Unsupervised Learning: Clustering, Kmeans, hierarchical clustering

James G. Shanahan [1,2,3]

[1]Church and Duncan Group,

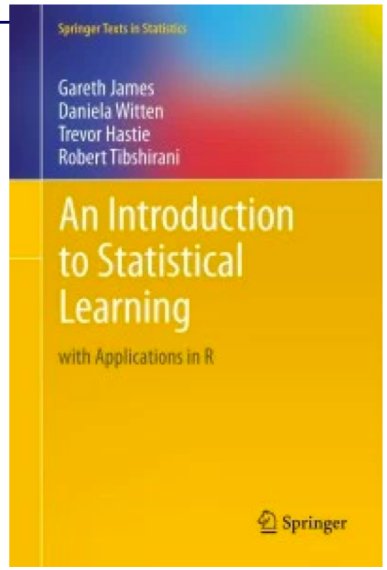
[2]Information School, UC Berkeley

[3]School of Informatics, Computing and Engineering, Indiana University

Reading material

ISLR Book, An Introduction to Statistical Learning with Applications in R (ISLR), Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani, 2013 Springer. Download for free at:
http://www-bcf.usc.edu/~gareth/ISL/

- **Chapter 10: Unsupervised Learning**



See The IR Book

- http://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html

Clustering wikipedia page

- **The following page provides a great and detailed overview of clustering**
  - https://en.wikipedia.org/wiki/Clus
  -

Kmeans Notebooks

- **Review Kmeans notebook (in MRJob) and improve quality of code**
  - http://nbviewer.jupyter.org/urls/dl.dropbox.com/s/9ehy0bg6r7bstm9/MrJobKmeans.ipynb
- **Extensive notebook on Kmeans (single core to MapReduce)**
  - http://nbviewer.jupyter.org/urls/dl.dropbox.com/s/vu20u1qz92andwx/K_Means_Unit_Test_Notebook.ipynb
- **Plot the knee curve (Kmeans with different numbers of clusters K)**
  - https://www.dropbox.com/s/hpjr0kmrkkmbk5p/D3.%20K-Means%20Clustering%20Analysis.ipynb?dl=0
- **Different version of Kmeans in MrJob (review)**
  - https://github.com/uchicago-cs/cmsc12300/tree/master/examples/data_analysis/src/cs123/mrjob

Kmeans Discussion and presentation

- **Chapter 8:**
- **Tan, Steinback, Kumar**
- **https://www.dropbox.com/s/wson7b4b1lizhj9/Kumar-Book-Clustering-kmeans-sim-measures-ch8.pdf?dl=0**

Live Session Outline

- **Introduction**
- **Unsupervised Learning**
- **Clustering**
  - Kmeans
    - Flat Clustering via Kmeans graphically
    - Flat Clustering via Kmeans in code
    - Hyperparameters: Different initializations
  - Hierarchical clustering
- **Similarity measures**
- **Evaluating Clustering**
- **Relatives of Kmeans, e.g., Expectation Maximization**
- **Summary**

From supervised to unsupervised learning

- In the previous lectures, we used supervised learning techniques to build machine learning models using data where the answer was already known—the class labels were already available in our training data.
- In this part of the course, we will switch gears and explore cluster analysis, a category of unsupervised learning techniques that allows us to discover hidden structures in data where we do not know the right answer upfront.
- The goal of clustering is to find a natural grouping in data so that items in the same cluster are more similar to each other than to those from different clusters.

Clustering Algorithms

- **Flat/Non-Hierarchical Clustering**
  - Exclusive Clustering (k-means) [In R, cluster package]
  - Overlapping Clustering (fuzzy c means) [In R, e1071]
  - Probabilistic Clustering (E.g., EM Mixture of Gaussians)

- **Hierarchical Clustering**
  - Agglomerative approach (bottom-up)
    - In R: hclust() and agnes()
  - 2. Divisive approach (top-down)
    - In R: diana()

Live Session Outline

- **Introduction**
- **Unsupervised Learning**
- **Clustering**
  - Kmeans
    - Flat Clustering via Kmeans graphically
    - Flat Clustering via Kmeans in code
    - Hyperparameters: Different initializations
  - Hierarchical clustering
- **Similarity measures**
- **Evaluating Clustering**
- **Relatives of Kmeans, e.g., Expectation Maximization**
- **Summary**

From supervised to unsupervised learning

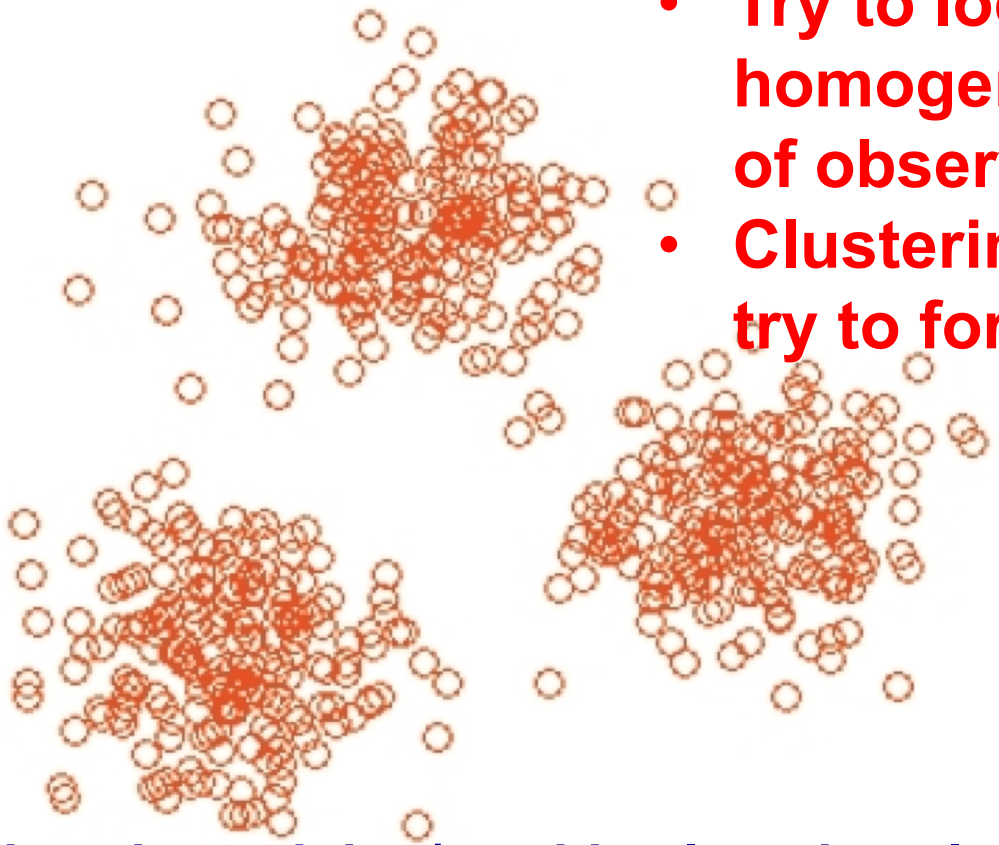
- The goal of clustering is to find a natural grouping in data so that items in the same cluster are more similar to each other than to those from different clusters.
- Dimensionality reduction

Unsupervised learning is broad

- **Unsupervised machine learning** is the machine learning task of inferring a function to describe hidden structure from "unlabeled" data (a classification or categorization is not included in the observations).
- Since the examples given to the learner are unlabeled, there is no evaluation of the accuracy of the structure that is output by the relevant algorithm—which is one way of distinguishing unsupervised learning from supervised learning and reinforcement learning.
- **Unsupervised learning is broad**
  - A central case of unsupervised learning is the problem of density estimation in statistics,[1] though unsupervised learning encompasses **many other problems (and solutions) involving summarizing and explaining key features of the data.**
  - **Dim reduction**

Clustering Algorithms

Many data points,



- Try to locate homogenous groups of observations
- Clustering algorithms try to formalize this

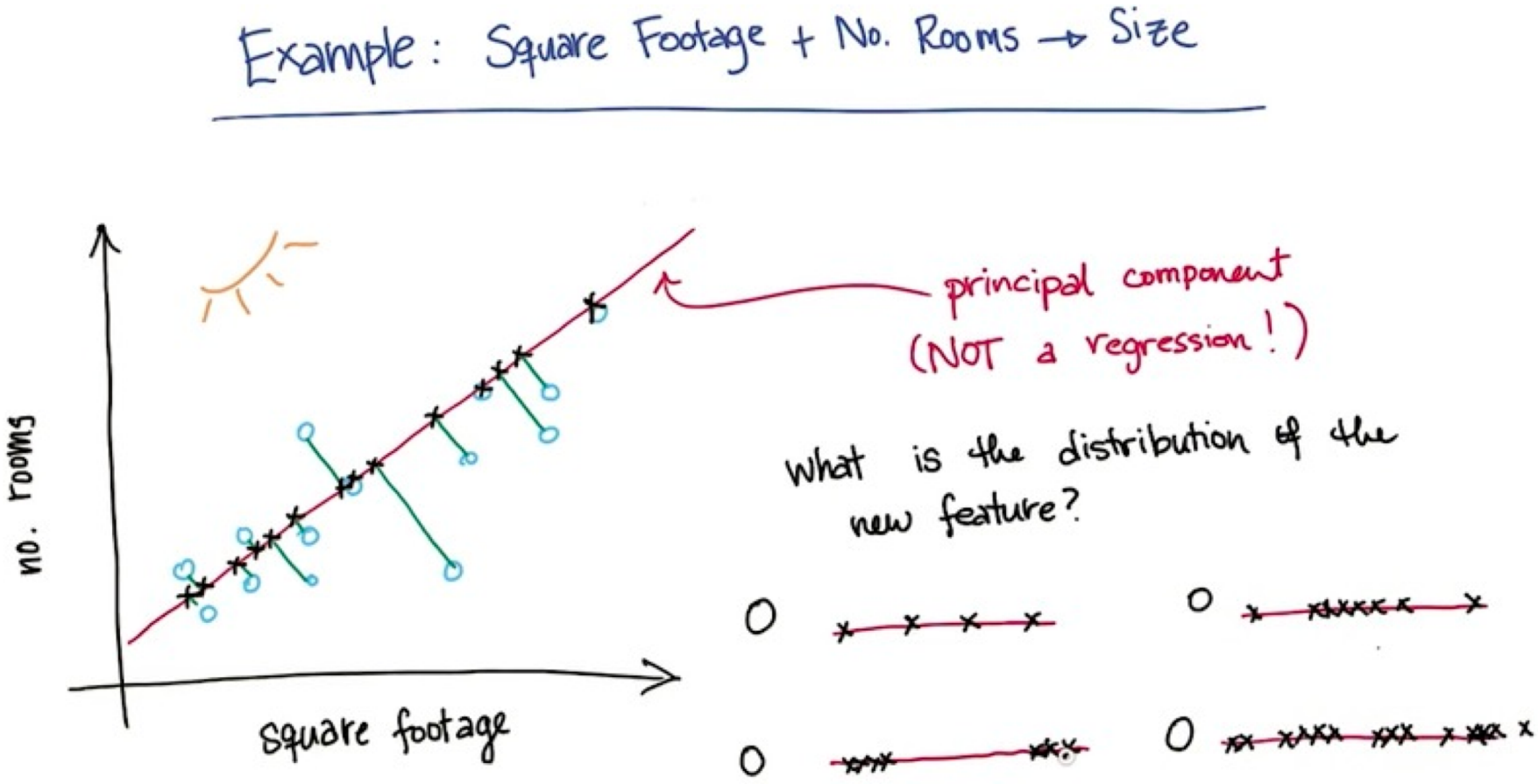
- Clustering is a data mining/machine learning algorithm used to cluster observations into groups of related observations without any prior knowledge of those relationships
- Try to locate homogenous groups of observations
- Clustering algorithms try to formalize this

Unsupervised learning approaches

- Approaches to unsupervised learning include:
- Clustering
  - k-means
  - mixture models
  - hierarchical clustering,[2]
- Anomaly detection
- Neural Networks
  - Hebbian Learning
  - Generative Adversarial Networks
- Approaches for learning latent variable models such as
  - Word2Vec
  - Expectation–maximization algorithm (EM)
  - Method of moments
  - Blind signal separation techniques, e.g.,
    - Principal component analysis,
    - Independent component analysis,
    - Non-negative matrix factorization,
    - Singular value decomposition.[3]

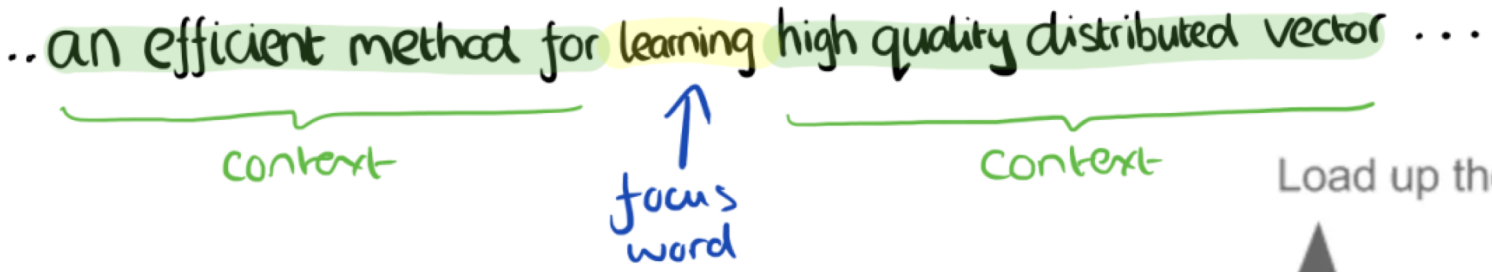
The k-means algorithm belongs to the category of prototype-based clustering. We will discuss two other categories of clustering, hierarchical and density-based clustering,

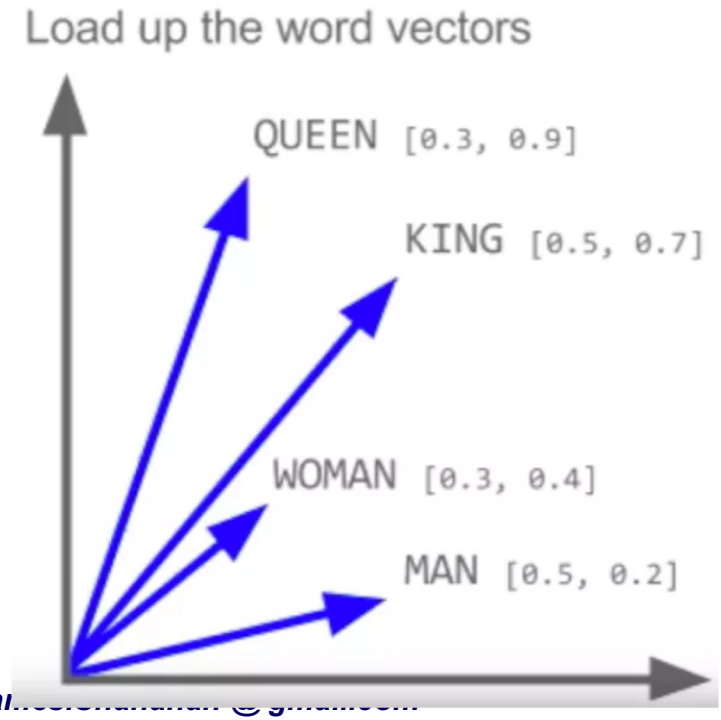
PCA


Example: Square Footage + No. Rooms → Size
no. rooms
square footage
principal component
(NOT a regression!)
What is the distribution of the new feature?

Word2Vec

- We have to convert these words to vectors via word embedding. Word embedding provides a lower-dimension vector representation of words while preserving the relationship / meaning between





Clustering

- In this lecture, we will learn about one of the most popular clustering algorithms, k-means, which is widely used in academia as well as in industry.
- Clustering (or cluster analysis) is a technique that allows us to find groups of similar objects, objects that are more related to each other than to objects in other groups.
- Examples of business-oriented applications of clustering include the grouping of documents, music, and movies by different topics, or finding customers that share similar interests based on common purchase behaviors as a basis for recommendation engines.

Clustering

Within group similarity

- Clustering is an unsupervised learning activity in that we do have not have labels associated
- Clustering a divide and conquer strategy to thinking about our world of examples
- Clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters).

EDA

- It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, and bioinformatics.

Clustering Algorithms

- Data mining is sorting through data to identify patterns and establish relationships.
- Extract useful knowledge from large datasets!
- Clustering is a data mining (machine learning) technique used to place data elements into related groups without advance knowledge of the group definitions.

Live Session Outline

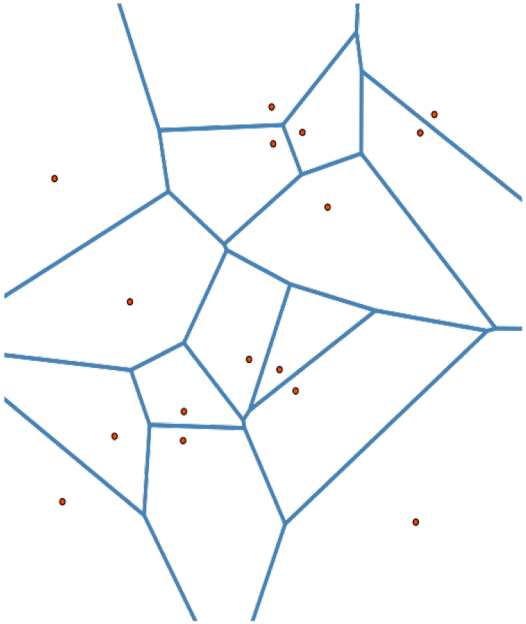
- **Introduction**
- **Unsupervised Learning**
- **Clustering**
  - Kmeans
    - Flat Clustering via Kmeans graphically
    - Flat Clustering via Kmeans in code
    - Hyperparameters: Different initializations
  - Hierarchical clustering
- **Similarity measures**
- **Evaluating Clustering**
- **Relatives of Kmeans, e.g., Expectation Maximization**
- **Summary**

The k-means algorithm is popular

- **The k-means algorithm is extremely easy to implement but is also computationally very efficient compared to other clustering algorithms, which might explain its popularity.**
- **The k-means algorithm belongs to the category of prototype-based clustering.**

Kmeans Intro.

- K-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster.
- This results in a partitioning of the data space into Voronoi cells.

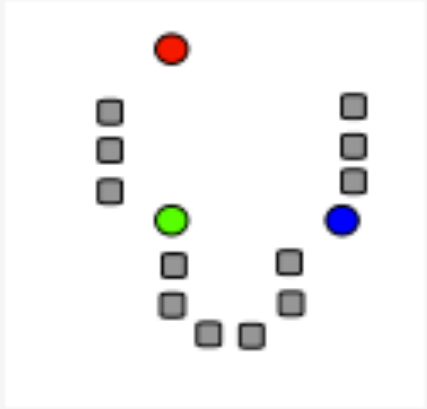


Kmeans Intro.

- **Clustering is computationally difficult (NP-hard);**
- **However, there are efficient heuristic algorithms that are commonly employed and converge quickly to a local optimum.**
- **EM is closely related to Kmeans**
  - These are usually similar to the expectation-maximization algorithm for mixtures of Gaussian distributions via an iterative refinement approach employed by both algorithms.
  - Additionally, they both use cluster centers to model the data; however, k-means clustering tends to find clusters of comparable spatial extent, while the expectation-maximization mechanism allows clusters to have different shapes.
- **The algorithm has a loose relationship to the k-nearest neighbor classifier, a popular machine learning technique for classification that is often confused with k-means because of the k in the name.**

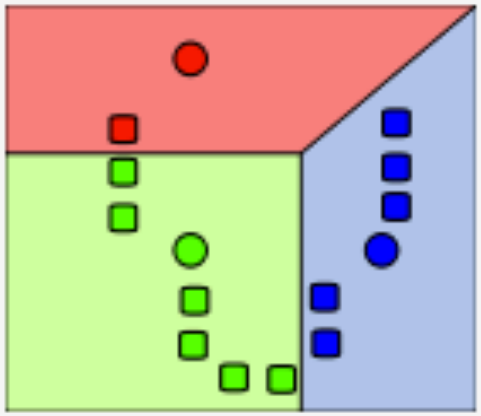
Kmeans Clustering: E-Step

Initialization



1) *k* initial "means" (in this case *k*=3) are randomly selected from the data set (shown in color).

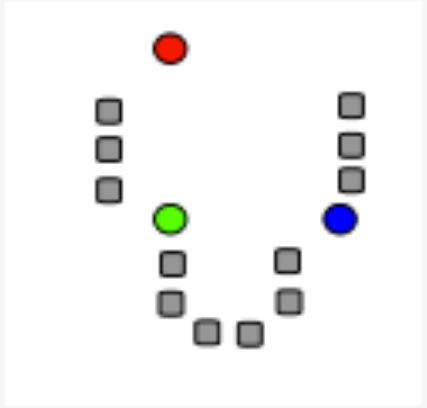
E-Step
"assignment" step



2) *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.

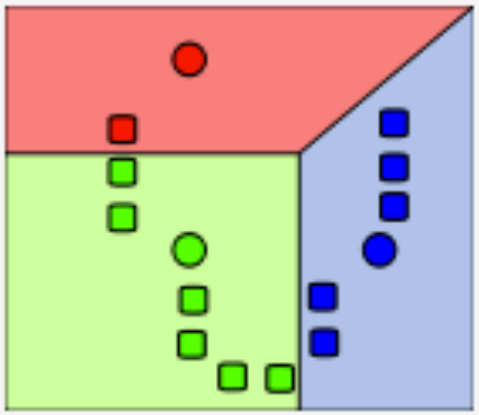
Kmeans Clustering: M-Step

Initialization



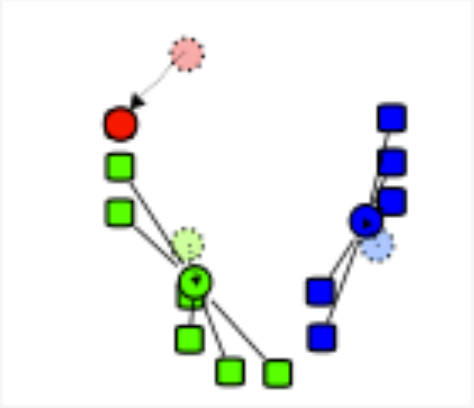
1) *k* initial "means" (in this case *k*=3) are randomly selected from the data set (shown in color).

E-Step
"assignment" step



2) *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.

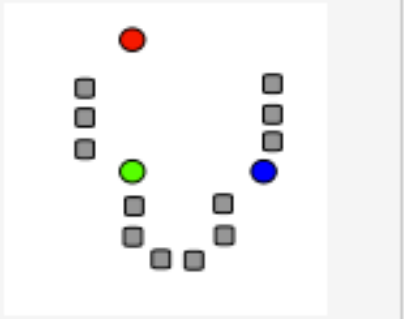
M-Step
update step



3) The centroid of each of the *k* clusters becomes the new means.

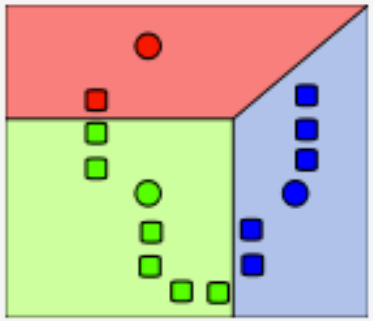
Kmeans Clustering

Initialization



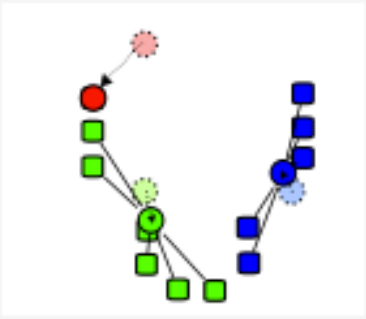
1) *k* initial "means" (in this case *k*=3) are randomly selected from the data set (shown in color).

E-Step "assignment" step



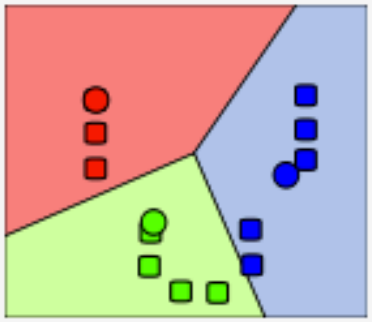
2) *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.

M-Step update step



3) The centroid of each of the *k* clusters becomes the new means.

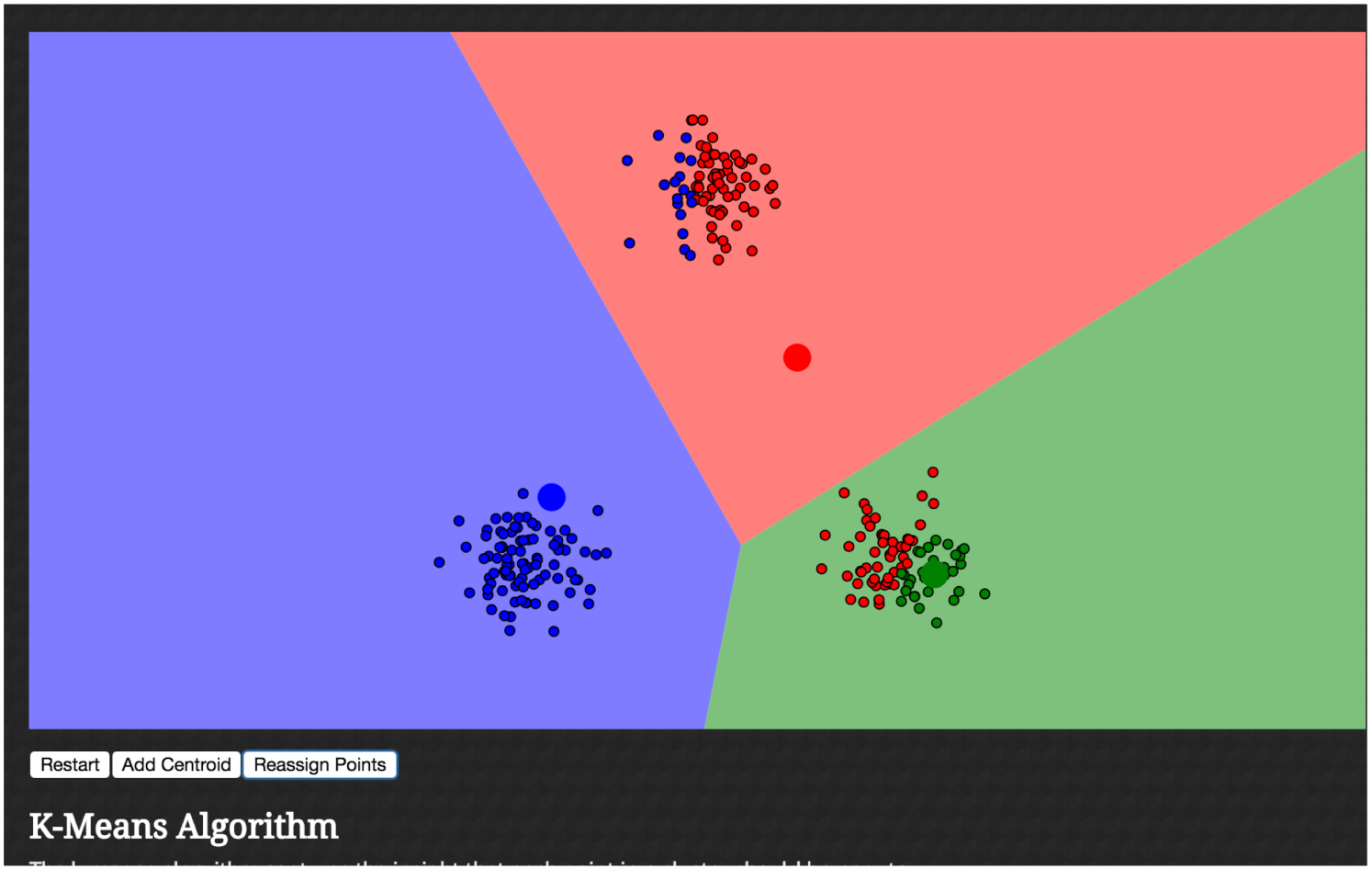
Converged



4) Steps 2 and 3 are repeated until convergence has been reached.

Vizualizing Clustering (Kmeans DEMO)

- http://www.naftaliharris.com/blog/visualizing-k-means-clustering/



K-Means Clustering Algorithm

- **Initialize our K cluster centers**
- **Perform K-Means Loop**

E

  - The "assignment" step is also referred to as expectation step,

M

  - The "update step" as maximization step, making this algorithm a variant of the *generalized* expectation-maximization algorithm.

- **Until Convergence**

K-Means

- **Partition data such that it minimizes the within group sum of squares over all variables**
- **N examples with k clusters**
- **Impractical to explore all possible partitions**

The k-means clustering technique seeks to partition a set of data into a specified number of groups, k, by minimising some numerical criterion, low values of which are considered indicative of a 'good' solution. The most commonly used approach, for example, is to try to find the partition of the n individuals into k groups, which minimises the within-group sum of squares over all variables. The problem then appears relatively simple; namely, consider every possible partition of the n individuals into k groups, and select the one with the lowest within-group sum of squares. Unfortunately, the problem in practise is not so straightforward. The numbers involved are so vast that complete enumeration of *every* possible partition remains impossible even with the fastest computer. The scale of the problem is illustrated by the numbers in Table 18.3.

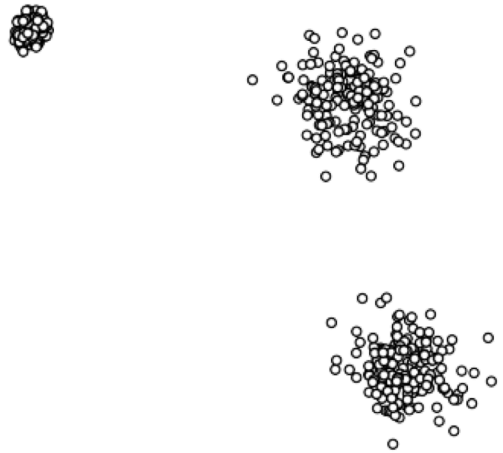
Table 18.3: Number of possible partitions depending on the sample size n and number of clusters k.

n	k	Number of possible partitions
15	3	2,375,101
20	4	45,232,115,901
25	8	690,223,721,118,368,580
100	5	10^{68}

The impracticability of examining every possible partition has led to the development of algorithms designed to search for the minimum values of the clustering criterion by rearranging existing partitions and keeping the new one only if it provides an improvement. Such algorithms do not, of course, guarantee finding the global minimum of the criterion. The essential steps in these algorithms are as follows:

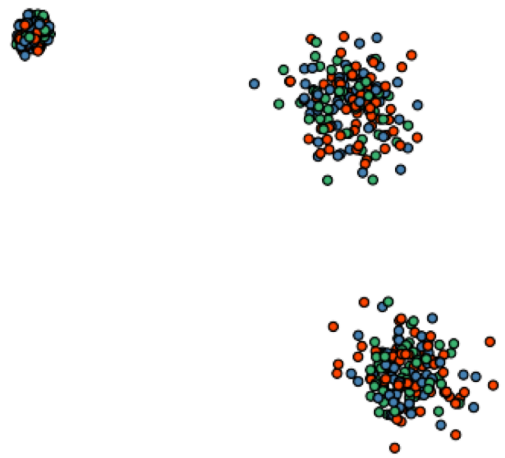
Input data

We would like 3 clusters (means), so set K = 3



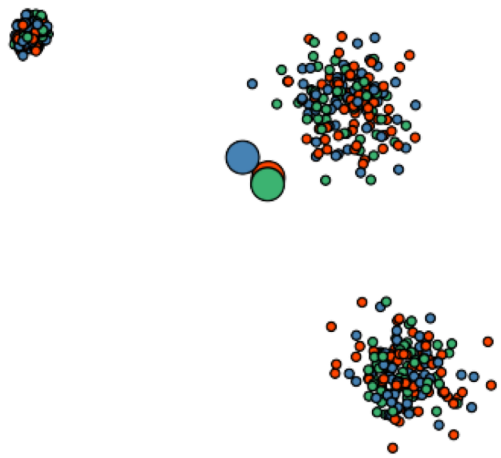
Initialization: alternative initialiazation

Points are tagged randomly



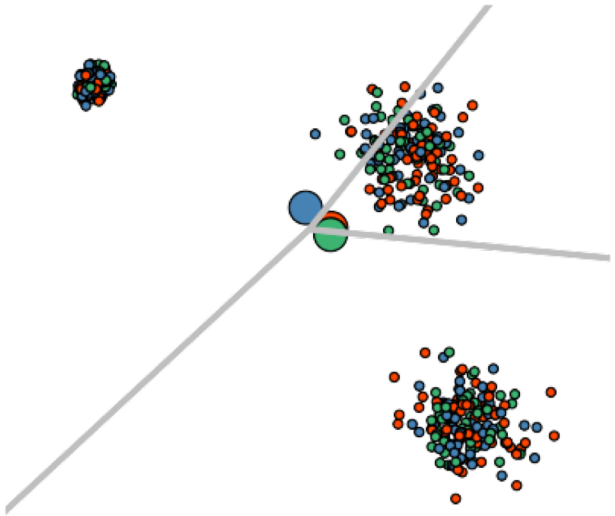
Calculate centres: Update

Computes center of *red*, *green* and *blue* points



Space partition

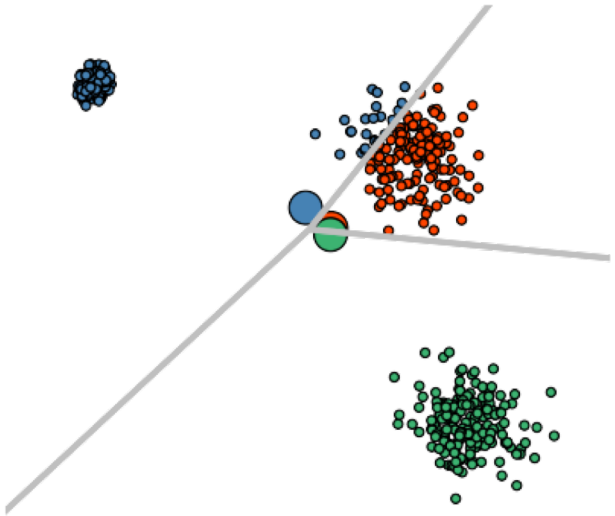
We can cut the space in 3 areas: a Voronoi diagram !



One area ⇒ area which points are closer to one center

Assignment

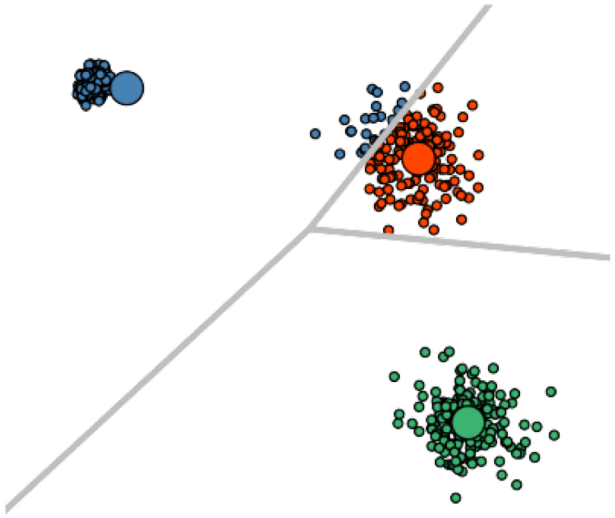
Change the color of the points



One point get the color of the closest cluster center

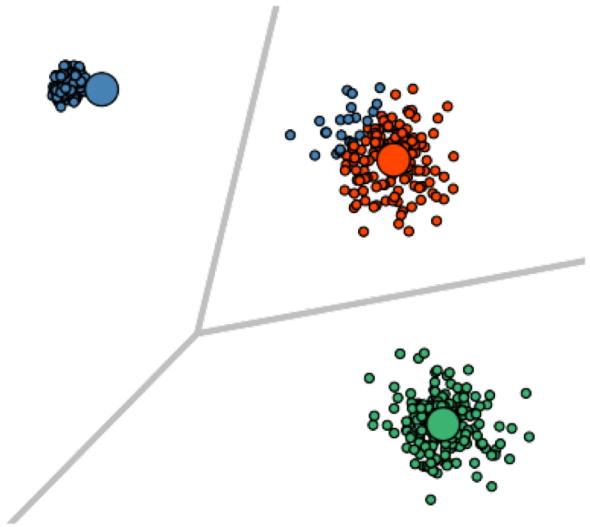
Update

Update center of *red*, *green* and *blue* points



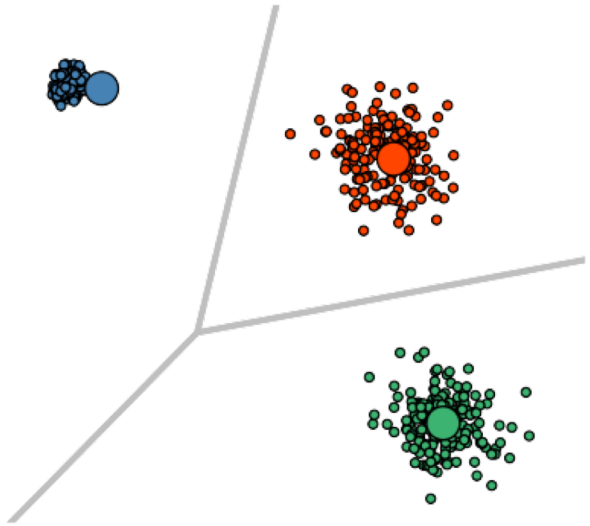
Space partition

The 3 areas changed



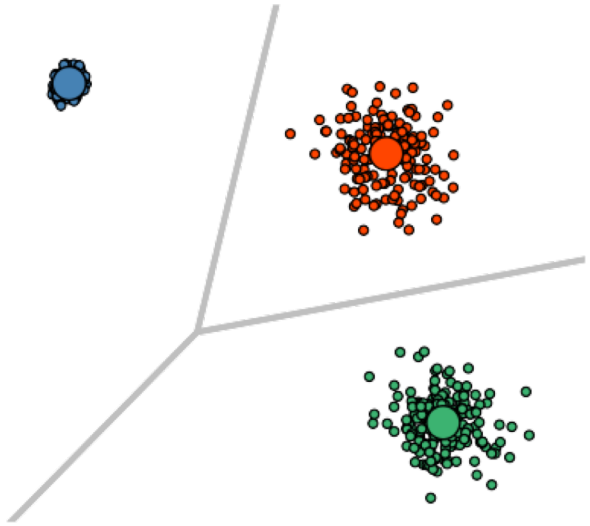
Assignment

Change the color of the points



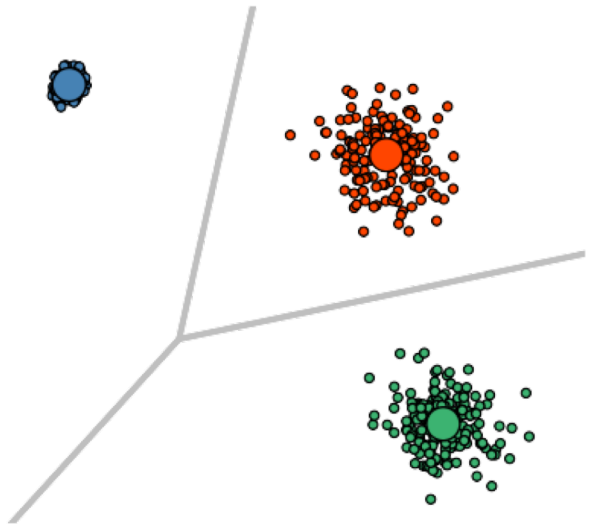
Update

Update center of *red*, *green* and *blue* points



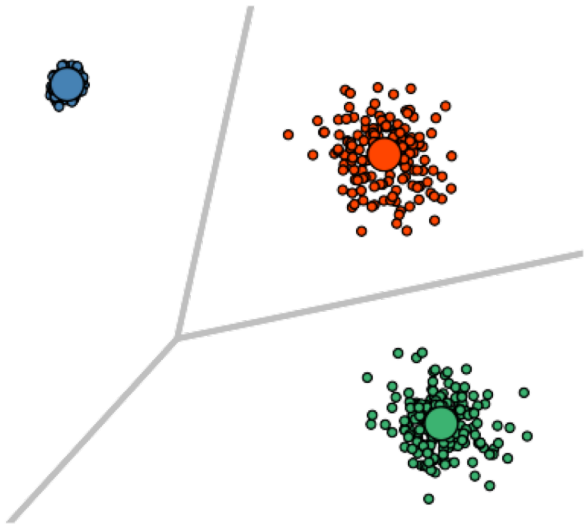
Space partition

The 3 areas changed (a little tiny bit)



Assignment

All points coloured properly already



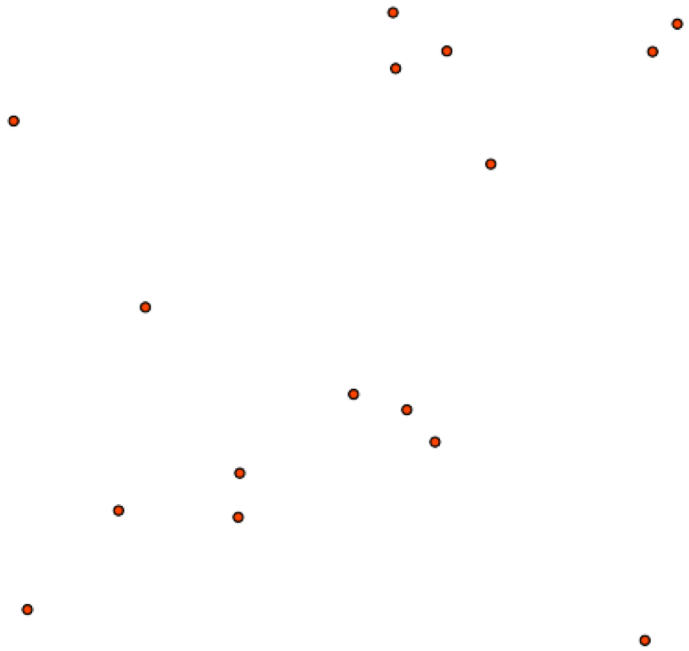
⇒ we are done !

Voronoi diagram

A little reminder/introduction to Voronoi diagrams

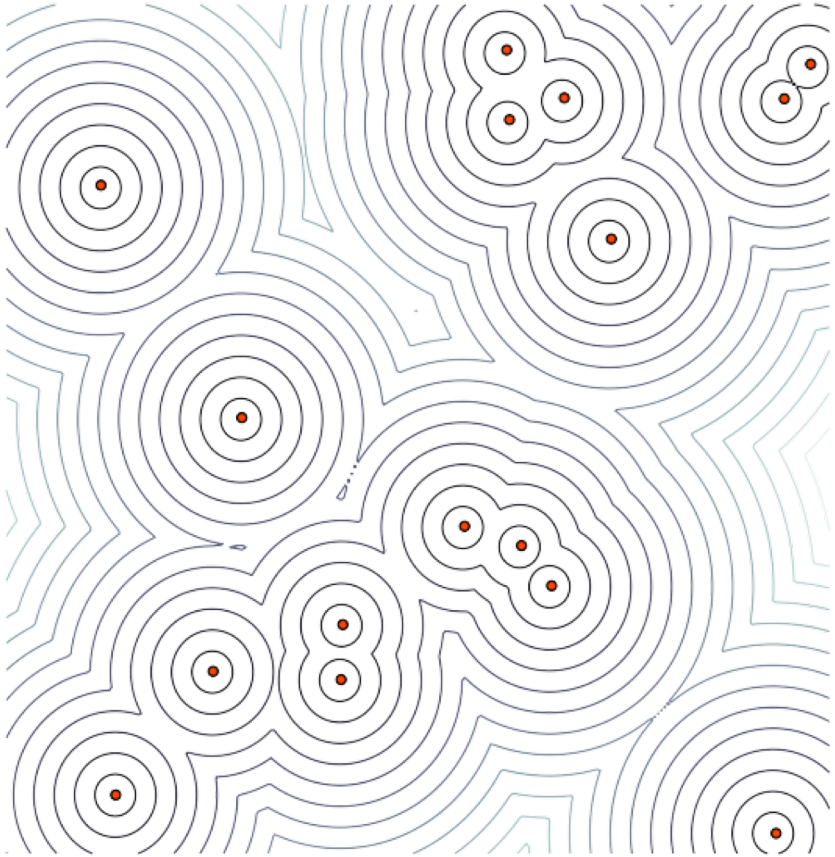
Voronoi diagram

Let's put some points on this slide



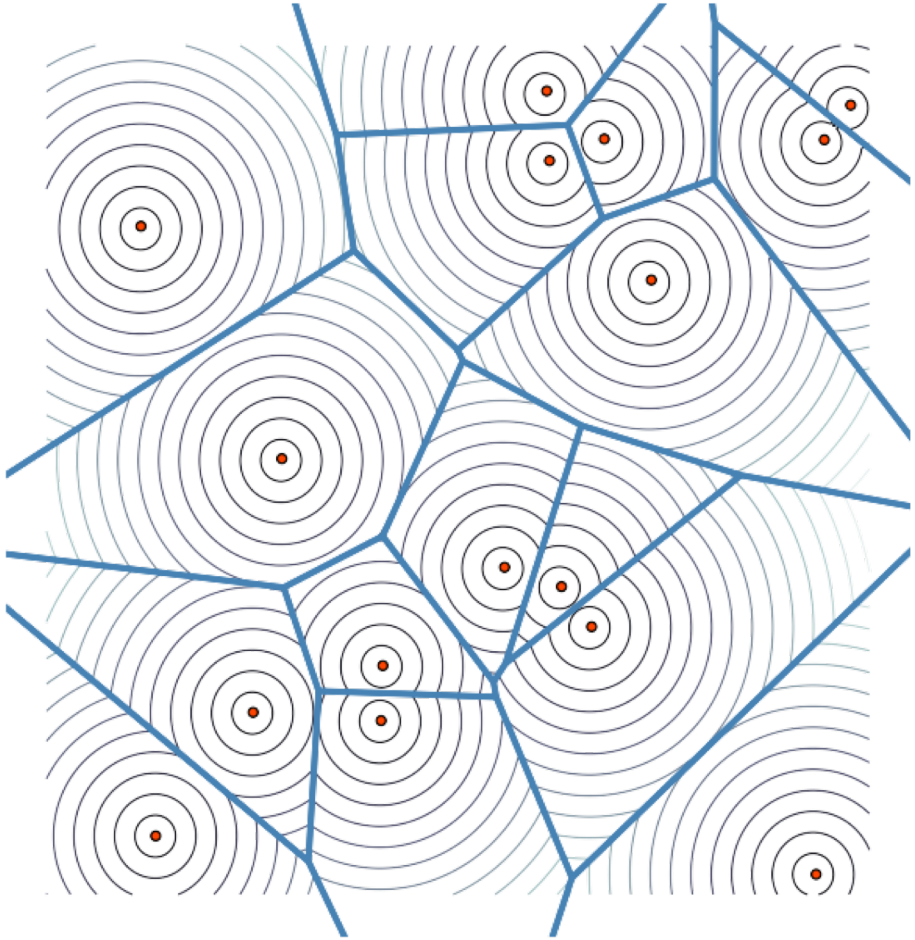
Voronoi diagram

Let's picture the distance to the closest point



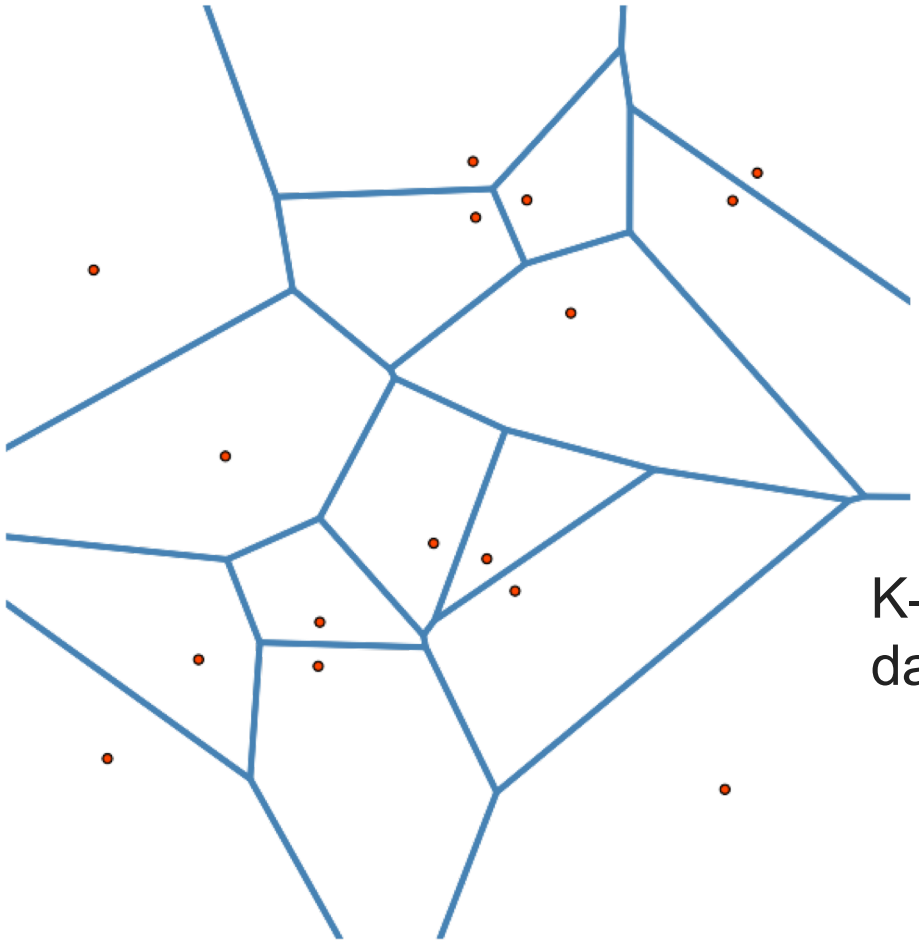
Voronoi diagram

Let's picture the places equidistant to several points



Voronoi diagram

The Voronoi diagram is made of the borders between cells



K-means separates data into Voronoi-cells

Kmeans Algorithm

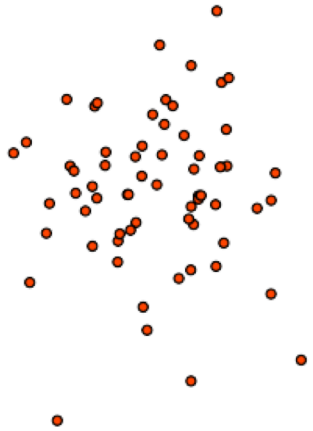
- **A 3 step loop: While not done yet**
  - ***Update*** $\Rightarrow$ **where are the clusters centres ?**
  - ***Assignment*** $\Rightarrow$ **who belongs to whom ?**
  - ***Stopping criteria*** $\Rightarrow$ **are we done yet ?**

Update Step

- **Update step: compute the center of each clusters. Center of a cluster can be**
  - 1 L2 norm ⇒ geometric center of the cluster's points
  - 2 median point of the cluster's points
  - 3 medoid point of the cluster's points
  - 4 whatever makes sense for your data The last point will have a lecture just for it !

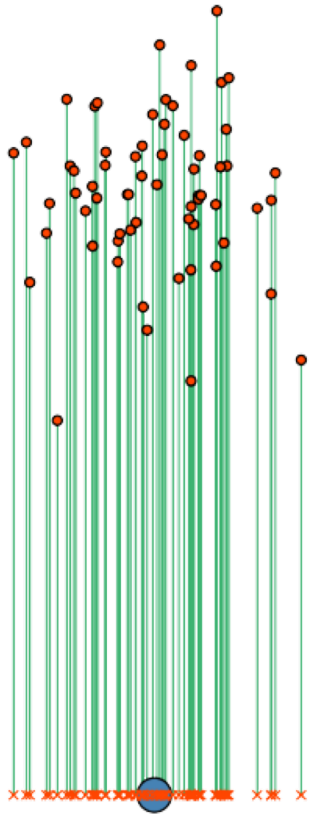
Center of a cluster

Let's compute the center of those points



Center of a cluster

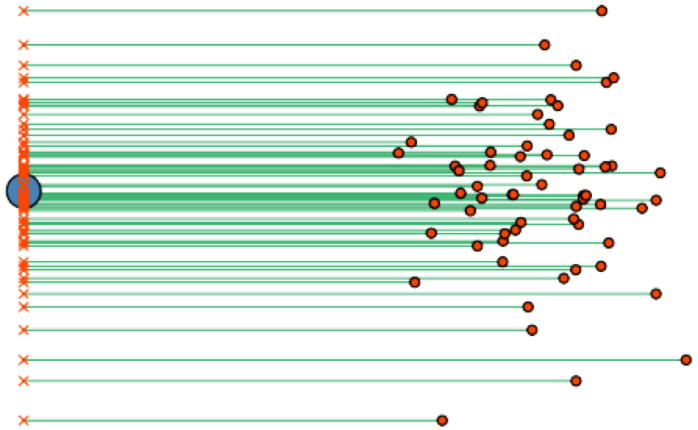
We can use the mean on each dimension



Calculate mean of x values

Center of a cluster

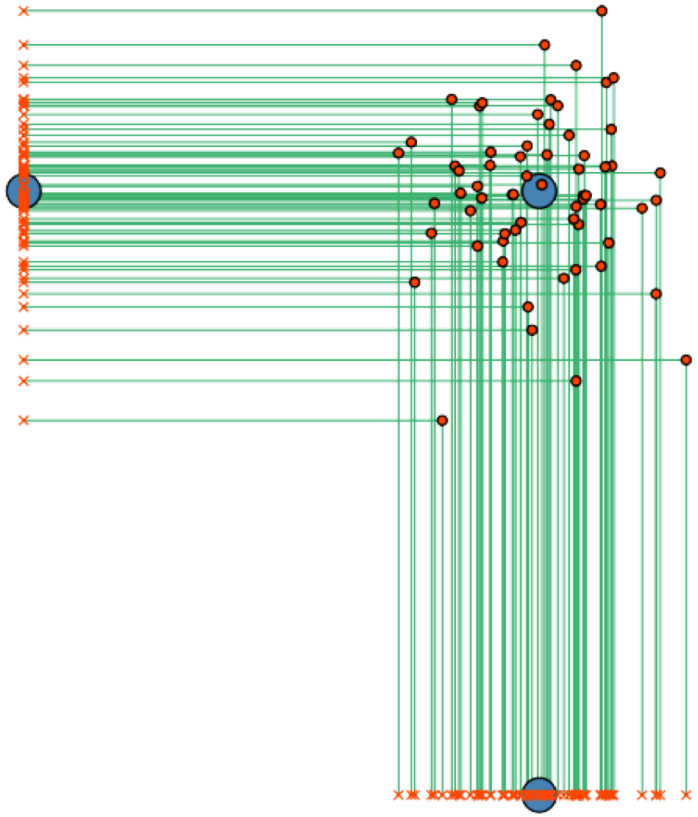
We can use the mean on each dimension



Calculate mean of y values

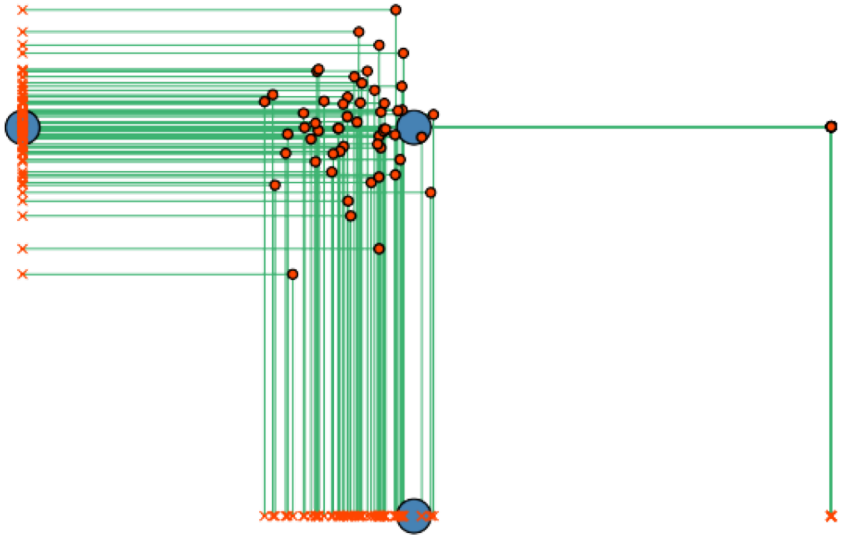
Center of a cluster

We can use the mean on each dimension



Center of a cluster

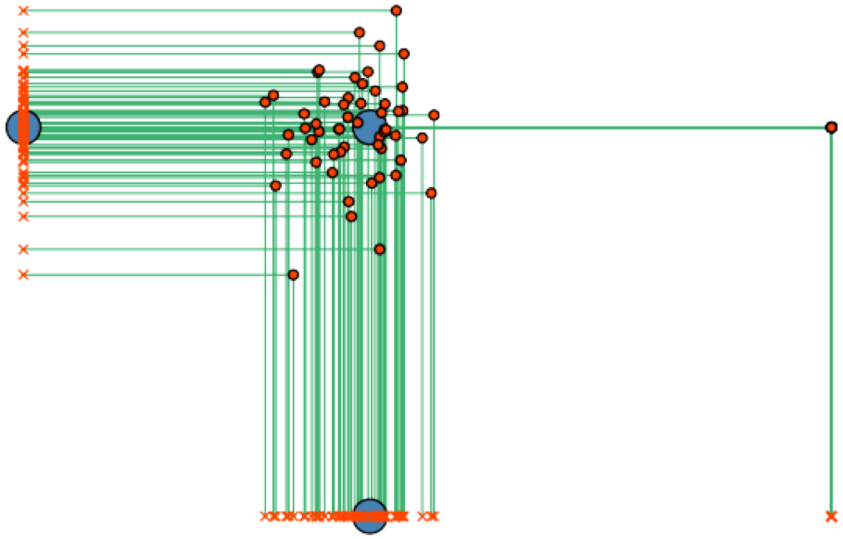
But the mean has trouble with outliers



Center of a cluster

Using the median on each dimension is more robust

https://en.wikipedia.org/wiki/K-medians_clustering



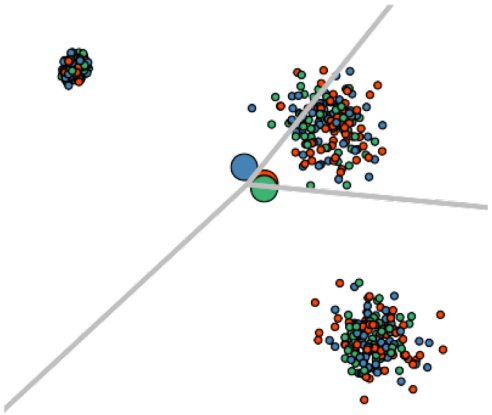
In statistics and data mining, k-medians clustering is a cluster analysis algorithm.

It is a variation of k-means clustering where instead of calculating the mean for each cluster to determine its centroid, one instead calculates the median.

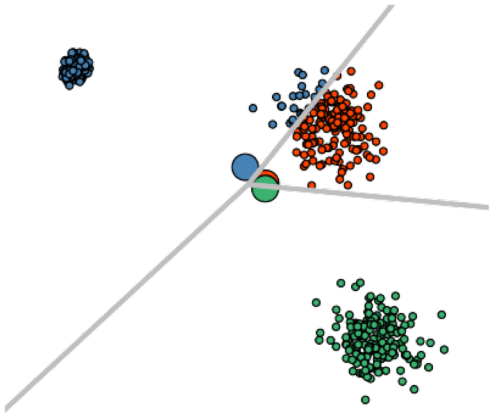
This has the effect of minimizing error over all clusters with respect to the 1-norm distance metric, as opposed to the square of the 2-norm distance metric (which k-means does.)

Assignment

***Assignment* step: a point belongs to the closest cluster center**



before

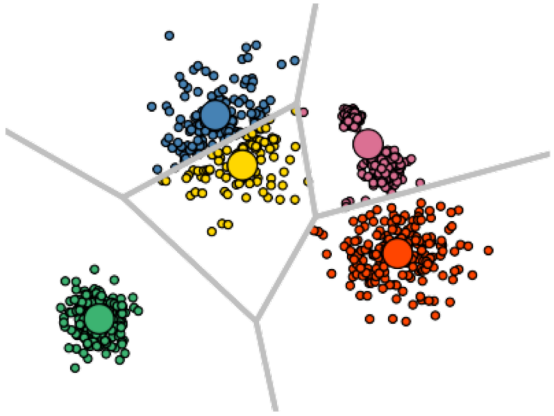


after

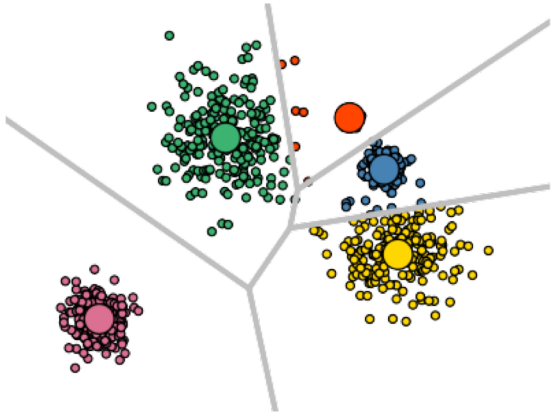
Stopping criteria

Stopping criteria: no more cluster assignments change

- ***k-means* always converges**
- **final clusters might not be ideal**



not so good clustering



better clustering

Live Session Outline

- **Introduction**
- **Unsupervised Learning**
- **Clustering**
  - Kmeans
    - Flat Clustering via Kmeans graphically
    - Flat Clustering via Kmeans in code
    - Hyperparameters: Different initializations
  - Hierarchical clustering
- **Similarity measures**
- **Evaluating Clustering**
- **Relatives of Kmeans, e.g., Expectation Maximization**
- **Summary**

Kmeans in Python NumPy

```
1   import numpy
2
3   # Parameters
4   points = numpy.loadtxt('myFile.dat')
5   nbClusters, nbPoints = 2, points.shape[0]
6
7   # Initialization
8   clustersId = numpy.random.randint(0, nbClusters, size=nbPoints)
9   clustersCenters = numpy.zeros((nbClusters, points.shape[1]))
10
11  # Iteration
12  converged = False
13  while not converged:
14    # Update the clusters center (mean)
15    for c in xrange(nbClusters):
16      numpy.mean(
17        [points[p] for p in xrange(nbPoints) if clustersId[p] == c],
18        axis=0, out=clustersCenters[c])
19
20    # Attribute each point to its closest cluster
21    oldClustersId = clustersId
22    clustersId = [
23      numpy.argmin([numpy.linalg.norm(p - c) for c in clustersCenters]) for p in points
24    ]
25
26    # Check if convergence is reached
27    converged = numpy.array_equal(oldClustersId, clustersId)
```

Does assignment and update in one statement using *numpy.mean(…)*

Numpy.mean manual page

numpy.mean

numpy.**mean**(*a*, *axis=None*, *dtype=None*, *out=None*, *keepdims=False*) [source]

Compute the arithmetic mean along the specified axis.

Returns the average of the array elements. The average is taken over the flattened array by default, otherwise over the specified axis. float64 intermediate and return values are used for integer inputs.

Parameters:

a *: array_like*
: Array containing numbers whose mean is desired. If *a* is not an array, a conversion is attempted.

axis *: None or int or tuple of ints, optional*
: Axis or axes along which the means are computed. The default is to compute the mean of the flattened array. If this is a tuple of ints, a mean is performed over multiple axes, instead of a single axis or all the axes as before.

dtype *: data-type, optional*
: Type to use in computing the mean. For integer inputs, the default is float64; for floating point inputs, it is the same as the input dtype.

out *: ndarray, optional*
: Alternate output array in which to place the result. The default is `None`; if provided, it must have the same shape as the expected output, but the type will be cast if necessary. See doc.ufuncs for details.

keepdims *: bool, optional*
: If this is set to True, the axes which are reduced are left in the result as dimensions with size one. With this option, the result will broadcast correctly against the original *arr*.

Returns:

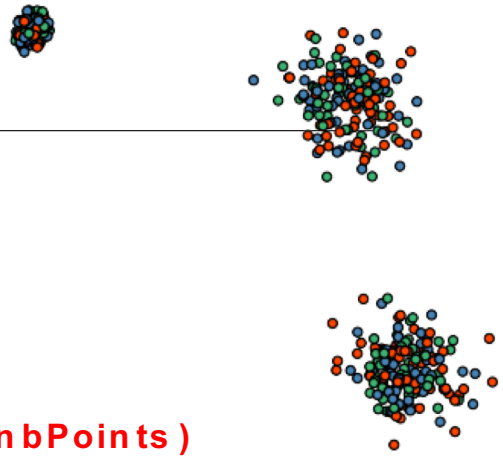
m *: ndarray, see dtype parameter above*
: If *out=None*, returns a new array containing the mean values, otherwise a reference to the output array is returned.

Kmeans in Python NumPy

```
1   import numpy
2
3   # Parameters
4   points = numpy.loadtxt('myFile.dat')
5   nbClusters, nbPoints = 2, points.shape[0]
6
7   # Initialization
8   clustersId = numpy.random.randint(0, nbClusters, size=nbPoints)
9   clustersCenters = numpy.zeros((nbClusters, points.shape[1]))
10
11  # Iteration
12  converged = False
13  while not converged:
14    # Update the clusters center (mean)
15    for c in xrange(nbClusters):
16      numpy.mean(
17        [points[p] for p in xrange(nbPoints) if clustersId[p] == c],
18        axis=0, out=clustersCenters[c])
19
20    # Attribute each point to its closest cluster
21    oldClustersId = clustersId
22    clustersId = [
23      numpy.argmin([numpy.linalg.norm(p - c) for c in clustersCenters]) for p in points
24    ]
25
26    # Check if convergence is reached
27    converged = numpy.array_equal(oldClustersId, clustersId)
```

Does assignment and update in one statement using *numpy.mean(…)*

Initialization: randomly assign an example to a cluster



```
1   import numpy 2
3 # Parameters
4   points = numpy.loadtxt('myFile.dat')
5   nbClusters,nbPoints = 2,points.shape[0] 6
7
8
9   # Initialization
    clustersId = numpy.random.randint(0,nbClusters,size=nbPoints)
    clustersCenters = numpy.zeros((nbClusters, points.shape[1]))
```

- line 1 ⇒ let's use Numpy
- line 4 and 5 ⇒ load data
- line 8 ⇒ points are assigned randomly to a cluster
- line 9 ⇒ matrix storing center of each cluster

Iterations

```
1# Iteration
2 converged = False
3 while not converged:
4     # Update the clusters center (mean)
6     # Attribute each point to its closest cluster
7
8     # Check if convergence is reached
```

Update step

```
1  # Update the clusters center (mean)
2  for c in xrange(nbClusters):
3    numpy.mean(
4      [points[p] for p in xrange(nbPoints) if clustersId[p] == c],
5      axis=0, out=clustersCenters[c])
```

- line 2 ⇒ for *c* from 0 to no. of clusters . . .
- line 3, 4 and 5 ⇒ center of cluster *c* is the mean of all the points which belong to cluster *c*

Assignment step

```
1# Attribute each point to its closest cluster
2   clusterIds = [
3       numpy.argmin([numpy.linalg.norm(p - c) for c in clustersCenters])
          for p in points]
4
5
6   ------------------------------------------------------------------
```

- line 4 ⇒ distance of cluster center $c_0, c_1, \ldots, c_n$ to point p
- line 3 and 5 ⇒ cluster of point p is the cluster with the closest center
- line 2 and 6 ⇒ for each point p

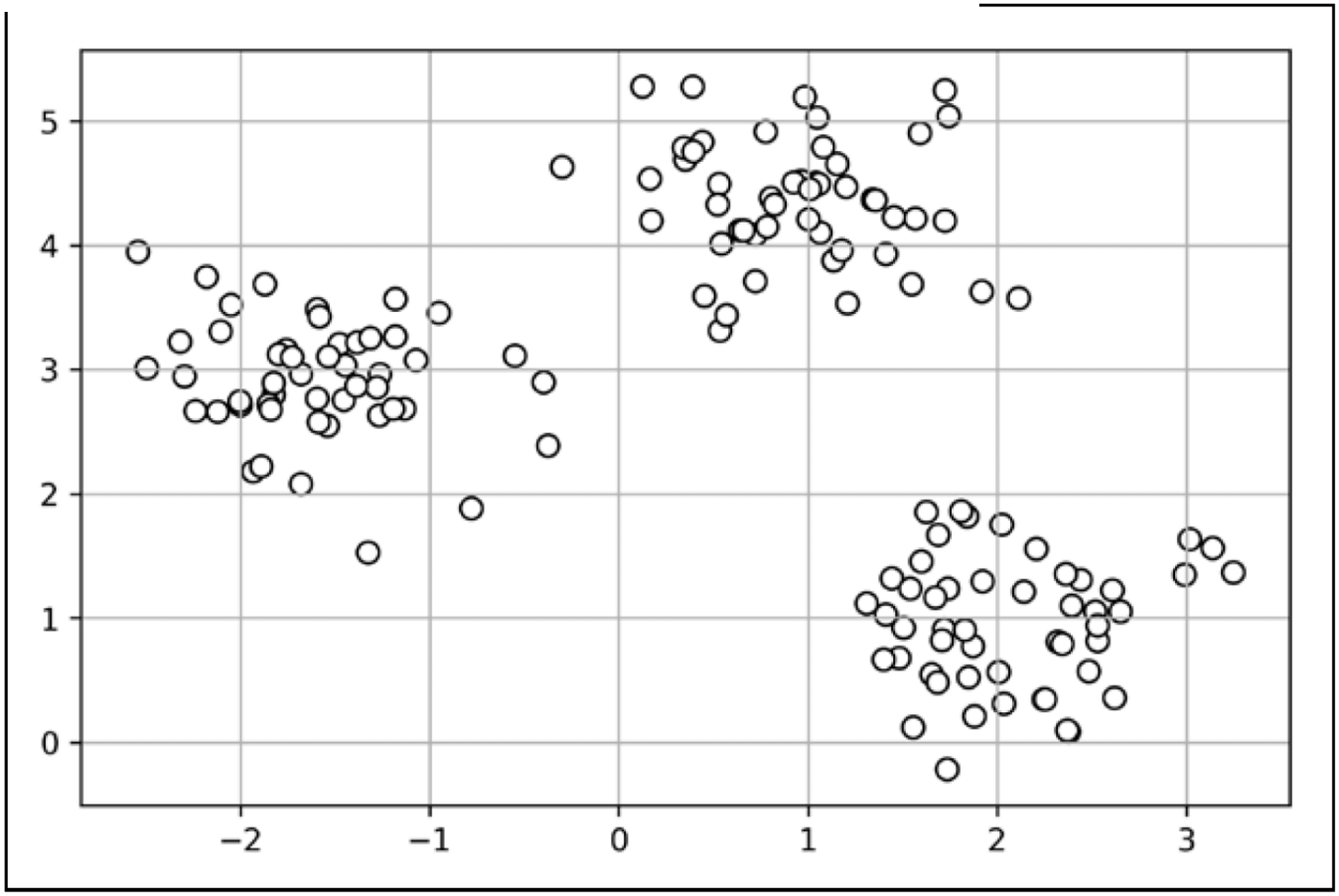
Stopping criteria

```
1# Attribute each point to its closest cluster
2   oldClustersId = clustersId
3   clustersId = [
4numpy.argmin([numpy.linalg.norm(p - c) for c in clustersCenters]) for p in points5
6   ]
7
8  # Check if convergence is reached
    converged = numpy.array equal(oldClustersId, clusters
    Id)
```

- **line 2 ⇒ keep previous cluster assignment of each point**
- **line 3, 4 and 5 ⇒ assignment step**
- **line 8 ⇒ if assignment did not changed, done**

3 blobs

```
>>> from sklearn.datasets import make_blobs
>>> X, y = make_blobs(n_samples=150,
...                   n_features=2,
...                   centers=3,
...                   cluster_std=0.5,
...                   shuffle=True,
...                   random_state=0)
```



Euclidean distance and SSE

- **similarity between objects**

$$d(\boldsymbol{x},\boldsymbol{y})^2 = \sum_{j=1}^{m}\left(x_j - y_j\right)^2 = \left\|\boldsymbol{x}-\boldsymbol{y}\right\|_2^2$$

- **an iterative approach for minimizing the within-cluster Sum of Squared Errors (SSE).**

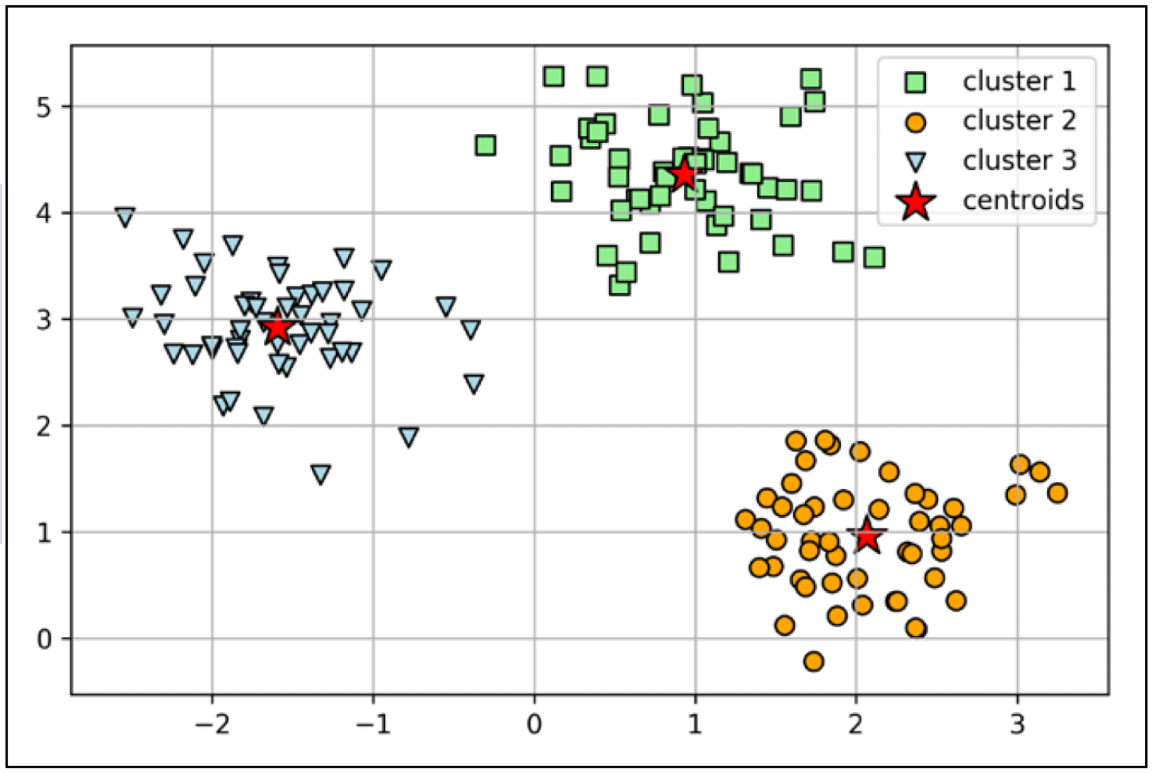
$$SSE = \sum_{i=1}^{n}\sum_{j=1}^{k} w^{(i,j)}\left\|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(j)}\right\|_2^2$$

Here $\boldsymbol{\mu}^{(j)}$ is the representative point (centroid) for cluster j, and $w^{(i,j)} = 1$ if the sample $\boldsymbol{x}^{(i)}$ is in cluster j; $w^{(i,j)} = 0$ otherwise.

Example

```
>>> from sklearn.cluster import KMeans
>>> km = KMeans(n_clusters=3,
...             init='random',
...             n_init=10,
...             max_iter=300,
...             tol=1e-04,
...             random_state=0)
>>> y_km = km.fit_predict(X)
```

Although k-means worked well on this toy dataset, we shall highlight another drawback of k-means: we have to specify the number of clusters, k, a priori.



Issues with KMeans

- **If a cluster is empty, the algorithm will search for the sample that is farthest away from the centroid of the empty cluster.**
  - Then it will reassign the centroid to be this farthest point.

- **Although k-means worked well on this toy dataset, we shall highlight another drawback of k-means: we have to specify the number of clusters, k, a priori.**
- **Must standardize the data**

Live Session Outline

- **Introduction**
- **Unsupervised Learning**
- **Clustering**
  - Kmeans
    - Flat Clustering via Kmeans graphically
    - Flat Clustering via Kmeans in code
    - Hyperparameters: Similarity measures and Different initializations
  - Hierarchical clustering
- **Similarity measures**
- **Evaluating Clustering**
- **Relatives of Kmeans, e.g., Expectation Maximization**
- **Summary**

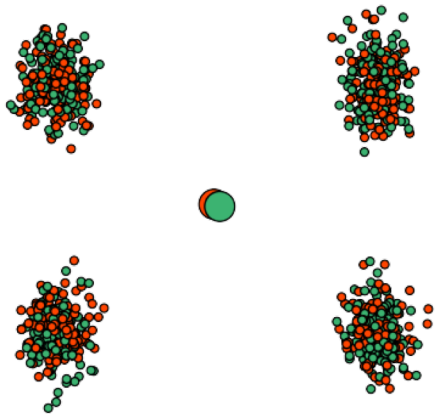
- **Lots of hyperparameters for clustering**
  - Init (address here)
  - Similarity measures (see later section)

Better initialization

- So far, we have discussed the classic k-means algorithm that uses a random seed to place the initial centroids, which can sometimes result in bad clusterings or slow convergence if the initial centroids are chosen poorly.
- One way to address this issue is to run the k-means algorithm multiple times on a dataset and choose the best performing model in terms of the SSE.
- Another strategy is to place the initial centroids far away from each other via the k-means++ algorithm

Random partition

random partition ⇒ end result not guaranteed

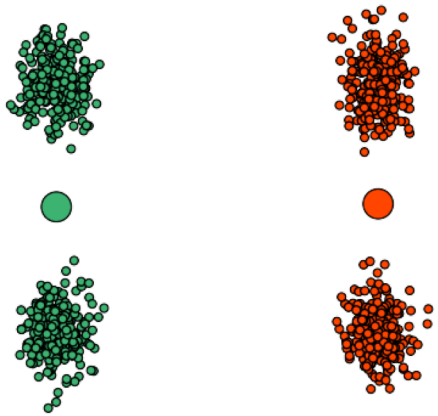


Kmeans++

1. Initialize an empty set **M** to store the k centroids being selected.
2. Randomly choose the first centroid $\boldsymbol{\mu}^{(j)}$ from the input samples and assign it to **M**.
3. For each sample $\boldsymbol{x}^{(i)}$ that is not in **M**, find the minimum squared distance $d\left(\boldsymbol{x}^{(i)}, \mathbf{M}\right)^2$ to any of the centroids in **M**.
4. To randomly select the next centroid $\boldsymbol{\mu}^{(\mathrm{p})}$, use a weighted probability distribution equal to $\dfrac{d\left(\boldsymbol{\mu}^{(\mathrm{p})}, \mathbf{M}\right)^2}{\sum_i d\left(\boldsymbol{x}^{(i)}, \mathbf{M}\right)^2}$.
5. Repeat steps 2 and 3 until k centroids are chosen.
6. Proceed with the classic k-means algorithm.

k-means++ initialization

***k-means*++ choose centres as maximally distant points**



k-means++ initialization

The center C_1 of the 1st cluster is picked randomly amongst the point $X_0, X_1, \ldots, X_n$

$$p_1(X_i) = \frac{1}{n}$$

$p_1(X_i)$ is the probability for X_i to be picked as C_1

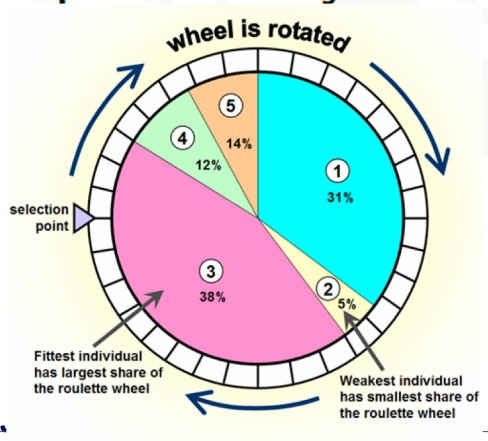
k-means++ initialization

The center C_2 of the 2nd cluster is picked with probability proportional to its distance to the center C_1 of the 1st cluster

Favors examples further away

$$p_2(X_i) = \frac{d(C_1, X_i)}{\sum_{j=1}^{n} d(C_1, X_j)}$$

$p_2(X_i)$ is the probability for X_i to be picked as C_2



k-means++ initialization

Distance is on the nearest cluster
But then Favors examples further away

The center C_3 of the 3nd cluster is picked with probability proportional to the distance to the closest center

$$p_3(X_i) = \frac{\min\left(d(C_1, X_i), d(C_2, X_i)\right)}{\sum_{j=1}^{n}\left(\min\left(d(C_1, X_j), d(C_2, X_j)\right)\right)}$$

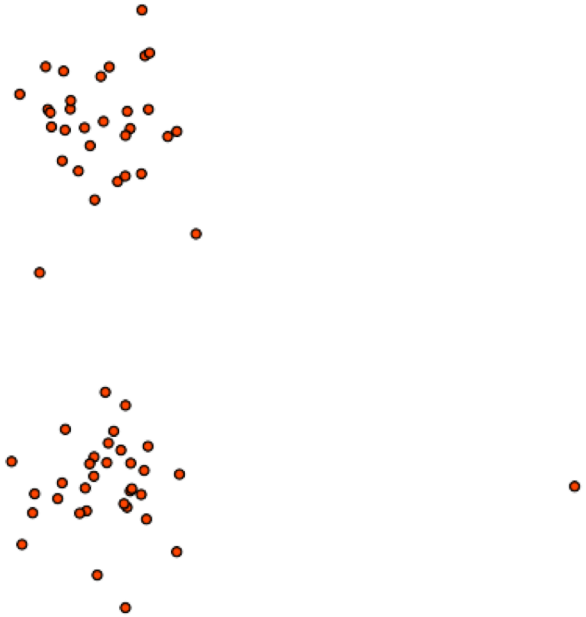
$p_3(X_i)$ is the probability for X_i to be picked as C_3

k-means++ initialization

Why using probability ? Why not using the farthest point?

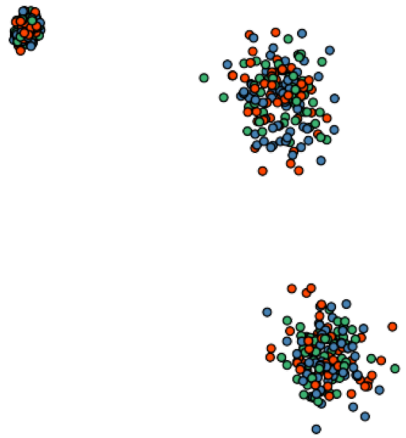
k-means++ initialization

⇒ **because we would select an outlier as a center**

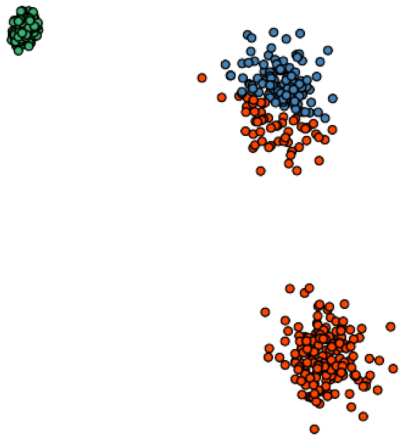


Smarter initialization

***k-means*++ initialization usually gives better initial guess**



random partition



***kmeans*++**

Better initial assignment

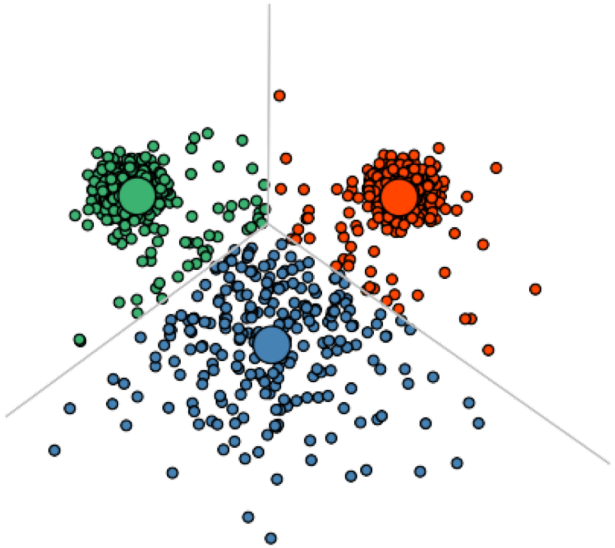
Multiple tries

⇒ **always run *k-means* several times**

- **very cheap anyway**
- **required to properly measure quality**
- **"unlucky initialization problem" less likely**

Boundaries

k-means **clusters space in a rigid & sharp fashion**



Kmeans in sklearn uses kmean++

- To use k-means++ with scikit-learn's KMeans object, we just need to set the init parameter to 'k-means++'. In fact, 'k-means++' is the default argument to the init parameter, which is strongly recommended in practice.

Live Session Outline

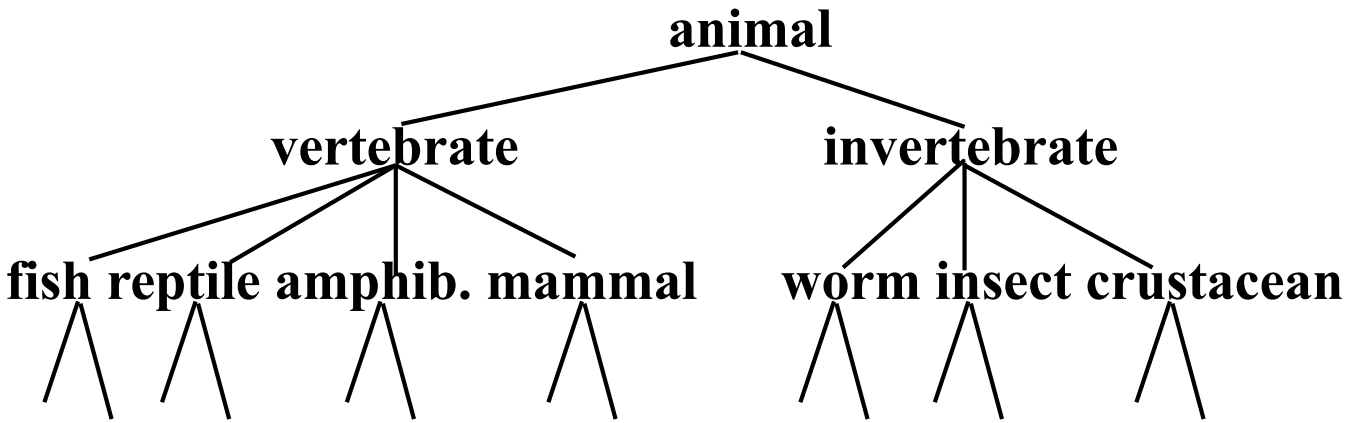
- **Introduction**
- **Unsupervised Learning**
- **Clustering**
  - Kmeans
    - Flat Clustering via Kmeans graphically
    - Flat Clustering via Kmeans in code
    - Hyperparameters: different initializations
  - Hierarchical clustering
- **Similarity measures**
- **Evaluating Clustering**
- **Relatives of Kmeans, e.g., Expectation Maximization**
- **Summary**

Clustering Algorithms

- **Flat/Non-Hierarchical Clustering (prototype-based clustering)**
  - Exclusive Clustering (k-means) [In R, cluster package]
  - Overlapping Clustering (fuzzy c means) [In R, e1071]
  - Probabilistic Clustering (E.g., EM Mixture of Gaussians)
- **Hierarchical Clustering**
  - Agglomerative approach (bottom-up)
    - In R: hclust() and agnes()
  - 2. Divisive approach (top-down)
    - In R: diana()
  - we do not need to specify the number of clusters up front.
  - plot dendrograms (visualizations of a binary hierarchical clustering), which can help with the interpretation of the results by creating meaningful taxonomies.

Hierarchical Clustering Example:

- **Build a tree-based hierarchical taxonomy (*dendrogram*) from a set of documents.**



How could you do this with k-means?

Hierarchical clustering algorithms

- **Two main types of hierarchical clustering**
  - **Agglomerative:**
    - Start with the points as individual clusters
    - At each step, merge the closest pair of clusters until only one cluster (or **k** clusters) left
  - **Divisive:**
    - Start with one, all-inclusive cluster
    - At each step, split a cluster until each cluster contains a point (or there are **k** clusters)
- **Does not require the number of clusters *k* in advance**
- **Needs a termination/readout condition**
- **Traditional hierarchical algorithms use a similarity or distance matrix**
  - Merge or split one cluster at a time

Hierarchical Agglomerative Clustering (HAC)
Cluster similarity

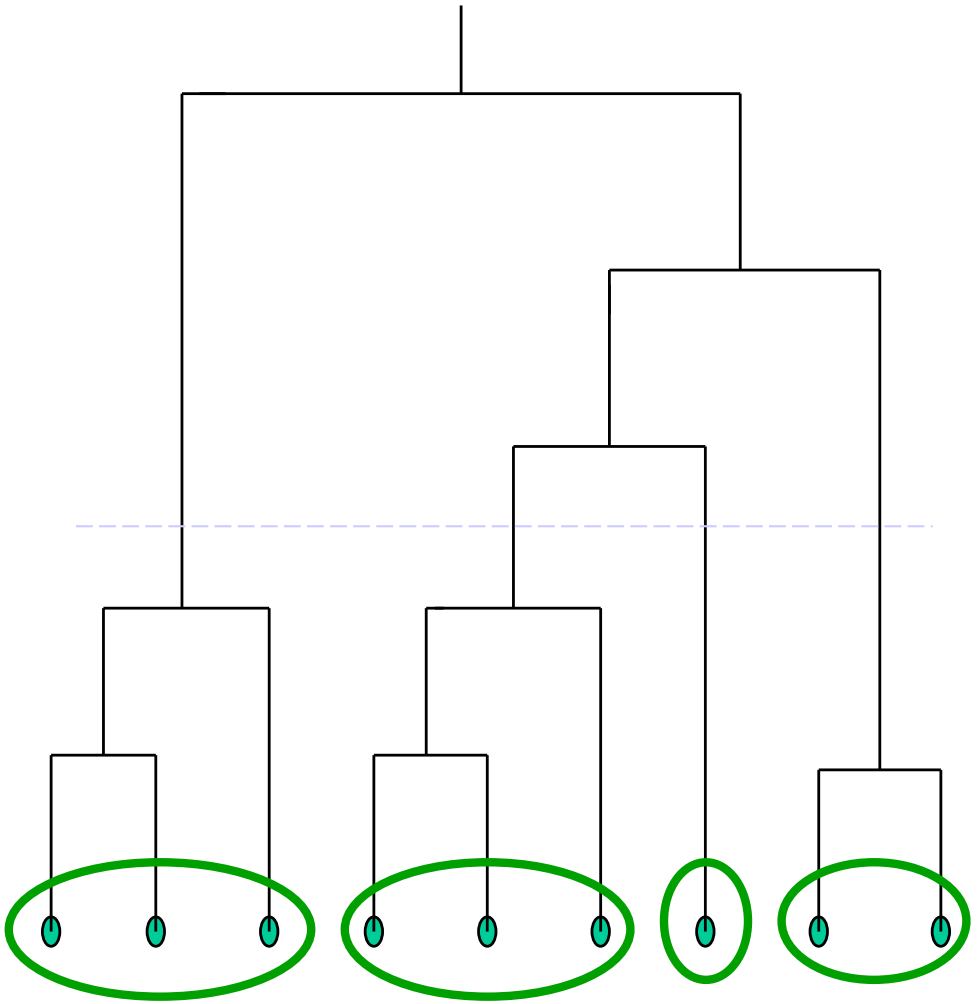
- **Starts with each doc in a separate cluster**
  - then repeatedly joins the *closest pair* of clusters, until there is only one cluster.
- **The history of merging forms a binary tree or hierarchy.**

How to measure distance of clusters??

- **Centroid-based similarity**
- **Pairs-based similarity**

Dendogram: Hierarchical Clustering

- **HAC: The history of merging forms a binary tree or hierarchy.**
- **Clustering obtained by cutting the dendrogram at a desired level: each connected component forms a cluster.**



Centroid-based Similarity

- **Always maintain average of vectors in each cluster:**

$$\vec{s}(c_j) = \frac{\sum_{\vec{x} \in c_j} \vec{x}}{|c_j|}$$

- **Compute similarity of clusters by:**

$$sim(c_i, c_j) = sim(s(c_i), s(c_j))$$

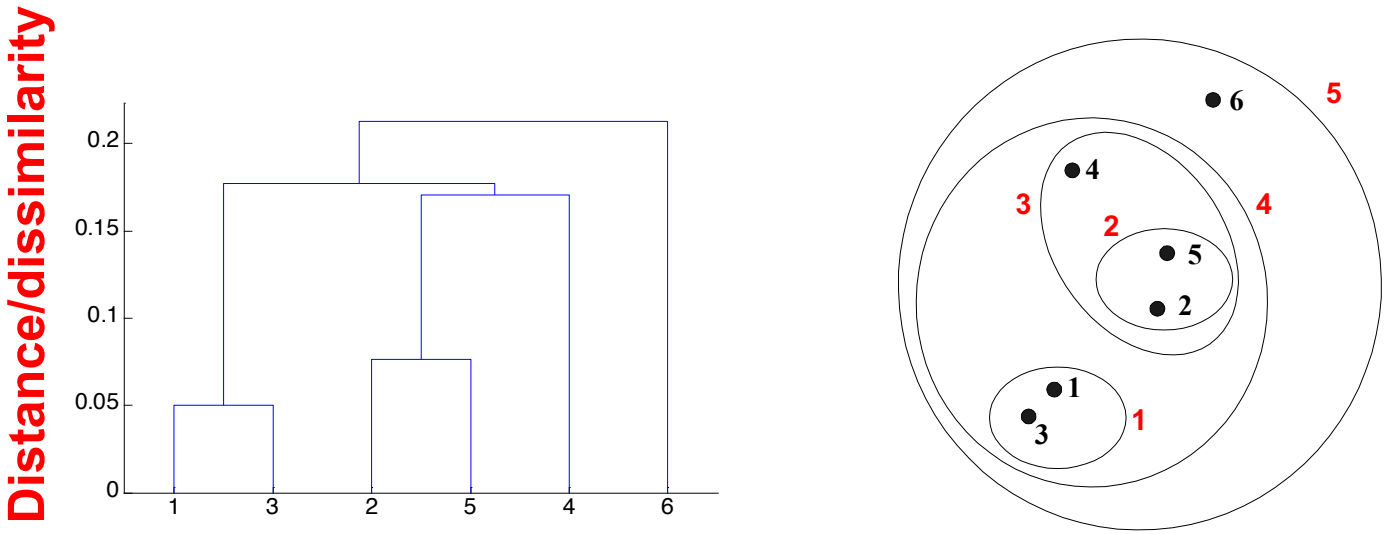
- **For non-vector data, can't always make a centroid**

Computational Complexity

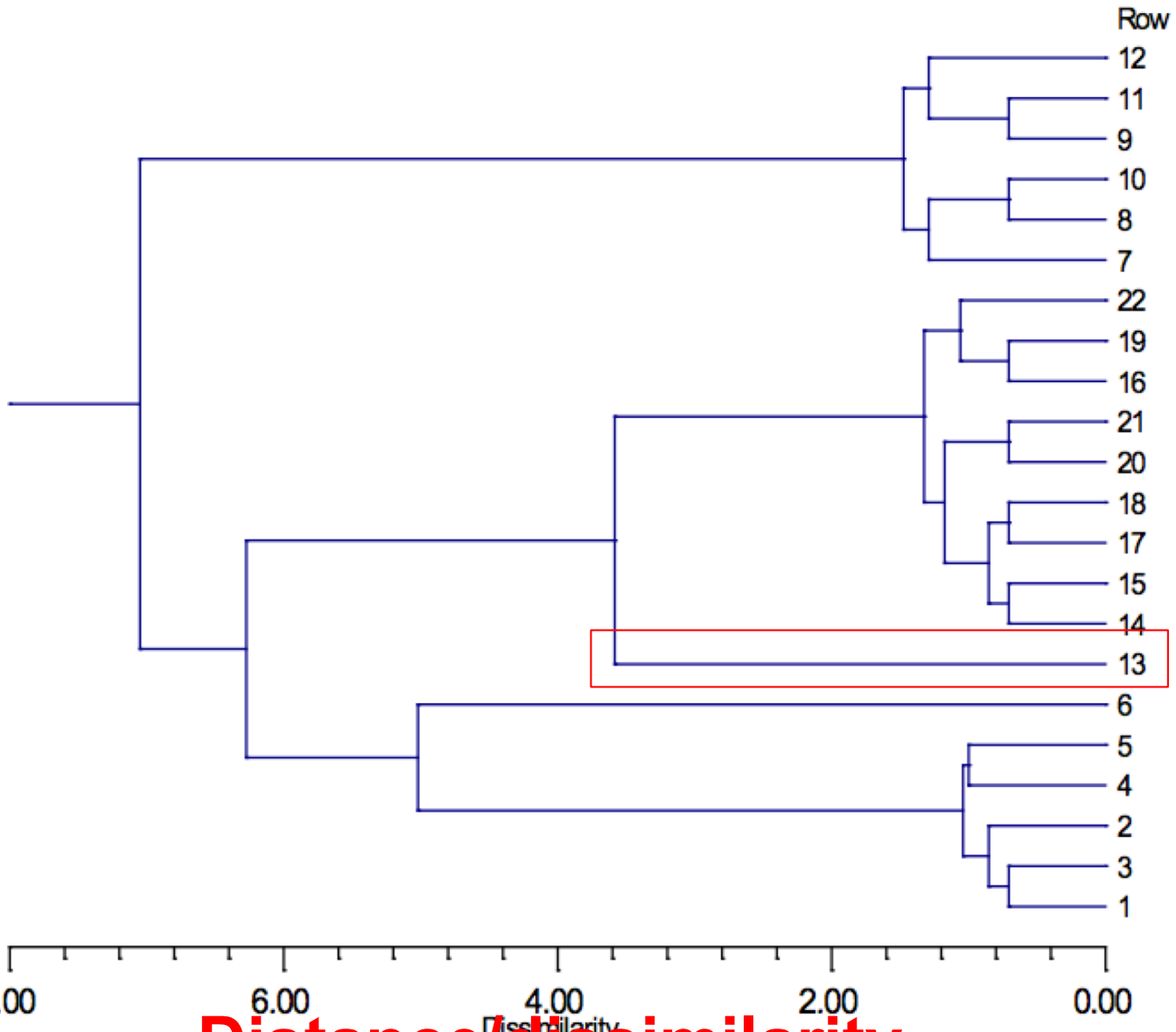
- In the first iteration, all HAC methods need to compute similarity of all pairs of *n* individual instances which is O(mn^2).
- In each of the subsequent $n-2$ merging iterations, compute the distance between the most recently created cluster and all other existing clusters.
- Maintaining of heap of distances allows this to be O($mn^2 \log n$)

Clustering Algorithms

- **Hierarchical Clustering**
  - Produces a set of ***nested clusters*** organized as a hierarchical tree
  - Can be visualized as a **dendrogram**
    - A tree-like diagram that records the sequences of merges or splits



HAC with group averaging: Dendogram

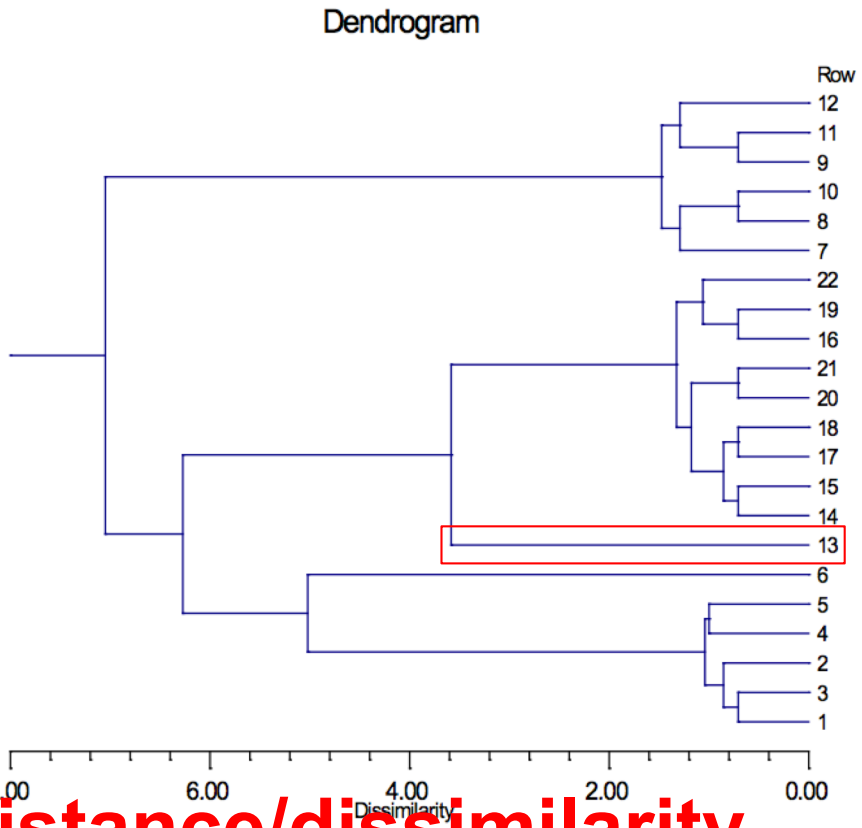


Outlier 13 & 6

Distance/dissimilarity

Hierarchical Clustering / Dendrograms

Following is a dendrogram of the results of running these data through the Group Average clustering algorithm.



Outlier 13

Outlier 6

Distance/dissimilarity

The horizontal axis of the dendrogram represents the distance or dissimilarity between clusters. The vertical axis represents the objects and clusters. The dendrogram is fairly simple to interpret. Remember that our main interest is in similarity and clustering. Each joining (fusion) of two clusters is represented on the graph by the splitting of a horizontal line into two horizontal lines. The horizontal position of the split, shown by the short vertical bar, gives the distance (dissimilarity) between the two clusters.

Looking at this dendrogram, you can see the three clusters as three branches that occur at about the same horizontal distance. The two outliers, 6 and 13, are fused in rather arbitrarily at much higher distances. This is the interpretation.

In this example we can compare our interpretation with an actual plot of the data. Unfortunately, this usually will not be possible because our data will consist of more than two variables.

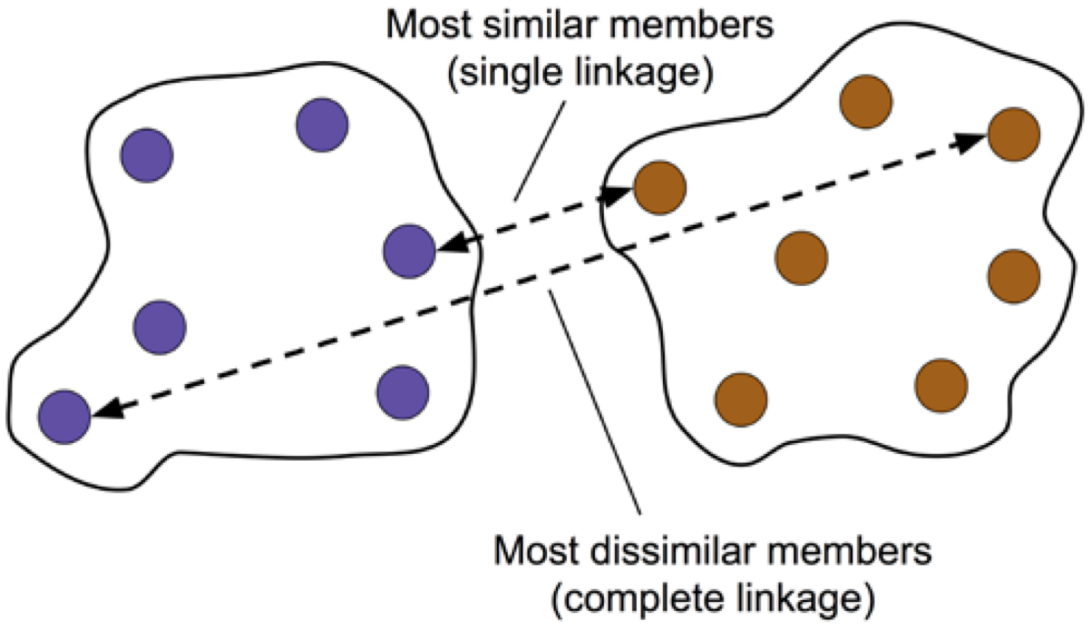
Hierarchical Agglomerative Clustering (HAC)
Cluster similarity

- **Starts with each doc in a separate cluster**
  - then repeatedly joins the *closest pair* of clusters, until there is only one cluster.
- **The history of merging forms a binary tree or hierarchy.**

How to measure distance of clusters??

- **Centroid-based similarity**
- **Pairs-based similarity**

Closest pair of clusters



Closest pair of clusters

Many variants to defining closest pair of clusters

- **Single-link**
  - Distance of the *"closest" points* (single-link)
- **Complete-link**
  - Distance of the "furthest" points
- **Centroid**
  - Distance of the centroids (centers of gravity)
- **(Average-link)**
  - Average distance between pairs of elements

Single Link Agglomerative Clustering

Worst link scenario

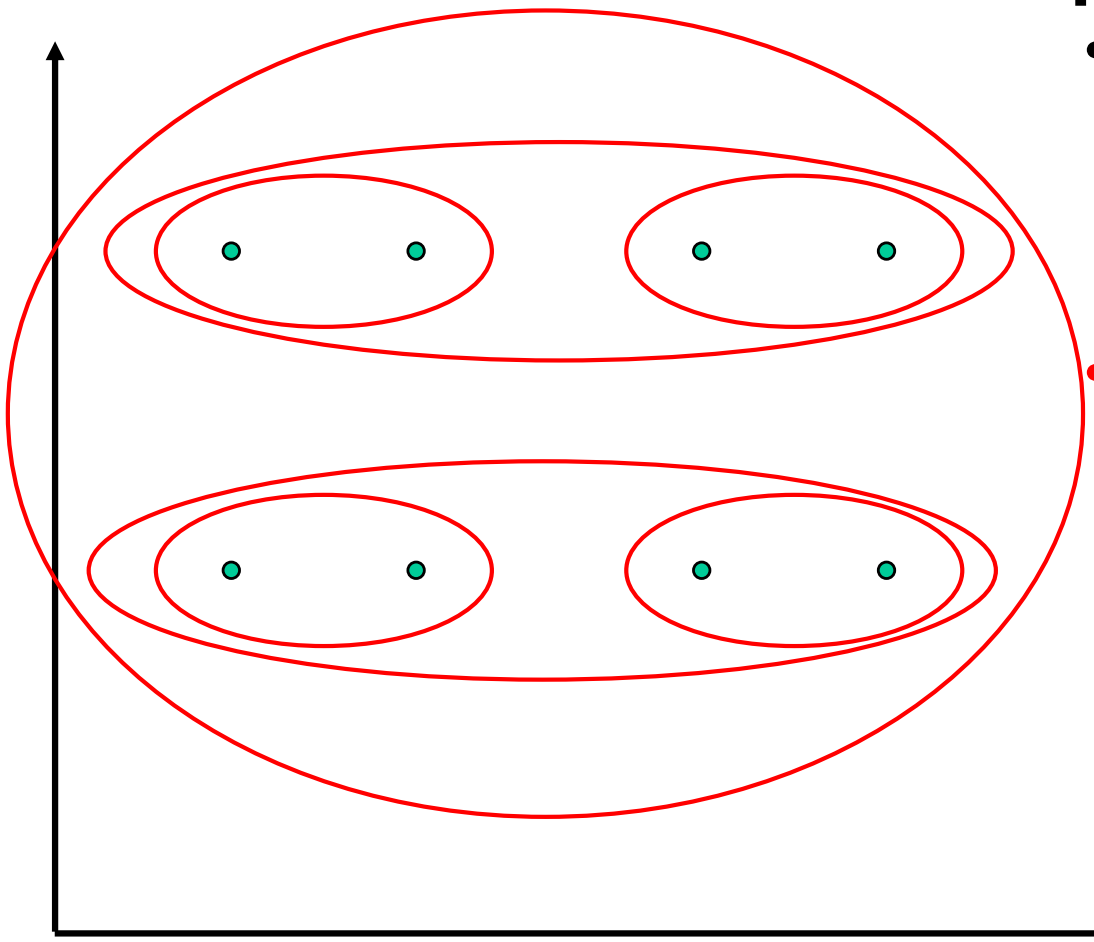
- **Use maximum similarity of pairs:**

$$sim(c_i, c_j) = \max_{x \in c_i, y \in c_j} sim(x, y)$$

- **Can result in "straggly" (long and thin) clusters due to chaining effect.**
- **After merging c_i and c_j, the similarity of the resulting cluster to another cluster, c_k, is:**

$$sim((c_i \cup c_j), c_k) = \max(sim(c_i, c_k), sim(c_j, c_k))$$

Single Link Example



Starts with each doc in a separate cluster

- **then repeatedly joins the *closest pair* of clusters, until there is only one cluster.**
- **Use maximum similarity of pairs:**

$$sim(c_i, c_j) = \max_{x \in c_i, y \in c_j} sim(x, y)$$

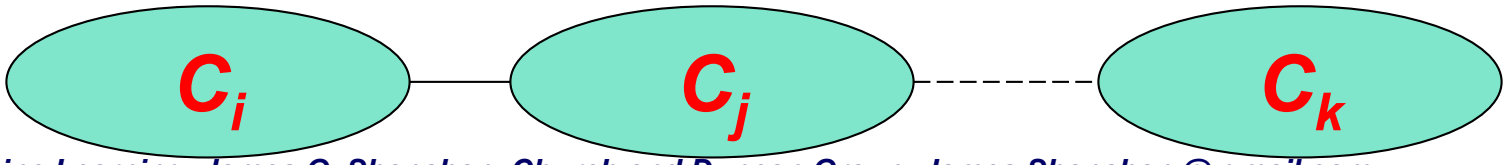
Complete Link Agglomerative Clustering

- **Use minimum similarity of pairs:**

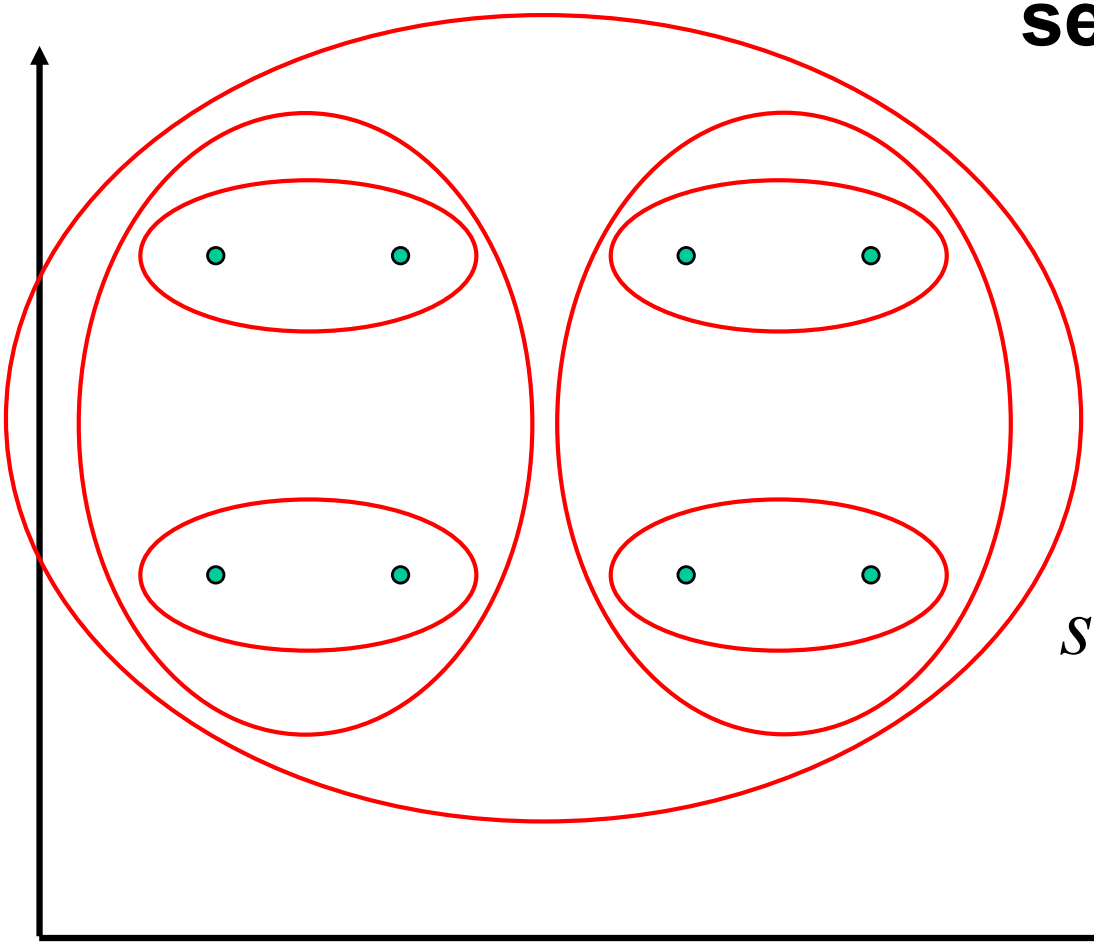
$$sim(c_i, c_j) = \min_{x \in c_i, y \in c_j} sim(x, y)$$

- **Makes "tighter," spherical clusters that are typically preferable.**
- **After merging c_i and c_j, the similarity of the resulting cluster to another cluster, c_k, is:**

$$sim((c_i \cup c_j), c_k) = \min(sim(c_i, c_k), sim(c_j, c_k))$$



Complete Link Example



Starts with each doc in a separate cluster

- **then repeatedly joins the *closest pair* of clusters, until there is only one cluster.**
- **Use Min similarity of pairs:**

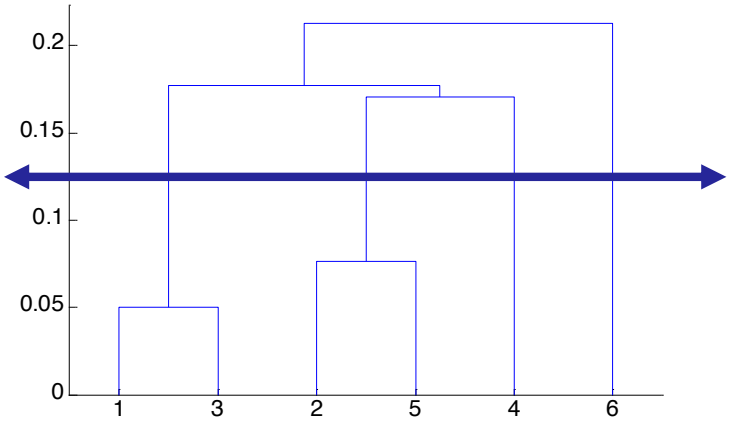
$$sim(c_i, c_j) = \min_{x \in c_i, y \in c_j} sim(x, y)$$

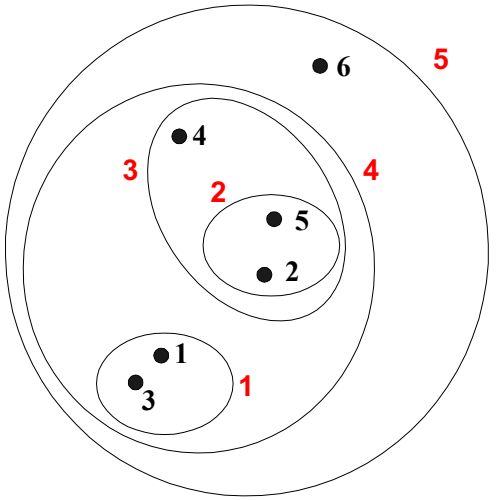
Key notion: *cluster representative*

- **We want a notion of a representative point in a cluster**
- **Representative should be some sort of "typical" or central point in the cluster, e.g.,**
  - point inducing smallest radii to docs in cluster
  - smallest squared distances, etc.
  - point that is the "average" of all docs in the cluster
    - Centroid or center of gravity

Clustering Algorithms

- **Hierarchical Clustering**
  - Produces a set of ***nested clusters*** organized as a hierarchical tree
  - Can be visualized as a **dendrogram**
    - A tree-like diagram that records the sequences of merges or splits





Clustering in SKLearn

- **Incorrect approach**: Using the `squareform` distance matrix shown in the following code snippet would lead to incorrect results:

```
>>> from scipy.cluster.hierarchy import linkage
>>> row_clusters = linkage(row_dist,
...                        method='complete',
...                        metric='euclidean')
```

- **Correct approach**: Using the condensed distance matrix as shown in the following code example yields the correct pairwise distance matrix:

```
>>> row_clusters = linkage(pdist(df, metric='euclidean'),
...                        method='complete')
```

- **Correct approach**: Using the complete input sample matrix as shown in the following code snippet also leads to a correct distance matrix similar to the preceding approach:

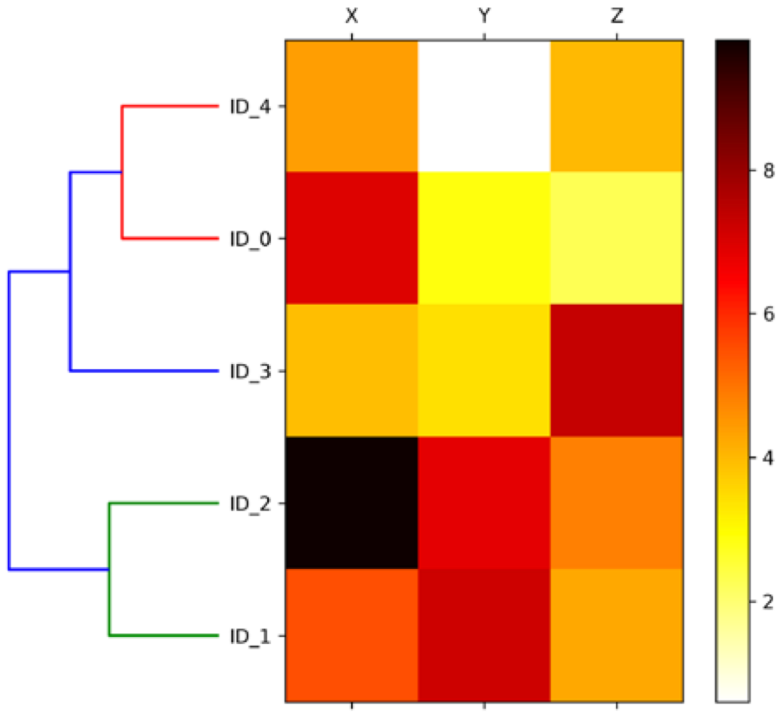
```
>>> row_clusters = linkage(df.values,
...                        method='complete',
...                        metric='euclidean')
```

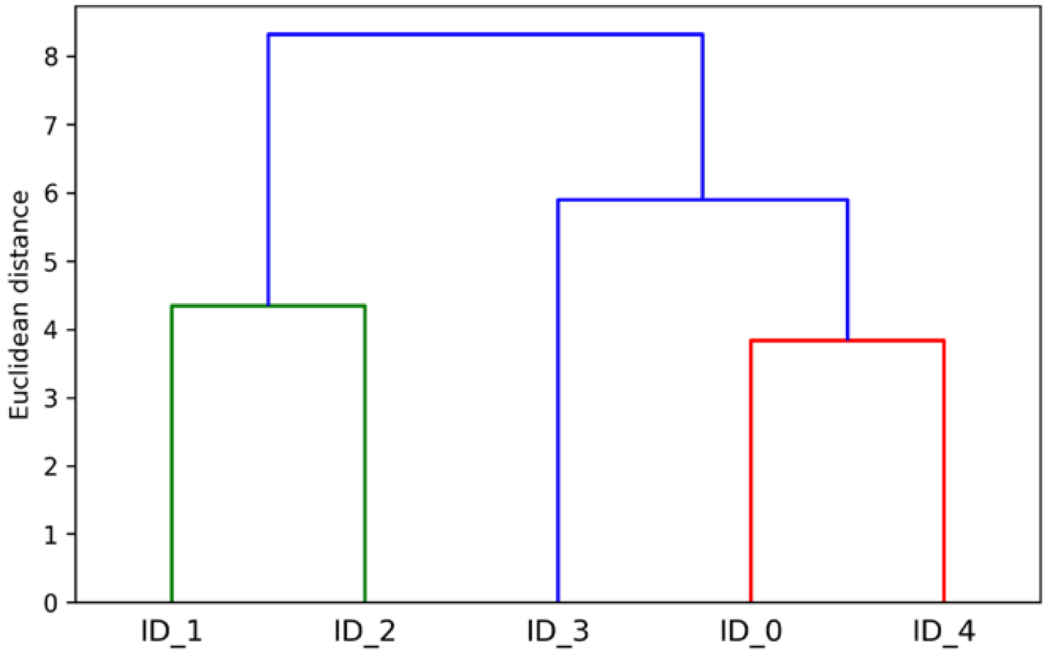
5 object, 3 features

	X	Y	Z
ID_0	6.964692	2.861393	2.268515
ID_1	5.513148	7.194690	4.231065
ID_2	9.807642	6.848297	4.809319
ID_3	3.921175	3.431780	7.290497
ID_4	4.385722	0.596779	3.980443

```
row_clusters = linkage(df.values,
                       method='complete',
                       metric='euclidean'
```

Heatmap of each Data value





Live Session Outline

- **Introduction**
- **Unsupervised Learning**
- **Clustering**
  - Kmeans
    - Flat Clustering via Kmeans graphically
    - Flat Clustering via Kmeans in code
    - Hyperparameters: different initializations
  - Hierarchical clustering
- **Similarity measures**
- **Evaluating Clustering**
- **Relatives of Kmeans, e.g., Expectation Maximization**
- **Summary**

distance metric is more important than the Clustering Algorithm

- Attach label to each observation or data points in a set
- You can say this "unsupervised classification"
- Intuitively, if you would want to assign same label to a data points that are "close" to each other
- Thus, clustering algorithms rely on a distance metric between data points
- Sometimes, it is said that the for clustering, the distance metric is more important than the clustering algorithm

metric or distance function is a function

- **A metric or distance function is a**
  - function d(x,y) that defines the distance between elements of a set as a non-negative real number.
  - If the distance is zero, both elements are equivalent under that specific metric.
  - Distance functions thus provide a way to measure how close two elements are, where elements do not have to be numbers but can also be vectors, matrices or arbitrary objects.
  - Distance functions are often used as error or cost functions to be minimized in an optimization problem.
  - There are multiple ways to define a metric on a set. A typical distance for real numbers is the absolute difference, $d:(x,y)\mapsto|x-y|$.
  - Every normed vector space induces a distance given by $d(x,y)=\|x-y\|$.

Distances

- **Euclidean Distance**

$$dist = \sqrt{\sum_{k=1}^{n}(p_k - q_k)^2}$$

  - Where n is the number of dimensions (attributes) and p_k and q_k are, respectively, the kth attributes (components) or data objects p and q.

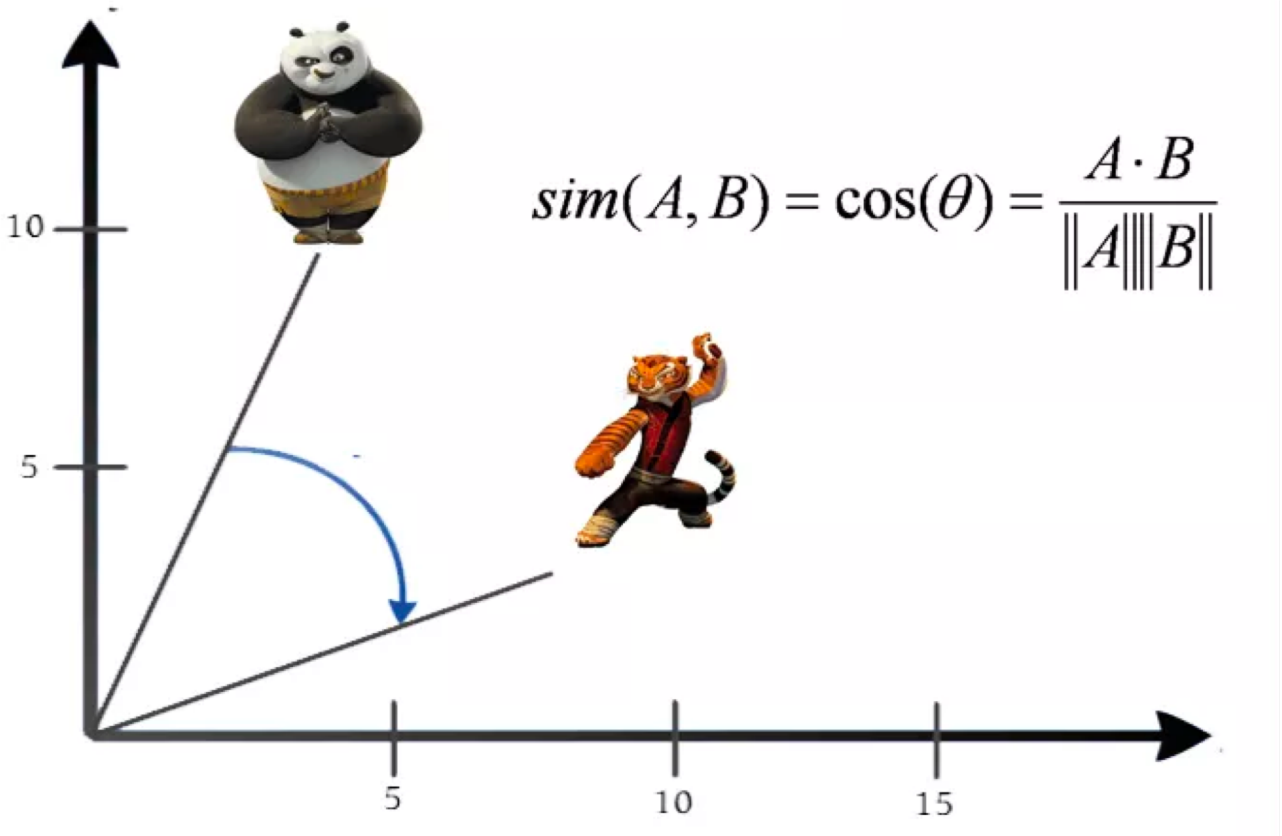
- **Minkowski Distance is a generalization of Euclidean Distance**

$$dist = (\sum_{k=1}^{n}| p_k - q_k |^r)^{\frac{1}{r}}$$

  - Where *r* is a parameter, *n* is the number of dimensions (attributes) and p_k and q_k are, respectively, the kth attributes (components) or data objects *p* and *q*.

1 – Cosine(A, B) as a distance

1 – Cosine(A, B)



$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

SAD, SSD of two vectors x, and y

Sum of Absolute Difference (SAD) AKA Manhanttan distance

The sum of absolute difference is equivalent to the L_1-norm of the difference, also known as Manhattan- or Taxicab-norm. The `abs` function makes this metric a bit complicated to deal with analytically, but it is more robust than SSD.

d_{SAD} ((1,3), (4,5)) = 2 + 2 = 4

$$d_{\mathrm{SAD}} : (x, y) \mapsto \|x - y\|_1 = \sum_{i=1}^{n} |x_i - y_i|$$

```
1:  double d = Distance.SAD(x, y);
```

Sum of Squared Difference (SSD)

The sum of squared difference is equivalent to the squared L_2-norm, also known as Euclidean norm. It is therefore also known as Squared Euclidean distance. This is the fundamental metric in least squares problems and linear algebra. The absence of the `abs` function makes this metric convenient to deal with analytically, but the squares cause it to be very sensitive to large outliers.

$$d_{\mathrm{SSD}} : (x, y) \mapsto \|x - y\|_2^2 = \langle x - y, x - y \rangle = \sum_{i=1}^{n} (x_i - y_i)^2$$

d_{SSD} ((1,3), (4,5)) = 2^2 + 2^2 = 8

http://numerics.mathdotnet.com/Distance.html

MAE and MSE of two vectors x, and y

```
1:  double d = Distance.SSD(x, y);
```

Mean-Absolute Error (MAE)

The mean absolute error is a normalized version of the sum of absolute difference.

$$d_{\mathrm{MAE}} : (x, y) \mapsto \frac{d_{\mathrm{SAD}}}{n} = \frac{\|x - y\|_1}{n} = \frac{1}{n}\sum_{i=1}^{n} |x_i - y_i|$$

```
1:  double d = Distance.MAE(x, y);
```

Mean-Squared Error (MSE)

The mean squared error is a normalized version of the sum of squared difference.

$$d_{\mathrm{MSE}} : (x, y) \mapsto \frac{d_{\mathrm{SSD}}}{n} = \frac{\|x - y\|_2^2}{n} = \frac{1}{n}\sum_{i=1}^{n} (x_i - y_i)^2$$

```
1:  double d = Distance.MSE(x, y);
```

d_{MSE} ((1,3), (4,5)) = (2^2 + 2^2)/2 = 8/2

Euclidean Distance

The euclidean distance is the L_2-norm of the difference, a special case of the Minkowski distance with p=2. It is the natural distance in a geometric interpretation.

$$d_2 : (x, y) \mapsto \|x - y\|_2 = \sqrt{d_{\mathrm{SSD}}} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

d_{MSE} ((1,3), (4,5)) = sqrt(2^2 + 2^2) = sqrt(8)

```
1:  double d = Distance.Euclidean(x, y);
```

Manhattan Distance

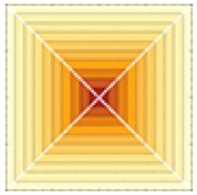
The Manhattan distance is the L_1-norm of the difference, a special case of the Minkowski distance with p=1 and equivalent to the sum of absolute difference.

$$d_1 \equiv d_{\mathrm{SAD}} : (x, y) \mapsto \|x - y\|_1 = \sum_{i=1}^{n} |x_i - y_i|$$

d_{SAD} ((1,3), (4,5)) = 2 + 2 = 4

```
1:  double d = Distance.Manhattan(x, y);
```

Chebyshev Distance



The Chebyshev distance is the L_∞-norm of the difference, a special case of the Minkowski distance where p goes to infinity. It is also known as Chessboard distance.

$$d_\infty : (x, y) \mapsto \|x - y\|_\infty = \lim_{p \to \infty} \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}} = \max_i |x_i - y_i|$$

```
1:  double d = Distance.Chebyshev(x, y);
```

Minkowski Distance

The Minkowski distance is the generalized L_p-norm of the difference. The contour plot on the left demonstrates the case of p=3.

$$d_{\mathbf{p}} : (x, y) \mapsto \|x - y\|_p = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

```
1:  double d = Distance.Minkowski(p, x, y);
```

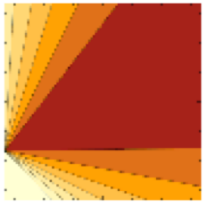
Canberra Distance

The Canberra distance is a weighted version of the Manhattan distance, introduced and refined 1967 by Lance, Williams and Adkins. It is often used for data scattered around an origin, as it is biased for measures around the origin and very sensitive for values close to zero.

$$d_{\mathrm{CAD}} : (x, y) \mapsto \sum_{i=1}^{n} \frac{|x_i - y_i|}{|x_i| + |y_i|}$$

```
1:  double d = Distance.Canberra(x, y);
```

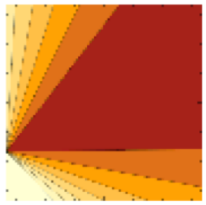
Cosine Distance



The cosine distance contains the dot product scaled by the product of the Euclidean distances from the origin. It represents the angular distance of two vectors while ignoring their scale.

$$d_{\cos} : (x, y) \mapsto 1 - \frac{\langle x, y \rangle}{\|x\|_2 \|y\|_2} = 1 - \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \sqrt{\sum_{i=1}^{n} y_i^2}}$$

Cosine Distance



The cosine distance contains the dot product scaled by the product of the Euclidean distances from the origin. It represents the angular distance of two vectors while ignoring their scale.

$$d_{\cos} : (x, y) \mapsto 1 - \frac{\langle x, y \rangle}{\|x\|_2 \|y\|_2} = 1 - \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \sqrt{\sum_{i=1}^{n} y_i^2}}$$

```
1:  double d = Distance.Cosine(x, y);
```

Pearson's Distance

The Pearson distance is a correlation distance based on Pearson's product-momentum correlation coefficient of the two sample vectors. Since the correlation coefficient falls between [-1, 1], the Pearson distance lies in [0, 2] and measures the linear relationship between the two vectors.

$$d_{\text{Pearson}} : (x, y) \mapsto 1 - \mathbf{Corr}(x, y)$$

```
1:  double d = Distance.Pearson(x, y);
```

Hamming Distance

The hamming distance represents the number of entries in the two sample vectors which are different. It is a fundamental distance measure in information theory but less relevant in non-integer numerical problems.

```
1:  double d = Distance.Hamming(x, y);
```

Distances: Quantitative Variables

Some examples

Identity (absolute) error

$$d_j(x_{ij}, x_{i'j}) = I(x_{ij} \neq x_{i'j})$$

Squared distance

$$d_j(x_{ij}, x_{i'j}) = (x_{ij} - x_{i'j})^2$$

L_q norms

$$L_{qii'} = \left[\sum_j |x_{ij} - x_{i'j}|^q\right]^{1/q}$$

Canberra distance

$$d_{ii'} = \sum_j \frac{|x_{ij} - x_{i'j}|}{|x_{ij} + x_{i'j}|}$$

Correlation

$$\rho(x_i, x_{i'}) = \frac{\sum_j (x_{ij} - \bar{x}_i)(x_{i'j} - \bar{x}_{i'})}{\sqrt{\sum_j (x_{ij} - \bar{x}_i)^2 \sum_j (x_{i'j} - \bar{x}_{i'})^2}}$$

Data point:

$$x_i = [x_{i1} \ldots x_{ip}]^T$$

Distances: Ordinal and Categorical Variables

- **Ordinal variables can be forced to lie within (0, 1) and then a quantitative metric can be applied:**

$$\frac{k-1/2}{M},\ k=1,2,\ldots,M$$

- **For categorical variables, distances must be specified by user between each pair of categories**
  - E.g., Distance in a hierarchy of categories
  - E.g., Jaccard distance between sets

Combining Distances

- **Different distance metrics for say, categorical variables and for numerical variables**
- **Often weighted sum is used:**

$$D(x_i, x_j) = \sum_{l=1}^{p} w_l d(x_{il}, x_{jl}), \quad \sum_{l=1}^{p} w_l = 1,\ w_l > 0.$$

Live Session Outline

- **Introduction**
- **Unsupervised Learning**
- **Clustering**
  - Kmeans
    - Flat Clustering via Kmeans graphically
    - Flat Clustering via Kmeans in code
    - Hyperparameters: different initializations
  - Hierarchical clustering
- **Similarity measures**
- **Evaluating Clustering**
- **Relatives of Kmeans, e.g., Expectation Maximization**
- **Summary**

- One of the main challenges in unsupervised learning is that we do not know the definitive answer.
- We don't have the ground truth class labels in our dataset that allow us to apply the techniques that we used for Model Evaluation and Hyperparameter Tuning, in order to evaluate the performance of a supervised model.

Quality Indexes: two families

- **Internal quality index**

$$\frac{Cluster\ Compactness}{Inter\ Cluster\ separtion}$$

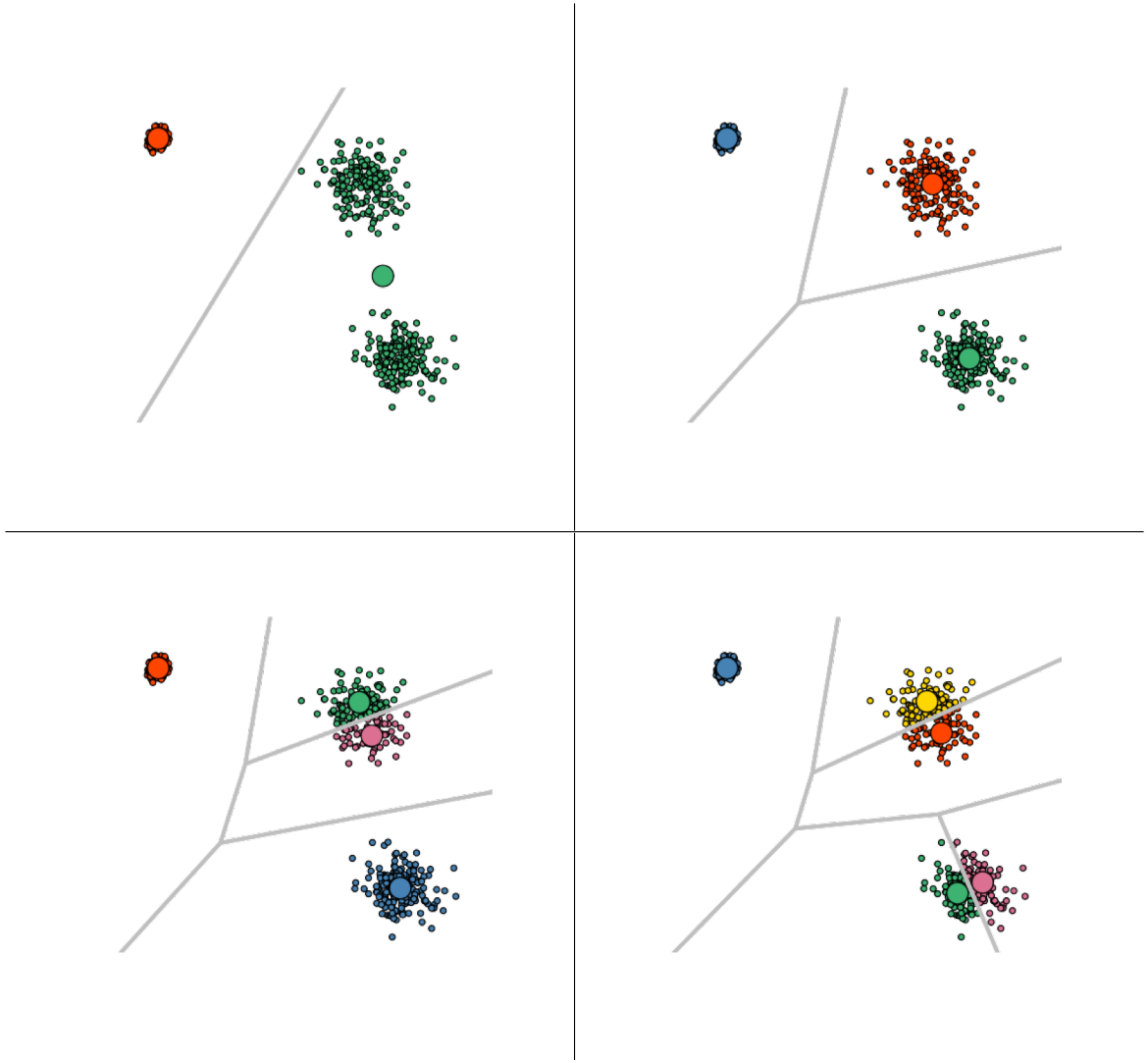
  - Thus, to quantify the quality of clustering, we need to use intrinsic metrics—such as the within-cluster SSE (AKA distortion) or average SSE
  - Dunn
  - Davis-Bouldin
- **External quality indexes**
  - Ground truth (known clusters; given labels for each example)

Thought experiment: What Is A Good Clustering?

- **Internal criterion: A good clustering will produce high quality clusters in which:**
  - the intra-class (that is, intra-cluster) similarity is high
  - the inter-class similarity is low
  - The measured quality of a clustering depends on both the document representation and the similarity measure used

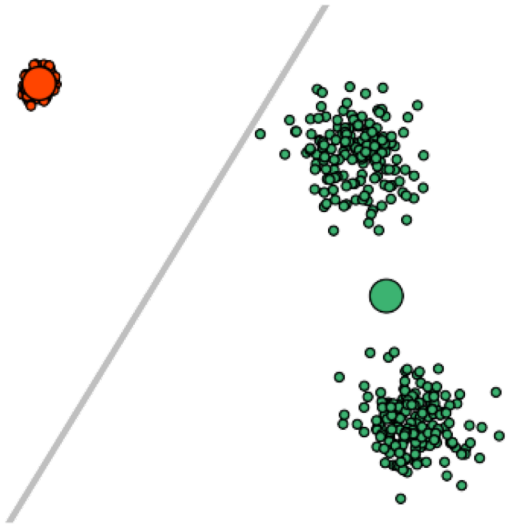
How many clusters k?

k-means can guess clusters, but not how many they are



Quality

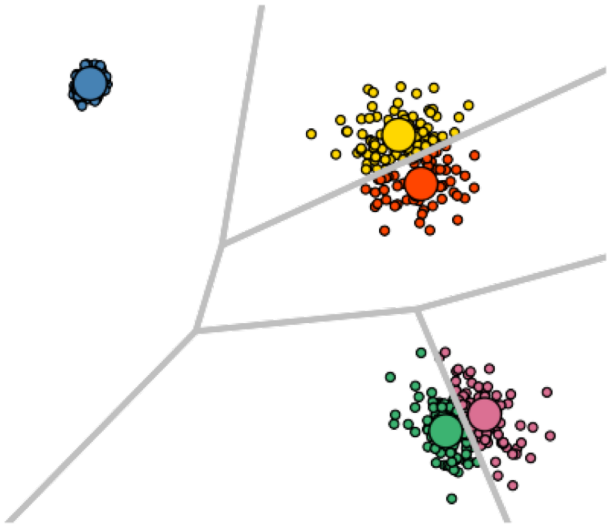
Wrong number of clusters ⇒ bad clustering



Not enough ⇒ big gaps within a single cluster

Quality

Wrong number of clusters ⇒ bad clustering



Too much ⇒ clusters very close to each other

Internal quality indexes

***Internal* quality indexes ⇒ uses the clustered data only**

Dunn index

Higher is better

$$\frac{\min_{1\le i\le n}\left(\min_{i<j\le n} d(c_i, c_j)\right)}{\max_{1\le i\le n} m_i}$$

Inter Cluster

Cluster Compactness

- $n \Rightarrow$ no. clusters
- $m_i \Rightarrow$ max. dist. of members of ith cluster with its center
- $d(c_i, c_j) \Rightarrow$ dist. between ith and kth cluster centres

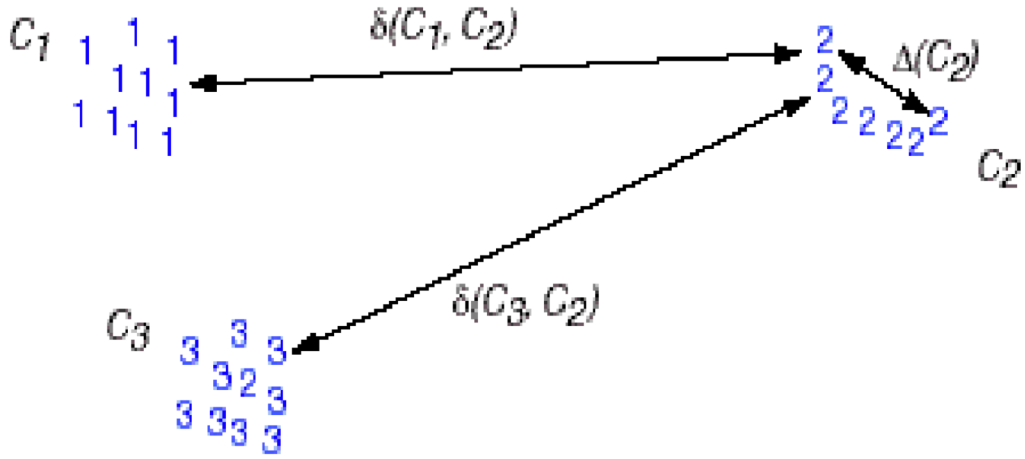
Cluster Validity

Definition

It reflects cluster **separability**

and formally depends on :

- Scatters inside clusters
- Separation between clusters



Indexes

It is formal characteristics of structure

- **Dunn** index
- **Davies Bouldin** index
- Hypervolume criterion (**Andre Hardy**)
- Density expected measure DEM (**Benno Stein**)

Dunn index (to be **max**)

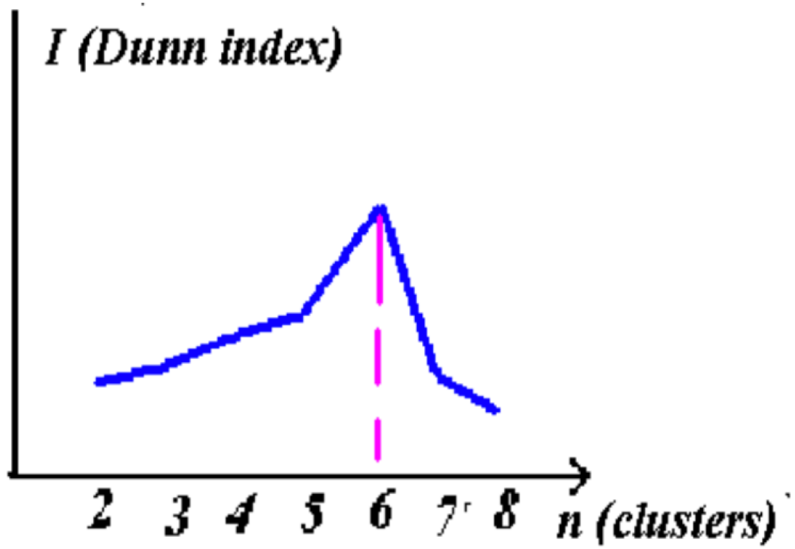
$$I(C) = \frac{\min_{i \neq j} \{\delta(C_i, C_j)\}}{\max_{1 \leq l \leq k} \{\Delta(C_l)\}}$$

Cluster Validity

Number of clusters

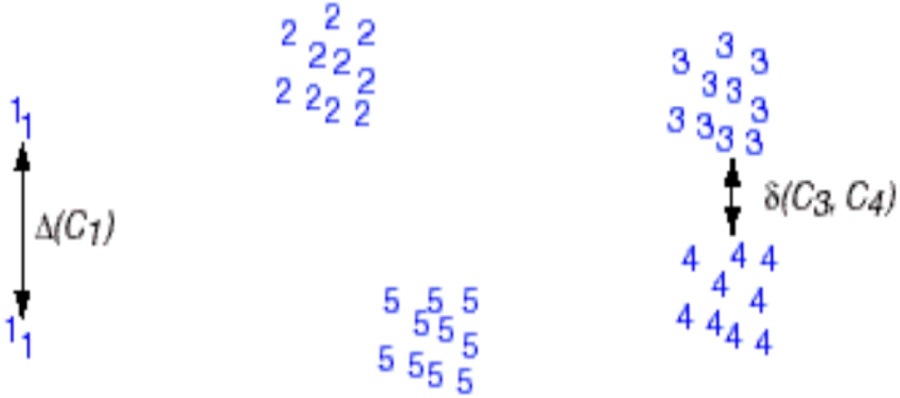
Geometrical approach, two variants:

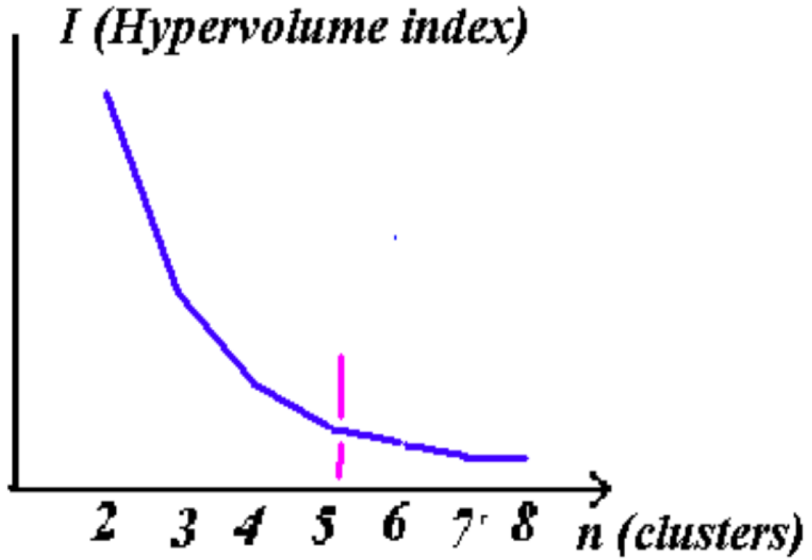
- **Optimum** (min, max) of curve
- **Jump** of curve



Dunn index (to be **max**) is too sensible to extremal cases

$$I(C) = \frac{\min_{i \neq j}\{\delta(C_i, C_j)\}}{\max_{1 \leq l \leq k}\{\Delta(C_l)\}}$$





Davies-Bouldin index

$$\frac{\text{Cluster Compactness}}{\text{Inter Cluster separtion}}$$

Lower is better

$$\frac{1}{n}\sum_{i=1}^{n}\max_{i\neq j}\left(\frac{m_i + m_j}{d(c_i, c_j)}\right)$$

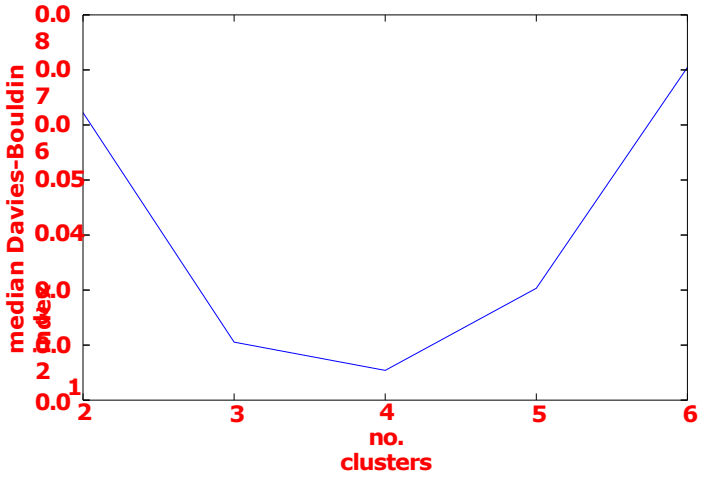
- $n \Rightarrow$ no. clusters
- $m_i \Rightarrow$ avg. dist. f members of ith cluster with its center
- $d(c_i, c_j) \Rightarrow$ dist. between ith and kth cluster centres

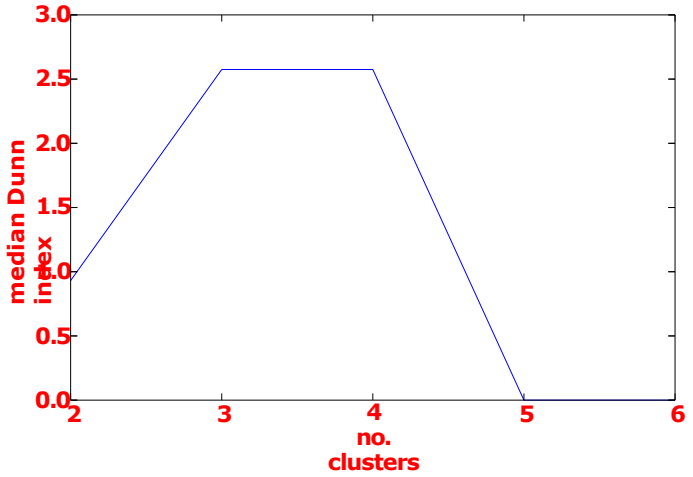
Using internal quality indexes

- **When computing internal quality indexes**
  - do it for different no. clusters
  - do it for several runs of k-means
  - take the result with a grain of salt !

Using internal quality indexes

Median *Dunn* & *Davies-Bouldin* indexes over 16 runs





According to the indices, 3 to 4 clusters seems ideal

Quality Indexes: two families

$$\frac{\textit{Cluster Compactness}}{\textit{Inter Cluster separtion}}$$

- **Internal quality index**
  - Thus, to quantify the quality of clustering, we need to use intrinsic metrics—such as the within-cluster SSE (AKA distortion) or average SSE
  - Dunn
  - Davis-Bouldin
- **External quality indexes**
  - Ground truth (known clusters; given labels for each example)
  - use clustering result on data not used to compute the clustering
    - Data used for clustering ⇒ training set A
    - Data used to test clustering ⇒ test set B

External criteria for clustering quality

- Quality measured by its ability to discover some or all of the hidden patterns or latent classes in gold standard data
- Assesses a clustering with respect to ground truth … requires *labeled data*
- Assume documents with C gold standard classes, while our clustering algorithms produce K clusters, $\omega_1, \omega_2, \ldots, \omega_K$ with n_i members.

External Evaluation of Cluster Quality

- **Simple measure: purity, the ratio between the dominant class in the cluster ω_i and the size of cluster ω_i**

$$Purity(\omega_i) = \frac{1}{n_i} \max_j (n_{ij}) \quad j \in C$$

- **Biased because having *n* clusters maximizes purity**
- **Others are entropy of classes in clusters (or mutual information between classes and clusters)**

Entropy

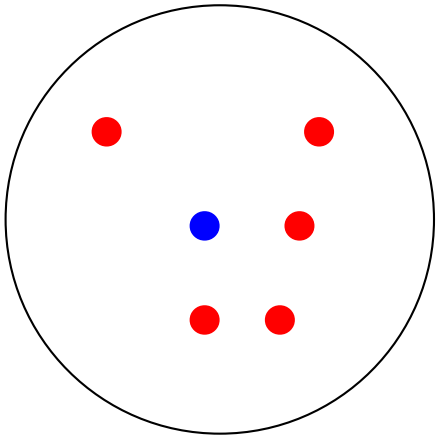
$$H = \sum_{i=1}^{N} -p(s_i)\log_2 p(s_i)$$

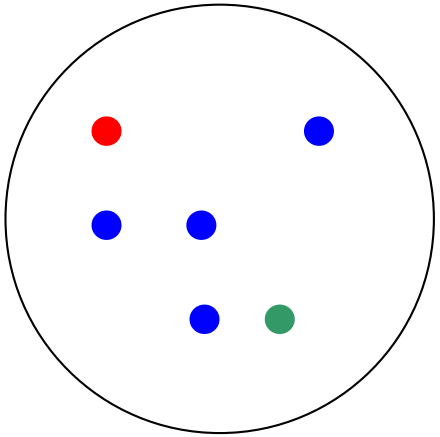
$$= -.25 * \log_2 .25 + -.30 * \log_2 .30 + -.12 * \log_2 .12 + -.15 * \log_2 .15 + -.18 * \log_2 .18$$

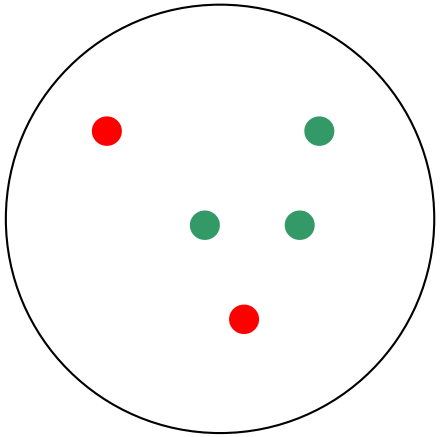
H = 2.24 bits

Symbol	P(S)	Code
A	0.25	11
B	0.30	00
C	0.12	010
D	0.15	011
E	0.18	10

Purity example: 3 gold classes; K=3







$$Purity(\omega_i) = \frac{1}{n_i}\max_j(n_{ij}) \quad j \in C$$

Cluster I

Cluster II

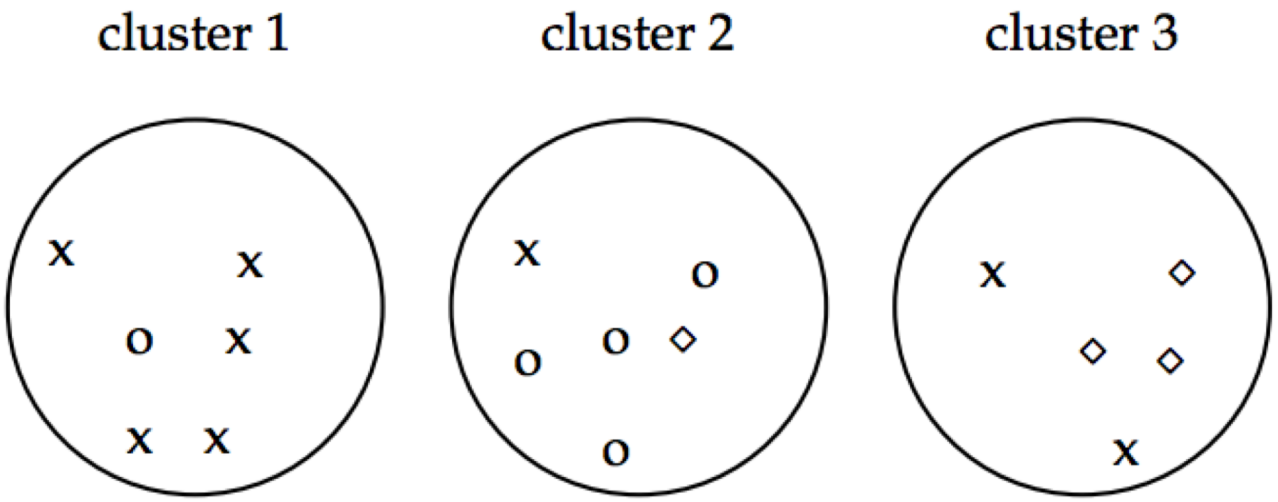
Cluster III

Cluster I: Purity = 1/6 (max(5, 1, 0)) = 5/6

Cluster II: Purity = 1/6 (max(1, 4, 1)) = 4/6

Cluster III: Purity = 1/5 (max(2, 0, 3)) = 3/5

Overall Purity



► **Figure 16.4** Purity as an external evaluation criterion for cluster quality. Majority class and number of members of the majority class for the three clusters are: x, 5 (cluster 1); o, 4 (cluster 2); and ◇, 3 (cluster 3). Purity is $(1/17) \times (5+4+3) \approx 0.71$.

	purity	NMI	RI	F_5
lower bound	0.0	0.0	0.0	0.0
maximum	1	1	1	1
value for Figure 16.4	0.71	0.36	0.68	0.46

Rand Index measures between pair decisions

An alternative to this information-theoretic interpretation of clustering is to view it as a series of decisions, one for each of the $N(N-1)/2$ pairs of documents in the collection. We want to assign two documents to the same cluster if and only if they are similar. A true positive (TP) decision assigns two similar documents to the same cluster, a true negative (TN) decision assigns two dissimilar documents to different clusters. There are two types of errors we can commit. A false positive (FP) decision assigns two dissimilar documents to the same cluster. A false negative (FN) decision assigns two similar documents to different clusters. The *Rand index* (RI) measures the percentage of decisions that are correct. That is, it is simply accuracy (Section 8.3, page 155).

$$\mathrm{RI} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN} + \mathrm{TN}}$$

TP = a pair of examples having the same label, are assigned to the same cluster
TN =a pair of examples having different labels, are assigned to different clusters
FP = a pair of examples having different labels, are assigned to the same cluster
FN = a pair of examples having the same label, are assigned to different clusters

Rand Index measures between pair decisions.

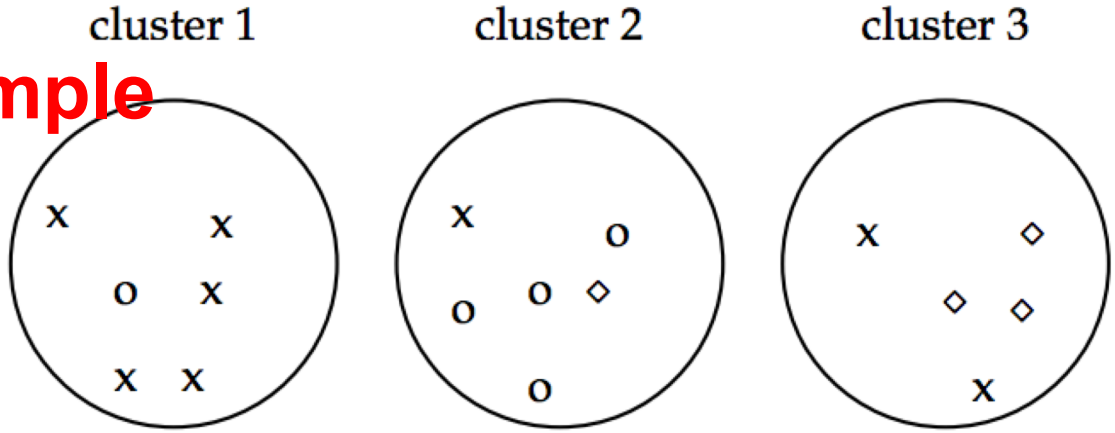
Here RI = 0.68

Sec. 16.3

0.6764 = 92/136

Number of point pairs	Same Cluster in clustering	Different Clusters in clustering
Same class in ground truth	20	24
Different classes in ground truth	20	72

Rand Index Example



As an example, we compute RI for Figure 16.4. We first compute TP + FP. The three clusters contain 6, 6, and 5 points, respectively, so the total number of "positives" or pairs of documents that are in the same cluster is:

$$\text{TP} + \text{FP} = \binom{6}{2} + \binom{6}{2} + \binom{5}{2} = 40$$

Of these, the x pairs in cluster 1, the o pairs in cluster 2, the ◇ pairs in cluster 3, and the x pair in cluster 3 are true positives:

$$\text{TP} = \binom{5}{2} + \binom{4}{2} + \binom{3}{2} + \binom{2}{2} = 20$$

Thus, FP = 40 − 20 = 20.

FN and TN are computed similarly, resulting in the following contingency table:

	Same cluster	Different clusters
Same class	TP = 20	FN = 24
Different classes	FP = 20	TN = 72

RI is then $(20 + 72)/(20 + 20 + 24 + 72) \approx 0.68$.

Rand index and Cluster F-measure

$$RI = \frac{A + D}{A + B + C + D}$$

Compare with standard Precision and Recall:

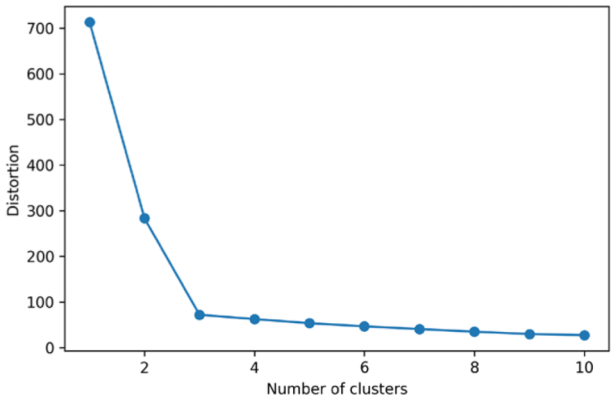
$$P = \frac{A}{A + B} \qquad R = \frac{A}{A + C}$$

People also define and use a cluster F-measure, which is probably a better measure.

elbow method: determine optimal k

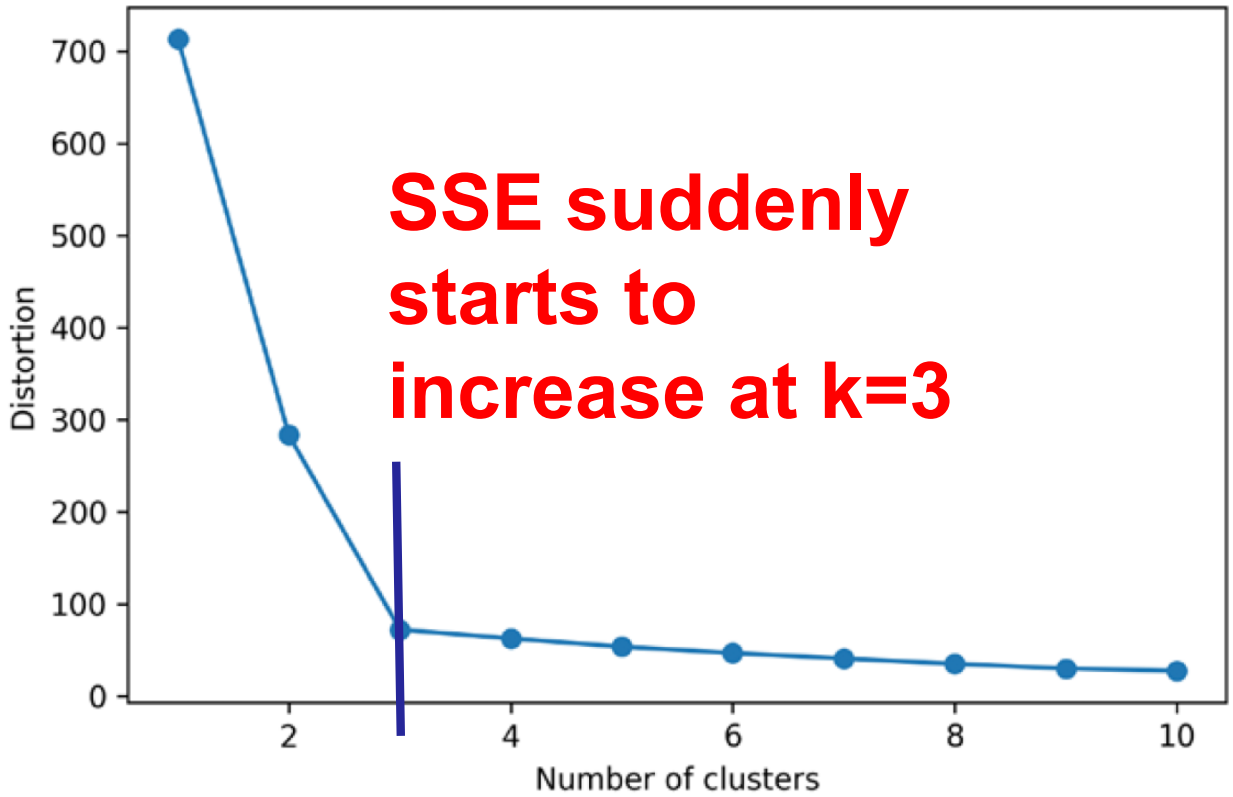
- **Based on the *within-cluster SSE*, we can use a graphical tool, the so-called elbow method, to estimate the optimal number of clusters k for a given task.**
- **Intuitively, we can say that, if k increases, the distortion will decrease.**
  - This is because the samples will be closer to the centroids they are assigned to.
- **The idea behind the elbow method is to identify the value of k where the distortion begins to increase most rapidly, which will become clearer if we plot the distortion for different values of k:**

SSE aka distortion



k versus SSE (aka distortion)

```
distortions = []
for i in range(1, 11):
    km = KMeans(n_clusters=i,
                init='k-means++',
                n_init=10,
                max_iter=300,
                random_state=0)
    km.fit(X)
    distortions.append(km.inertia_)
```




```
plt.plot(range(1,11), distortions, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Distortion')
plt.show()
```

) MIDS-MLS-2016-Spring - Google Groups
s://groups.google.com/forum/#!forum/mids-2016-spring

get-2016-04/Classes-India-2016-08-29/Lecture08-Kmeans/D3.%20K-Means%20Clustering%20Analysis.ipynb

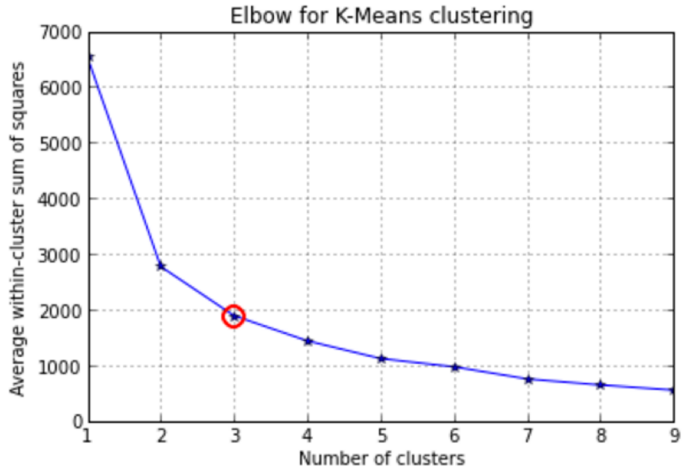
intro... nbviewer.ipython.org Bookmarks (2) MIDS-MLS-201... Stanford Machine L... Getting Started Statistical Analysis... eBay/Google 2013:...

jupyter D3. K-Means Clustering Analysis Last Checkpoint: 06/19/2016 (autosaved)

File Edit View Insert Cell Kernel Help Python [default]

Markdown CellToolbar

In [10]:
```
kIdx = 2
# plot elbow curve
fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(K, avgWithinSS, 'b*-')
ax.plot(K[kIdx], avgWithinSS[kIdx], marker='o', markersize=12,
      markeredgewidth=2, markeredgecolor='r', markerfacecolor='None')
plt.grid(True)
plt.xlabel('Number of clusters')
plt.ylabel('Average within-cluster sum of squares')
tt = plt.title('Elbow for K-Means clustering')
```



See notes for link

So we see that a good K to use in our model would be 3. We now use the KMeans modeling software to fit clusters to our data and then plot how the software has clustered our data. In this case we look at how the data cluster when we plot Infant Mortality against GDP.

Determine the knee of a curve using 2nd derivatives

https://www.dropbox.com/s/k56q48vviugax0x/Find-theknee-in-a-curve-kmeans.R?dl=0

```
library(ggplot2)
p.corners <- data.frame(rbind(c(0, 6), c(80, 12), c(100, 100)))
colnames(p.corners) <- c("x", "y")
x.a <- 1:p.corners[2, "x"]
y.a <- (p.corners[2, "y"]-p.corners[1, "y"])/(p.corners[2, "x"]-p.corners[1, "x"])*x.a+p.corners[1, "y"]
x.b <- (p.corners[2, "x"]+1):100
y.b <- (p.corners[3, "y"]-p.corners[2, "y"])/(p.corners[3, "x"]-p.corners[2, "x"])*x.b+p.corners[2, "y"]-(((p.corners[3, "y"]-p.corners[2, "y"])/(p.corners[3, "x"]-p.corners[2, "x"]))*p.corners[2, "x"])
x <- c(x.a, x.b)
y <- c(y.a, y.b)
p.random <- data.frame(cbind(x, y))
p.random$y.random <- y + 20*(runif(20)-0.5)

p <- ggplot(p.random, aes(x=x, y=y.random))
p <- p + geom_point()
p <- p + xlim(0, 100)
p <- p + ylim(0, 100)
p <- p + labs(title="Measured samples")
p
p <- ggplot(p.random, aes(x=x, y=y))
p <- p + geom_line()
p <- p + xlim(0, 100)
p <- p + ylim(0, 100)
p <- p + labs(title="Underlying curve")
p


> p.random$y
  [1]   6.075   6.150   6.225   6.300   6.375   6.450   6.525   6.600   6.675   6.750
 [11]   6.825   6.900   6.975   7.050   7.125   7.200   7.275   7.350   7.425   7.500
 [21]   7.575   7.650   7.725   7.800   7.875   7.950   8.025   8.100   8.175   8.250
 [31]   8.325   8.400   8.475   8.550   8.625   8.700   8.775   8.850   8.925   9.000
 [41]   9.075   9.150   9.225   9.300   9.375   9.450   9.525   9.600   9.675   9.750
 [51]   9.825   9.900   9.975  10.050  10.125  10.200  10.275  10.350  10.425  10.500
 [61]  10.575  10.650  10.725  10.800  10.875  10.950  11.025  11.100  11.175  11.250
 [71]  11.325  11.400  11.475  11.550  11.625  11.700  11.775  11.850  11.925  12.000
 [81]  16.400  20.800  25.200  29.600  34.000  38.400  42.800  47.200  51.600  56.000
 [91]  60.400  64.800  69.200  73.600  78.000  82.400  86.800  91.200  95.600 100.000
> diff(p.random$y, differences = 1)
 [1] 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075
[15] 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075
[29] 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075
[43] 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075
[57] 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075
[71] 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075 4.400 4.400 4.400 4.400 4.400
[85] 4.400 4.400 4.400 4.400 4.400 4.400 4.400 4.400 4.400 4.400 4.400 4.400 4.400 4.400 4.400
[99] 4.400



which.max((diff(diff(p.random$y, differences = 1))))
#[1] 79
```

Final word and resources

- **In clustering, clusters are inferred from the data without human input (unsupervised learning)**
- **However, in practice, it's a bit less clear: there are many ways of influencing the outcome of clustering: number of clusters, similarity measure, representation of documents, . . .**

- **Resources**
  - IR Book IIR 16 except 16.5
  - IIR 17.1–17.3

Live Session Outline

- **Introduction**
- **Unsupervised Learning**
- **Clustering**
  - Kmeans
    - Flat Clustering via Kmeans graphically
    - Flat Clustering via Kmeans in code
    - Hyperparameters: different initializations
  - Hierarchical clustering
- **Similarity measures**
- **Evaluating Clustering**
- **Relatives of Kmeans, e.g., Expectation Maximization**
- **Summary**

K-Means Clustering Algorithm

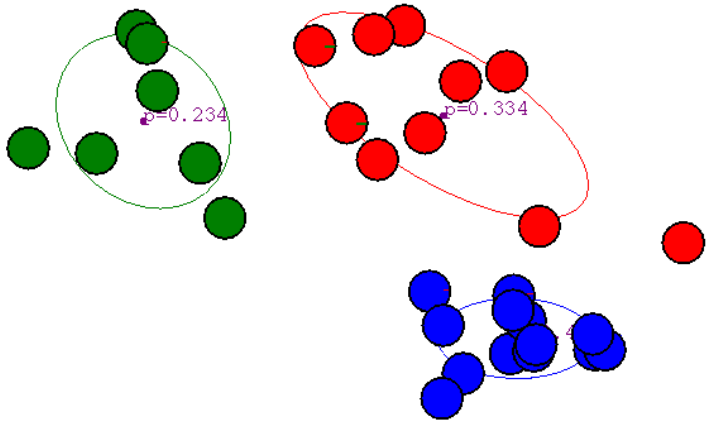
- **Initialize our K cluster centers**
- **Perform K-Means Loop**

E – The "assignment" step is also referred to as expectation step,

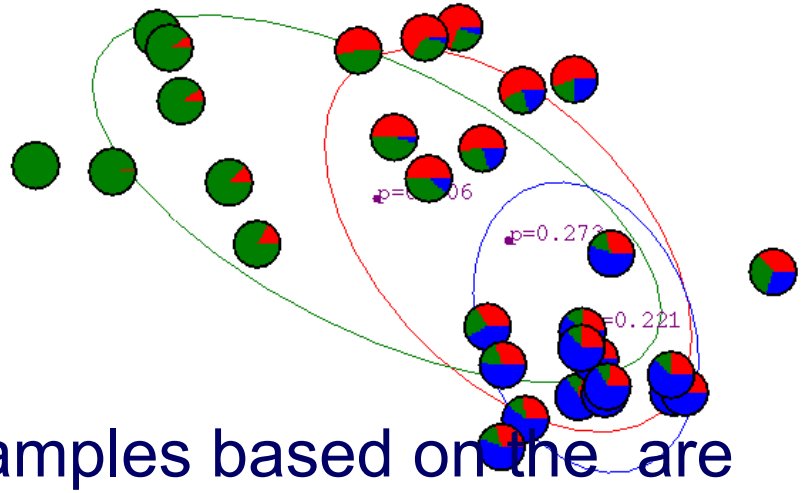
M – The "update step" as maximization step, making this algorithm a variant of the *generalized* expectation-maximization algorithm.

- **Until Convergence**

Flat Clustering: Hard versus Soft



- **Hard Clustering**
  - Hard assignment; Kmeans (EM-like)
- **Soft Clustering**
  - This set of assignment probabilities (aka responsibilities) defines a soft clustering.
  - Model-based Clustering using EM and Maximum Likelihood



  - Weighted EM-like
    - EM Centroids are weighted examples based on the are the cluster/class probabilities assignment probabilities (aka responsibilites)

EM Algorithm: distances vs Distributions

16.5 Model-based clustering

In this section, we describe a generalization of *K*-means ... EM algorithm. It can be applied to a larger variety of document r... ...ns and distributions than *K*-means.

In *K*-means, we attempt to find centroid... ...good representatives. We can view the set of *K* centroids as a m... ...enerates the data. Generating a document in this model co... ...picking a centroid at random and then adding some noise. ... ...is normally distributed, this procedure will result in clusters ... ...al shape. *Model-based clustering* assumes that the data were generated by a model and tries to recover the original model from the data. The model that we recover from the data then defines clusters and an assignment of documents to clusters.

MODEL-BASED CLUSTERING

Cover in a later lecture

A commonly used criterion for estimating the model parameters is maximum likelihood.

Hard clustering versus soft clustering

- **Soft clustering via FCM**
  - In contrast, algorithms for soft clustering (sometimes also called fuzzy clustering) assign a sample to one or more clusters.
- **History of Fuzzy C-Means (FCM)**
  - A popular example of soft clustering is the fuzzy C-means (FCM) algorithm (also called soft k-means or fuzzy k-means). The original idea goes back to the 1970s, when Joseph C. Dunn first proposed an early version of fuzzy clustering to improve k-means (A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters, J. C. Dunn, 1973).
  - Almost a decade later, James C. Bedzek published his work on the improvement of the fuzzy clustering algorithm, which is now known as the FCM algorithm (Pattern Recognition with Fuzzy Objective Function Algorithms, J. C. Bezdek, Springer Science+Business Media, 2013).

FCM: 3 cluster scenario

- **The FCM procedure is very similar to k-means.**
  - However, we replace the hard cluster assignment with probabilities for each point belonging to each cluster.
  - In k-means, we could express the cluster membership of a sample x with a sparse vector of binary values

$$\begin{bmatrix} \mu^{(1)} \rightarrow 0 \\ \mu^{(2)} \rightarrow 1 \\ \mu^{(3)} \rightarrow 0 \end{bmatrix}$$

Here, the index position with value 1 indicates the cluster centroid $\mu^{(j)}$ the sample i assigned to (assuming $k = 3,\ j \in \{1, 2, 3\}$). In contrast, a membership vector in FCM could be represented as follows:

$$\begin{bmatrix} \mu^{(1)} \rightarrow 0.10 \\ \mu^{(2)} \rightarrow 0.85 \\ \mu^{(3)} \rightarrow 0.05 \end{bmatrix}$$

FCM Algorithm: m is the fuzzy coefficient

1. Specify the number of k centroids and randomly assign the cluster memberships for each point.
2. Compute the cluster centroids $\mu^{(j)}$, $j \in \{1,\ldots,k\}$.
3. Update the cluster memberships for each point.
4. Repeat steps 2 and 3 until the membership coefficients do not change, or a user-defined tolerance or maximum number of iterations is reached.

Kmeans Objective function

$$\text{Minimize } SSE = \sum_{i=1}^{n}\sum_{j=1}^{k} w^{(i,j)} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(j)} \right\|_2^2$$

FCM Objective function

$$\text{Minimize } J_m = \sum_{i=1}^{n}\sum_{j=1}^{k} w^{m(i,j)} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(j)} \right\|_2^2$$

The exponent m, any number greater than or equal to one (typically m=2), is the so-called fuzziness coefficient (or simply fuzzifier) that controls the degree of fuzziness.
The larger the value of m the smaller the cluster membership w(i, j) becomes, which leads to fuzzier clusters

- **The cluster membership probability itself is calculated as follows:**

$$w^{(i,j)} = \left[\sum_{p=1}^{k} \left(\frac{\left\| \boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(j)} \right\|_2}{\left\| \boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(p)} \right\|_2} \right)^{\frac{2}{m-1}} \right]^{-1}$$

Membership of $x^{(i)}$ in cluster $\mu^{(j)}$

For example, if we chose three cluster centers as in the previous k-means example, we could calculate the membership of the $x^{(i)}$ sample belonging to the $\mu^{(j)}$ cluster as follows:

Distance from $x^{(i)}$ in cluster $\mu^{(j)}$

Distance from $x^{(i)}$ in cluster $\mu^{(1)}$

$$w^{(i,j)} = \left[\left(\frac{\left\| \boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(j)} \right\|_2}{\left\| \boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(1)} \right\|_2} \right)^{\frac{2}{m-1}} + \left(\frac{\left\| \boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(j)} \right\|_2}{\left\| \boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(2)} \right\|_2} \right)^{\frac{2}{m-1}} + \left(\frac{\left\| \boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(j)} \right\|_2}{\left\| \boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(3)} \right\|_2} \right)^{\frac{2}{m-1}} \right]^{-1}$$

The center $\boldsymbol{\mu}^{(j)}$ of a cluster itself is calculated as the mean of all samples weighted by the degree to which each sample belongs to that cluster ($w^{m(i,j)}$):

$$\boldsymbol{\mu}^{(j)} = \frac{\sum_{i=1}^{n} w^{m(i,j)} \boldsymbol{x}^{(i)}}{\sum_{i=1}^{n} w^{m(i,j)}}$$

FCM is not available in SKLearn

- It is intuitive to say that each iteration in FCM is more expensive than an iteration in k-means.
- However, FCM typically requires fewer iterations overall to reach convergence.

- Unfortunately, the FCM algorithm is currently not implemented in scikit-learn

Live Session Outline

- **Introduction**
- **Unsupervised Learning**
- **Clustering**
  - Kmeans
    - Flat Clustering via Kmeans graphically
    - Flat Clustering via Kmeans in code
    - Hyperparameters: different initializations
  - Hierarchical clustering
- **Similarity measures**
- **Evaluating Clustering**
- **Relatives of Kmeans, e.g., Expectation Maximization**
- **Summary**

End of lecture