Unsupervised learning

Clustering and Dimensionality Reduction

Ramon Ferrer-i-Cancho
rferrericancho@upc.edu
(Marta Arias)

Dept. CS, UPC

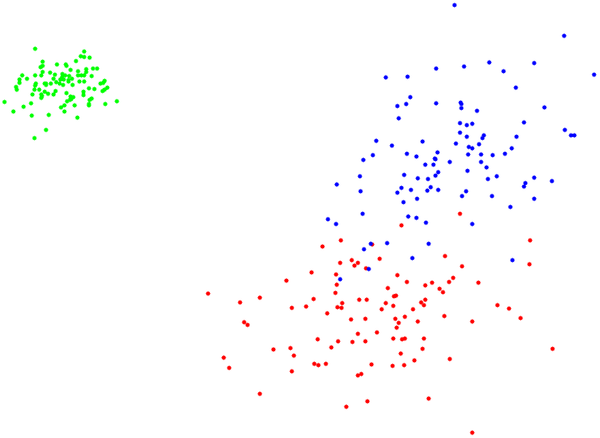
Fall 2024

Clustering

Partition input examples into *similar* subsets

Clustering

Partition input examples into *similar* subsets



Clustering

Main challenges

- How to measure similarity?
- How many clusters?
- How do we evaluate the clusters?

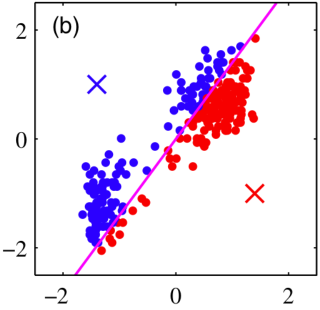
Algorithms we will cover

- K-means
- Hierarchical clustering

K-means clustering

Intuition

- Input data are:
  - m examples $x^1, .., x^m$, and
  - K, the number of desired clusters
- Clusters represented by cluster centers $\mu_1, .., \mu_K$
- Given centers $\mu_1, .., \mu_K$, each center defines a cluster: the subset of inputs x^i that are closer to it than to other centers



K-means clustering

Intuition

The aim is to find

- cluster centers $\mu_1, .., \mu_K$ and
- a cluster assignment $\mathbf{z} = (z^1, .., z^m)$ where $z^i \in \{1, .., K\}$
  - z^i is the cluster assigned to example x^i

such that $\mu_1, .., \mu_K, \mathbf{z}$ *minimize* the cost function

$$J(\mu_1, .., \mu_K, \mathbf{z}) = \sum_i \|\mathrm{x}^i - \mu_{z^i}\|^2.$$

K-means clustering

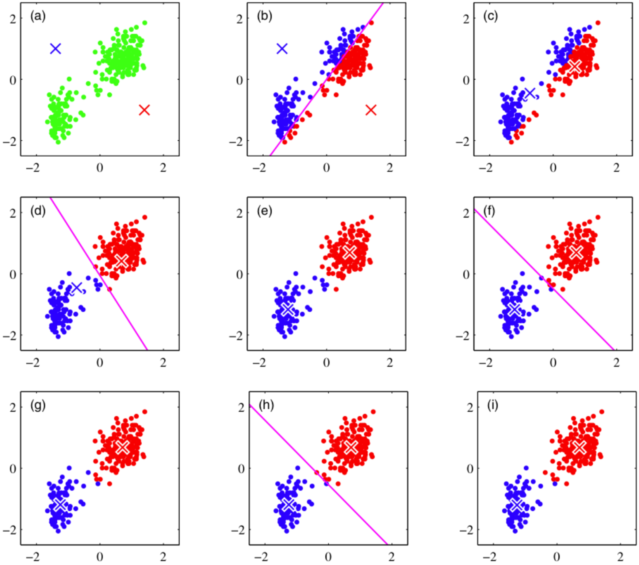
Cost function

$$J(\mu_1, .., \mu_K, \mathbf{z}) = \sum_i \|\mathbf{x}^i - \mu_{z^i}\|^2$$

Pseudocode

- Pick initial centers $\mu_1, .., \mu_K$ at random
- Repeat until convergence
  - Optimize $\mathbf{z}$ in $J(\mu_1, .., \mu_K, \mathbf{z})$ keeping $\mu_1, .., \mu_K$ fixed
    - Set z^i to closest center: $z^i = \arg\min_k \|\mathbf{x}^i - \mu_k\|^2$
  - Optimize $\mu_1, .., \mu_K$ in $J(\mu_1, .., \mu_K, \mathbf{z})$ keeping $\mathbf{z}$ fixed
    - For each $k = 1, .., K$, set $\mu_k = \frac{1}{|\{i|z^i = k\}|} \sum_{i:z^i=k} \mathbf{x}^i$

K-Means illustrated



Limitations of k-Means

K-Means works well if..

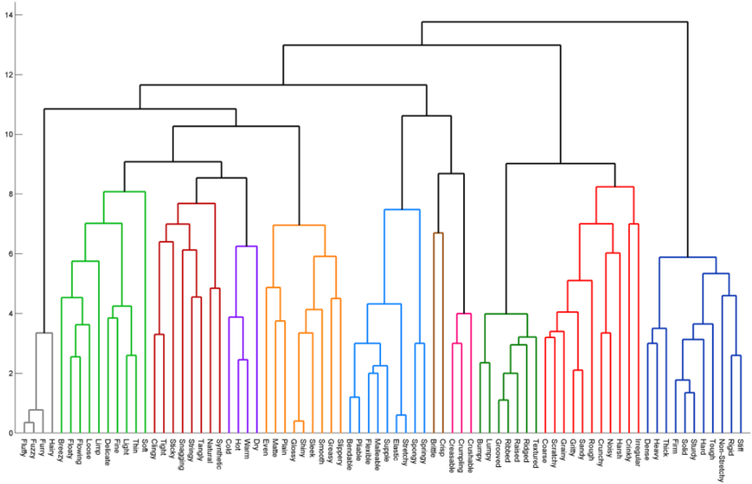
- Clusters are spherical
- Clusters are well separated
- Clusters are of similar volumes
- Clusters have similar number of points

.. so improve it with more general model

- Mixture of Gaussians:
- Learn it using *Expectation Maximization*

Hierarchical clustering

Output is a *dendogram*



Agglomerative hierarchical clustering

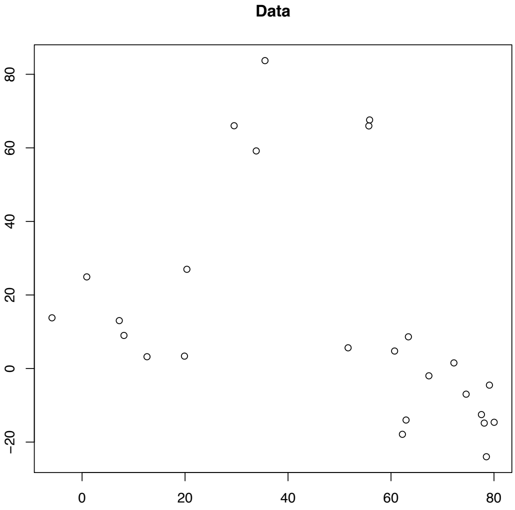
Bottom-up

Pseudocode

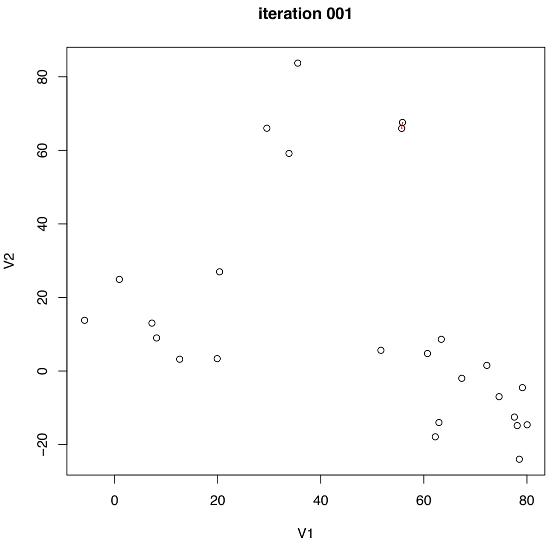
1. Start with one cluster per example
2. Repeat until all examples in one cluster
   - merge two closest clusters

(Next example from D. Blei's course at Princeton)

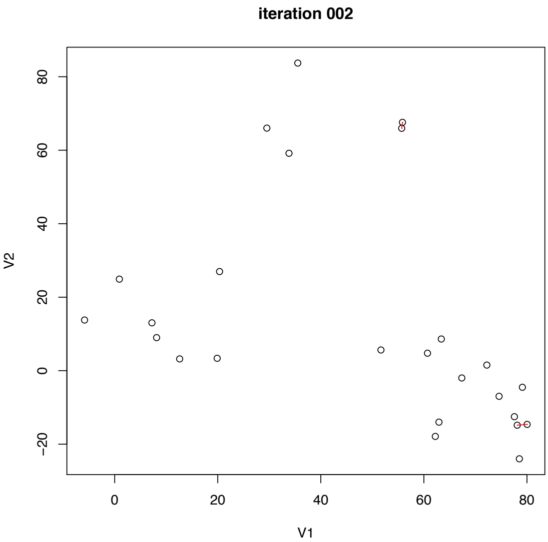
Example



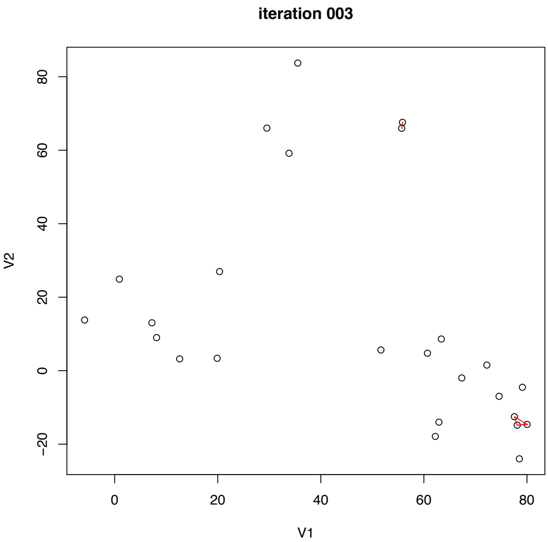
Example



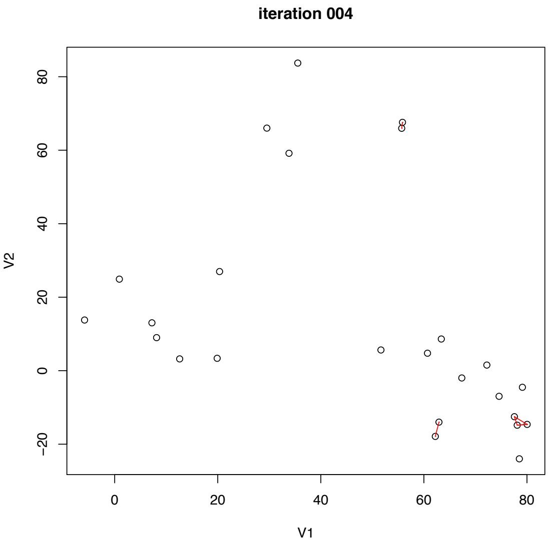
Example



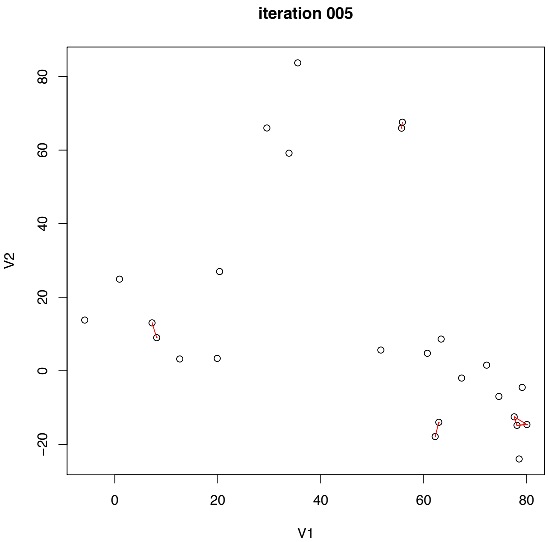
Example



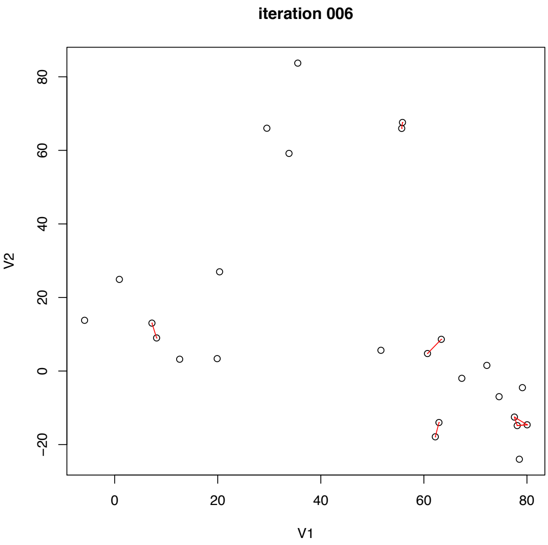
Example



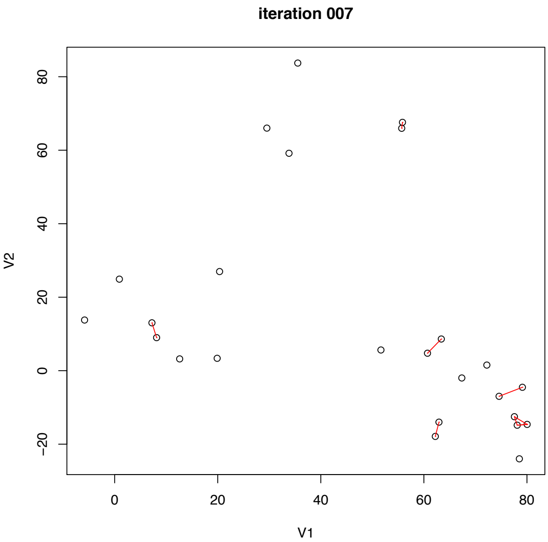
Example



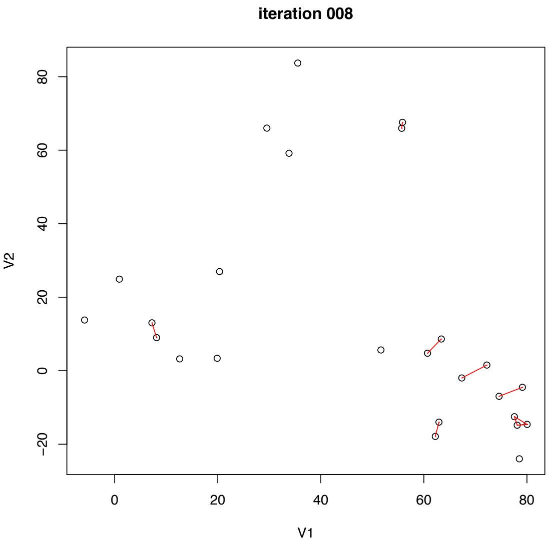
Example



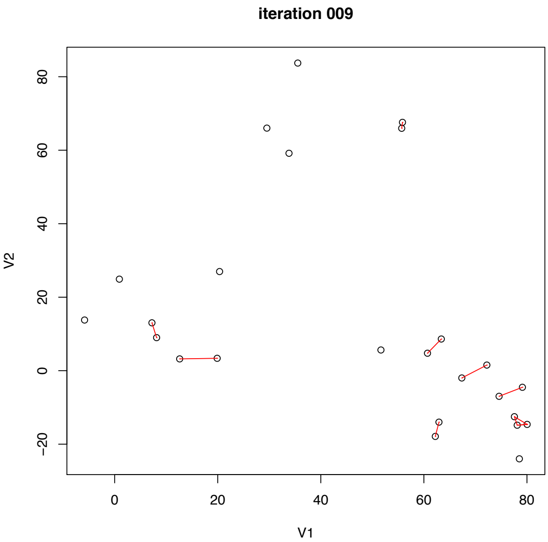
Example



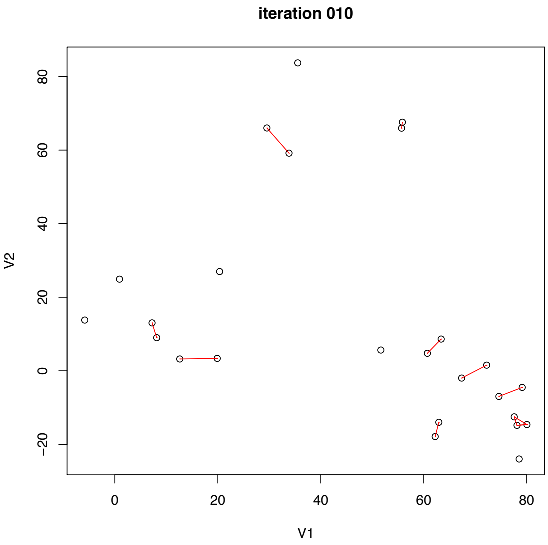
Example



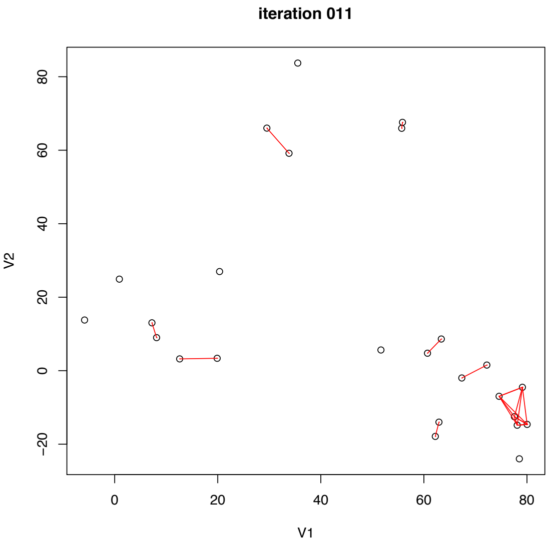
Example



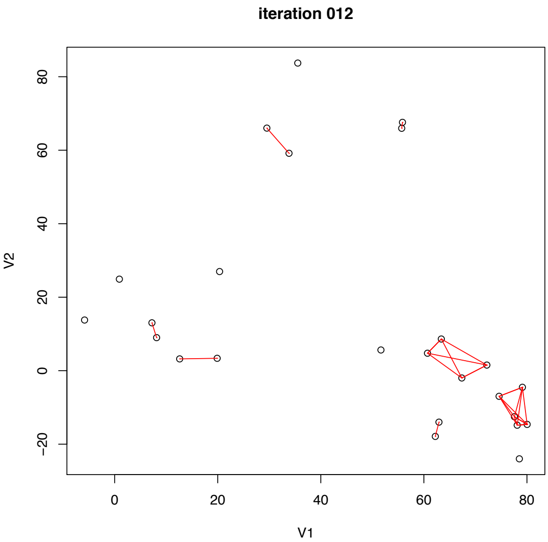
Example



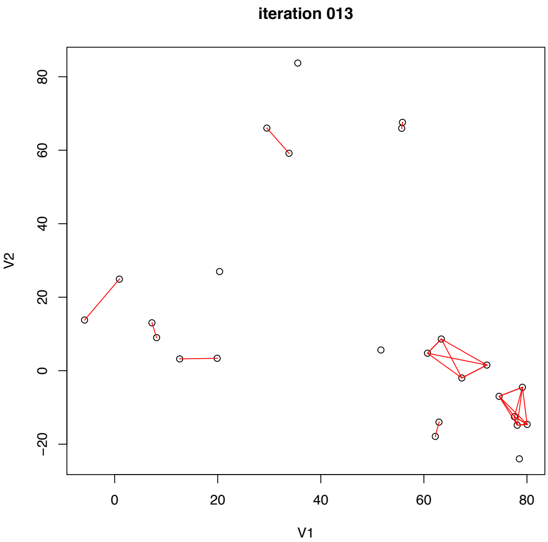
Example



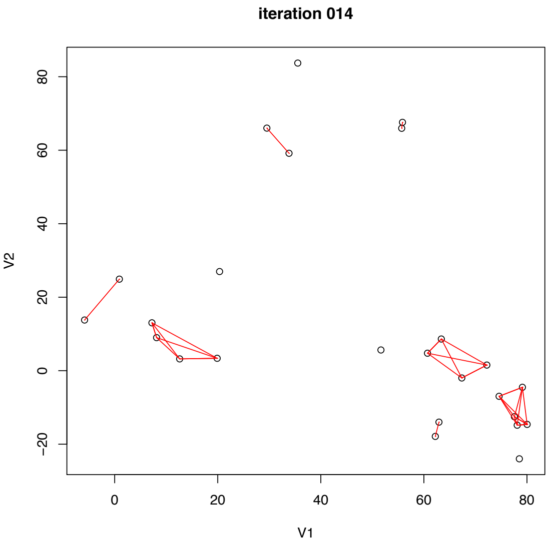
Example



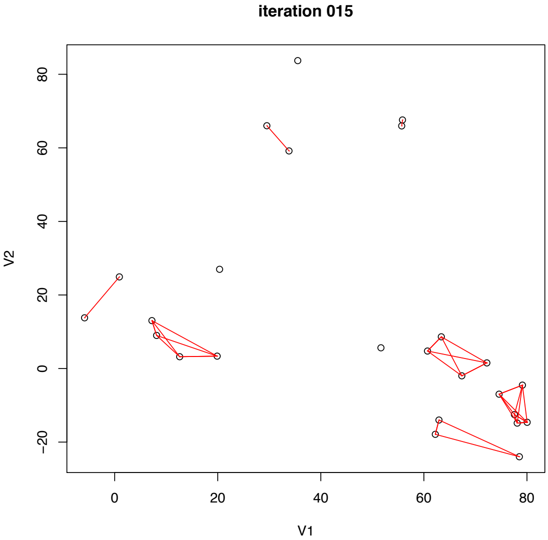
Example



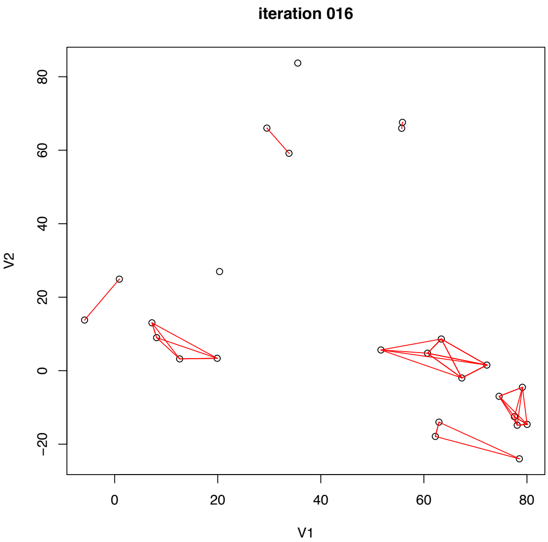
Example



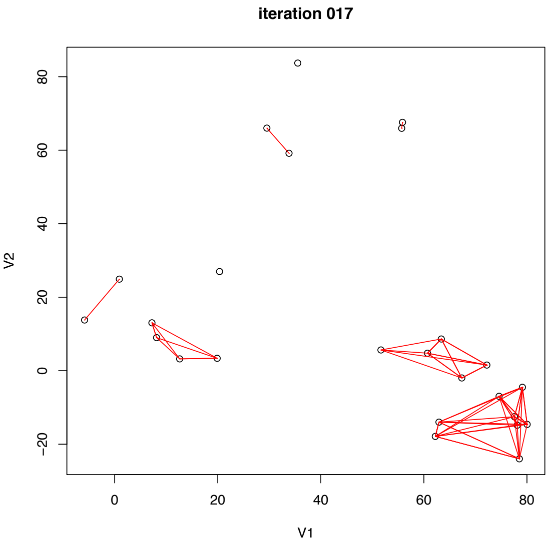
Example



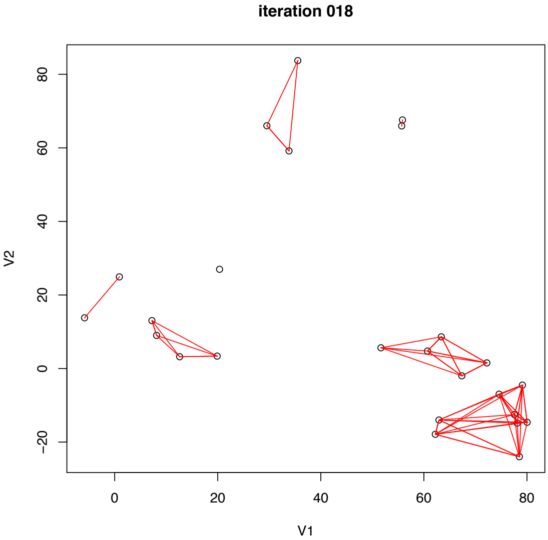
Example



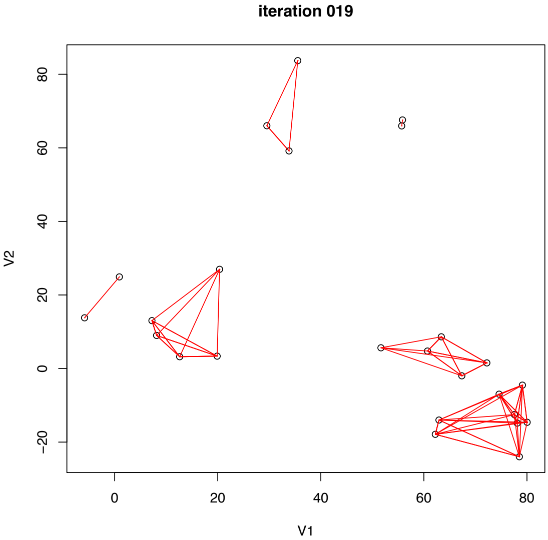
Example



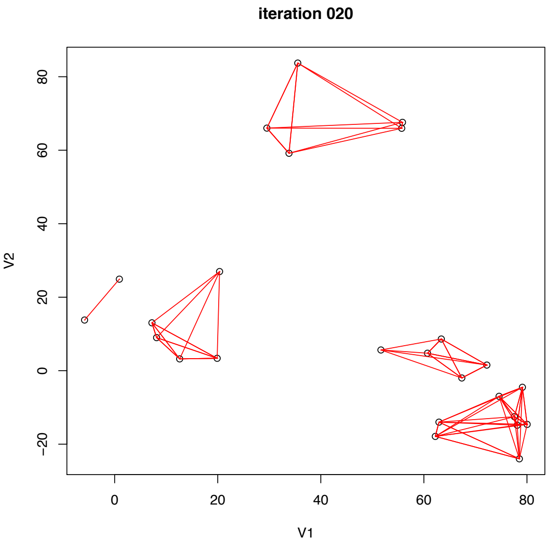
Example



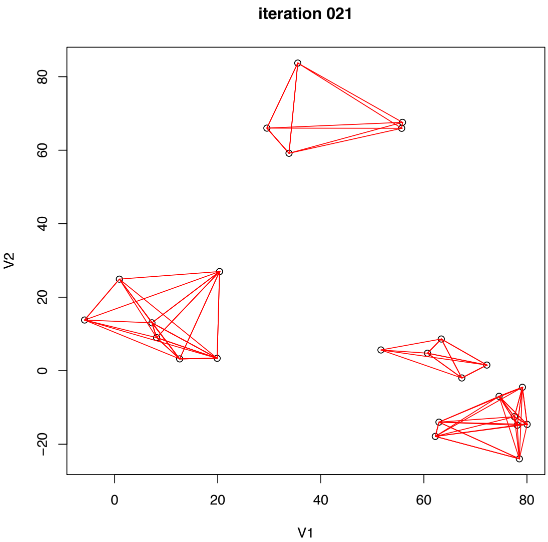
Example



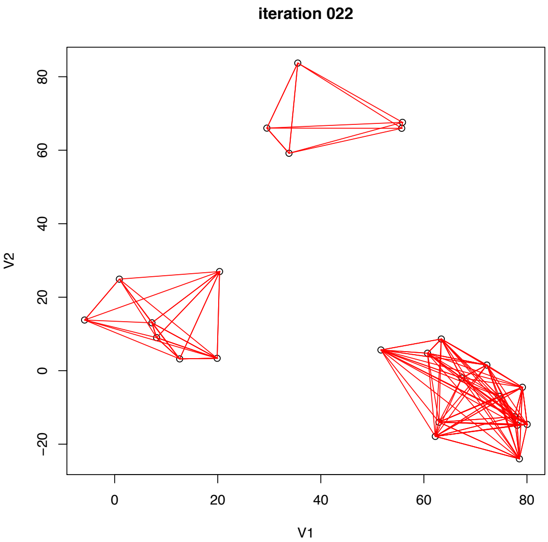
Example



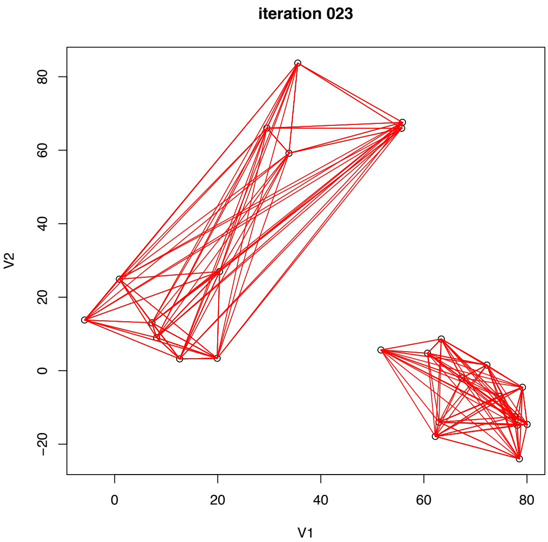
Example



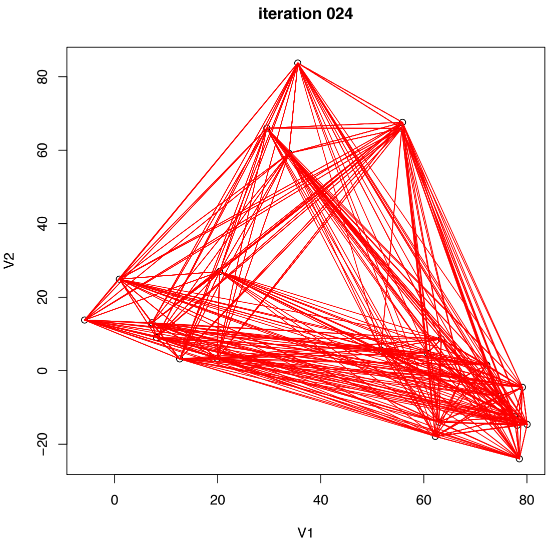
Example



Example



Example



Agglomerative hierarchical clustering

Bottom-up

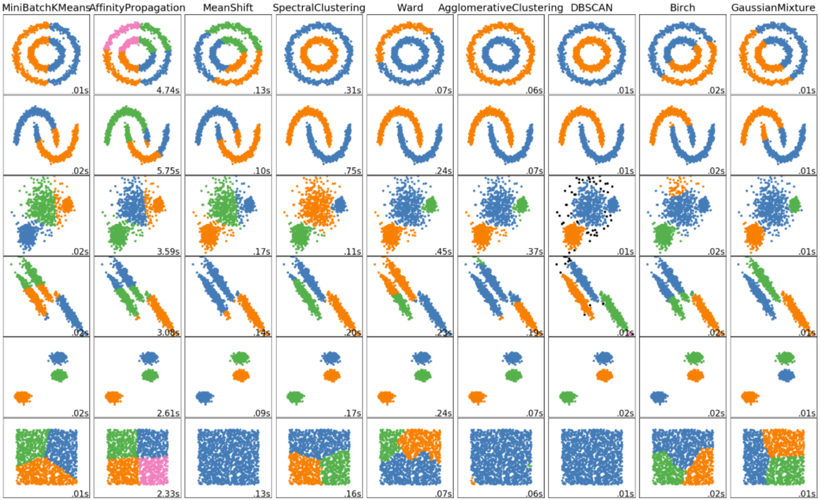
Pseudocode

1. Start with one cluster per example
2. Repeat until all examples in one cluster
   - merge two closest clusters

Defining distance between clusters (i.e. *sets* of points)

- Single Linkage: $d(X, Y) = \min_{x \in X, y \in Y} d(x, y)$
- Complete Linkage: $d(X, Y) = \max_{x \in X, y \in Y} d(x, y)$
- Group Average: $d(X, Y) = \dfrac{\sum_{x \in X, y \in Y} d(x, y)}{|X| \times |Y|}$
- Centroid Distance: $d(X, Y) = d\left(\dfrac{1}{|X|} \sum_{x \in X} x, \dfrac{1}{|Y|} \sum_{y \in Y} y\right)$

Many, many, many other algorithms available ..



Clustering with *scikit-learn* I

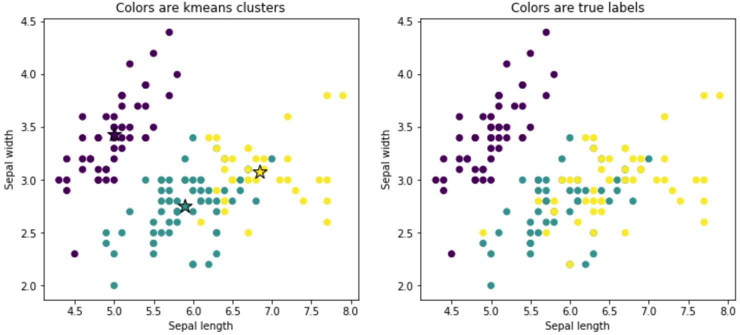
K-means: an example with the Iris dataset

```
In [33]: %matplotlib inline
         import numpy as np
         import matplotlib.pyplot as plt

         from sklearn.cluster import KMeans
         from sklearn.datasets import make_blobs, load_iris

         iris = load_iris()
         kmeans = KMeans(n_clusters = 3)
         kmeans.fit(iris.data)

         kmeans.cluster_centers_

Out[33]: array([[5.006     , 3.428     , 1.462     , 0.246     ],
                [5.9016129 , 2.7483871 , 4.39354839, 1.43387097],
                [6.85      , 3.07368421, 5.74210526, 2.07105263]])

In [34]: kmeans.labels_

Out[34]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 2, 2, 2, 1, 2, 2, 2,
                2, 2, 2, 1, 1, 2, 2, 2, 2, 1, 2, 1, 2, 1, 2, 2, 1, 1, 2, 2, 2, 2,
                2, 1, 2, 2, 2, 2, 1, 2, 2, 2, 1, 2, 2, 2, 1, 2, 2, 1], dtype=int32)

In [35]: import pandas as pd
         pd.crosstab(iris.target, kmeans.labels_)

Out[35]:
```

col_0	0	1	2
row_0			
0	50	0	0
1	0	48	2
2	0	14	36

Clustering with *scikit-learn* II

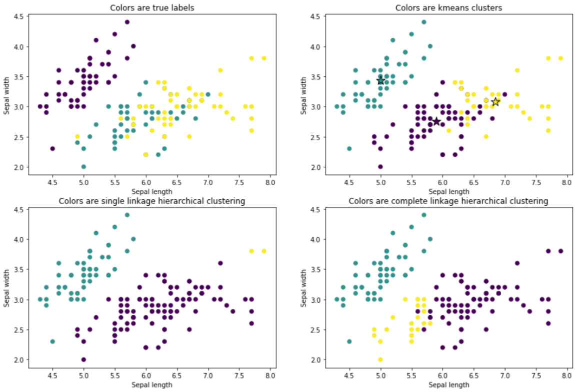
K-means: an example with the Iris dataset



Clustering with *scikit-learn* I

Hierarchical clustering: an example with the Iris dataset

```
from sklearn.cluster import AgglomerativeClustering
clustering1 = AgglomerativeClustering(linkage='single', n_clusters=3)
clustering1.fit(iris.data)
clustering2 = AgglomerativeClustering(linkage='complete', n_clusters=3)
clustering2.fit(iris.data)
```



Dimensionality reduction I

The curse of dimensionality

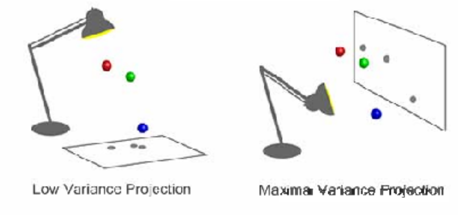
- When dimensionality increases, data becomes increasingly sparse in the space that it occupies
- Definitions of density and distance between points (critical for many tasks!) become less meaningful
- Visualization and qualitative analysis becomes impossible

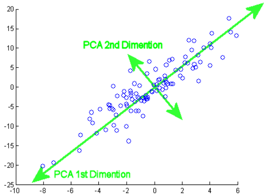
And so dimensionality reduction methods..

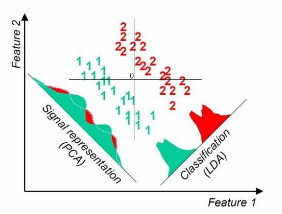
- avoid or at least mitigate the curse of dimensionality
- reduce time and memory required
- allow data to be more easily visualized
- may help eliminate irrelevant features
- may reduce noise

Principal Components Analysis

Find linear projections of original coordinates that maximize variance

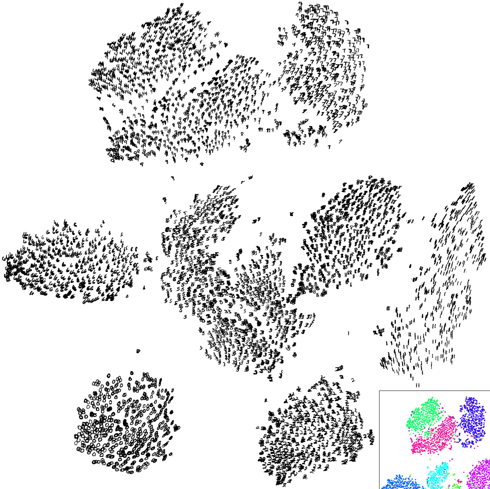


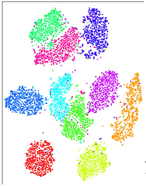




t-SNE: t-distributed stochastic neighbor embedding

A non-linear distance-preserving method



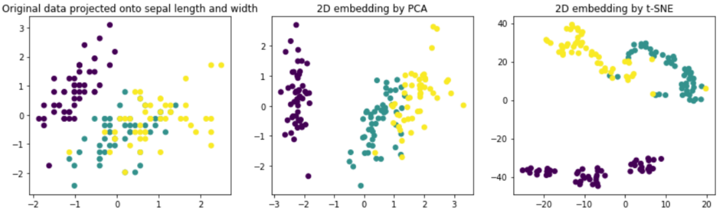

[1]

[1] From https://lvdmaaten.github.io/tsne/

Dimensionality reduction with *scikit-learn*

```
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
from sklearn.preprocessing import StandardScaler

X_normalized = StandardScaler().fit_transform(iris.data)
X_pca = PCA(n_components=2).fit_transform(X_normalized)
X_tsne = TSNE(n_components=2, perplexity=10).fit_transform(X_normalized)

print(X_pca.shape, X_tsne.shape)
```

```
(150, 2) (150, 2)
```



So, we are done for this course

Lots of important things we have left out!

- Online and incremental learning; data mining for streams
- Important models: Support Vector Machines, Neural Nets (and Deep learning)
- Kernel methods and learning from structured objects
- Ensemble methods: random forests, boosting, bagging, etc.
- Spatial and temporal learning
- Feature selection methods
- many many more...

To finish

Reading assignments: check
https://www.cs.upc.edu/~csi/

Exam: check https://www.cs.upc.edu/~csi/ and
corresponding notice at Racó.