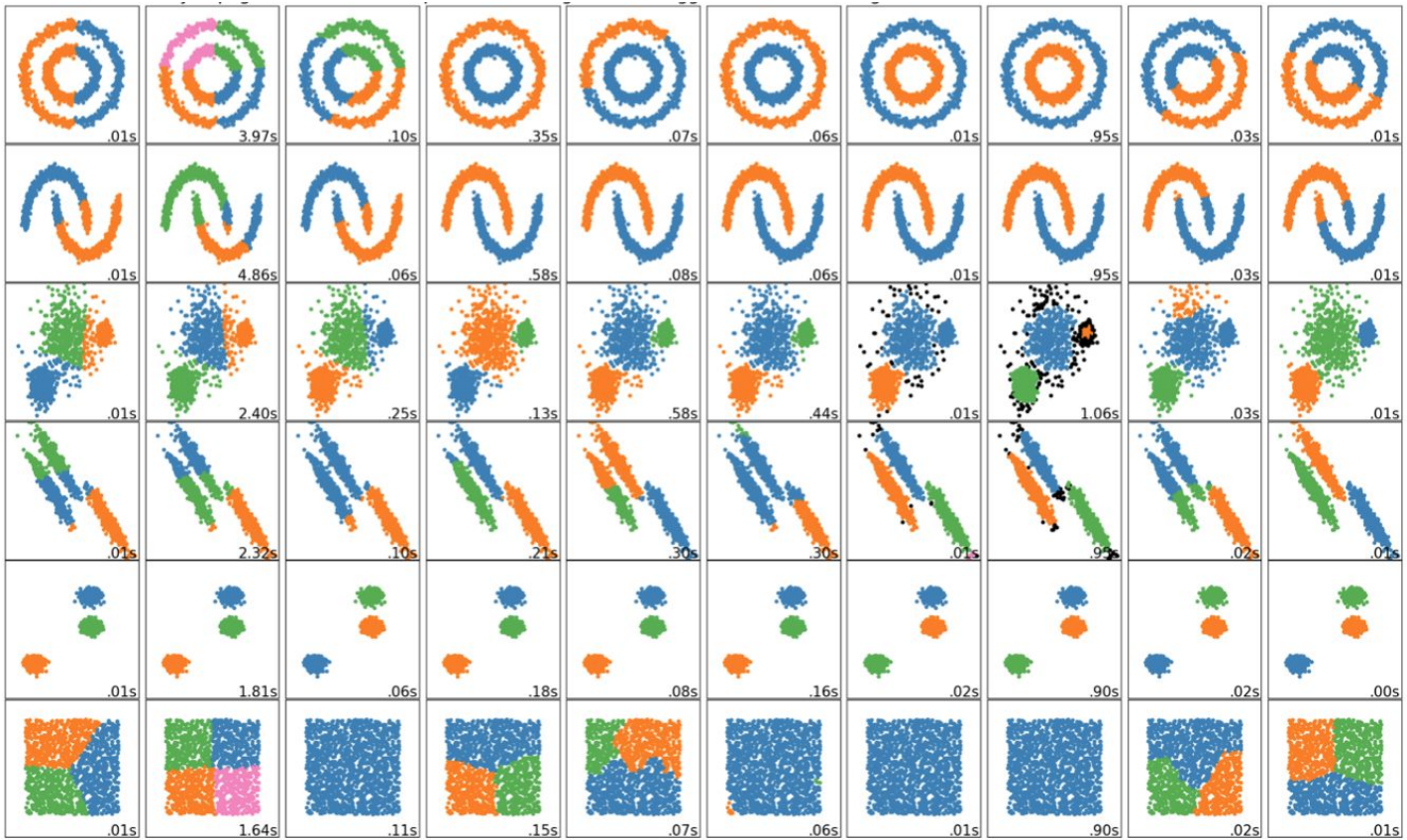
ME 536

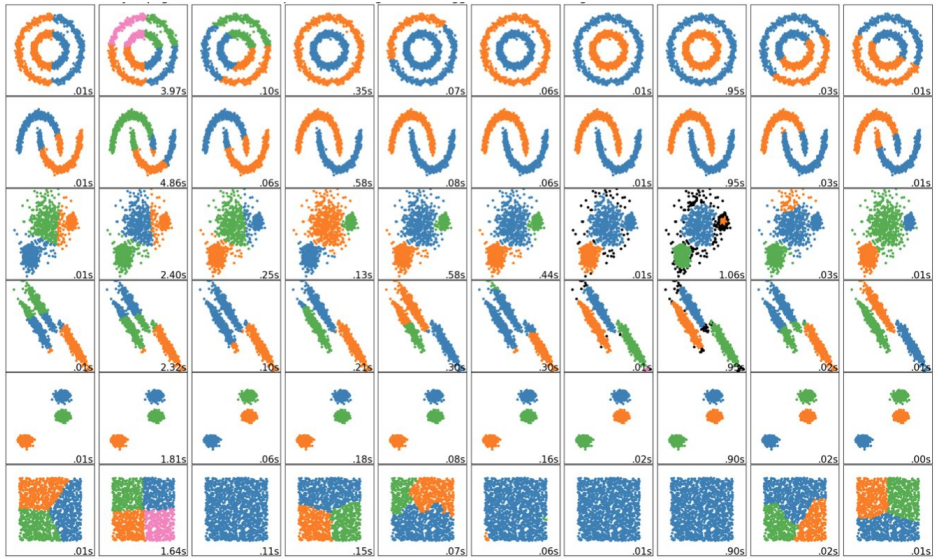
Week 7-8: Clustering

Problems in different forms

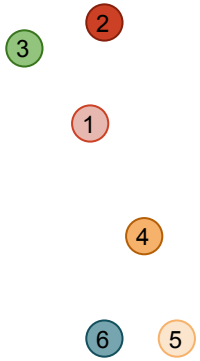


Cluster ‘em all

- 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*

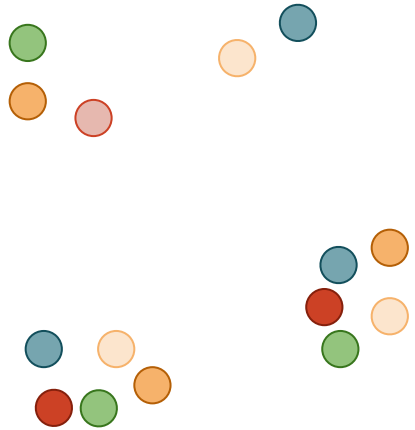
What do you expect for $k = 1, 2, 3$

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:

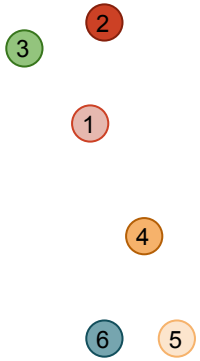
http://user.ceng.metu.edu.tr/~akifakkus/courses/ceng574/k-means/

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

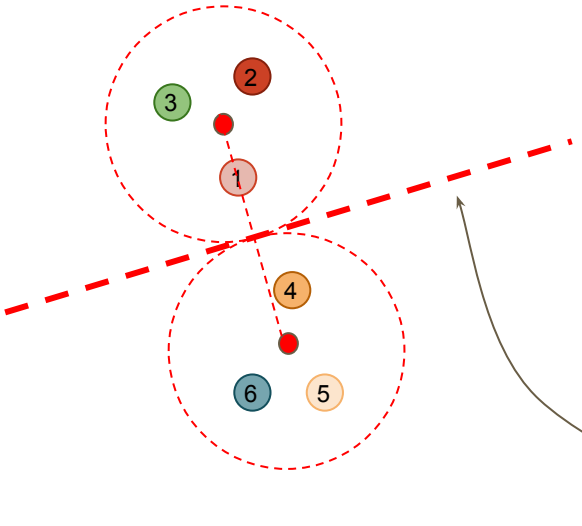
- Better guess than random?
  - Preprocess data ?

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

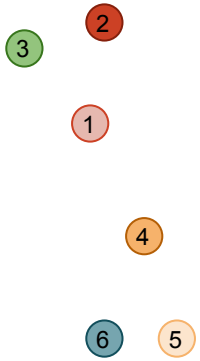
equidistant lines for pairs of CMs: i.e. *Decision boundary*

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

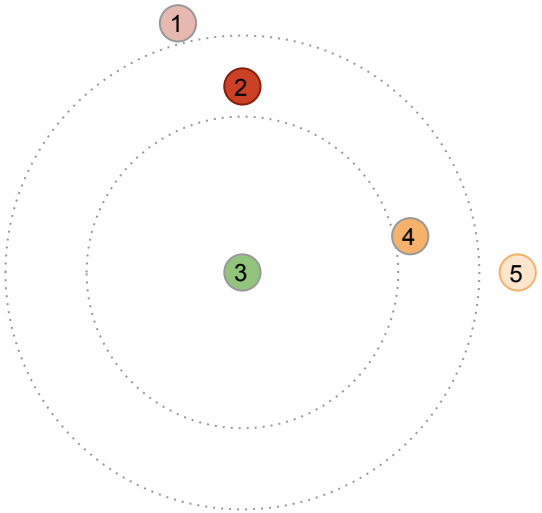
How to improve k-means initial guesses

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:

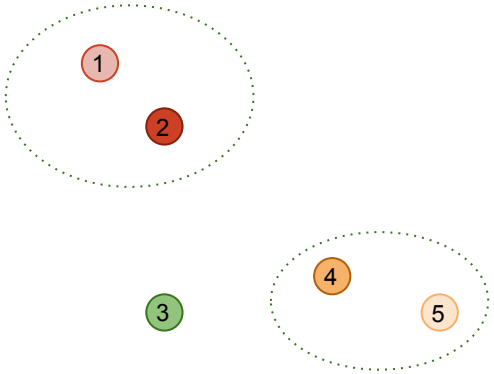
- Stop when there is 1 cluster left ??? Genius why not put them into a single cluster to start with?
- What does *closest* mean?

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

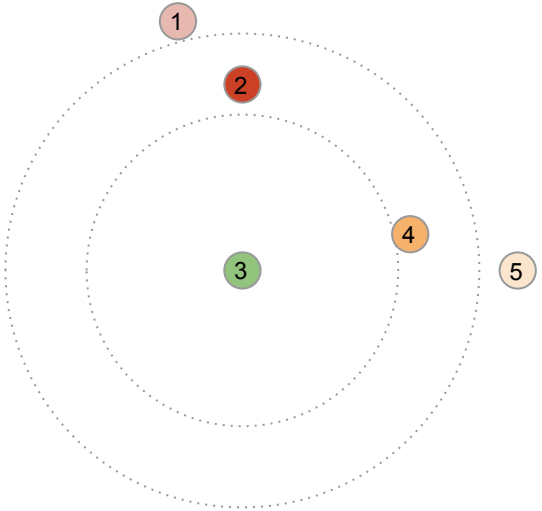
Algorithm:

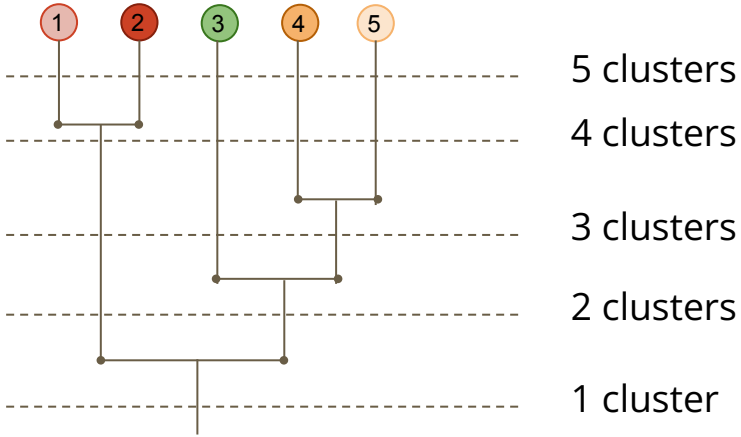
- Assign each datapoint in a cluster
- Join *closest* two clusters
- **Stop when there is 1 cluster left**

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

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

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

Distance between cities

City	ANKARA	BALIKESİR	BİLECİK	BOLU	BURSA	ÇANKIRI	KIRŞEHİR	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
KIRŞEHİR	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

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

Cluster into 2: using k-Means

How will you represent a city as a data point?

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

Distance between cities

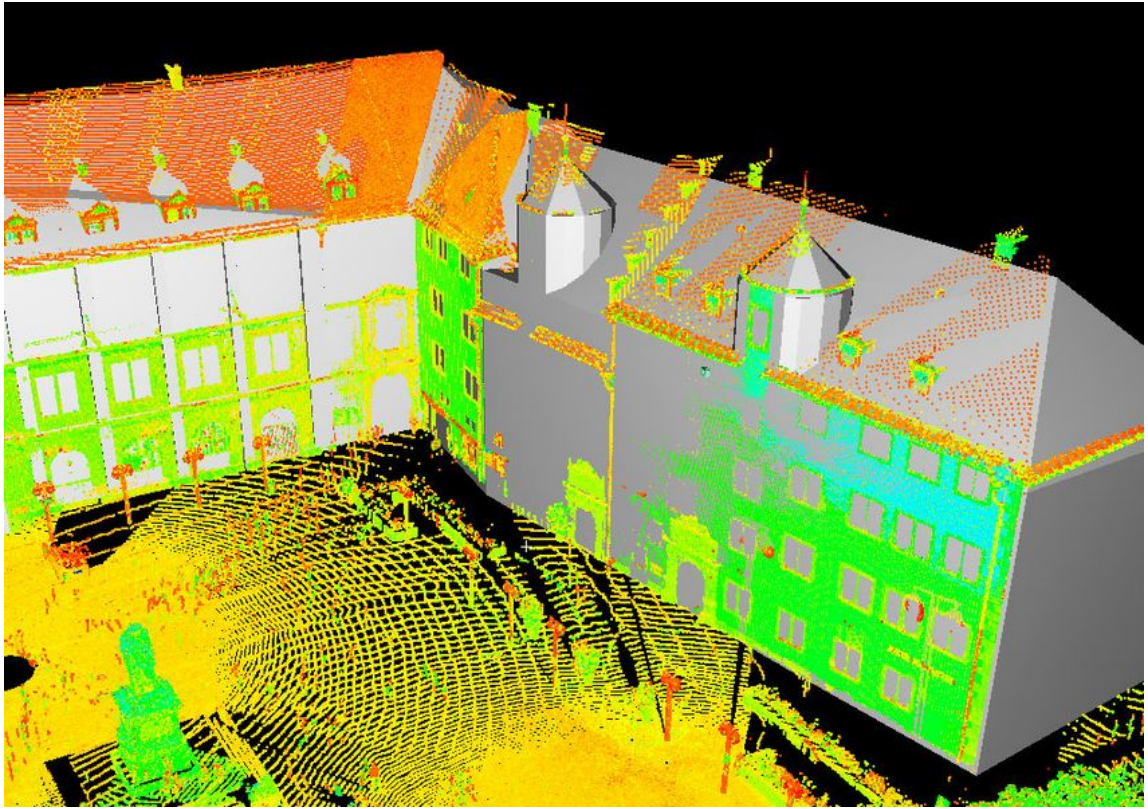
City	ANKARA	BALIKESİR	BİLECİK	BOLU	BURSA	ÇANKIRI	KIRŞEHİR	KONYA	AKSARAY
ANKARA	0	536	316	191	385	130	184	258	225
BALIKESİR	536	0	245	422	151	655	706	551	693
BİLECİK	316	245	0	213	94	446	486	421	526
BOLU	191	422	213	0	271	233	378	456	423
BURSA	385	151	94	271	0	504	555	490	595
ÇANKIRI	130	655	446	233	504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



Becker, Susanne & Haala, Norbert. (2007). Refinement of Building Facades by Integrated Processing of LIDAR and Image Data. In: Proceedings of ISPRS Workshop on Photogrammetric Image Analysis, Munic, Germany, September 19-21, 2007, pp. 1-6. 36.

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

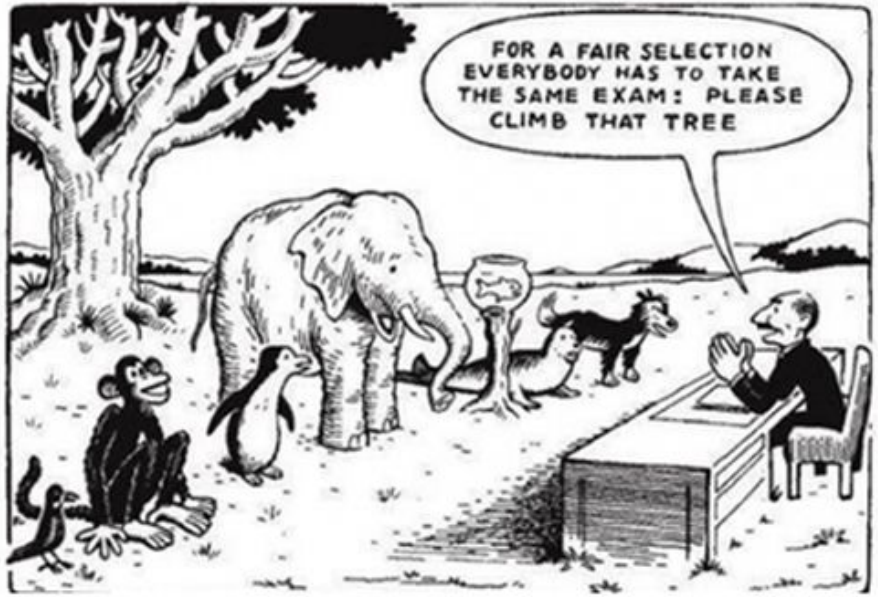
Larger range for a feature might dominate

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 values have **zero-mean and unit standard deviation**

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

just normalization

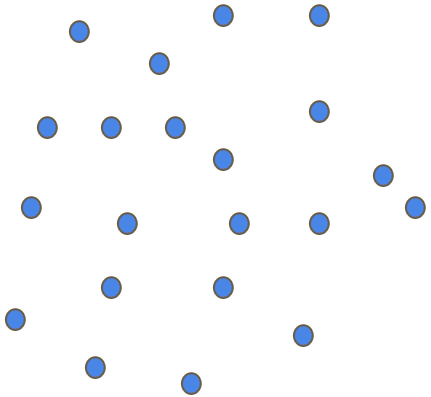
all values values have **zero-mean** scaled *over the range of values*

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

Example - clusters live separately in their own ways?

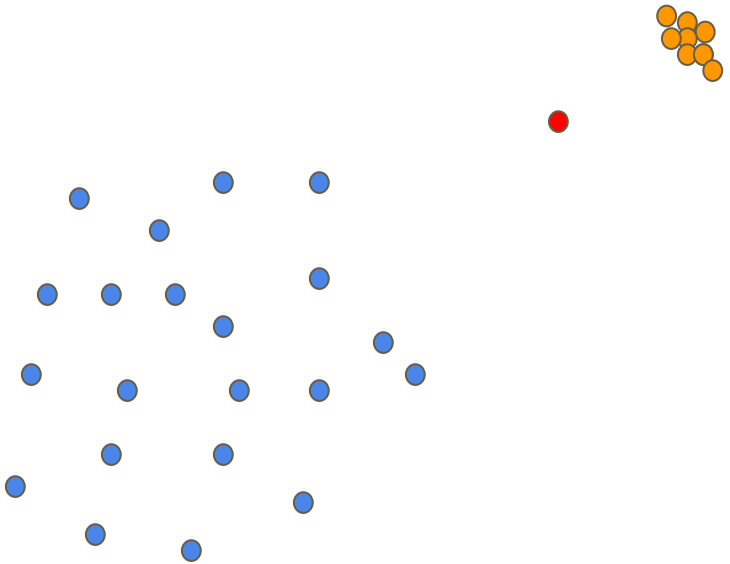
Moving beyond ℓ_1 , ℓ_2 etc

How many clusters do you see?

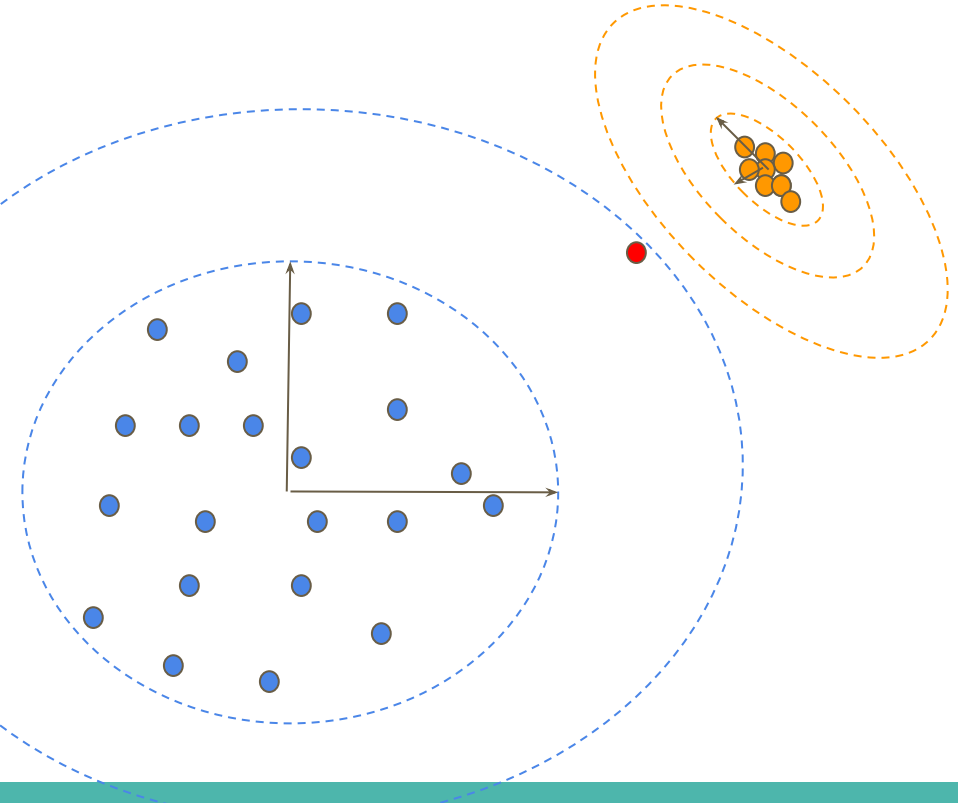


Mahalanobis distance - say who

Where does the new point belong to?



such as - Mahalanobis distance - say who



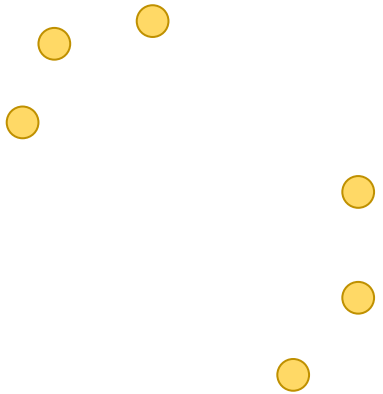
What if distances are **normalized** along **principal axis**?

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



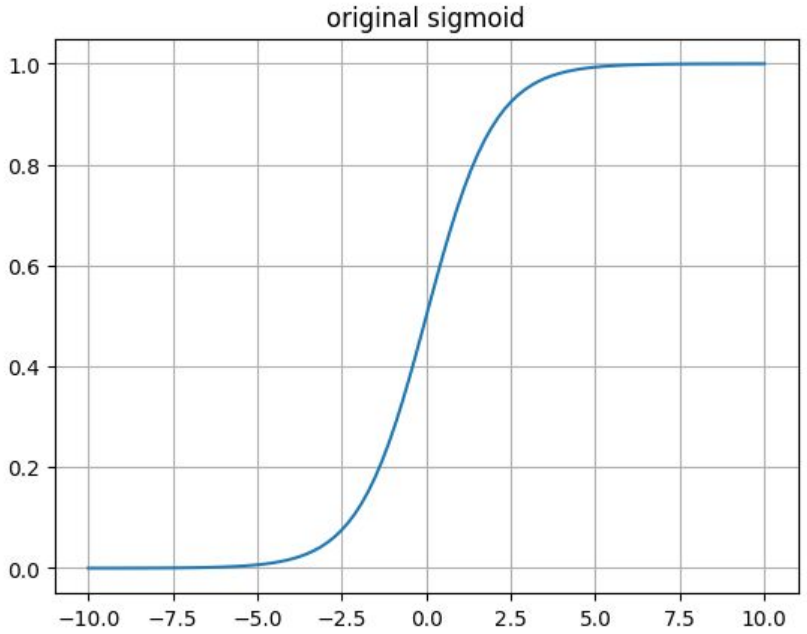
How do I write a function that switch between

0 and 1?

A useful function: Sigmoid

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

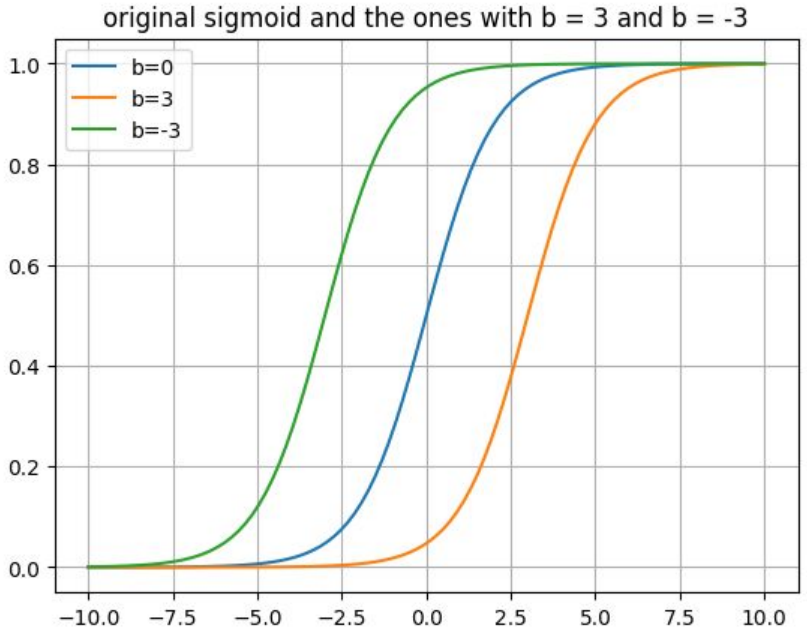
In its original form it is a switching function, similar to a step function, but smooth



A useful function: Sigmoid

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

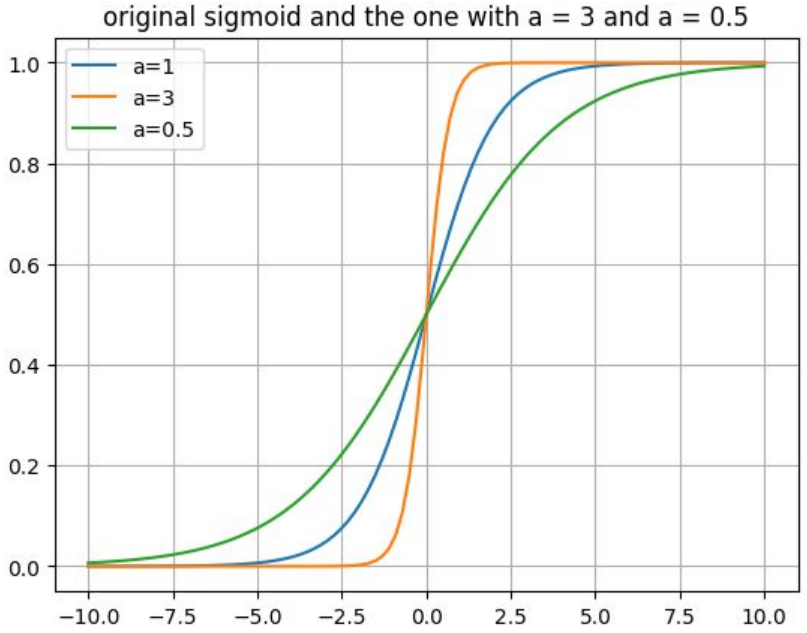
What if the transition should be somewhere else?



A useful function: Sigmoid

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

What if the transition should be faster?



A useful function: Sigmoid

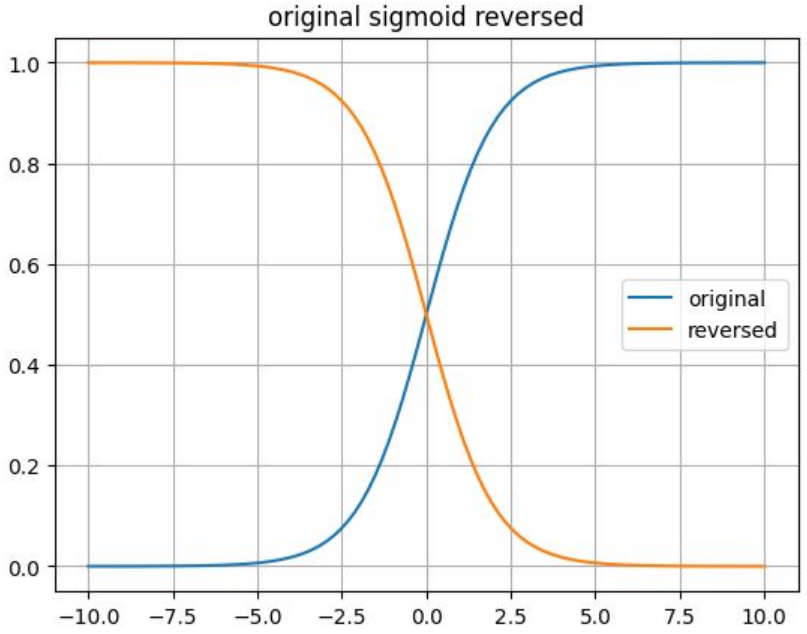
All together

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

A useful function: Sigmoid

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

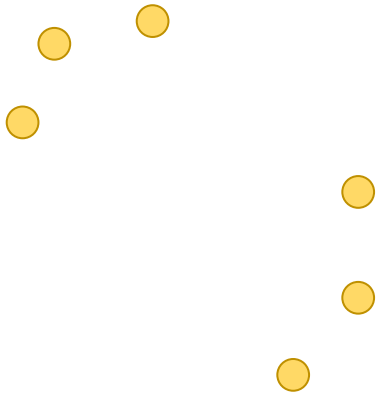
What if the switching logic should be reversed?



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

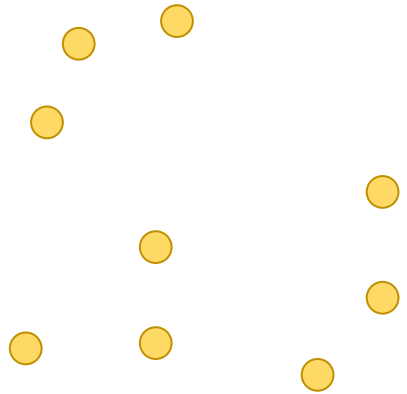
Connects 2 ***adjacent*** vertices → **edge**

Connects a vertex to itself → self-loop

No edge → no relationship

Edges might be directed → directed graph

Otherwise → **undirected graph**



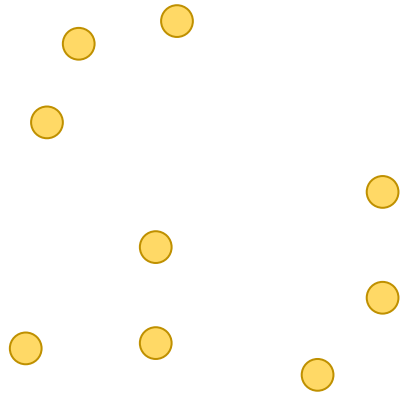
Edges might have (*commonly positive*) values, if not, edge values are assumed to be equal (1)

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 $V=\{v_1, v_2, ..., v_n\}$ and edges $E=\{e_{12}, e_{23}, ..., e_{mk}\}$.

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

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

Formal Definitions

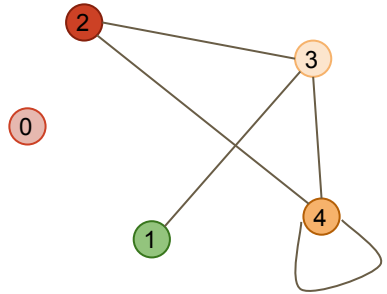
$G=\{V,E\}$ defines a graph with vertices $V=\{v_1, v_2, ..., v_n\}$ and edges $E=\{e_{12}, e_{23}, ..., e_{mk}\}$.

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!

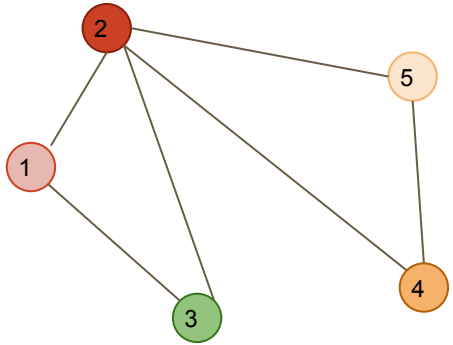


Formal Definitions

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

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 →

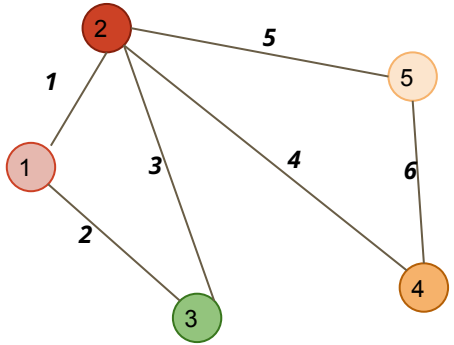
$$L := D - A$$

Formal Definitions

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

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

$$J_{6x5} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$



Laplacian Matrix of G → $L := J^T J$

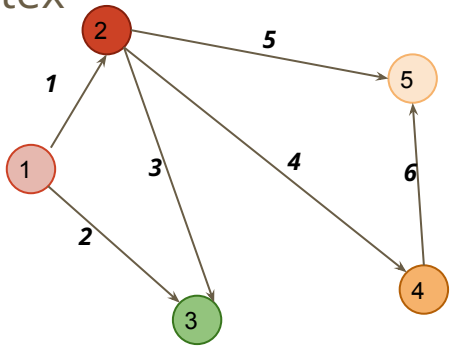
Formal Definitions

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

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

-1 for the left vertex, 1 for the arrived vertex

$$J_{6x5} = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$



Laplacian Matrix of G →

$$L := J^T J$$

Formal Definitions

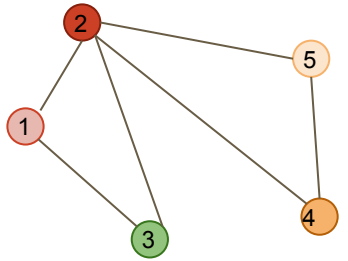
$G=\{V,E\}$ is a graph with vertices $V=\{v_1, v_2, ..., v_n\}$ and edges $E=\{e_{12}, e_{23}, ..., e_{mk}\}$.

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**

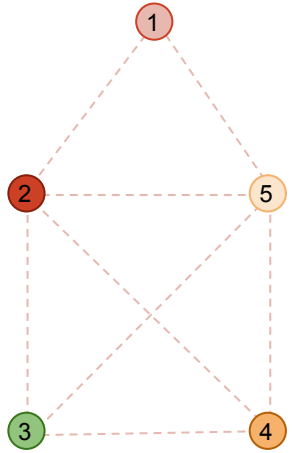


Path length: Sum of edge values.
In case all edges have the same value, *number of edges on the path.*

Puzzle from childhood years

Can you connect all vertices with a single:

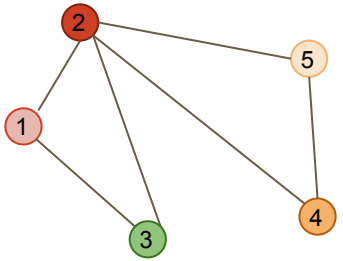
- Path = ?
- Trail = ?
- Walk = ?



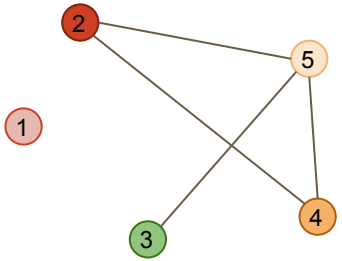
Formal Definitions

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

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 $V=\{v_1, v_2, ..., v_n\}$ and edges $E=\{e_{12}, e_{23}, ..., e_{mk}\}$.

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

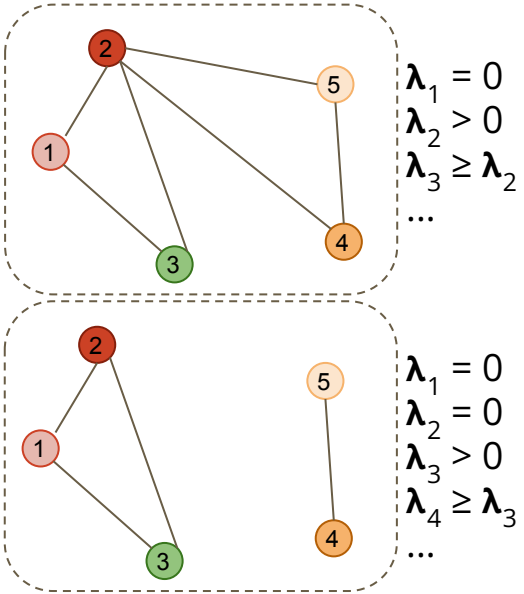
***L is beautiful*:**

Eigenvalues of L are first sorted in ascending order

Real eigenvalues with orthogonal eigenvectors

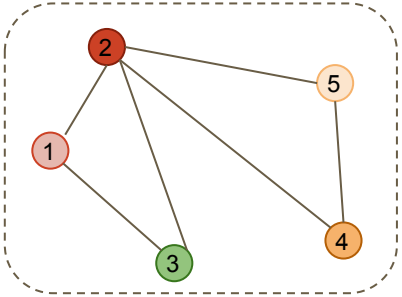
Number of eigenvalues that are 0

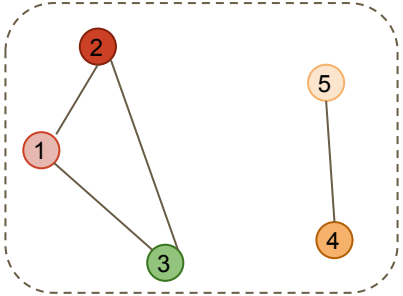
→ number of connected sub-graphs



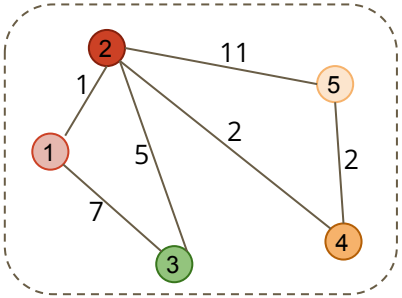
NOTE: Definition of Laplacian Matrix L is not unique

Self Study: Find the graph Laplacian & eigenvalues





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



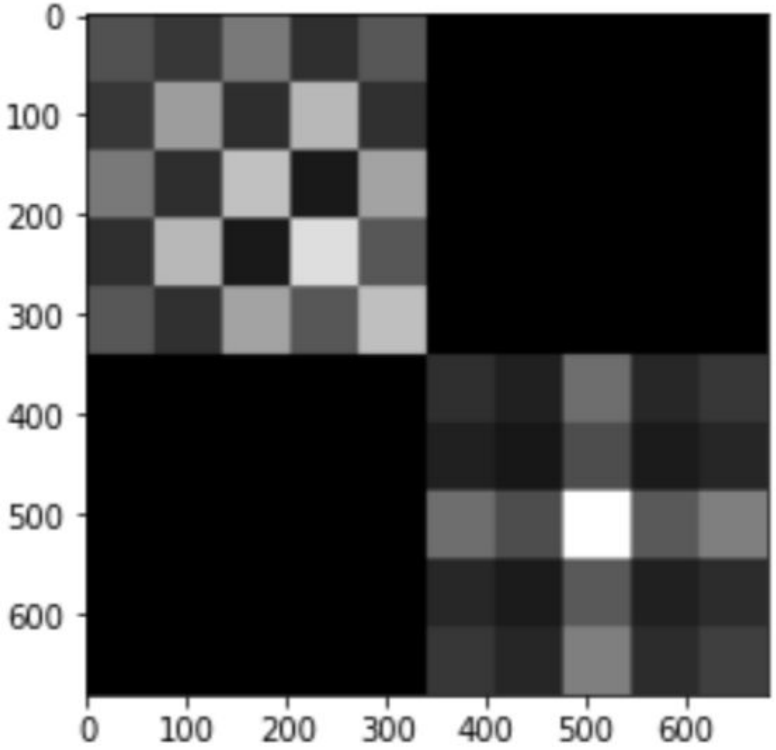
Which edges would you eliminate?

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 $\mathbf{M}$?



Shape Interaction Matrix: SIM

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

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

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

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

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

A good paper: https://arxiv.org/pdf/1509.02649.pdf

One use of SIM:

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

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:

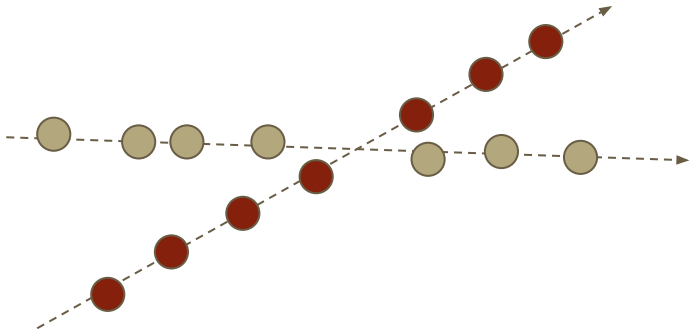
- 2 subspaces: 1 line & 1 plane
- 2 subspaces: 2 lines
- 3 subspaces: 3 lines

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

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

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

Why does SIM work: Some linear algebra

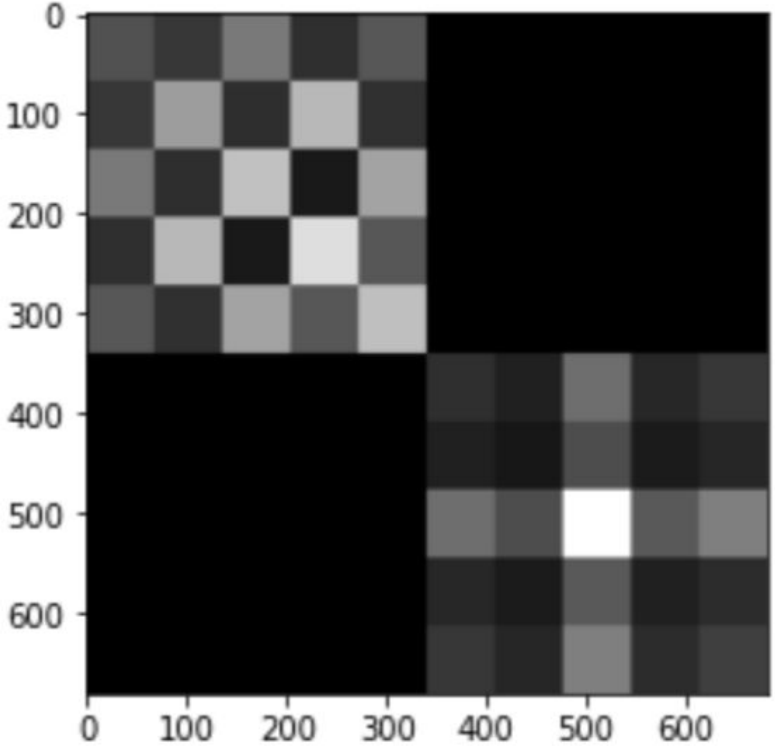
Theorem 3

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* $\mathscr{U}$ *as in Assumptions 1. Let the skinny SVD of* $\mathbf{W}$ *be given by* $\mathbf{W} = U\Sigma V^T$, *and define* $Q = abs(VV^T)$. *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.
- Calulate 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 $0 = \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \ldots \lambda_n$.
- 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 $\mathbf{R}$.

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

- 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.

Sci-kit has it implemented, wiki for sure, geeks are welcomed, A good resource online, And a good paper on the matter

Trial and error: more iterations

There is noise

No noise models advanced filters etc in play

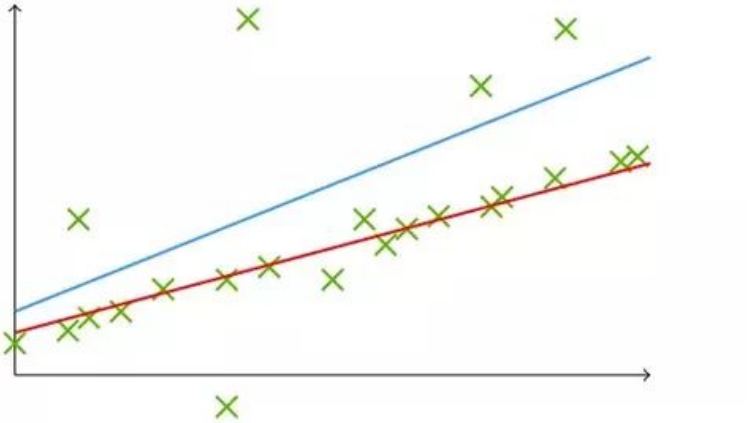
I need a quick, practical approach to find structures in data?

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

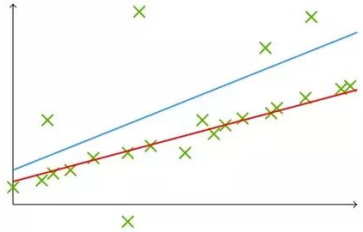


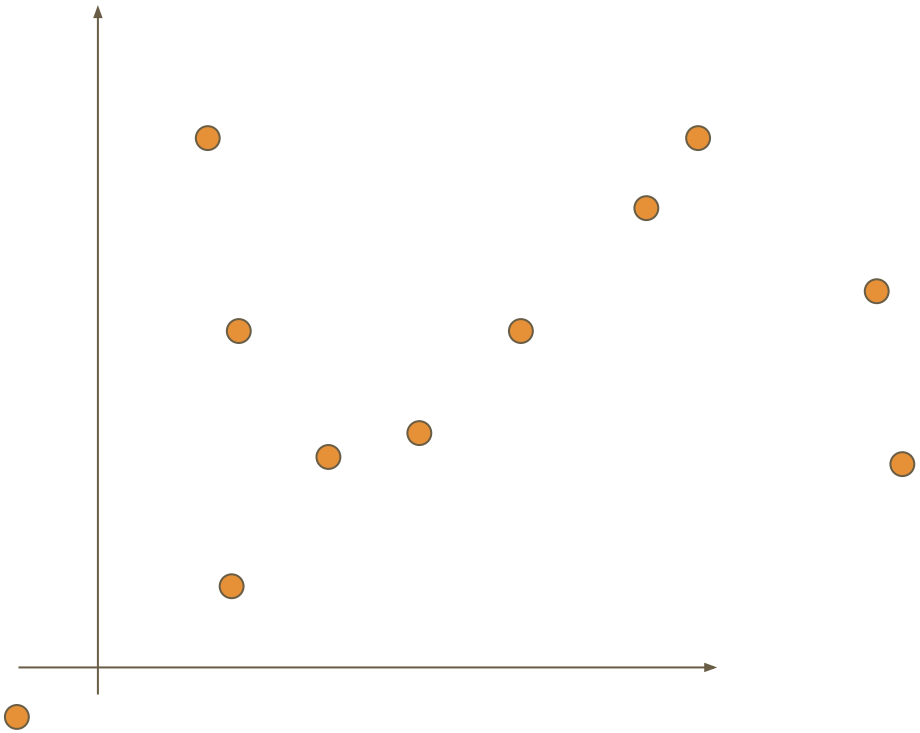
Outliers: An iterative perspective

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





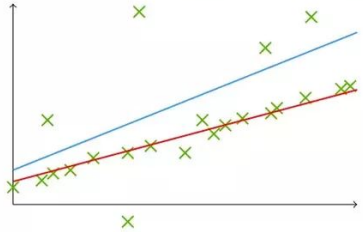
1. Sample 2 points randomly

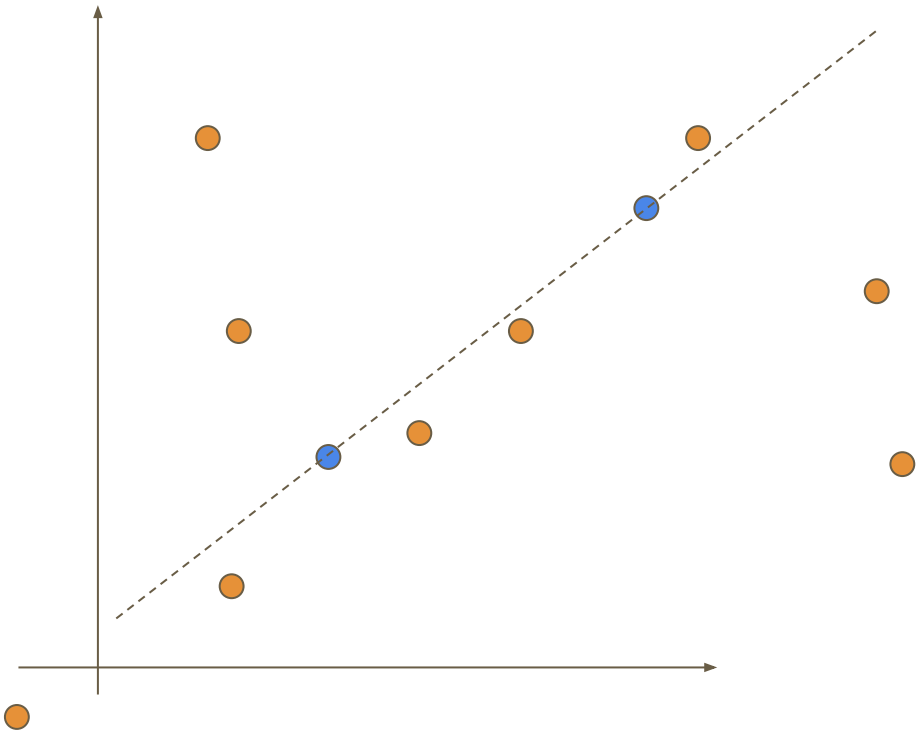
Outliers: An iterative perspective

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





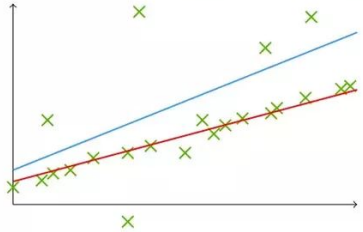
1. Sample 2 points randomly
2. Find the line

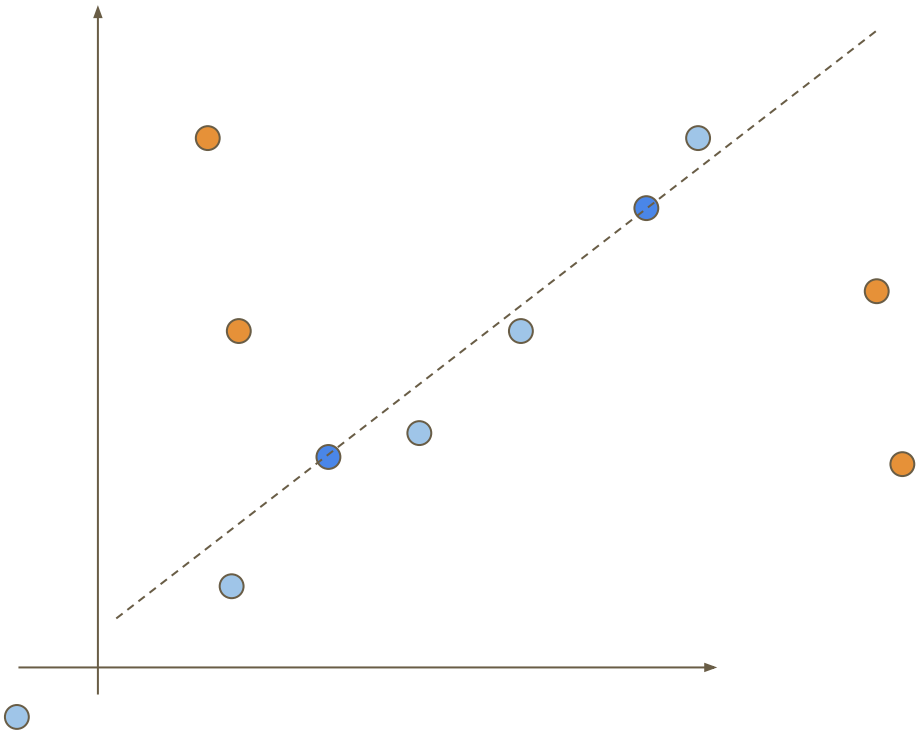
Outliers: An iterative perspective

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

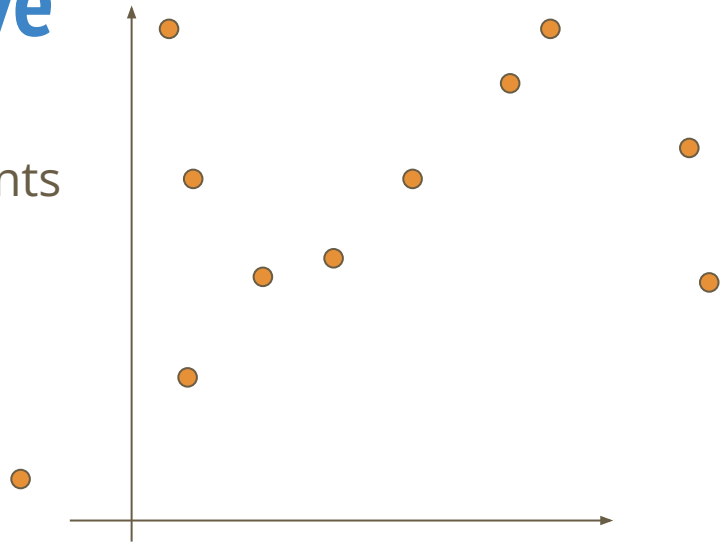




1. Sample 2 points randomly
2. Find the line
3. Find the **good** points

Repeat ***N*** times or until?

RANSAC: An iterative perspective

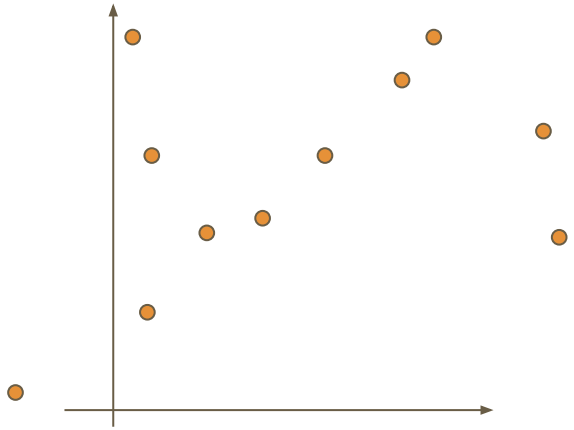


Select a model and determine number of data points needed for the model: **s**

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

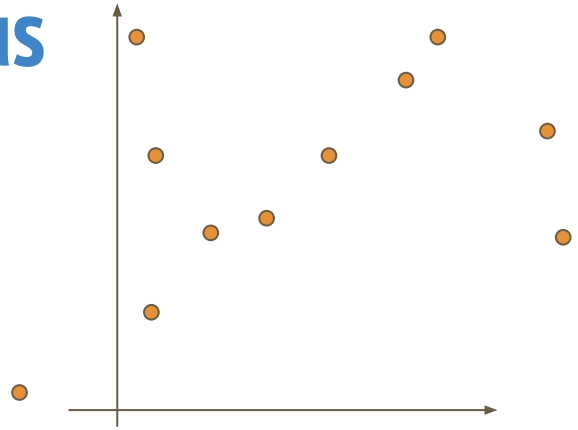
RANSAC: An iterative perspective



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

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
    - 99.9% chance that I will end up with the best model: ***p*** = 0.999
4. ***e***: probability that a point is an outlier
    - ***e*** = 0.15 : 15% of my data is contaminated with outliers
    - 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 → 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