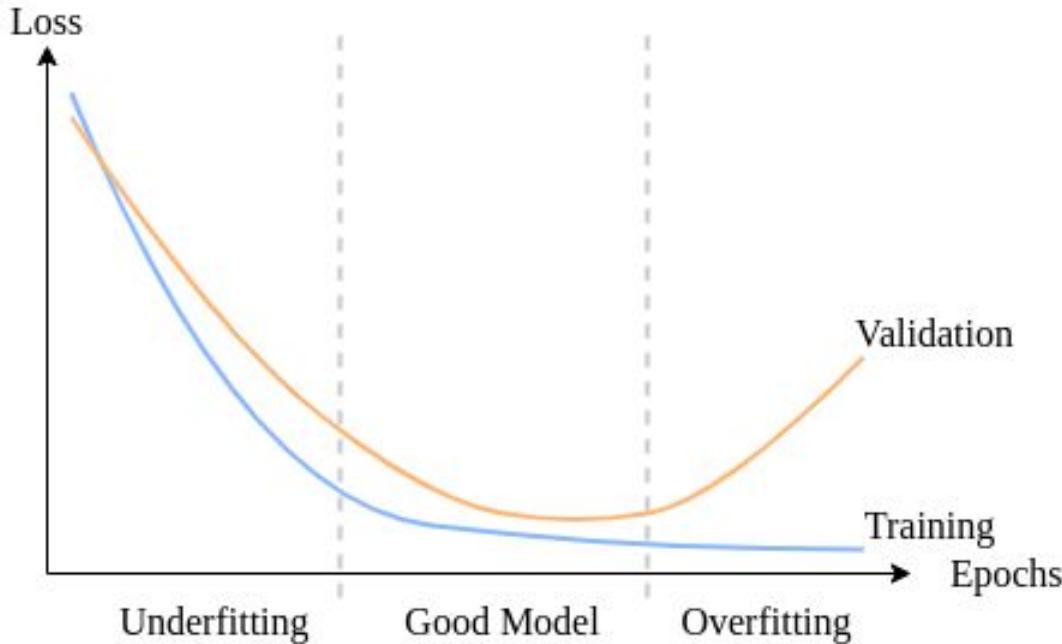


Low training error
Low test error



Low training error
High test error

Big Picture: How to avoid overfit?



A simple case: Try backpropagation manually

$\&$

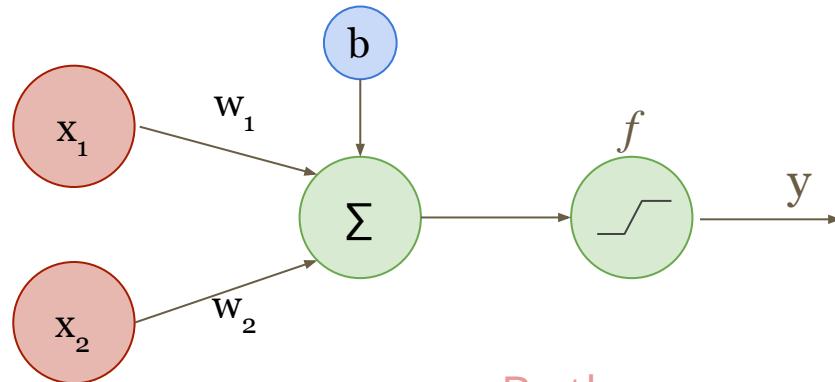
| | | x_1 |
|-------|---|-------|
| | | 0 |
| x_2 | 0 | 0 |
| | 1 | 1 |

Initialize w, b as you like

choose $f(\cdot)$, loss function and learning rate

Given $y = f(x^T w + b)$

Run gradient descent



By the way, you can implement this in numpy

Good news

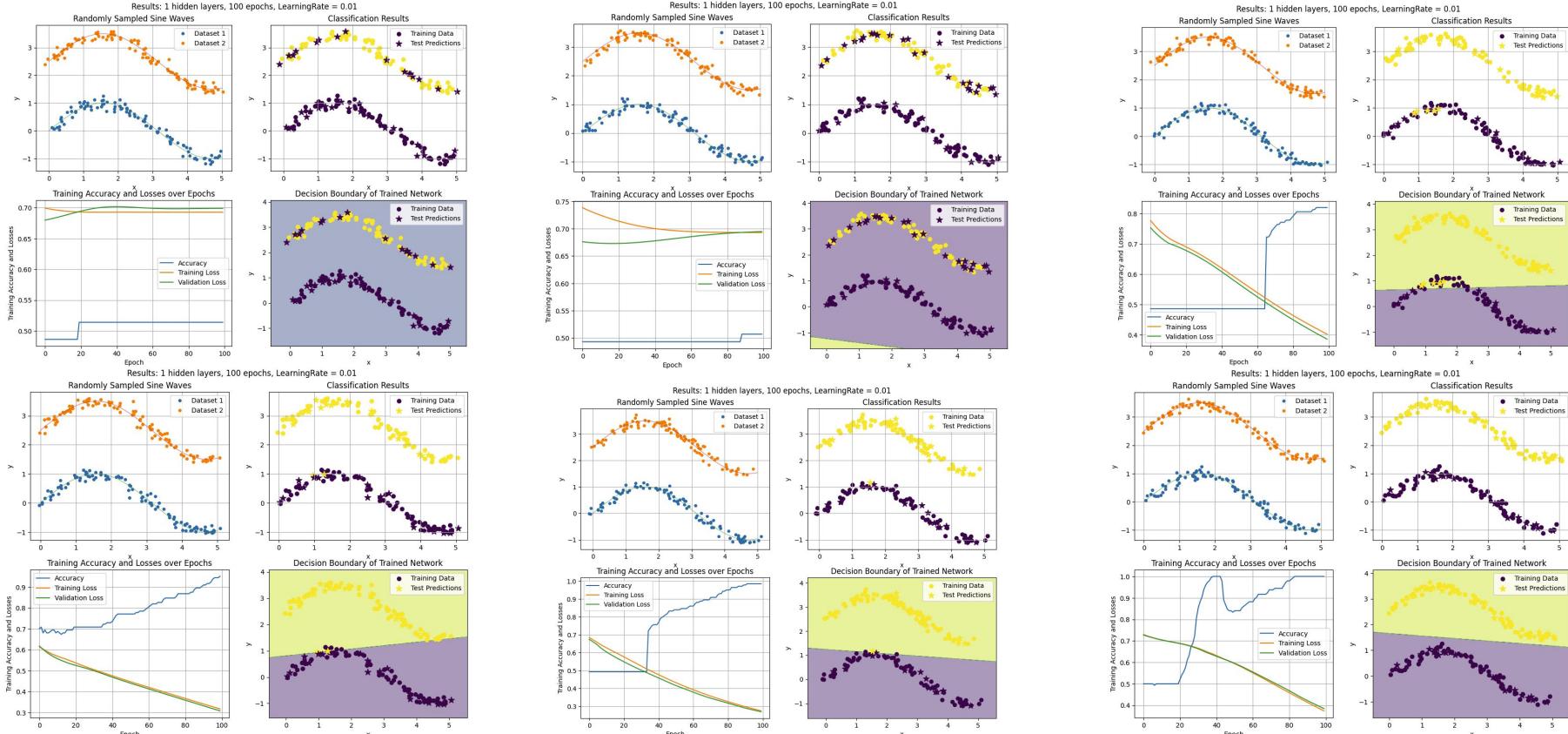
13 lines of code → A Neural Network: A good read, a good practice

You won't need to code a ANN from scratch:

- TF, pyTorch, etc. exit
- LLMs assist

Check out: Tensorflow playground

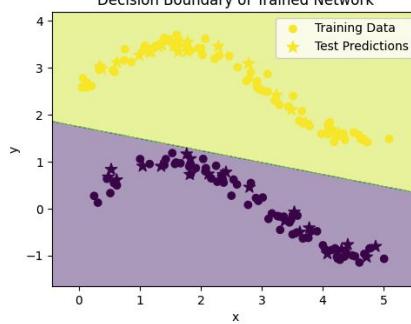
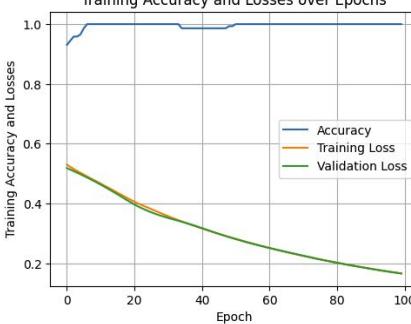
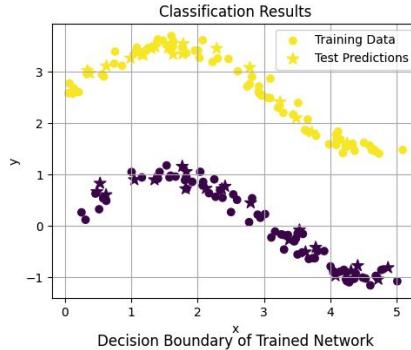
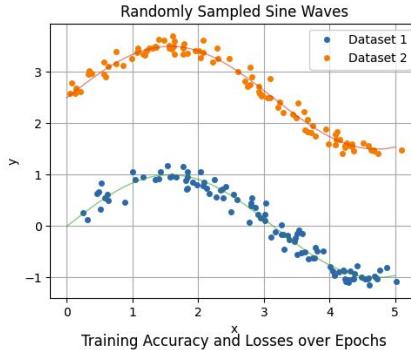
Same conditions, Just re-runs from scratch



Let's try together

Check this [colab notebook](#)

Results: 1 hidden layers, 100 epochs, LearningRate = 0.01

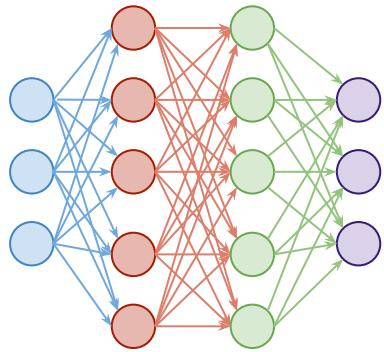


to be continued...

ME 536

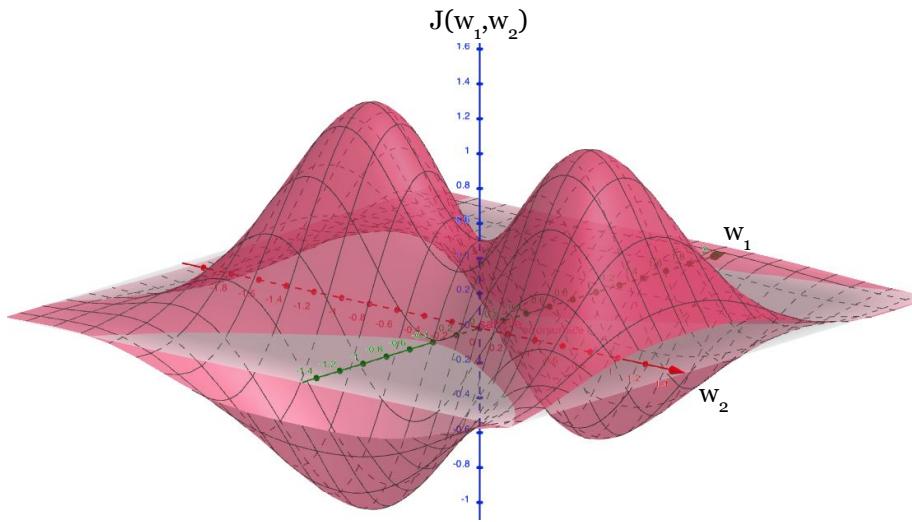
— Week 13: How deep is your
learning network? —

Recall: Fully Connected Networks

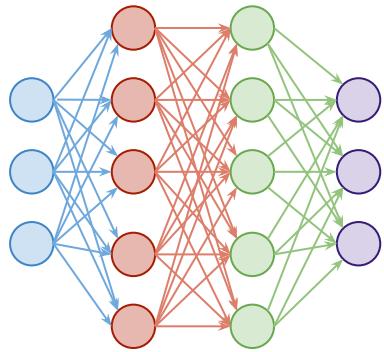


Too many parameters to tune?

The more parameters, the more complex the search space / surface



Recall: Fully Connected Networks



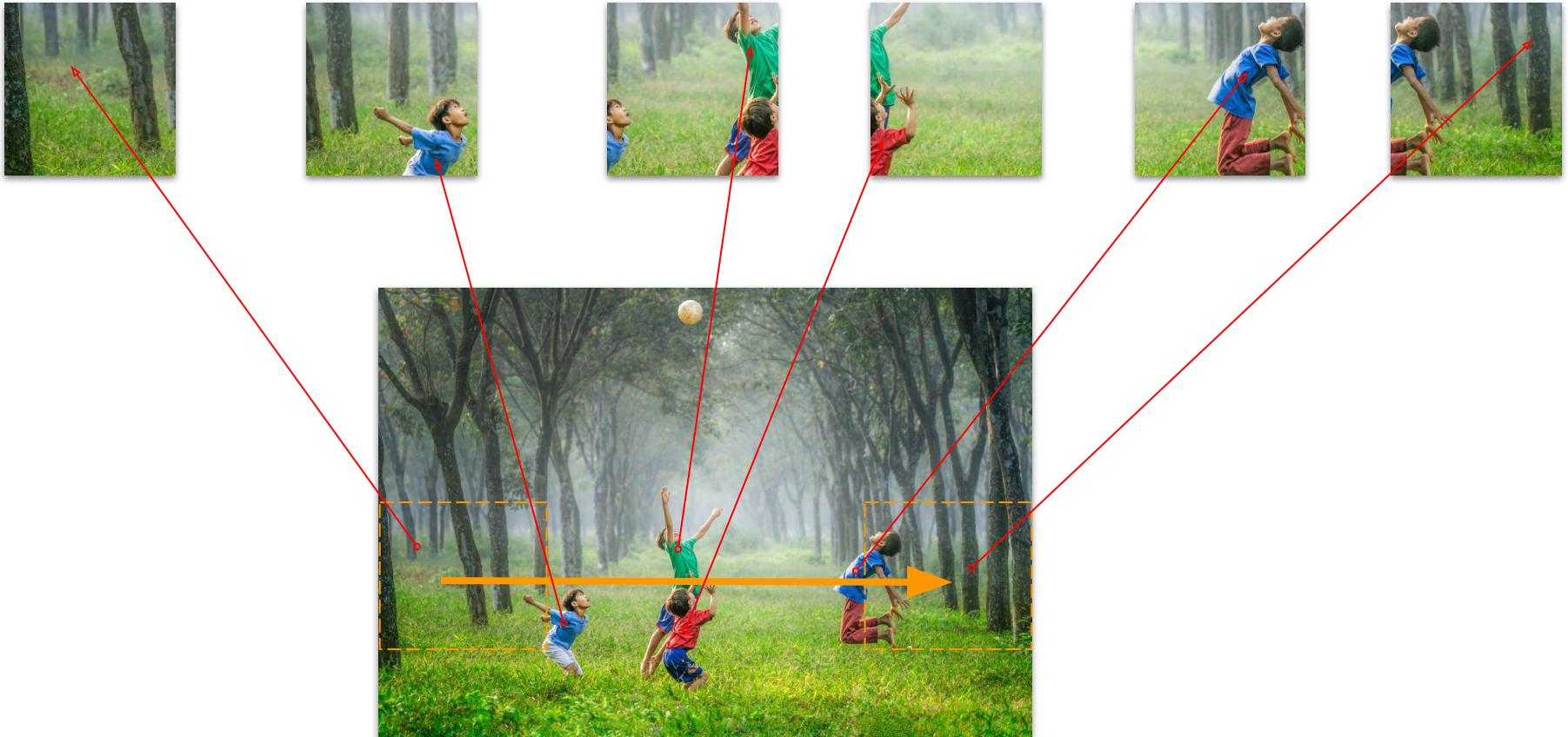
or DENSE networks

All outputs of a layer are connected to all inputs of the next.

Are all pixels in an image related?



Who cares: about that pixel



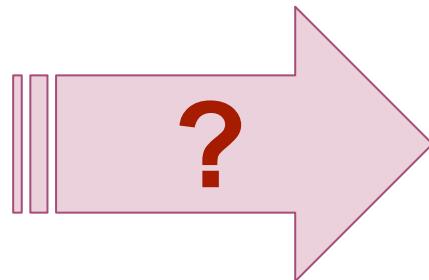
Seeing the whole: part by part



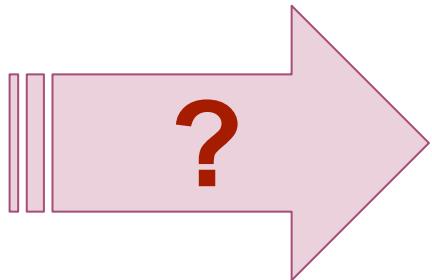
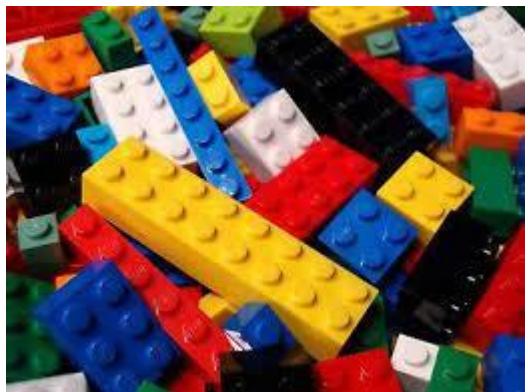
Old but still useful tactic: Divide and conquer



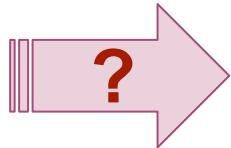
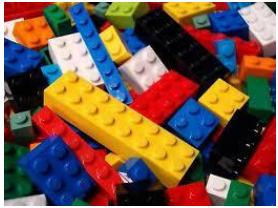
Old but still useful tactic: match and conquer



Reality Check: Divide and but how to conquer?



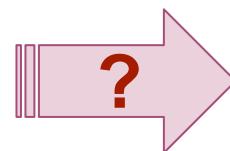
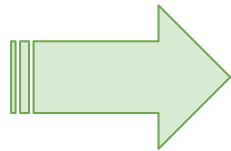
Hmmm: Divide and but how to conquer?



Note that the problem is:

- NOT about a path from piece to whole
- About
 - finding pieces given the whole
 - And identifying the whole form a subset of all possible pieces

May be: Divide and join to conquer



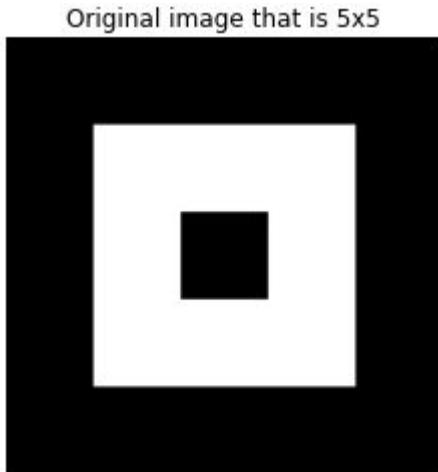
Tear down the model and by just looking at the pieces:

guess which model we had to start with?

Note that you can form new small pieces by using the ones you have

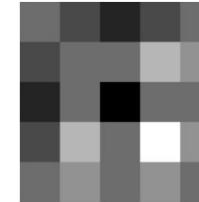


Recall Convolution: Moving a window over an image



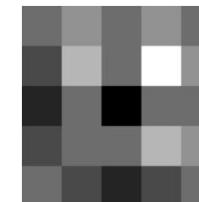
Kernel 1

| | | |
|-------|-------|-------|
| -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | 1.00 |
| -1.00 | 1.00 | -1.00 |



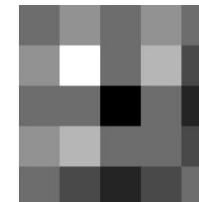
Kernel 2

| | | |
|-------|-------|-------|
| -1.00 | 1.00 | -1.00 |
| -1.00 | 1.00 | 1.00 |
| -1.00 | -1.00 | -1.00 |



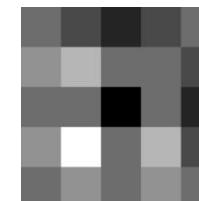
Kernel 3

| | | |
|-------|-------|-------|
| -1.00 | 1.00 | -1.00 |
| 1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 |



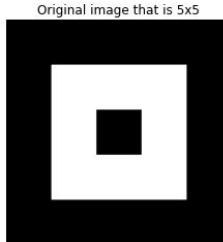
Kernel 4

| | | |
|-------|-------|-------|
| -1.00 | -1.00 | -1.00 |
| 1.00 | 1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |



More on Convolution: 1 kernel + ReLu + Pooling

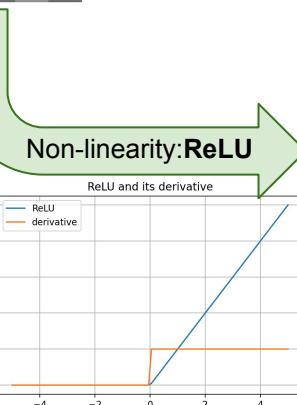
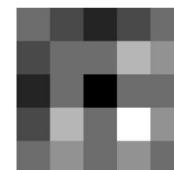
Image I → 5x5



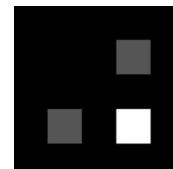
Kernel 1

$$\begin{vmatrix} -1.00 & -1.00 & -1.00 \\ -1.00 & 1.00 & 1.00 \\ -1.00 & 1.00 & -1.00 \end{vmatrix}$$

feature map → 5x5



Maxpooling:
3x3 into 1
with stride of 3



result → 2x2



Maxpooled

| | | |
|------|------|------|
| 0.00 | 1.00 | 1.00 |
| 1.00 | 3.00 | 3.00 |

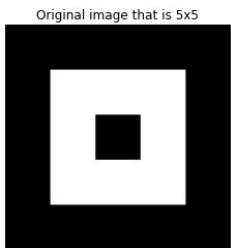
Feature Map after ReLU

| | | | | |
|------|------|------|------|------|
| 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.00 | 1.00 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |

| | | | | |
|------|------|------|------|------|
| 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 1.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 3.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |

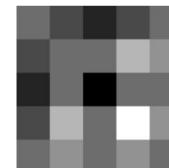
Convolution: Multiple Kernels + ReLU + Maxpooling

Image I convolved with Kernel $i \rightarrow$ feature map \rightarrow ReLU \rightarrow Maxpooling



Kernel 1

$$\begin{vmatrix} -1.00 & -1.00 & -1.00 \\ -1.00 & 1.00 & 1.00 \\ -1.00 & 1.00 & -1.00 \end{vmatrix}$$

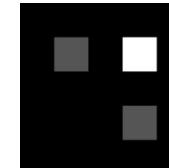
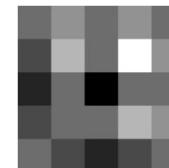


Maxpooled

$$\begin{vmatrix} 0.00 & 1.00 \\ 1.00 & 3.00 \end{vmatrix}$$

Kernel 2

$$\begin{vmatrix} -1.00 & 1.00 & -1.00 \\ -1.00 & 1.00 & 1.00 \\ -1.00 & -1.00 & -1.00 \end{vmatrix}$$

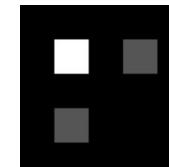
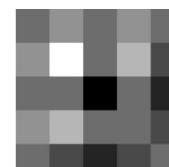


Maxpooled

$$\begin{vmatrix} 1.00 & 3.00 \\ 0.00 & 1.00 \end{vmatrix}$$

Kernel 3

$$\begin{vmatrix} -1.00 & 1.00 & -1.00 \\ 1.00 & 1.00 & -1.00 \\ -1.00 & -1.00 & -1.00 \end{vmatrix}$$

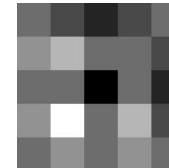


Maxpooled

$$\begin{vmatrix} 3.00 & 1.00 \\ 1.00 & 0.00 \end{vmatrix}$$

Kernel 4

$$\begin{vmatrix} -1.00 & -1.00 & -1.00 \\ 1.00 & 1.00 & -1.00 \\ -1.00 & 1.00 & -1.00 \end{vmatrix}$$

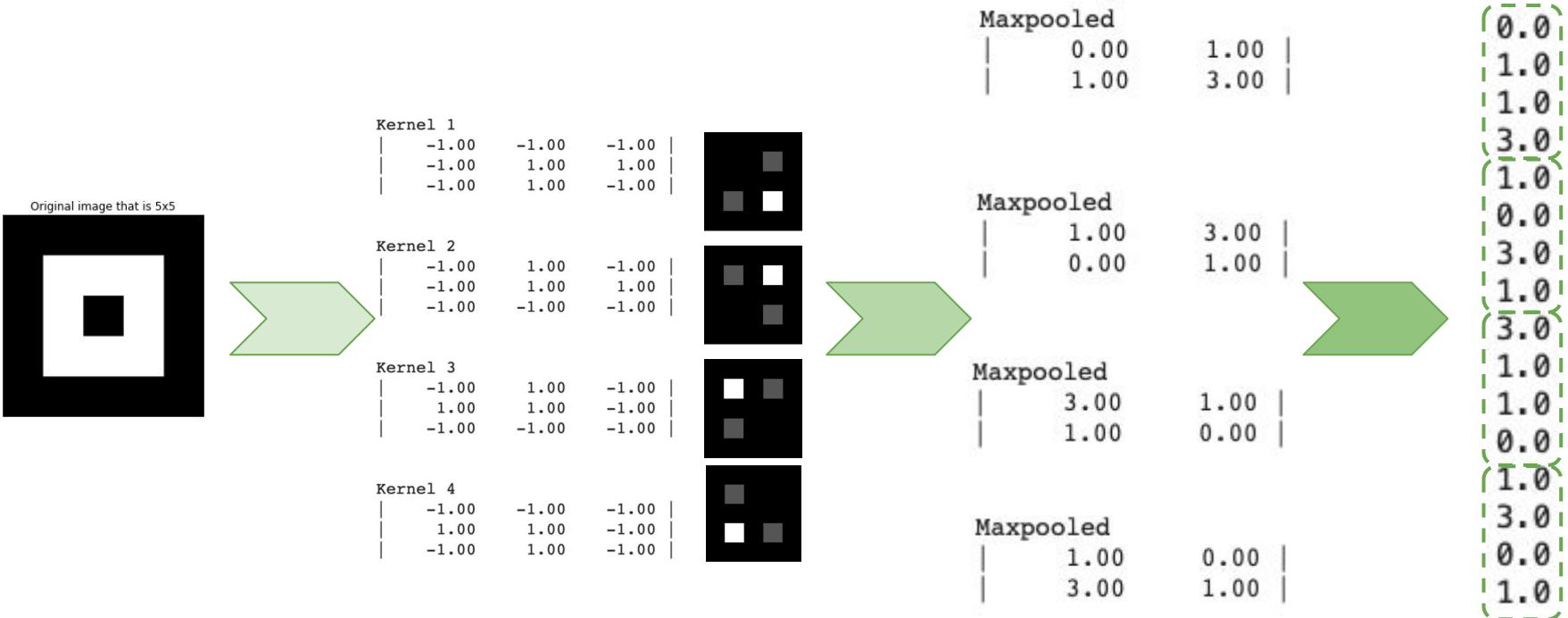


Maxpooled

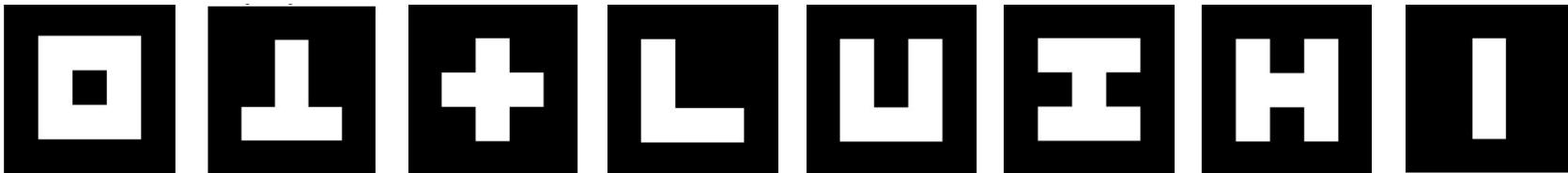
$$\begin{vmatrix} 1.00 & 0.00 \\ 3.00 & 1.00 \end{vmatrix}$$

More after Convolution: ... feature vector ...

Vectorize result:
i.e. flatten the
feature matrix



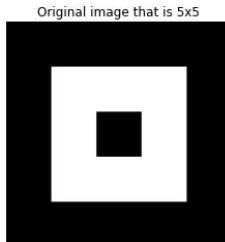
More after Convolution: ... feature vectors ...



| | | | | | | | | |
|----------|--|-------------------------------|--------------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| Kernel 1 | -1.00 -1.00 -1.00 -1.00 | -1.00 1.00 1.00 1.00 | -1.00 1.00 1.00 -1.00 | 0.0 1.0 1.0 0.0 | 2.0 1.0 0.0 1.0 | 1.0 1.0 0.0 2.0 | 1.0 1.0 2.0 0.0 | 2.0 2.0 0.0 0.0 |
| | 3.0 | 1.0 | 1.0 | 2.0 | 3.0 | 1.0 | 1.0 | 0.0 |
| | 1.0 | 2.0 | 1.0 | 2.0 | 2.0 | 2.0 | 1.0 | 2.0 |
| Kernel 2 | -1.00 -1.00 -1.00 | 1.00 1.00 -1.00 | -1.00 1.00 1.00 | 0.0 0.0 3.0 | 0.0 0.0 0.0 | 0.0 0.0 2.0 | 0.0 0.0 2.0 | 0.0 0.0 0.0 |
| | 1.0 | 1.0 | 0.0 | 2.0 | 1.0 | 1.0 | 0.0 | 0.0 |
| | 3.0 | 2.0 | 1.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 |
| Kernel 3 | -1.00 1.00 -1.00 | 1.00 1.00 -1.00 | -1.00 -1.00 -1.00 | 1.00 1.00 1.00 | 1.00 0.0 0.0 | 1.00 2.0 0.0 | 0.0 1.0 0.0 | 0.0 0.0 0.0 |
| | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| | 1.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 2.0 | 1.0 |
| Kernel 4 | -1.00 1.00 -1.00 | -1.00 1.00 1.00 | -1.00 -1.00 -1.00 | 3.0 0.0 1.00 | 2.0 0.0 0.0 | 3.0 0.0 0.0 | 2.0 0.0 0.0 | 1.0 0.0 0.0 |
| | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 |

Recall the workflow:

Image I → 5x5

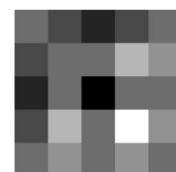


Kernel 1

| | | |
|-------|-------|-------|
| -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | 1.00 |
| -1.00 | 1.00 | -1.00 |

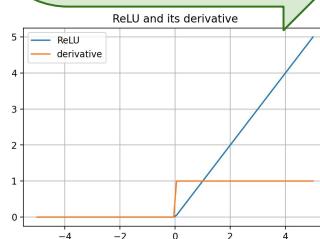
Convolve with kernels

feature map → 5x5



ReLU

Activation Func:ReLU



+ Pooling

- Max
- Average
- ...

result → 2x2



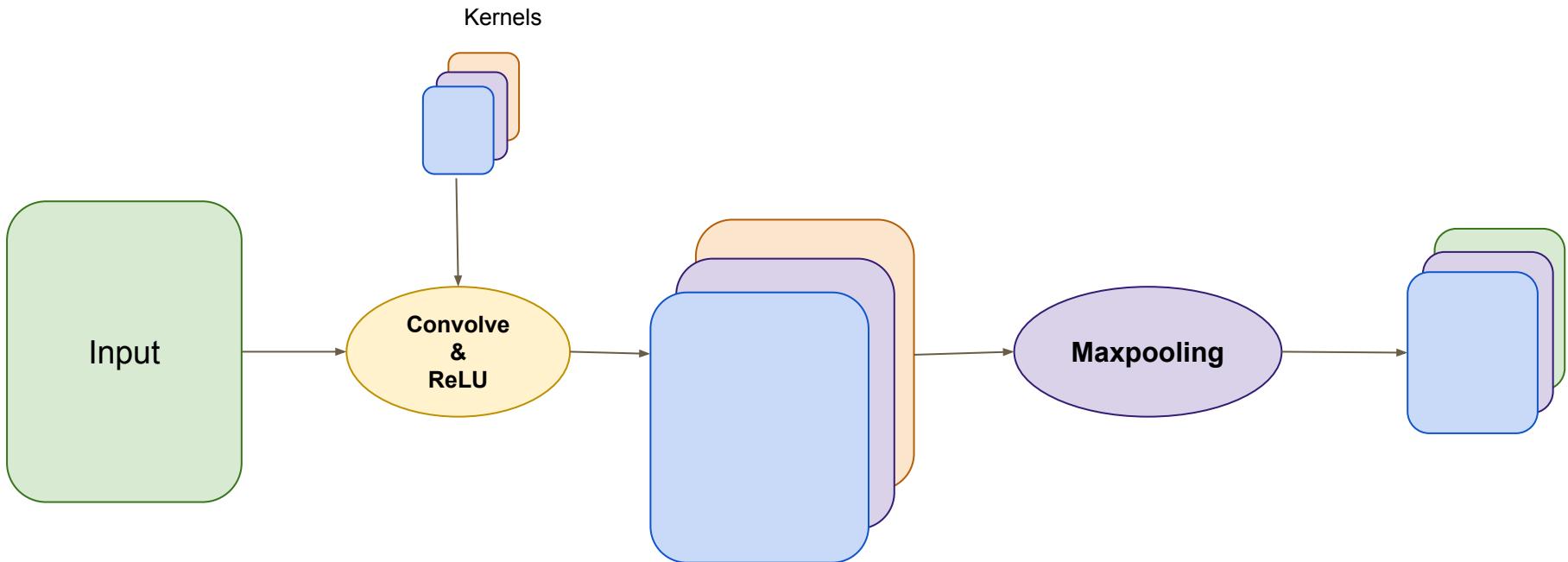
Maxpooling:
3x3 into 1
with stride of 3

| | | |
|------|------|------|
| 0.00 | 1.00 | 3.00 |
|------|------|------|

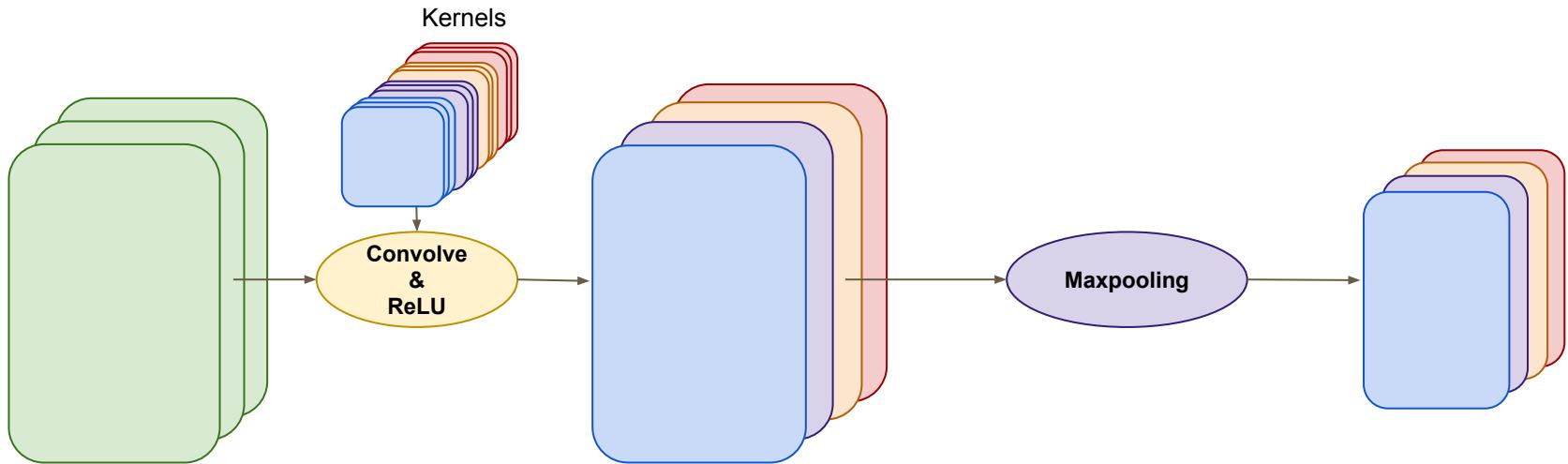
Feature Map after ReLU

| | | | | |
|------|------|------|------|------|
| 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.00 | 1.00 | 0.00 | 3.00 | 0.00 |
| 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |

CNN layers: a typical case at the input layer



CNN layers: a typical case after input



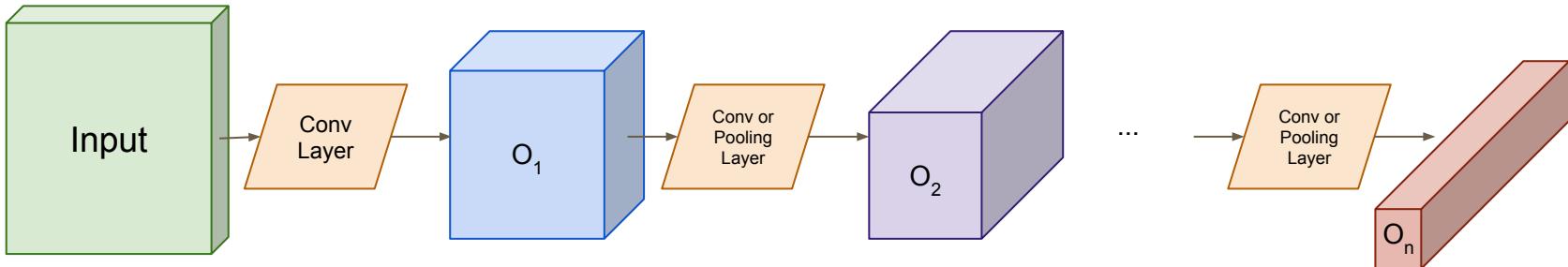
Observe that input of a layer gets a **smaller** footprint while getting **taller**

Also note that this might be the input layer for an RGB image

A deep CNN: feature extractor

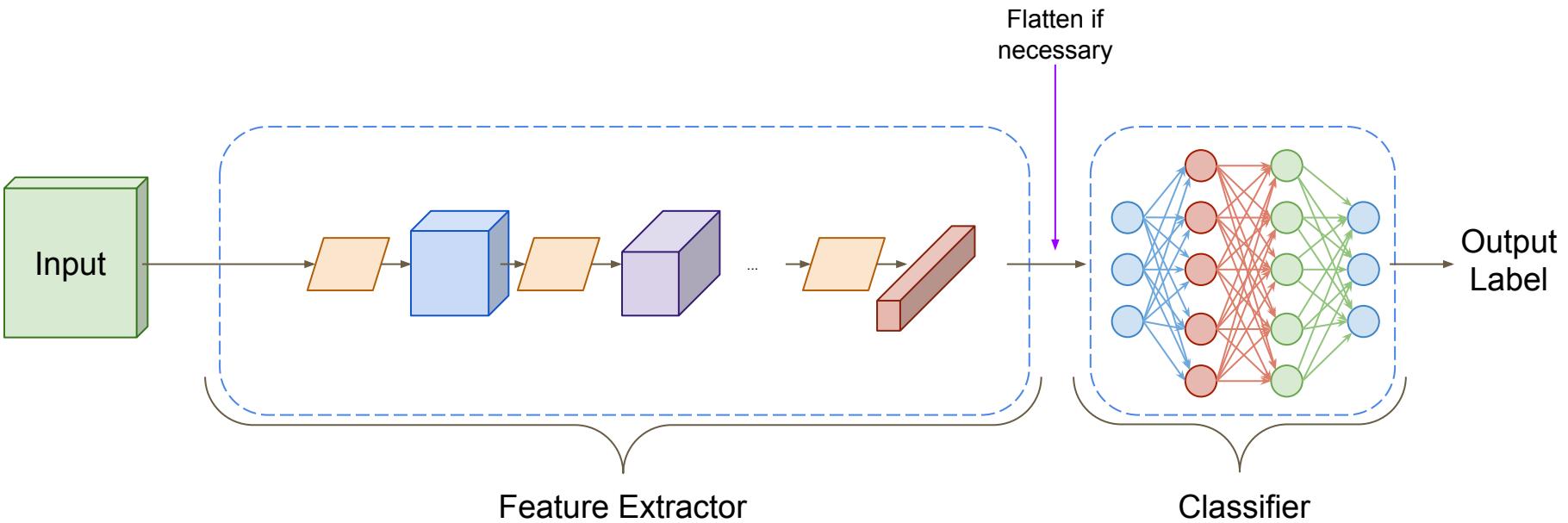
After n layers (convolutional or pooling) we expect a *vectorized output* from the CNN, i.e. **a feature vector**

If not, *flatten* the *feature tensor* into the *feature vector*.

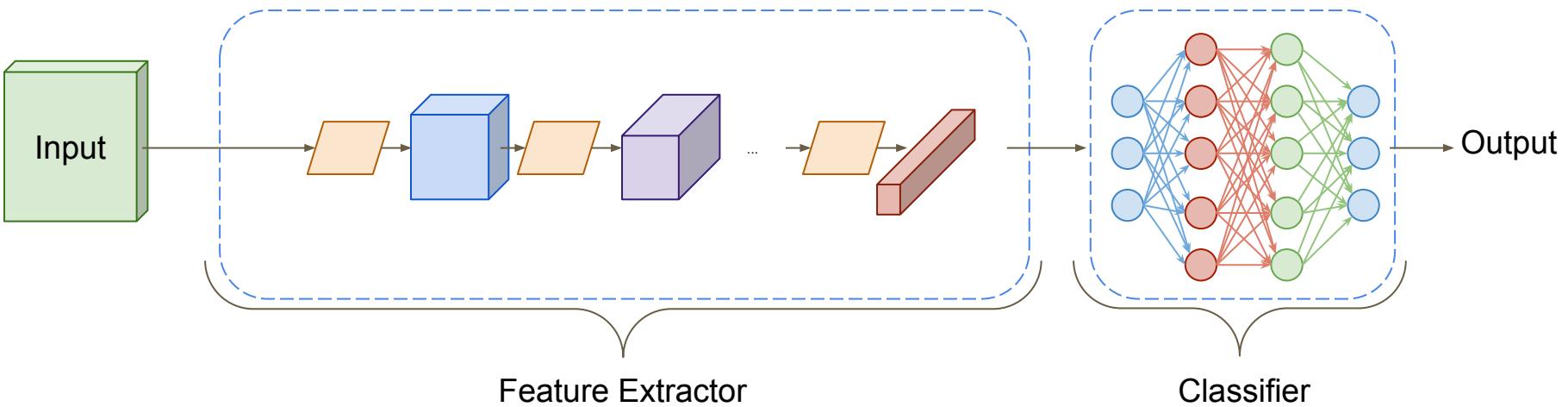


Note that both a convolutional and pooling layers can use stride other than 1 and hence can scale the input down

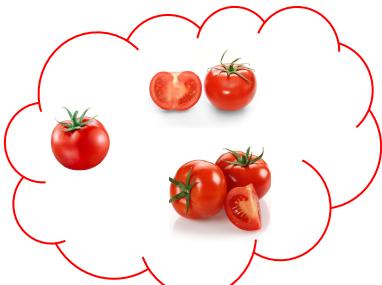
A typical deep CNN



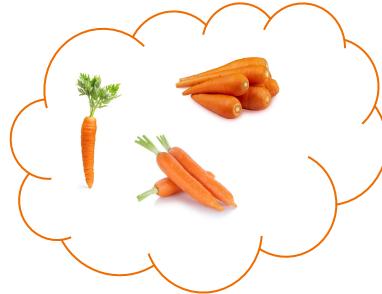
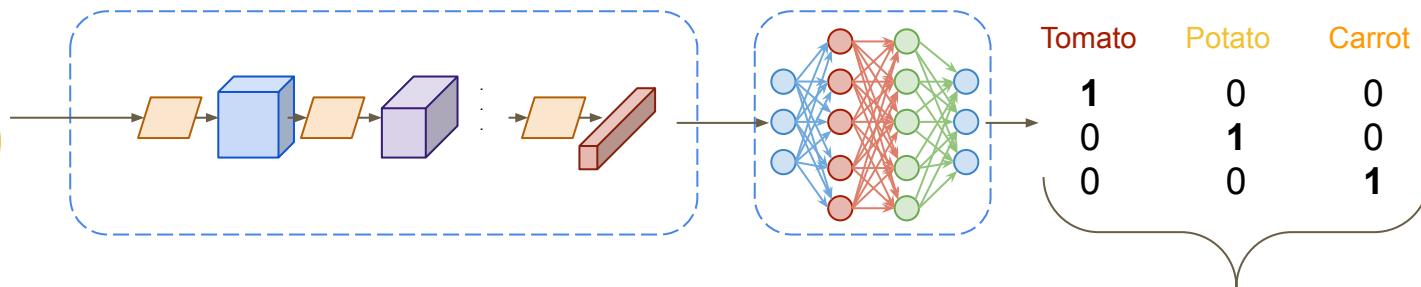
A typical deep CNN



Training a deep CNN: Inputs and expected output



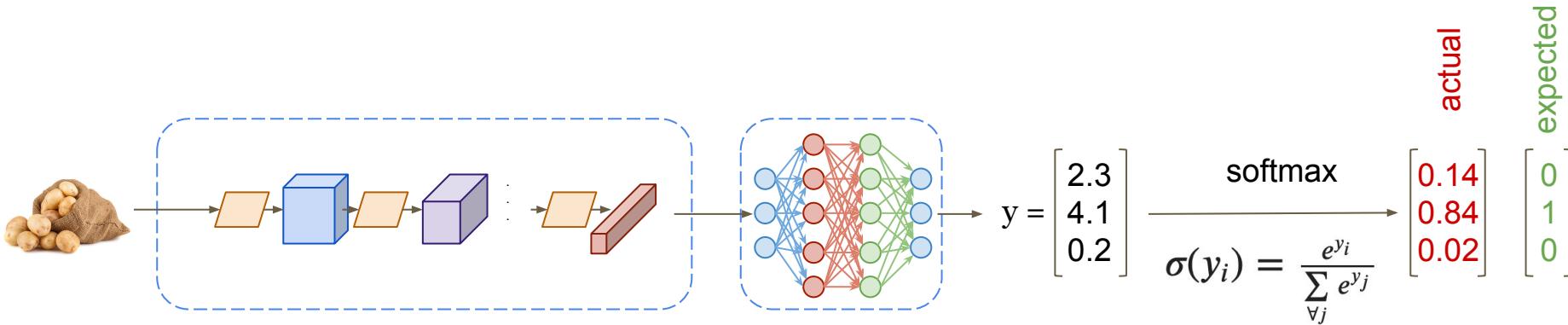
During training a ground truth / correct labels are provided with every input



One-hot
encoded
vectors

Using a trained deep CNN: hope for the best

A FCN (fully connected network) outputs a vector of values when an image is presented at the input which is *not necessarily* one-hot encoded



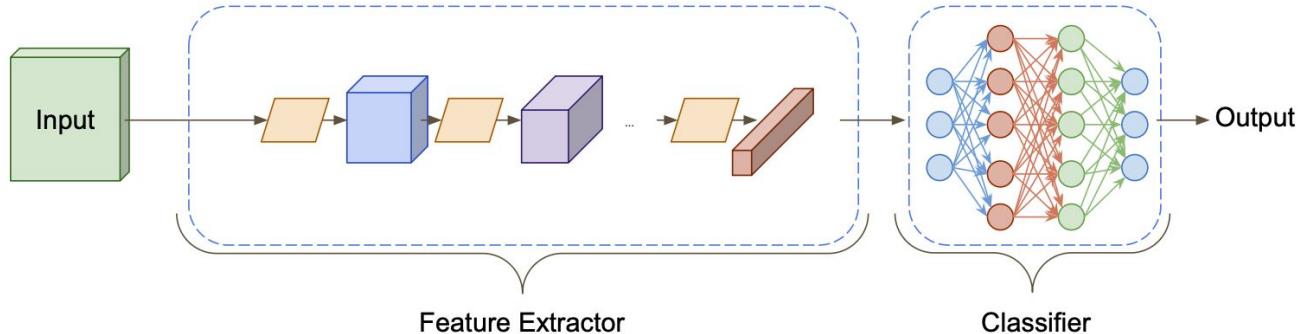
Note that:

After softmax sum of the output vector is 1. Result is more like a PDF

In practice:

Top label or top n labels are used to assess the success of the trained network

More on CNNs...



Following links are provided to assist you in your project...

and also to show you the breadth and depth of this topic

yet we are touch the very basics in a nutshell today...

- [Deep Lizard tutorials](#)
- [Deep learning course videos @ MIT](#)

Recall: Linear Algebra

Let

$$\mathbf{AB} = \mathbf{A}$$

where \mathbf{B} is not identity and neither matrices are *null*

What are possible \mathbf{B} s?

Among all possible ones, which one(s) are more preferable?

Recall: Linear Algebra

Let

$$\mathbf{AB} = \mathbf{A}$$

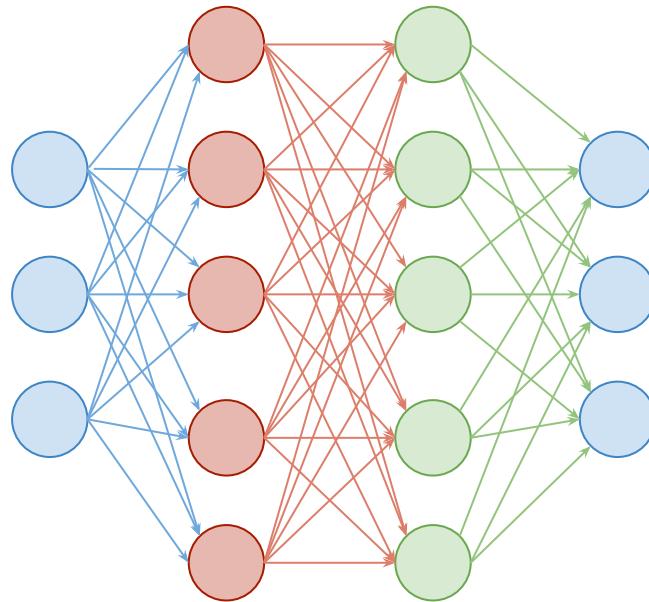
Assume columns of \mathbf{A} are coming from union of 2 subspaces?

A weird network: output learns to match input

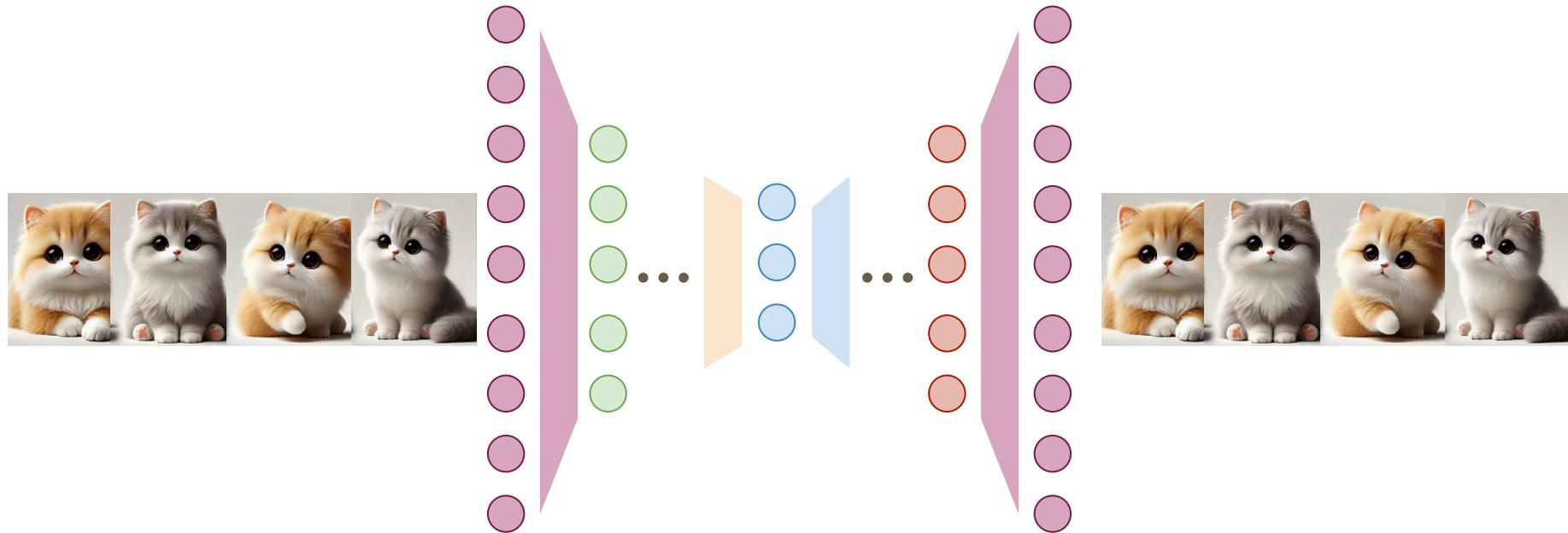
Say what?

Why on earth?

What good does it do?



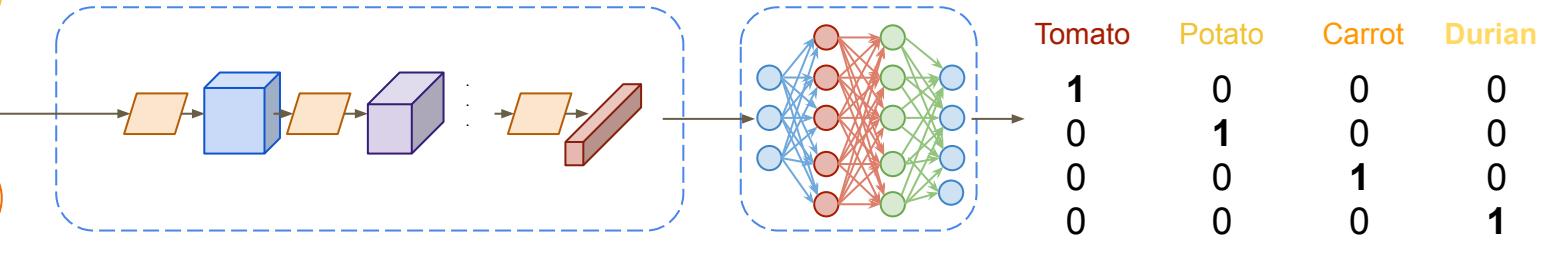
A weird network: How about this one



Transfer Learning: transfer what?

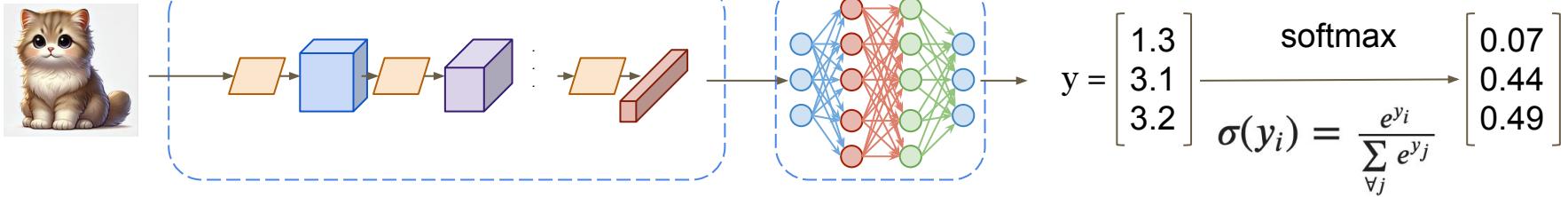


After training with 3 classes, a new class of object can be added
Training with already trained network yields better results



What if: network sees something NEW

Network will still output some value !!!
Shame on you deep network



Hot question:



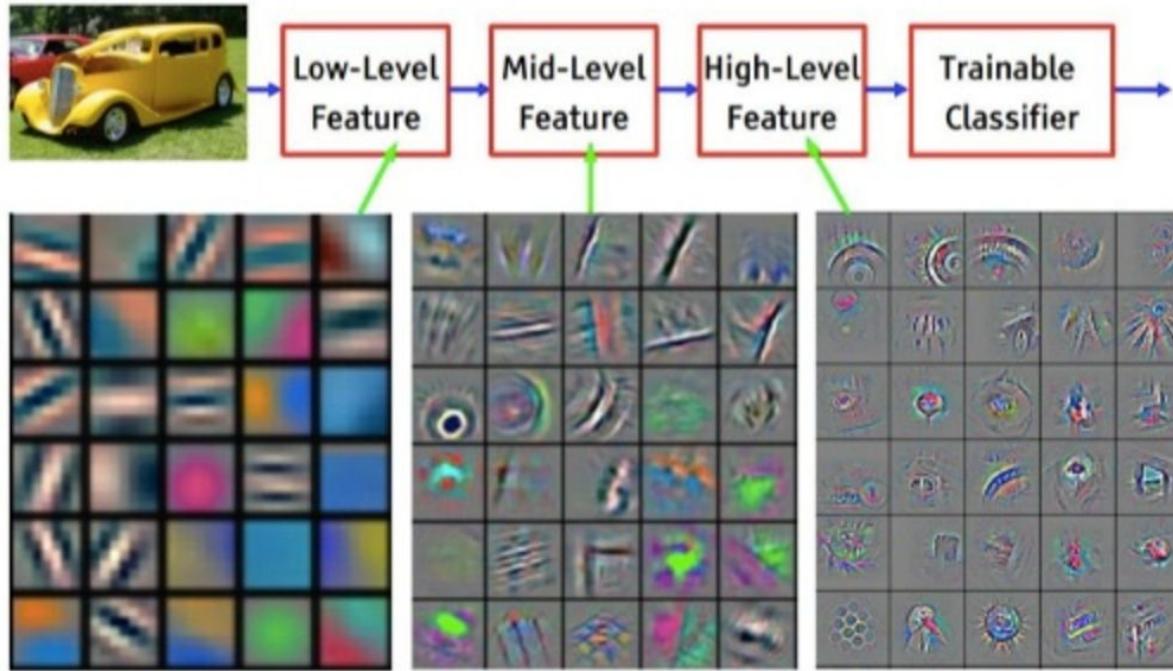
How can we detect new?

A shameless case: adversarial examples

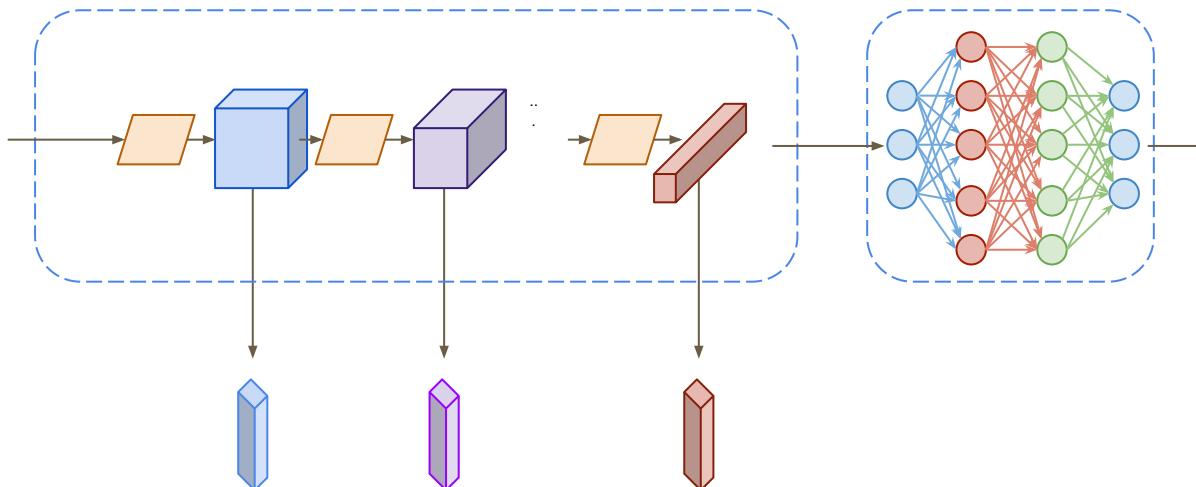


<https://arxiv.org/pdf/1312.6199.pdf>

What is happening: to kernels as it learns



What if: output is not at the output?

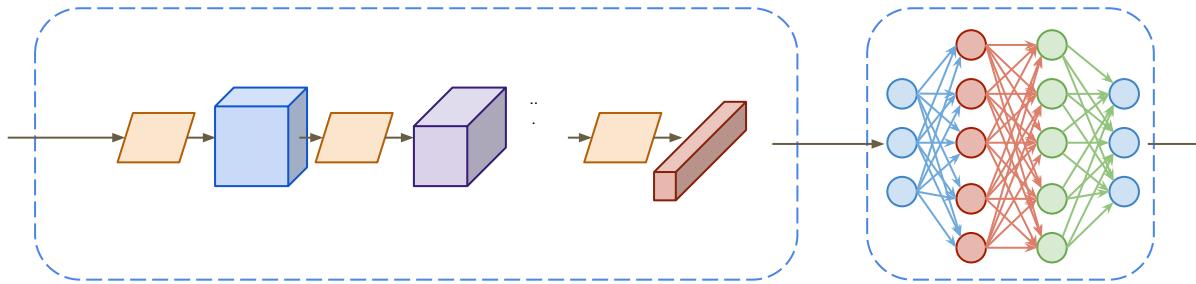


Hot question:

HOT! Can the output layer be too specific?

What if we are not interested in the output but an intermediate output?

What if: a network has seen everything?



Hot question:

Can a CNN be a UNIVERSAL feature extractor?



Self-learning follows...

The project will require you to:

- Define a problem
- Select proper approach
 - Learn tools for the proposed approach
 - Implement and demonstrate

Datasets are already available: Free practice

An easy way to start testing is using existing publicly available datasets.

tensorflow: <https://www.tensorflow.org/datasets/catalog/overview>

pyTorch: https://pytorch.org/tutorials/beginner/basics/data_tutorial.html

NOT to be continued...

This was the last official lecture