Unsupervised Learning I

Dr. Alex Williams

October 19, 2020

THE UNIVERSITY OF TENNESSEE KNOXVILLE

COSC 425: Introduction to Machine Learning
Fall 2020 (CRN: 44874)

Next Time

We will address:

1. kMeans
2. Hierarchical Clustering

Supervised Learning: Classification

Use-Case Criteria:

- You have output variables, i.e. y_i..
- Your OVs are **discrete / categorical**.

Example: Spam Filtering

- ***Goal***: Learn a function from categorical output.
- e.g. {spam, not spam}

	isUTKEmail	HeaderKeyword	Word 1	Word 2		isSpam
x1	Yes	CS425	Hi	Prof	…	No
x2	Yes	Orientation	Alex	You	…	No
x2	No	urgent	Dear	Sir	…	Yes
x4	No	cash	hello	I	…	Yes
x5	No	help	are	you	…	Yes
x6	Yes	Survey	Faculty	this	…	No
			…			

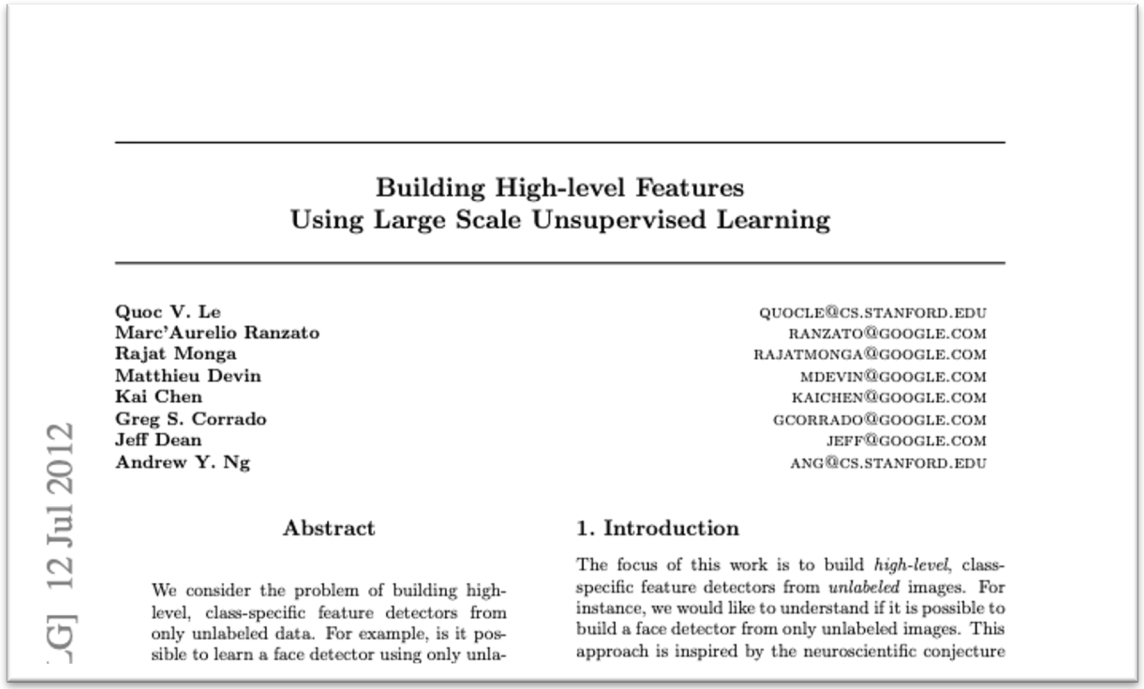
Unsupervised Learning: Feature Selection

Long-Term Goal.

- Figure out which inputs matter.

Feasible, but Challenging.

- Data, data, and more data.



Building High-level Features
Using Large Scale Unsupervised Learning

Quoc V. Le QUOCLE@CS.STANFORD.EDU
Marc'Aurelio Ranzato RANZATO@GOOGLE.COM
Rajat Monga RAJATMONGA@GOOGLE.COM
Matthieu Devin MDEVIN@GOOGLE.COM
Kai Chen KAICHEN@GOOGLE.COM
Greg S. Corrado GCORRADO@GOOGLE.COM
Jeff Dean JEFF@GOOGLE.COM
Andrew Y. Ng ANG@CS.STANFORD.EDU

Abstract

We consider the problem of building high-level, class-specific feature detectors from only unlabeled data. For example, is it possible to learn a face detector using only unla-

1. Introduction

The focus of this work is to build *high-level*, class-specific feature detectors from *unlabeled* images. For instance, we would like to understand if it is possible to build a face detector from only unlabeled images. This approach is inspired by the neuroscientific conjecture

[+2000 Citations!](https://arxiv.org/pdf/1112.6209.pdf)
https://arxiv.org/pdf/1112.6209.pdf

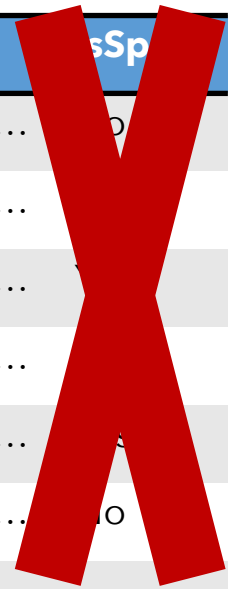
Supervised Learning: Deterrents

Challenges in Practice

→ Finding a ”labeler” may be difficult, expensive or impossible.

→ Unsupervised learning is concerned with learning without a teacher.

	isUTKEmail	HeaderKeyword	Word 1	Word 2		sSp
x1	Yes	CS425	Hi	Prof	…	o
x2	Yes	Orientation	Alex	You	…	
x2	No	urgent	Dear	Sir	…	
x4	No	cash	hello	I	…	
x5	No	help	are	you	…	
x6	Yes	Survey	Faculty	this	…	o
		…				



Unsupervised Learning

In unsupervised learning, data consists only of examples and not the corresponding labels.

→ Our job is to make sense of or find some pattern of regularity in the data even though no one has provided correct labels.

For example, we might want to do:

- **Clustering**: Automatically partition the data into groups.

- **Dimensionality Reduction**: Project high dimension data into lower dimension space, so it can be more easily visualized.

Overview

1. **kMeans**

2. **Hierarchical Clustering**

A Simple Clustering Example

A fruit merchant approaches you with a set of apples to classify according to their variety.

→ Tells you there are five varieties of apples in the basket.

→ Tells you the weight and color of each apple in the basket.

Can you label each apple with the correct variety?

→ What would you need to know / assume?

A Simple Clustering Example

Representation

Data = $\langle x_1, ? \rangle, \langle x_2, ? \rangle, \ldots, \langle x_n, ? \rangle$

You know there are 5 varieties

→ $y = \{y_1, y_2, y_3, y_4, y_5\}$

Assume each variety generates apples according to some 2D Gaussian distribution.

→ If you know μ_i, σ_i^2 for each class, it's easy to classify the apples.

→ If you know the class of each apple, it's easy to estimate μ_i, σ_i^2.

What if we know neither?

Chicken and Egg Problem.

In unsupervised clustering, the goal is to find clusters in the data.

We represent each cluster by its center, i.e. its centroid.

→ If we know the cluster's center, we can assign each point to its nearest cluster.

→ If we know which points belong to which clusters, then we can compute the center.

This is a chicken-and-egg problem, which can be solved iteratively.

1. Guess the centroids.
2. Assign point closest to centers.
3. Recompute centers.
4. Repeat until clusters stop moving.

→ This is the **K-means algorithm**.

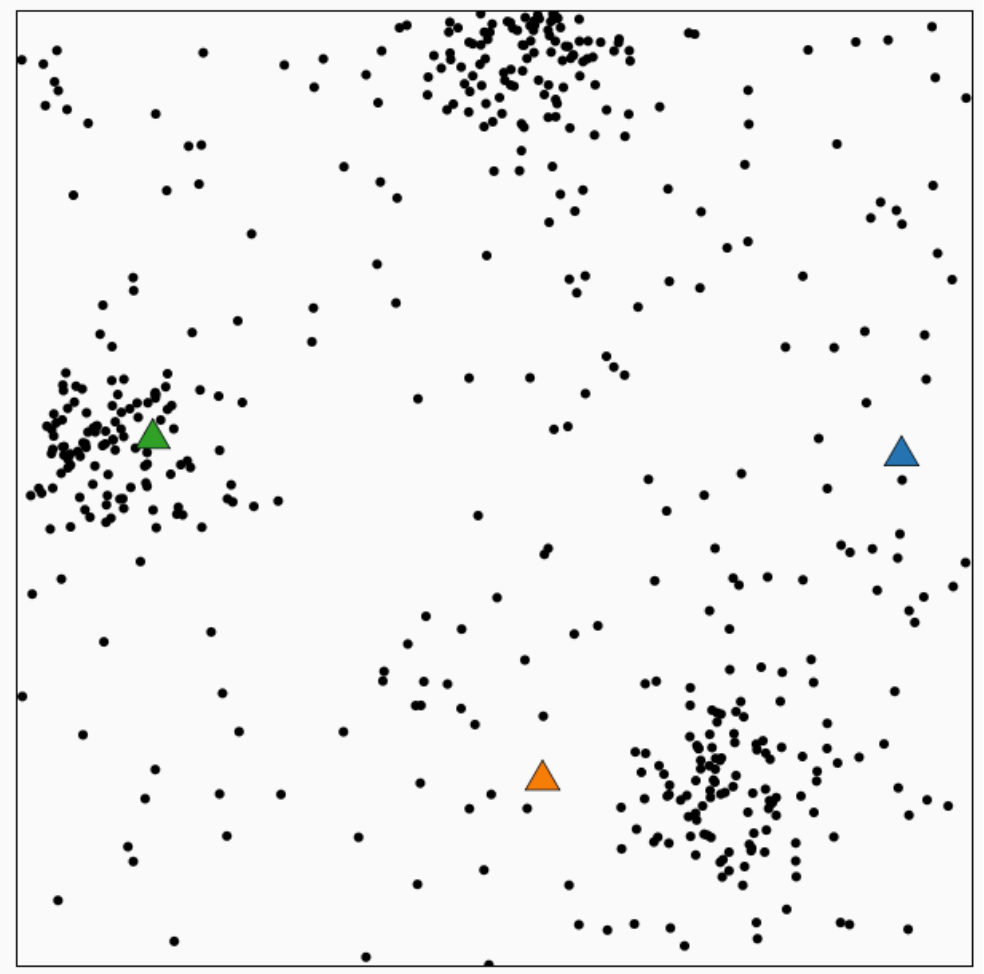
A Simple Algorithm: K-means clustering

Objective: Cluster *n* instances into K distinct classes.

Preliminaries:
Step 1: Pick the desired number of clusters, K.
Step 2: Assume a parametric distribution for each class (e.g. Gaussian)
Step 3: Randomly estimate the parameters of the *K* distributions.

Iterate, until convergence:
Step 4: Assign instances to the most likely classes based on the current parametric distributions.
Step 5: Estimate the parametric distribution of each class based on the latest assignment.

kMeans: Visualization



https://stanford.edu/class/engr108/visualizations/kmeans/kmeans.html

Procedure

Step 1: Ask user how many clusters.

Step 2: Randomly guess ***k*** centers.

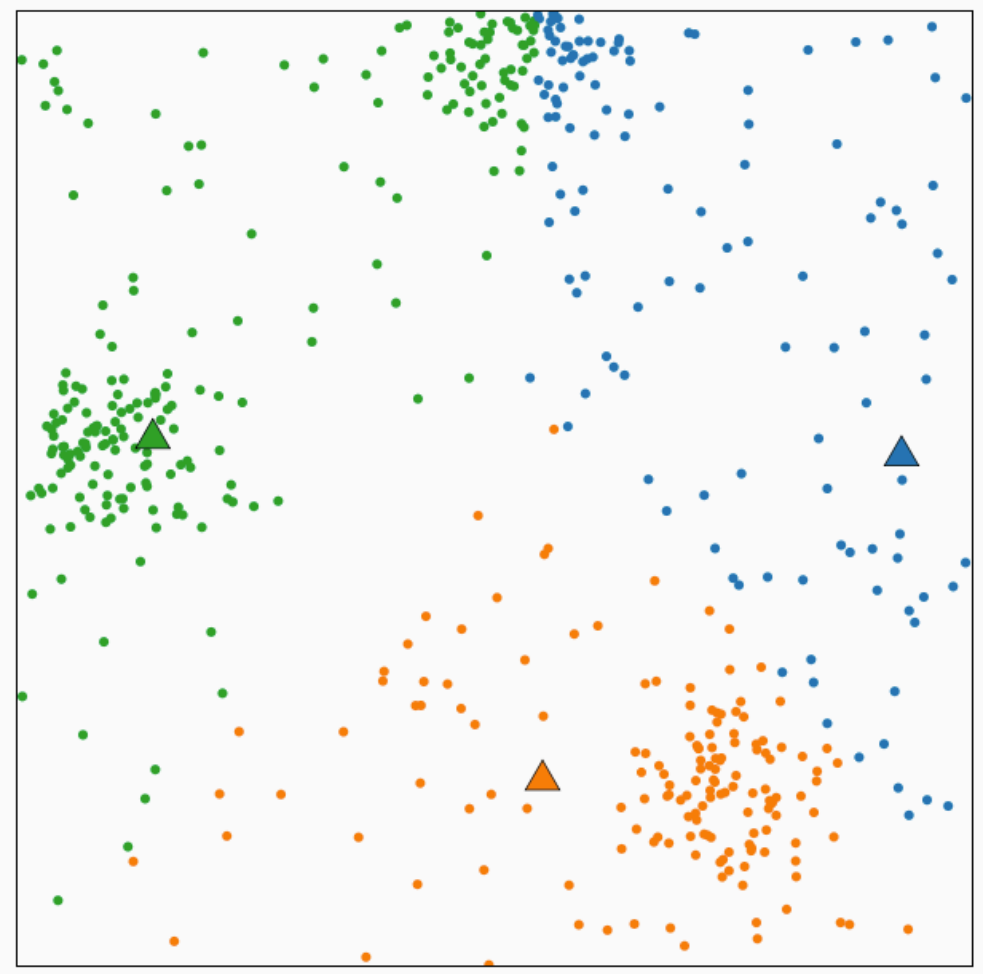
Step 3: Assign each data point to the closest center.

Step 4: Each center finds the centroid of the point it owns.

Step 5: Repeat

kMeans: Visualization

Iteration: 1



https://stanford.edu/class/engr108/visualizations/kmeans/kmeans.html

Procedure

Step 1: Ask user how many clusters.

Step 2: Randomly guess ***k*** centers.

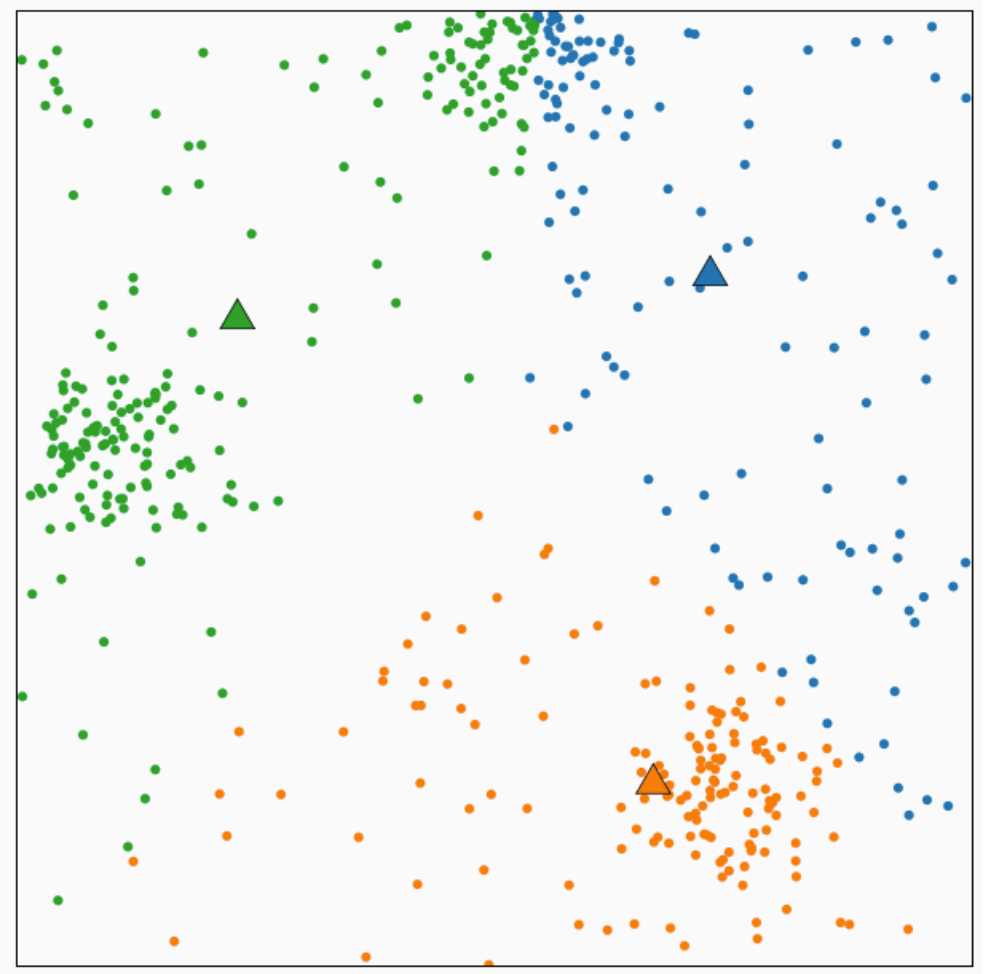
Step 3: Assign each data point to the closest center.

Step 4: Each center finds the centroid of the point it owns.

Step 5: Repeat

kMeans: Visualization

Iteration: 1



https://stanford.edu/class/engr108/visualizations/kmeans/kmeans.html

Procedure

Step 1: Ask user how many clusters.

Step 2: Randomly guess ***k*** centers.

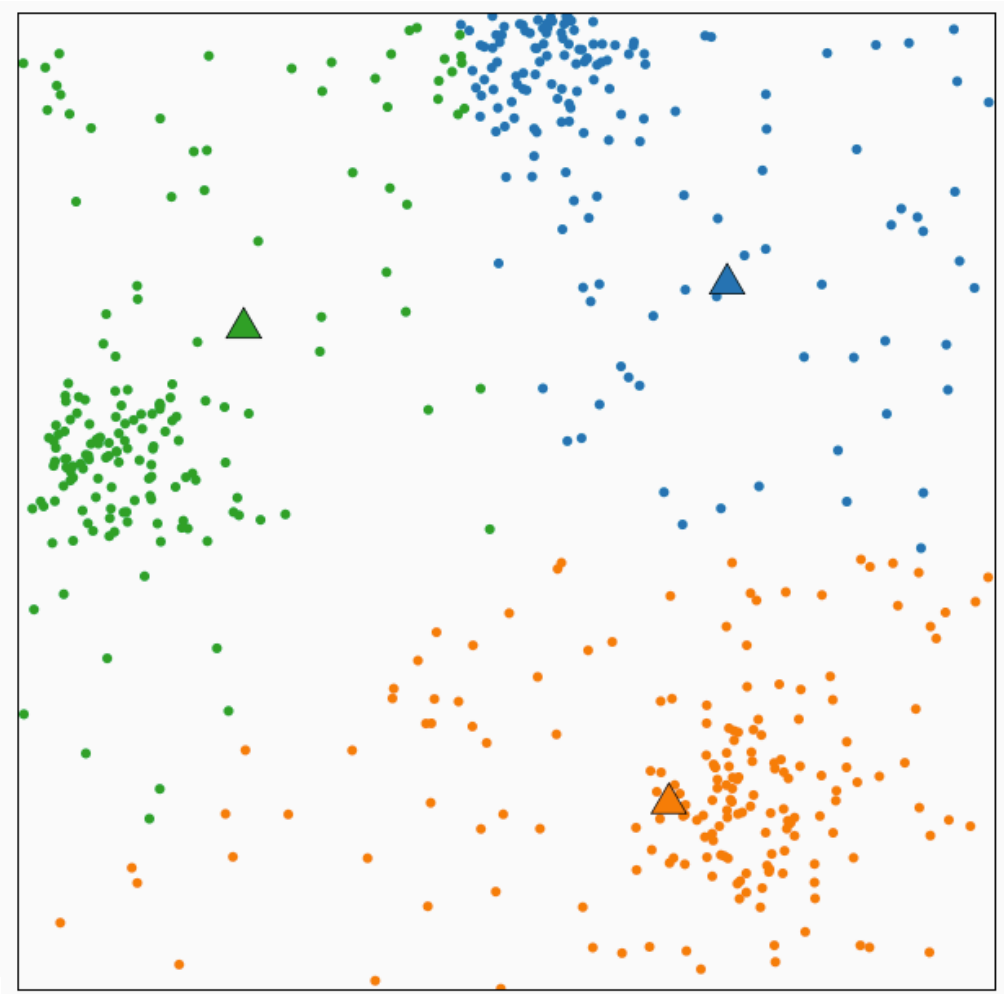
Step 3: Assign each data point to the closest center.

Step 4: Each center finds the centroid of the point it owns.

Step 5: Repeat

kMeans: Visualization

Iteration: 2



https://stanford.edu/class/engr108/visualizations/kmeans/kmeans.html

Procedure

Step 1: Ask user how many clusters.

Step 2: Randomly guess ***k*** centers.

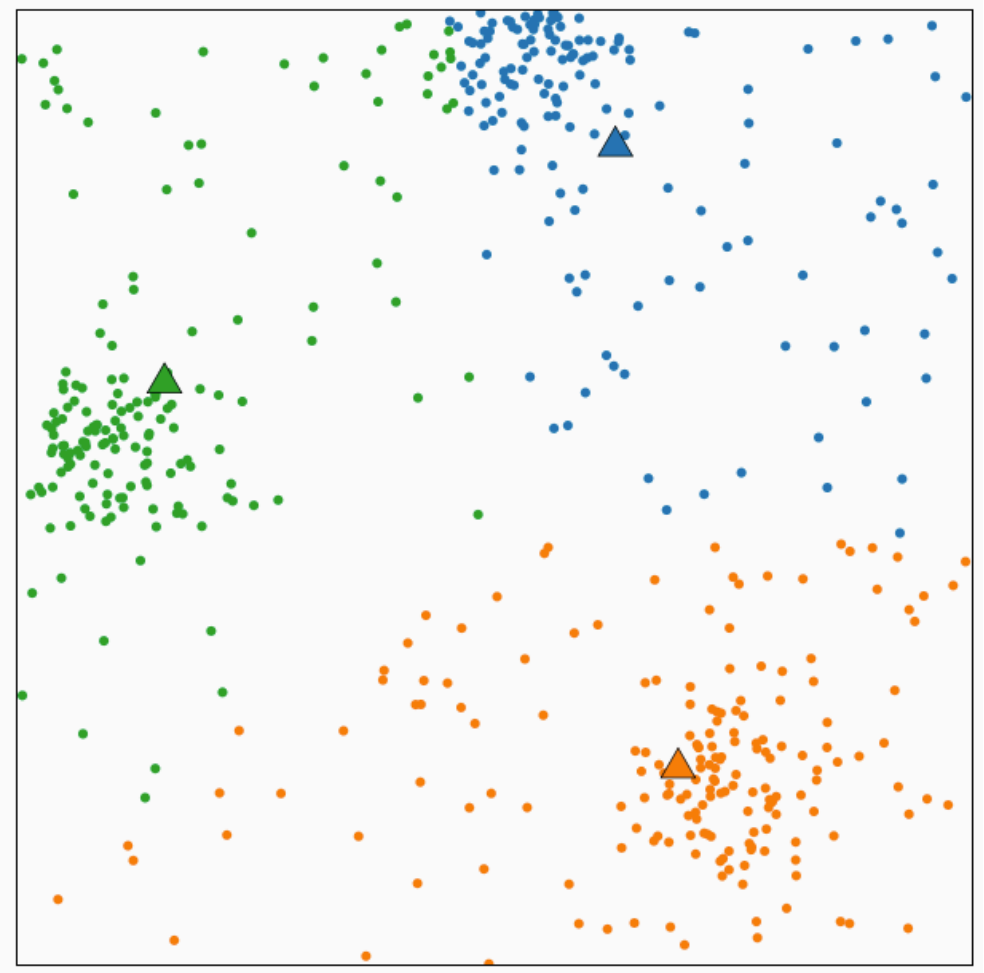
Step 3: Assign each data point to the closest center.

Step 4: Each center finds the centroid of the point it owns.

Step 5: Repeat

kMeans: Visualization

Iteration: 2



https://stanford.edu/class/engr108/visualizations/kmeans/kmeans.html

Procedure

Step 1: Ask user how many clusters.

Step 2: Randomly guess ***k*** centers.

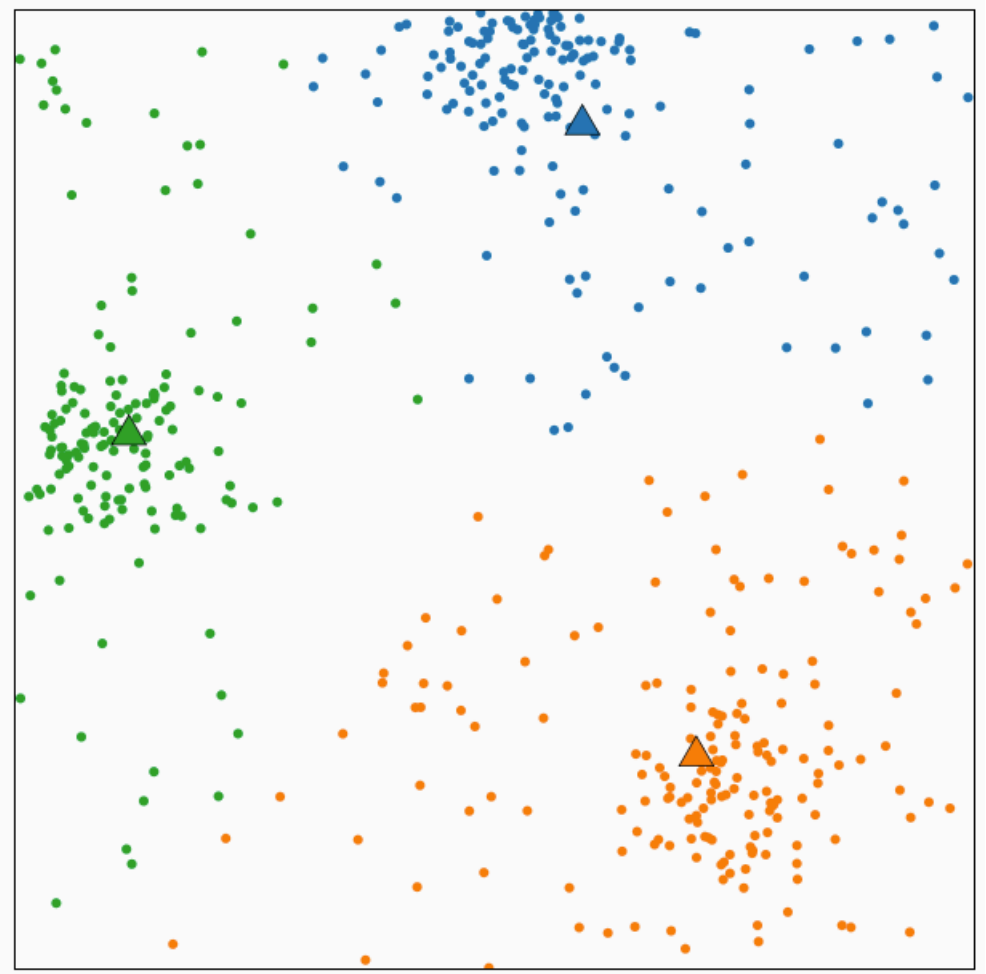
Step 3: Assign each data point to the closest center.

Step 4: Each center finds the centroid of the point it owns.

Step 5: Repeat

kMeans: Visualization

Iteration: 3



https://stanford.edu/class/engr108/visualizations/kmeans/kmeans.html

Procedure

Step 1: Ask user how many clusters.

Step 2: Randomly guess ***k*** centers.

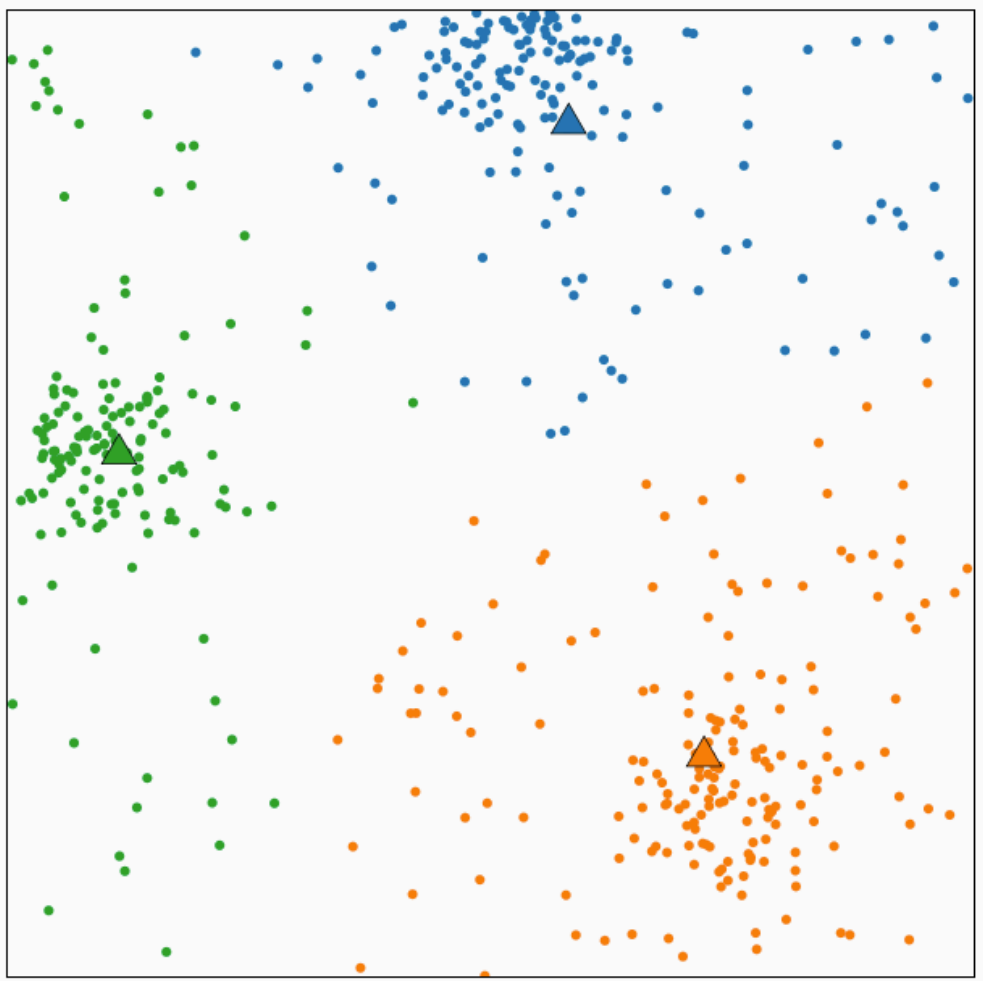
Step 3: Assign each data point to the closest center.

Step 4: Each center finds the centroid of the point it owns.

Step 5: Repeat

kMeans: Visualization

Iteration: 5



https://stanford.edu/class/engr108/visualizations/kmeans/kmeans.html

Procedure

Step 1: Ask user how many clusters.

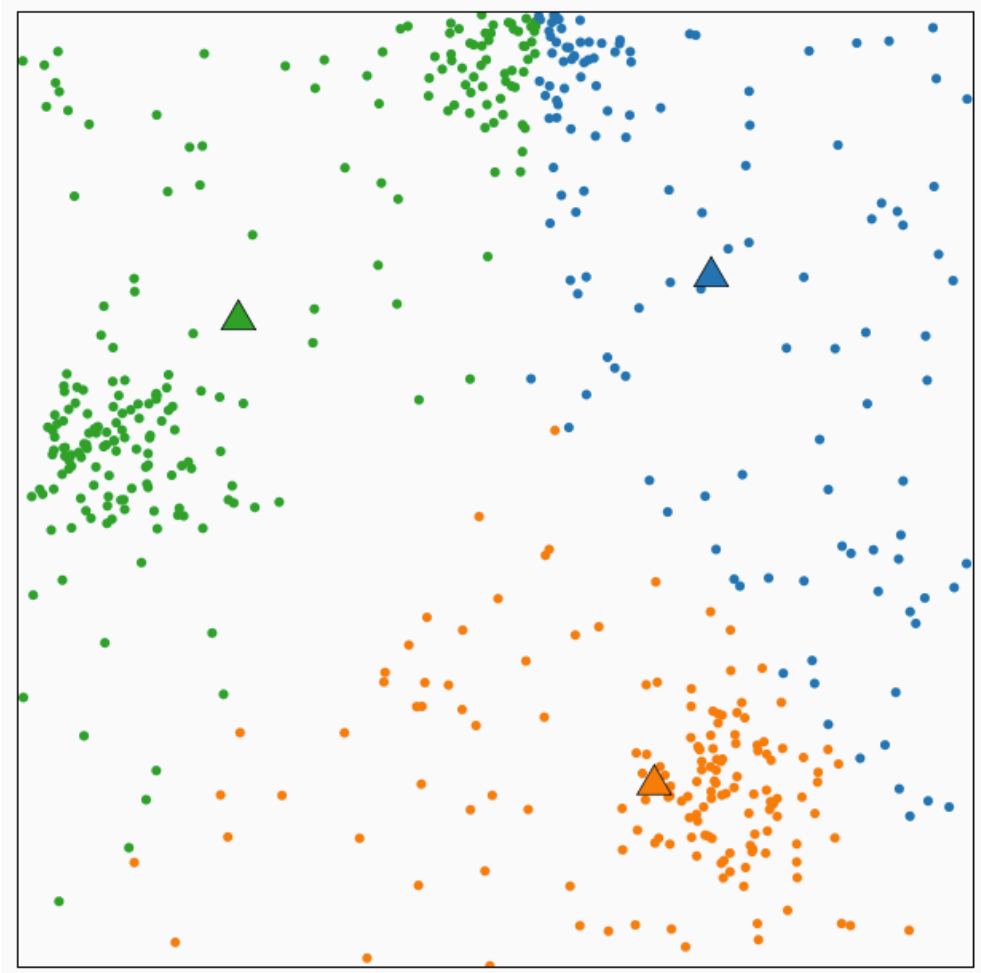
Step 2: Randomly guess ***k*** centers.

Step 3: Assign each data point to the closest center.

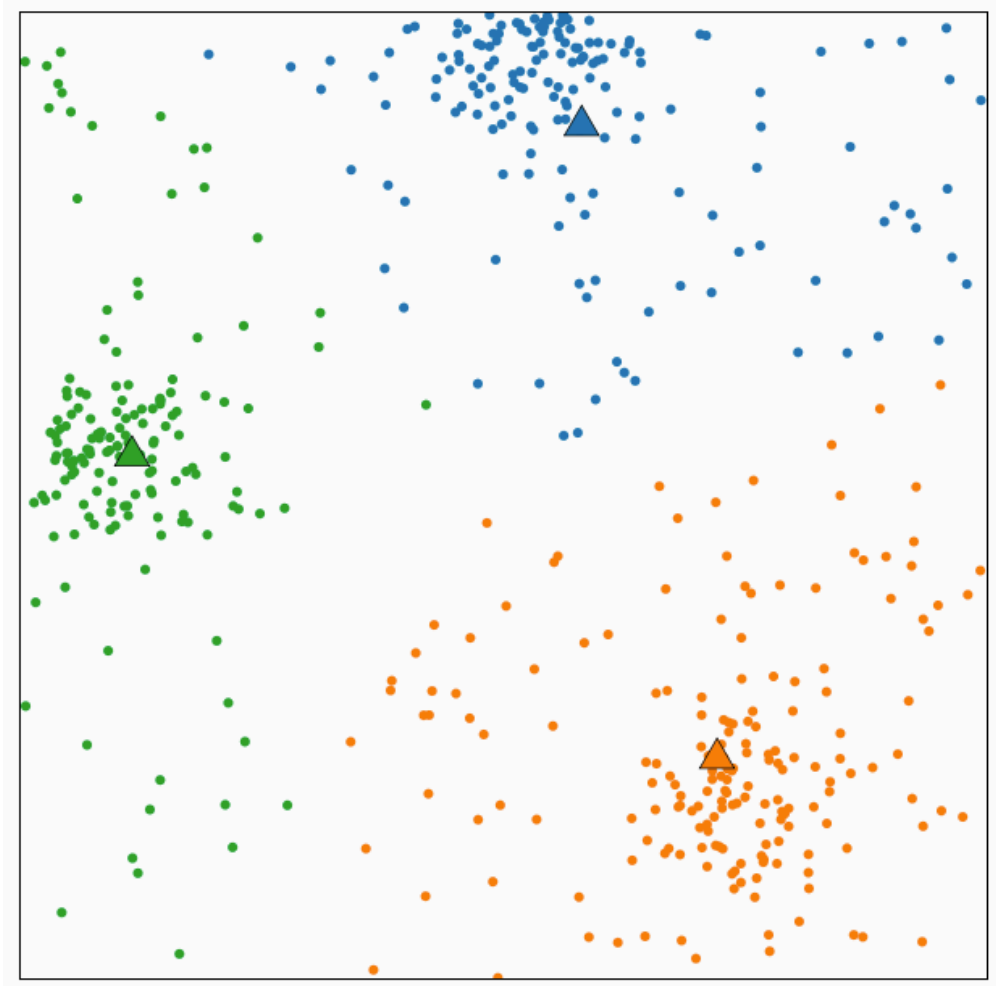
Step 4: Each center finds the centroid of the point it owns.

Step 5: Repeat

kMeans: Visualization



Iteration: 1



Iteration: 5

A Simple Algorithm: K-means clustering

Objective: Cluster *n* instances into K distinct classes.

Preliminaries:
Step 1: Pick the desired number of clusters, K.
Step 2: Assume a parametric distribution for each class (e.g. Gaussian)
Step 3: Randomly estimate the parameters of the *K* distributions.

Iterate, until convergence:
Step 4: Assign instances to the **most likely** classes based on the current parametric distributions. → **Expectation Step**
Step 5: Estimate the parametric distribution of each class based on the latest assignment. → **Maximization Step**

Properties of kMeans

Does it converge?

→ Yes, but to a local optimum. (Proof in Daume).

How long does it take to converge?

→ In practice, very quickly (usually fewer than 20 iterations).

→ In theory, O(***knm***) i.e., exponential in the number of data points.

- → k = number of centers
- → n = number of data points
- → m = dimensionality of data points

Does it converge to the right answer?

→ No guarantees, because we often don't know what the right answer is.

Properties of kMeans

Rapid convergence is hugely dependent on **initialization**.

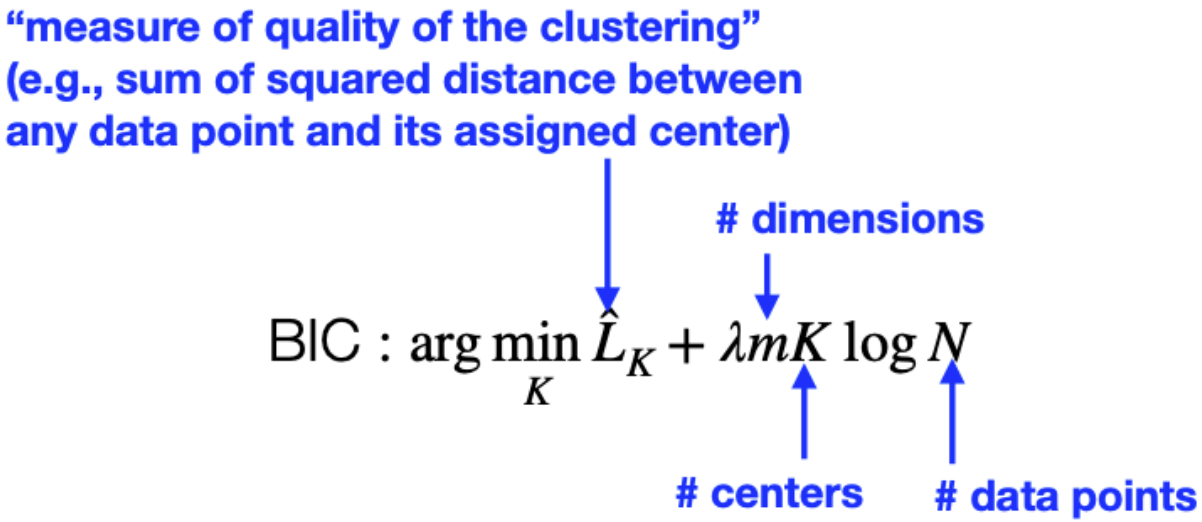
Can use random re-starts to get a better local optimum.
→ Run the algorithm 10 times with different initializations.

Alternatively, you can choose your initial centers carefully.
→ Place your first center on top of a randomly chosen point.
→ Place your second center on top of a datapoint that is furthest from the 1st.
→ Place your third center on top of a datapoint furthest from the 1st and 2nd.

kMeans: Choosing K

A common approach is to search over many solutions (i.e. with different K) and find the one that minimizes a certain criteria.

e.g. Bayes Information Criterion (BIC)



http://www.cs.cmu.edu/~./awm/tutorials/kmeans11.pdf

kMeans: Choosing K "Optimally"

within-cluster scatter

$$W(K) = \sum_{k=1}^{K} \sum_{i \in I_k} \|x_i - \bar{x}_k\|^2$$

between-cluster scatter

$$B(K) = \sum_{k=1}^{K} n_k \|\bar{x}_k - \bar{x}\|^2$$

I_k is the set of indicies of the data points belonging to cluster C_k.

Intuition: Think of the characteristics of the clusters that maximize the "separability" of your data. Ideally:

→ A cluster's points are very tightly packed together. (Within-Cluster Scatter)

→ A cluster, as an entity, is isolated from other clusters. (Between-Cluster Scatter)

Goal: Choose a K that has a small "W" and a large "B".

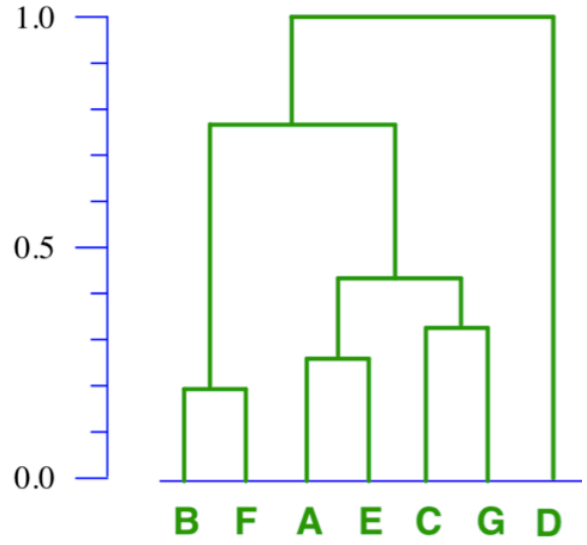
Overview

1. **kMeans**

2. **Hierarchical Clustering**

Hierarchical Clustering

A hierarchy of clusters, where the cluster at each level is created by merging clusters from the next lower level.



Two Approaches

→ Bottom-up: Recursively merge a pair of clusters.

→ Top-down: Recursively split the existing clusters.

Use Dissimilarity Measures to select split/merge pairs.

→ Measure pairwise distance between any points in the 2 clusters.

→ e.g. Euclidean distance, Manhattan distance

→ Measure distance over entire clusters using "linkage" criterion.

→ e.g. Min/Max/Mean over pairs of points.

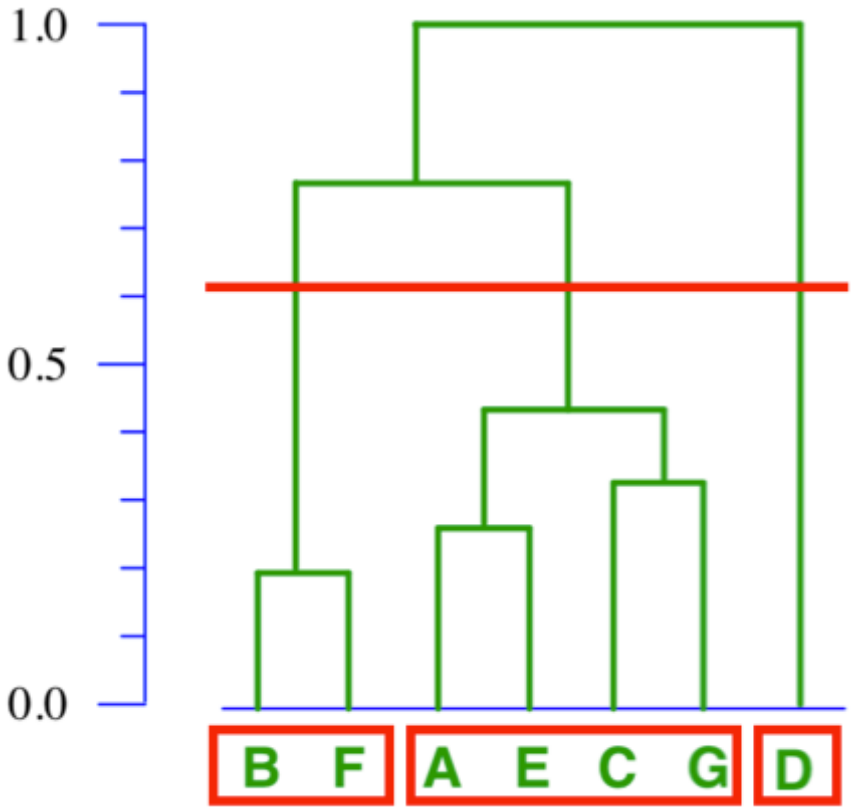
Hierarchical Clustering: Dendrograms

A Dendrogram is a tree where each node represents a group / cluster.

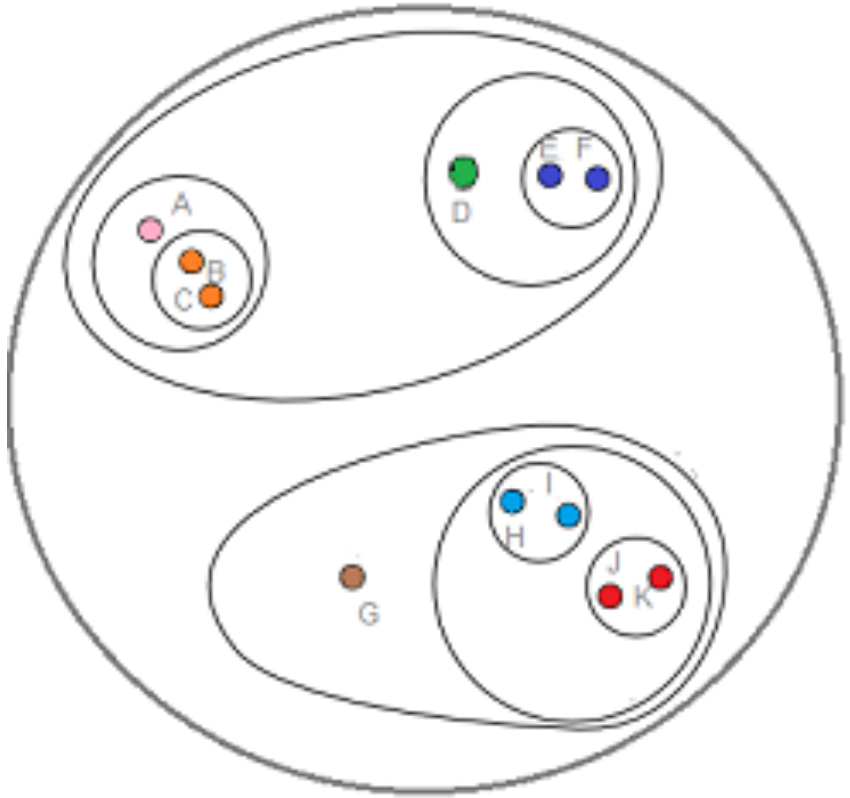
→ **Leaf:** a group with a single data point.
→ **Root:** A group containing a whole dataset.
→ **Internal node:** Has two child nodes representing the groups that were merged to form it.

Each internal node is drawn at a height proportional the dissimilarity between its two children.

→ Assume that the leaf nodes are at height zero.



Hierarchical Clustering: Dendrograms

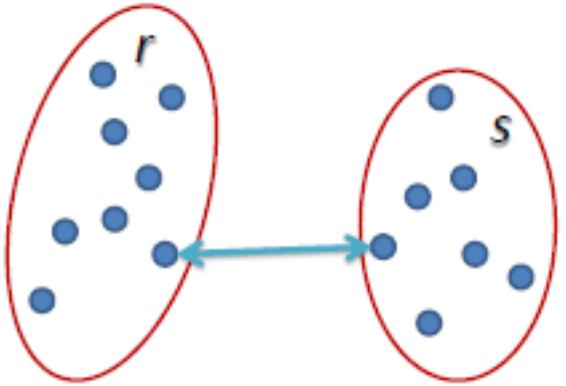


https://www.statisticshowto.com/hierarchical-clustering/

HC: Linkage Functions

Linkage Function
A function d(G,H) that takes two groups, G, H, as input and computes a dissimilarity score between them.

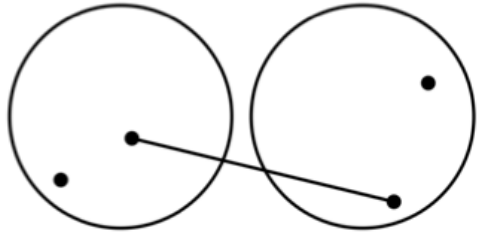
The Effects of Linkage Functions
The clustering process will result in different dendrograms depending on the choice of linkage function we use to measure dissimilarity between groups.



$$L(r,s) = \min(D(x_{ri}, x_{sj}))$$

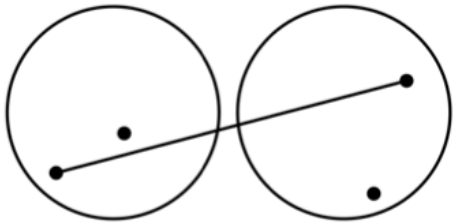
HC: Linkage Functions

Single Linkage (i.e. nearest-neighbor linkage)
The dissimilarity between G and H is the smallest dissimilarity between two points in the opposite groups.



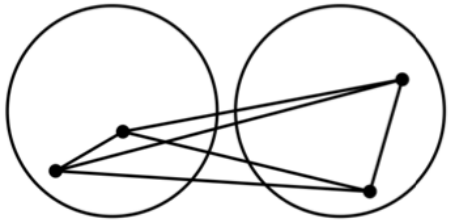
$$d_{single}(G,H) = \min_{i \in G, j \in H} d_{ij}$$

Complete Linkage (i.e. further-neighbor linkage)
The dissimilarity between G and H is the largest dissimilarity between two points in the opposite groups.



$$d_{complete}(G,H) = \max_{i \in G, j \in H} d_{ij}$$

Average Linkage
The dissimilarity between G and H is the smallest dissimilarity between two points in the opposite groups.



$$d_{average}(G,H) = \frac{1}{n_G \cdot n_H} \sum_{i \in G, j \in H} d_{ij}$$

Example: Complete Linkage

Exhibit 5.6 Presence–absence data of 10 species in 7 samples.

Samples	*Species*									
	sp1	sp2	sp3	sp4	sp5	sp6	sp7	sp8	sp9	sp10
A	1	1	1	0	1	0	0	1	1	1
B	1	1	0	1	1	0	0	0	0	1
C	0	1	1	0	1	0	0	1	0	0
D	0	0	0	1	0	1	0	0	0	0
E	1	1	1	0	1	0	1	1	1	0
F	0	1	0	1	1	0	0	0	0	1
G	0	1	1	0	1	1	0	1	1	0

Raw Dataset

http://www.econ.upf.edu/~michael/stanford/maeb7.pdf

Jaccard Similarity

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

$$J(A,B) = \frac{Num\ of\ A{=}1\ AND\ B{=}1}{(Num\ of\ A{=}1) + (Num\ of\ B{=}1) - Num\ of\ A{=}1\ AND\ B{=}1}$$

$$J(A,B) = \frac{4}{(7)+(5)-4} = \frac{4}{8} = 0.5$$

Jaccard Index

d(A,B) = **1 – J(A,B)** = 0.5

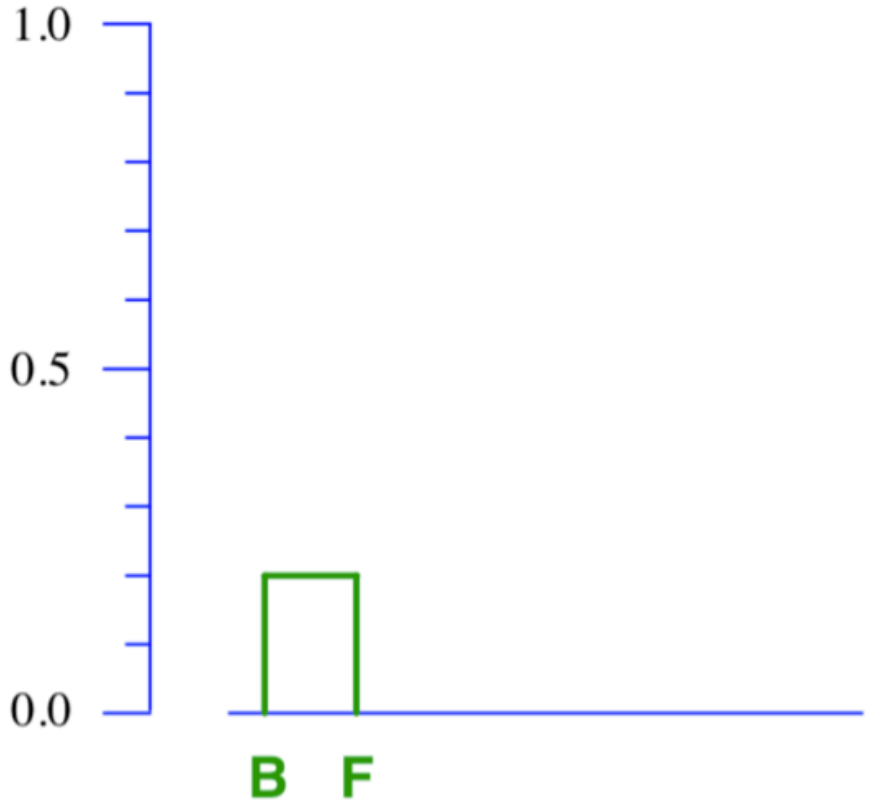
Example: Complete Linkage

samples	A	B	C	D	E	F	G
A	0	0.5000	0.4286	1.0000	0.2500	0.6250	0.3750
B	0.5000	0	0.7143	0.8333	0.6667	0.2000	0.7778
C	0.4286	0.7143	0	1.0000	0.4286	0.6667	0.3333
D	1.0000	0.8333	1.0000	0	1.0000	0.8000	0.8571
E	0.2500	0.6667	0.4286	1.0000	0	0.7778	0.3750
F	0.6250	0.2000	0.6667	0.8000	0.7778	0	0.7500
G	0.3750	0.7778	0.3333	0.8571	0.3750	0.7500	0

Step 1: Look for the most similar pairs (lowest similarity scores)

http://www.econ.upf.edu/~michael/stanford/maeb7.pdf

Example: Complete Linkage



Step 2: Join B and F at level 0.20. This forms a node.

Example: Complete Linkage

samples	A	B	C	D	E	F	G
A	0	0.5000	0.4286	1.0000	0.2500	0.6250	0.3750
B	0.5000	0	0.7143	0.8333	0.6667	0.2000	0.7778
C	0.4286	0.7143	0	1.0000	0.4286	0.6667	0.3333
D	1.0000	0.8333	1.0000	0	1.0000	0.8000	0.8571
E	0.2500	0.6667	0.4286	1.0000	0	0.7778	0.3750
F	0.6250	0.2000	0.6667	0.8000	0.7778	0	0.7500
G	0.3750	0.7778	0.3333	0.8571	0.3750	0.7500	0

Step 3: Calculate the similarity score between each data point x and the merged pair (B, F)

→ Complete linkage means dissimilarity = max of **d(x, B)** and **d(x, F)**

→ E.g. **d(A,B)** = 0.5, **d(A,F)** = 0.6250), therefore **d(A, (B,F))** = 0.6250.

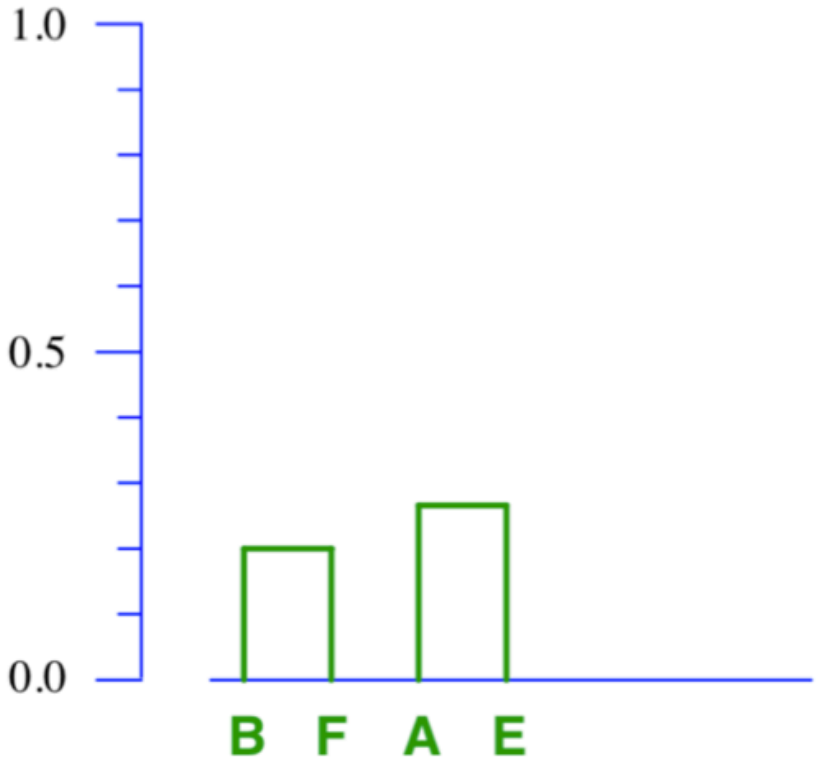
Example: Complete Linkage

samples	A	(B,F)	C	D	E	G
A	0	0.6250	0.4286	1.0000	0.2500	0.3750
(B,F)	0.6250	0	0.7143	0.8333	0.7778	0.7778
C	0.4286	0.7143	0	1.0000	0.4286	0.3333
D	1.0000	0.8333	1.0000	0	1.0000	0.8571
E	0.2500	0.7778	0.4286	1.0000	0	0.3750
G	0.3750	0.7778	0.3333	0.8571	0.3750	0

This is what the table looks like after re-calculating the similarity scores between each point and the merged (B, F) pair.

Step 4: Repeat the process. Find the smallest similarity score.
→ d(A,E) is lowest at 0.25

Example: Complete Linkage



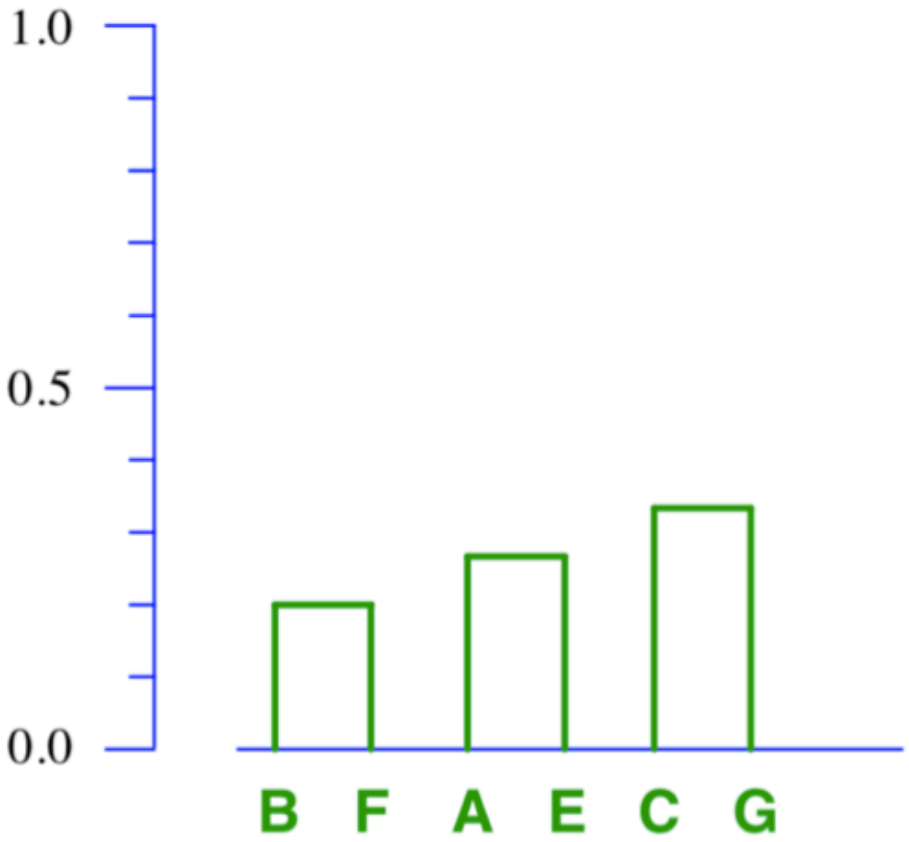
Step 5: Join A and E at level 0.25. This forms a node.

Example: Complete Linkage

samples	(A,E)	(B,F)	C	D	G
(A,E)	0	0.7778	0.4286	1.0000	0.3750
(B,F)	0.7778	0	0.7143	0.8333	0.7778
C	0.4286	0.7143	0	1.0000	0.3333
D	1.0000	0.8333	1.0000	0	0.8571
G	0.3750	0.7778	0.3333	0.8571	0

Step 6: Recompute the table to reflect post-calculations for similarity scores between each point and the merged (A,E) pair.
→ (C, G) is lowest at 0.3333.

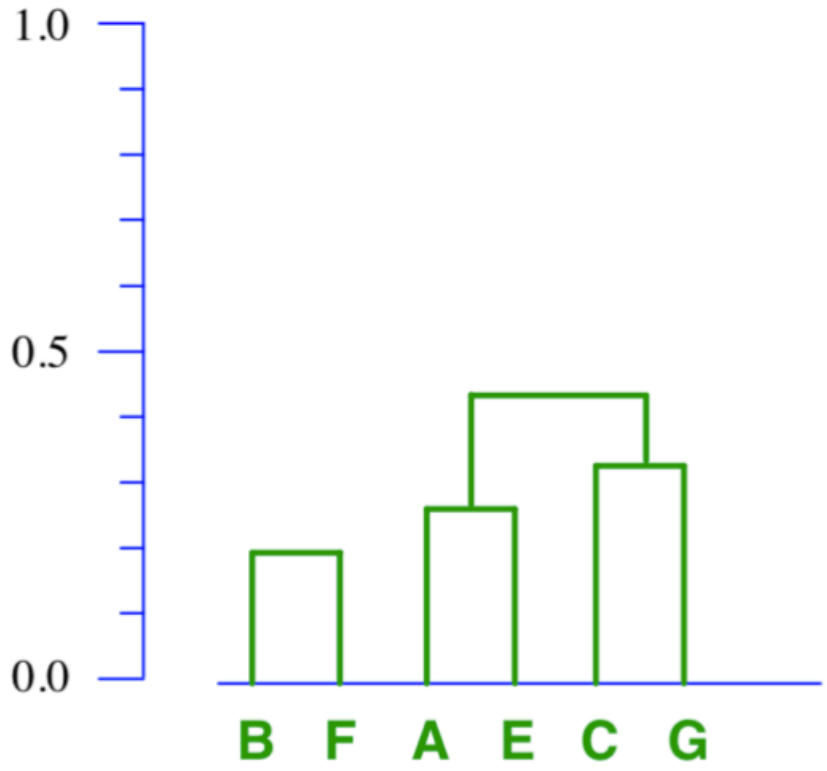
Example: Complete Linkage



Example: Complete Linkage

samples	(A,E)	(B,F)	(C,G)	D
(A,E)	0	0.7778	0.4286	1.0000
(B,F)	0.7778	0	0.7778	0.8333
(C,G)	0.4286	0.7778	0	1.0000
D	1.0000	0.8333	1.0000	0

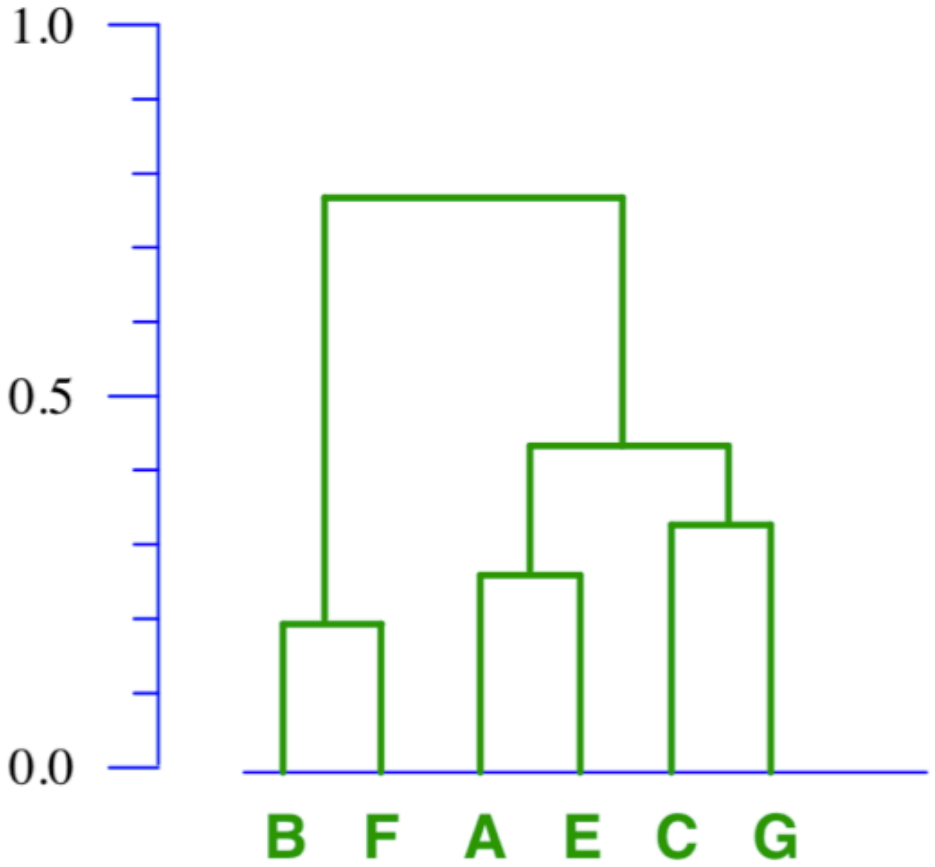
Example: Complete Linkage



Example: Complete Linkage

samples	(A,E,C,G)	(B,F)	D
(A,E,C,G)	0	0.7778	1.0000
(B,F)	0.7778	0	0.8333
D	1.0000	0.8333	0

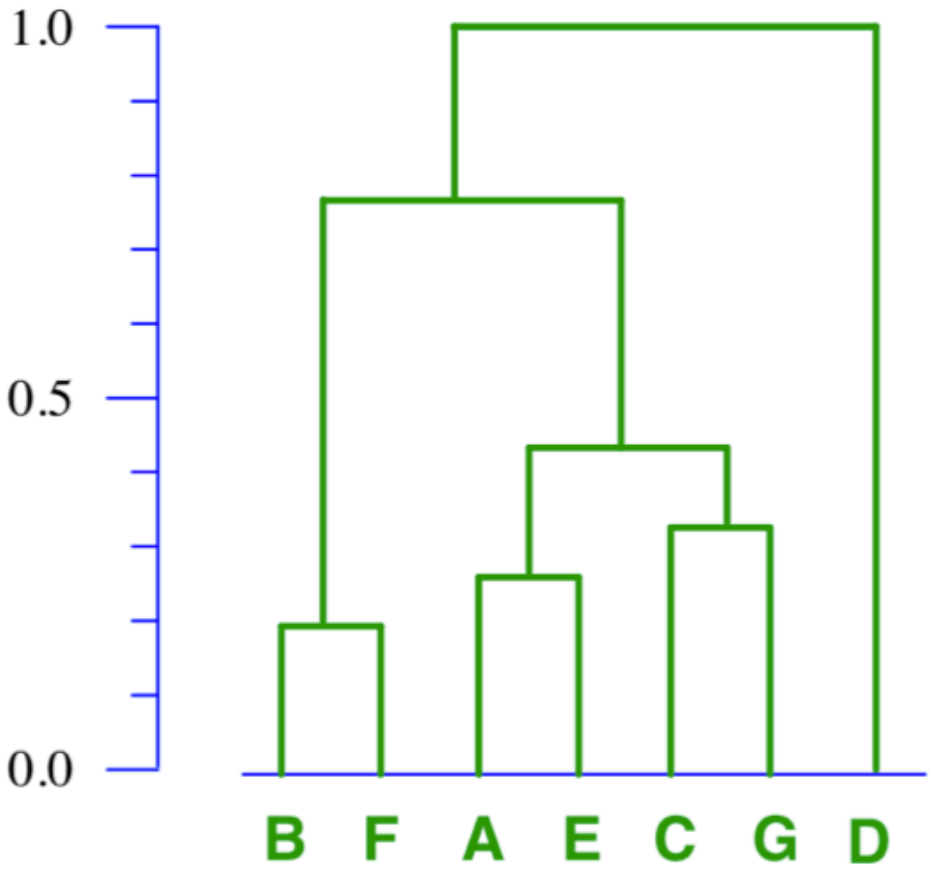
Example: Complete Linkage



Example: Complete Linkage

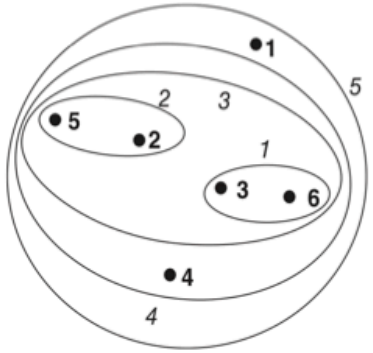
samples	(A,E,C,G,B,F)	D
(A,E,C,G,B,F)	0	1.0000
D	1.0000	0

Example: Complete Linkage

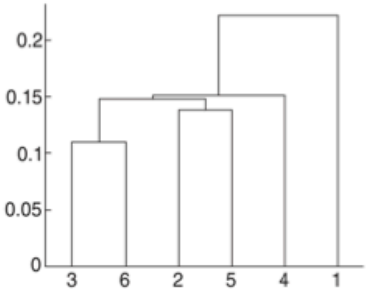


Single vs. Complete Linkage

Single Linkage
→ Sensitive to noise / outliers.
→ Clusters tend to be elliptical, long, skinny.

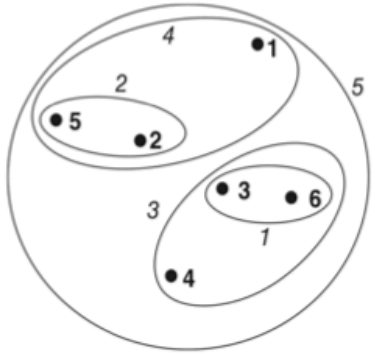


(a) Single link clustering.

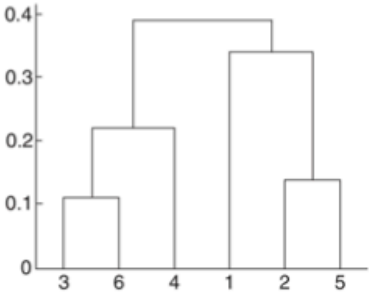


(b) Single link dendrogram.

Complete Linkage
→ Less sensitive to noise / outliers.
→ Clusters tend to be tight, compact, globular.



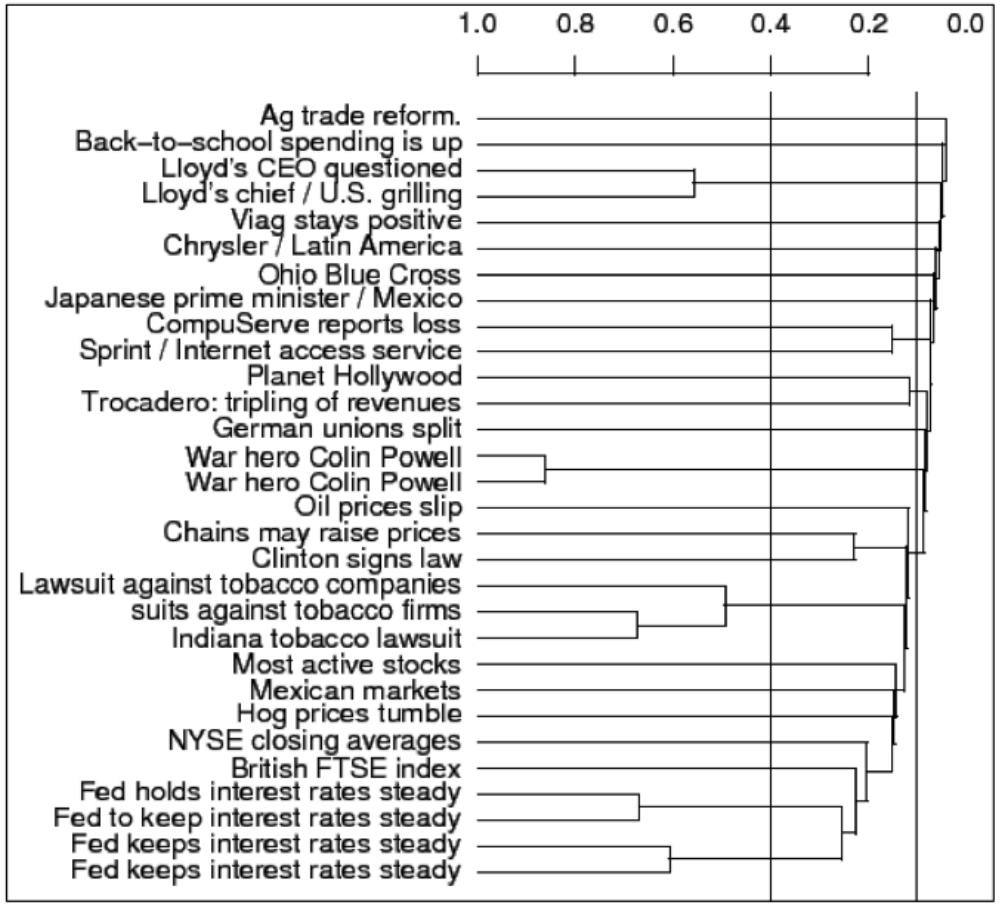
(a) Complete link clustering.



(b) Complete link dendrogram.

https://www-users.cs.umn.edu/~kumar001/dmbook/ch7_clustering.pdf

Hierarchical Clustering of News Articles



http://nlp.stanford.edu/IR-book/html/htmledition/hierarchical-agglomerative-clustering-1.html

Next Time

We will address:

1. Dimensionality Reduction