

Unsupervised Learning: Clustering

Some material adapted from slides by Andrew Moore, CMU. See <http://www.autonlab.org/tutorials/> for a repository of Data Mining tutorials

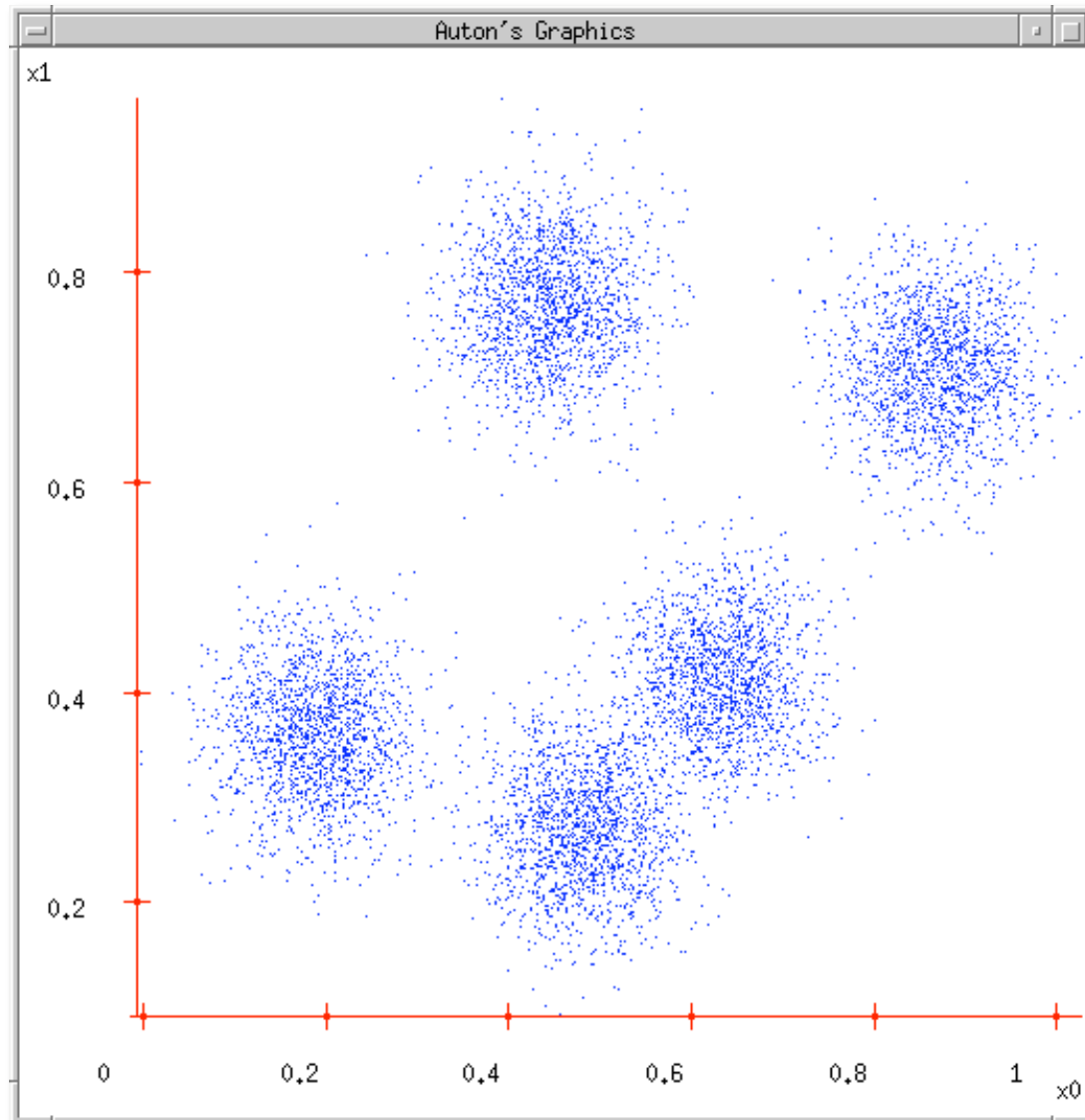
Unsupervised Learning

- Supervised learning used labeled data pairs (x, y) to learn a function $f : X \rightarrow Y$.
- But, what if we don't have labels?
- No labels = **unsupervised learning**
- Only some points are labeled = **semi-supervised learning**
 - Getting labels may be expensive, so we only get a few
- **Clustering** is the unsupervised grouping of data points. It can be used for **knowledge discovery**

Clustering algorithms

- Many clustering algorithms
- Clustering typically done using a distance measure defined between instances
- Distance defined in the instance feature space
- Agglomerative approach works bottom up:
 - Treat each instance as a cluster
 - Merge two closest clusters
 - Repeat until stop condition is met
- Top-down approach starts cluster with all instances
 - Find a cluster to split into two or more smaller clusters
 - Repeat until stop condition met

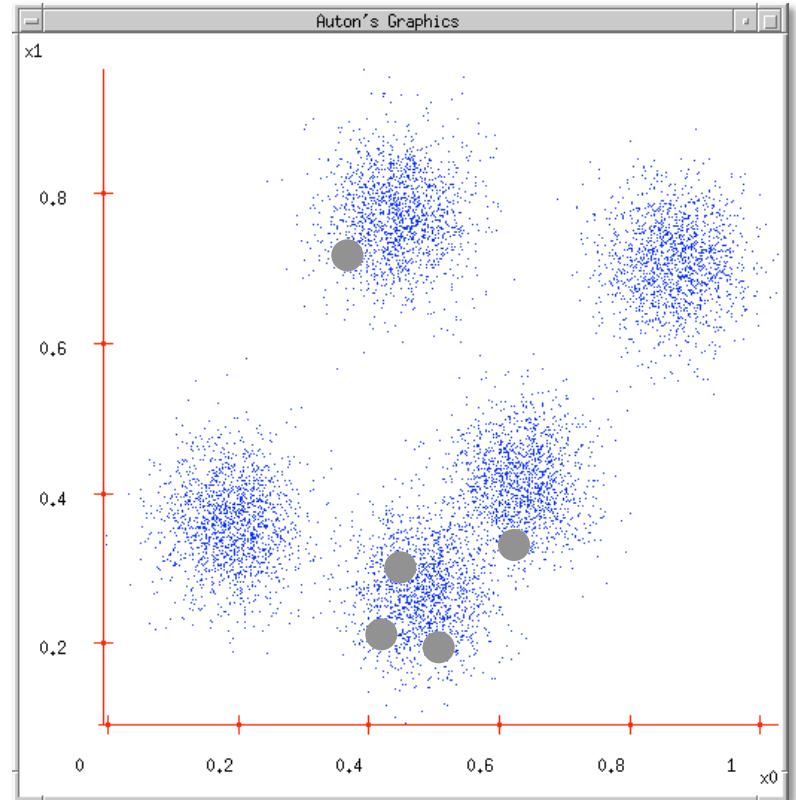
Clustering Data



K-Means Clustering

K-Means (k , data)

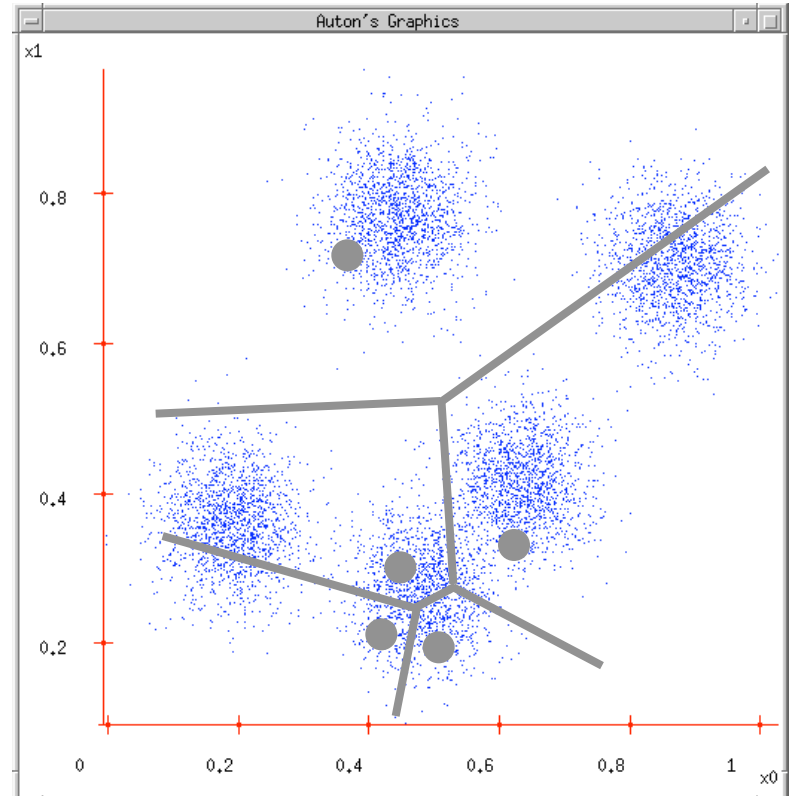
- Randomly choose k cluster center locations (centroids)
- Loop until convergence
 - Assign each point to the cluster of the closest centroid
 - Re-estimate the cluster centroids based on the data assigned to each
- Convergence: no point is assigned to a different cluster



K-Means Clustering

K-Means (k , data)

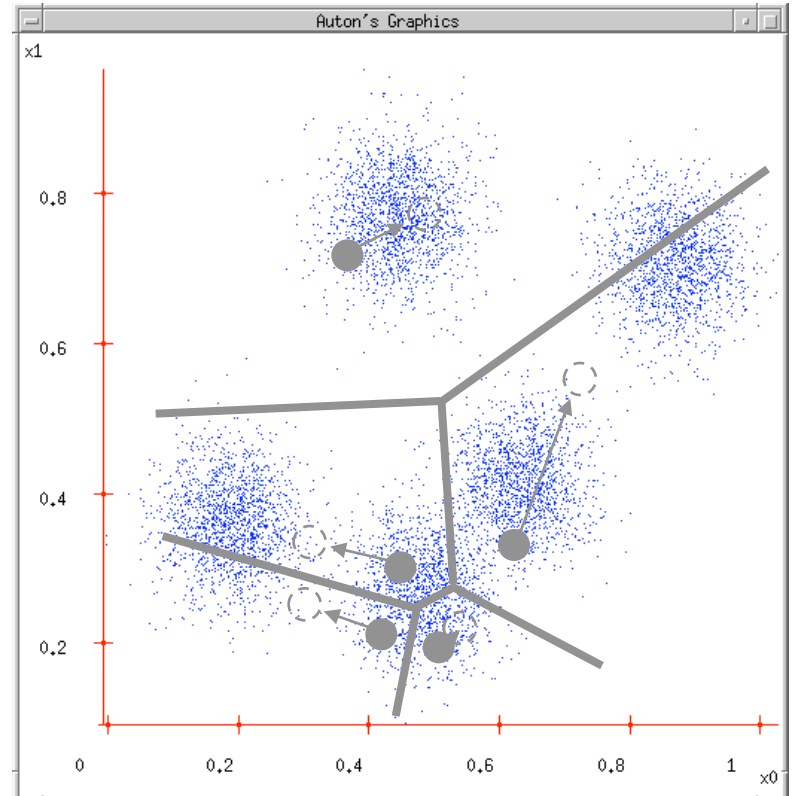
- Randomly choose k cluster center locations (centroids)
- Loop until convergence
 - Assign each point to the cluster of the closest centroid.
 - Re-estimate the cluster centroids based on the data assigned to each
- Convergence: no point is assigned to a different cluster



K-Means Clustering

K-Means (k , data)

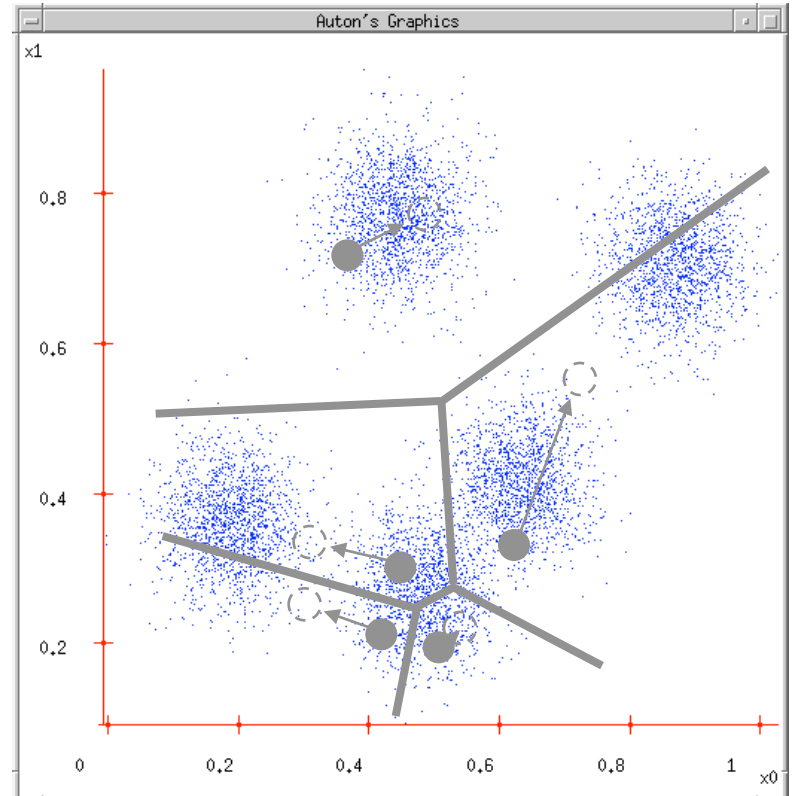
- Randomly choose k cluster center locations (centroids)
- Loop until convergence
 - Assign each point to the cluster of the closest centroid
 - Re-estimate the cluster centroids based on the data assigned to each
- Convergence: no point is assigned to a different cluster



K-Means Clustering

K-Means (k , data)

- Randomly choose k cluster center locations (centroids)
- Loop until convergence
 - Assign each point to the cluster of the closest centroid
 - Re-estimate the cluster centroids based on the data assigned to each
- Convergence: no point is assigned to a different cluster



Preprocess Classify **Cluster** Associate Select attributes Visualize

Clusterer

Choose **SimpleKMeans** -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 3 -A "wek

Cluster mode

- Use training set
- Supplied test set
- Percentage split %
- Classes to clusters evaluation
-
- Store clusters for visualization

Ignore attributes

Start

Stop

Result list (right-click for options)

13:11:45 - EM

13:12:30 - SimpleKMeans

14:09:05 - SimpleKMeans

14:09:57 - SimpleKMeans

14:10:01 - SimpleKMeans

14:10:03 - SimpleKMeans

14:10:40 - SimpleKMeans

Clusterer output

=== Run information ===

```
Scheme:      weka.clusterers.SimpleKMeans -init 0 -max-candidate
Relation:    iris
Instances:   150
Attributes:  5
              sepallength
              sepalwidth
              petallength
              petalwidth
```

Ignored:

class

Test mode: evaluate on training data

=== Clustering model (full training set) ===

kMeans

=====

Number of iterations: 6

Within cluster sum of squared errors: 6.9981140048267605

Status

OK

Log



x 0

Using scikit-learn for K-means clustering on Fisher's Iris dataset with a 3D graph

```
In [38]: %matplotlib inline

from sklearn import datasets, cluster
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Random seed chosen so cluster labeling corresponds to the original y values
np.random.seed(2)

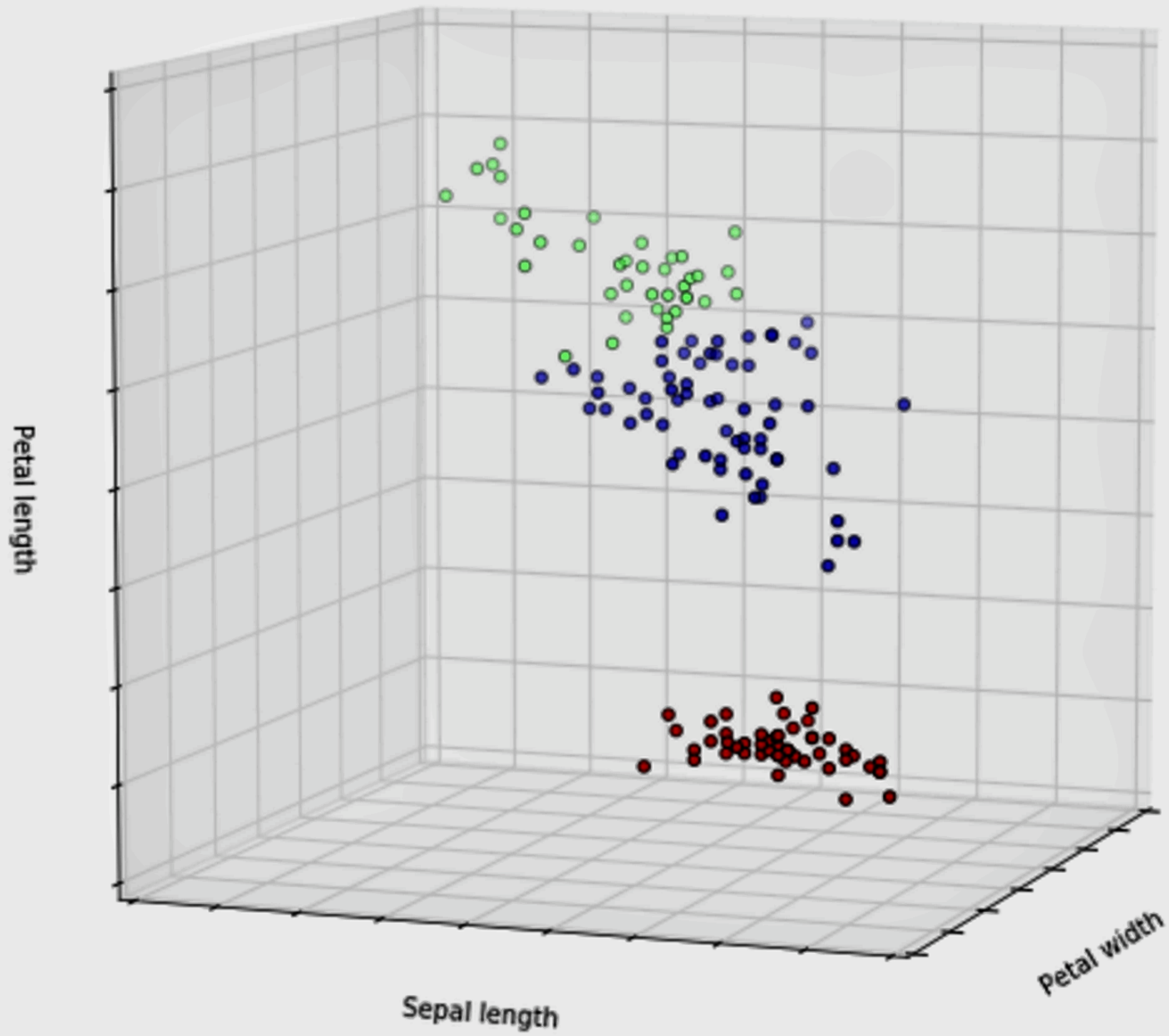
In [39]: # load data
iris = datasets.load_iris()
X_iris = iris.data
y_iris = iris.target

In [40]: # Form three clusters
k_means = cluster.KMeans(n_clusters=3)
k_means.fit(X_iris)
labels = k_means.labels_

In [41]: # How many of the samples correctly labeled?
correct_labels = sum(y_iris == labels)

print("Result: %d of %d samples correctly labeled, inertia:%f." % (correct_labels, y_iris.size, k_means.inertia_))

Result: 134 of 150 samples correctly labeled, inertia:78.940841.
```

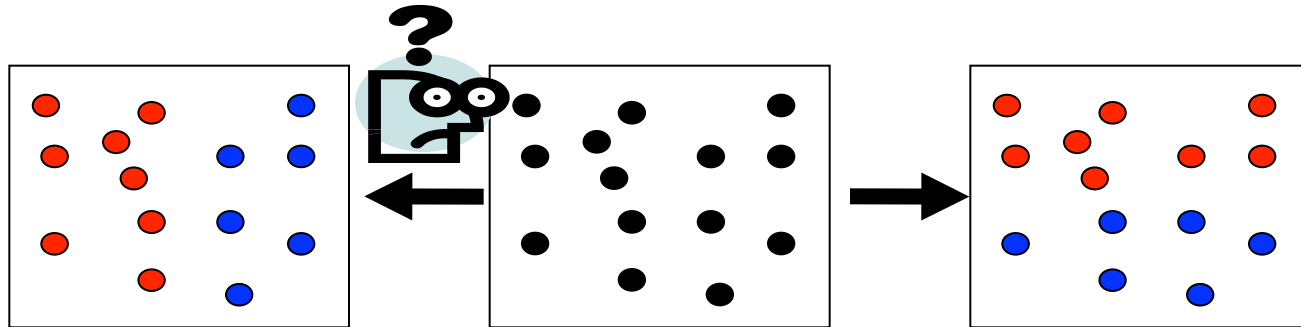


Problems with K-Means

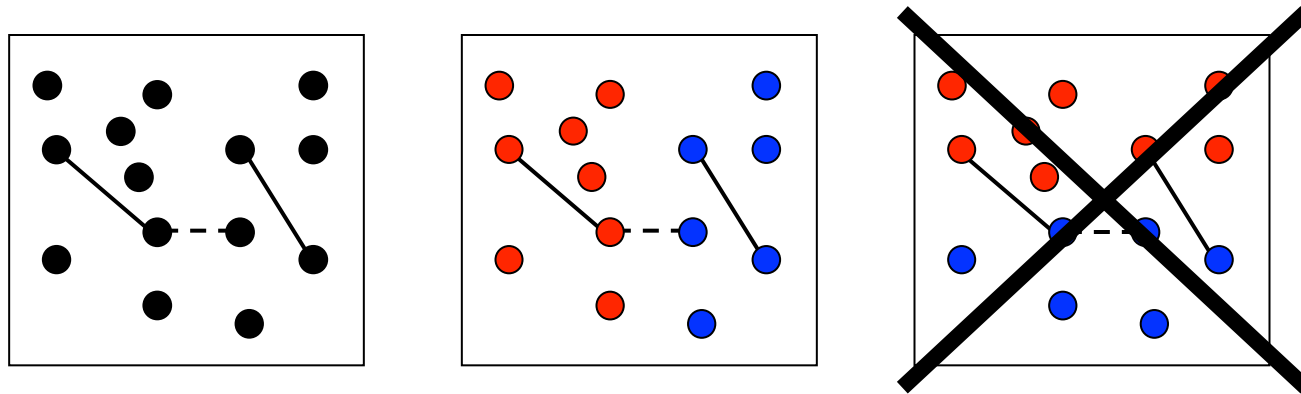
- Only works for numeric data (typically reals)
- **Very** sensitive to the initial points
 - Do many runs of k-Means, each with different initial centroids
 - Seed the centroids using a better method than random. (e.g., Farthest-first sampling)
- Must manually choose k
 - Learn the optimal k for the clustering. (Note that this requires a performance measure)

Problems with K-Means

- How do you tell it which clustering you want?



– Constrained clustering techniques

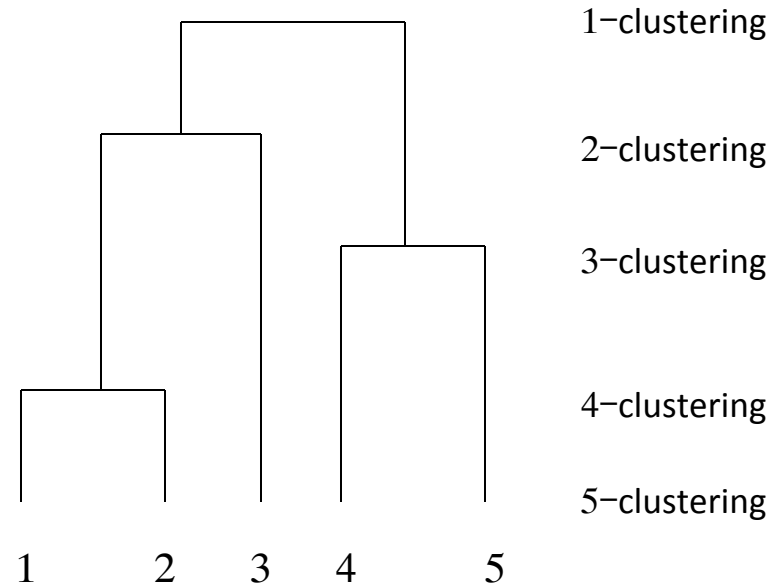
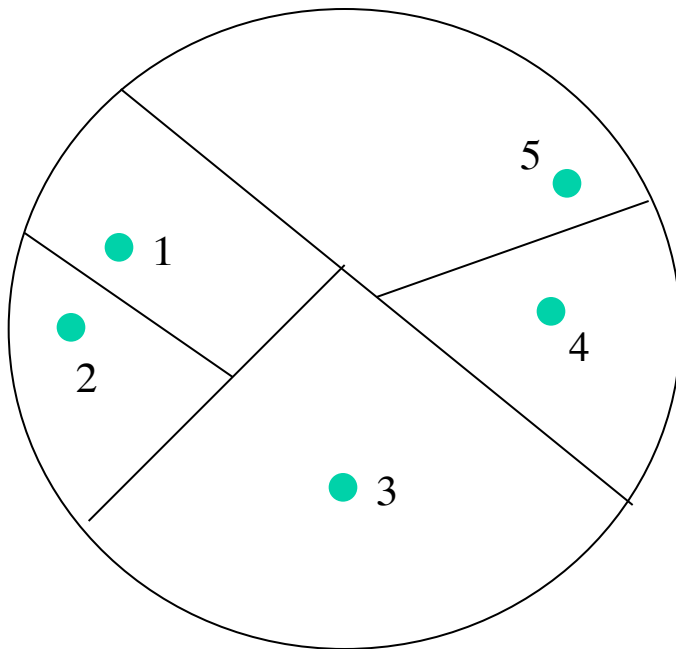


— Same-cluster constraint
(must-link)

- - - Different-cluster constraint
(cannot-link)

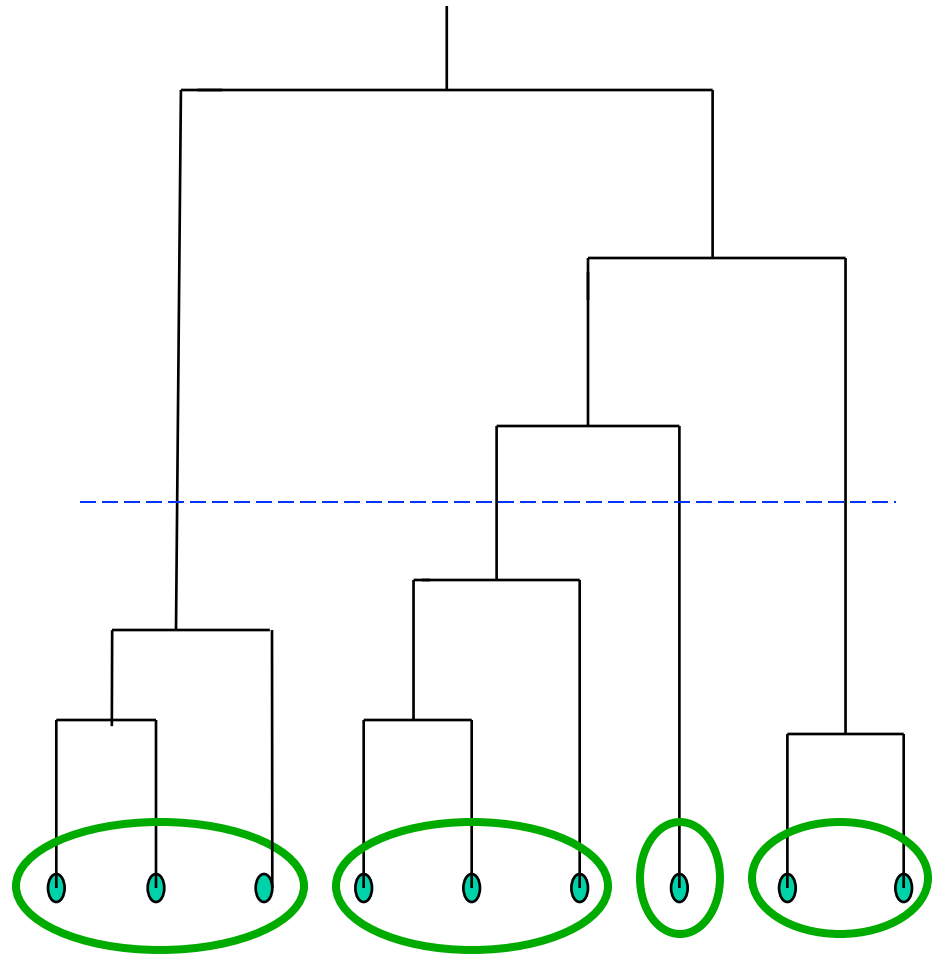
Hierarchical Clustering

Recursive partitioning/merging of a data set



Dendrogram

- Represents all partitionings of data
- Can get a K clustering by looking at **connected** components at any given level
- Frequently binary dendograms, but n-ary dendograms easy to obtain with minor changes to algorithms



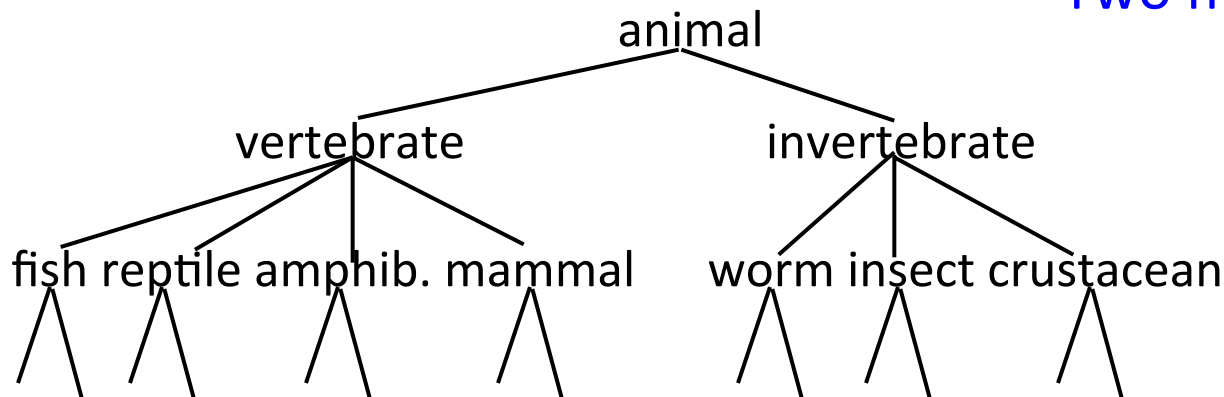
Advantages of hierarchical clustering

- Don't need to specify the number of clusters
- Good for data visualization
 - See how the data points interact at many levels
 - Can view the data at multiple levels of granularity
 - Understand how all points interact
- Specifies all of the K clusterings/partitions

Hierarchical Clustering

- Common in many domains
 - Biologists and social scientists
 - Gene expression data
 - Document/web page organization
 - DMOZ
 - Yahoo directories

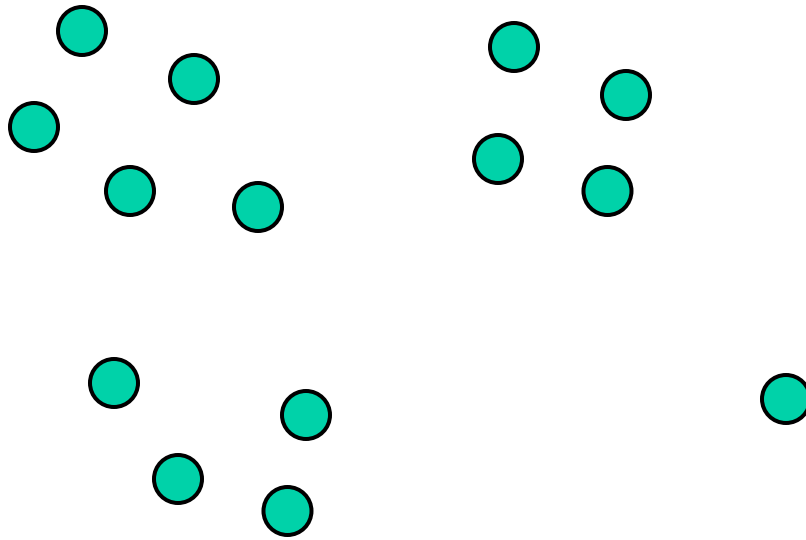
Two main approaches...



Divisive hierarchical clustering

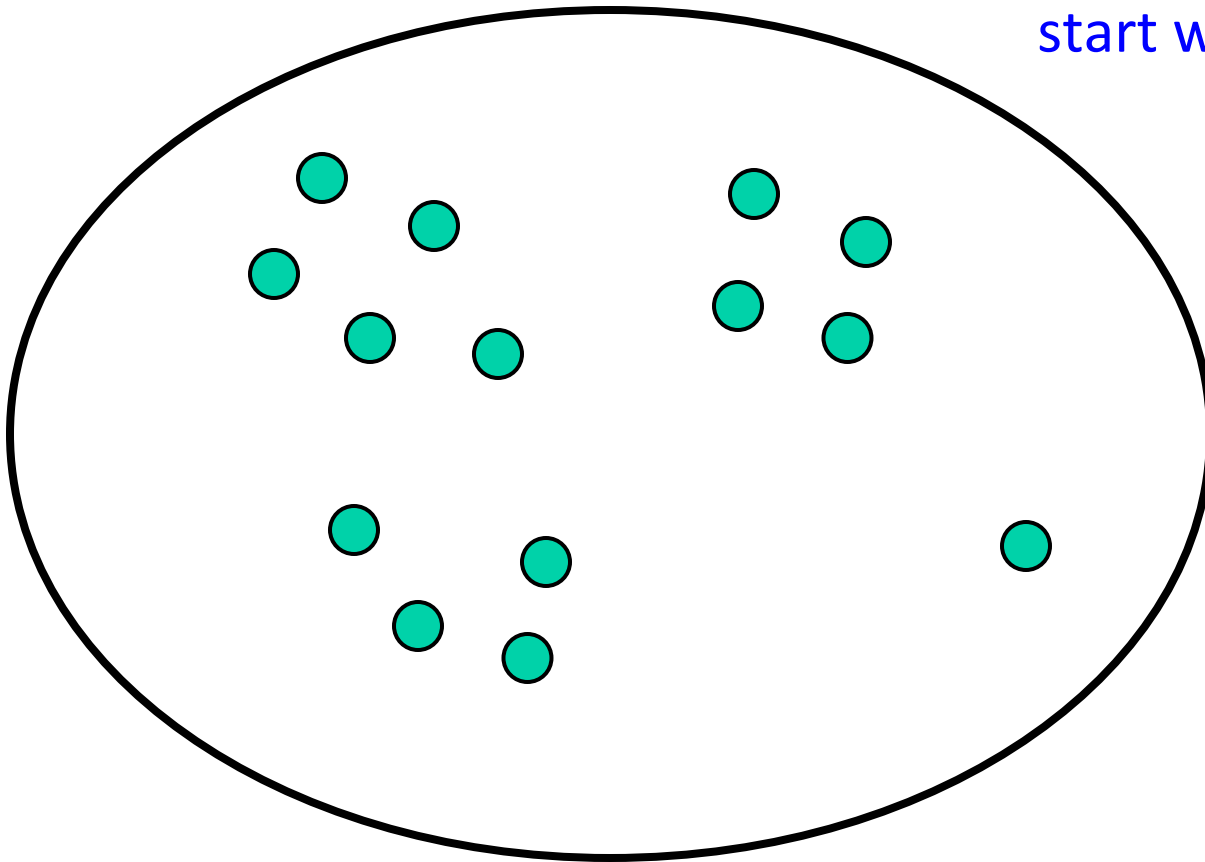
- Top-down
- Finding the best partitioning of the data is generally exponential in time
- Common approach:
 - Let \mathbf{C} be a set of clusters
 - Initialize \mathbf{C} to be the one-clustering of the data
 - While there exists a cluster c in \mathbf{C}
 - remove c from \mathbf{C}
 - partition c into 2 clusters using a flat clustering algorithm, c_1 and c_2
 - Add to c_1 and c_2 \mathbf{C}
- Bisecting k-means

Divisive clustering



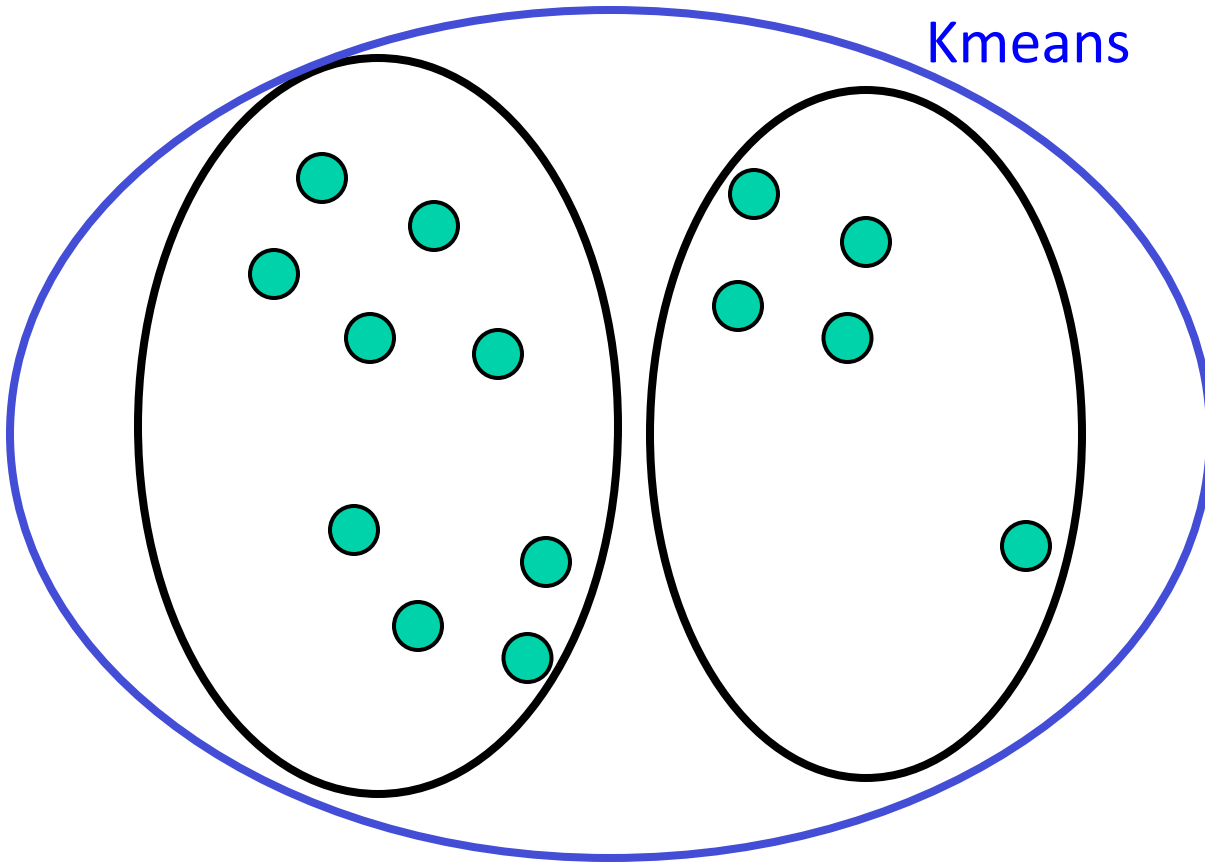
Divisive clustering

start with one cluster

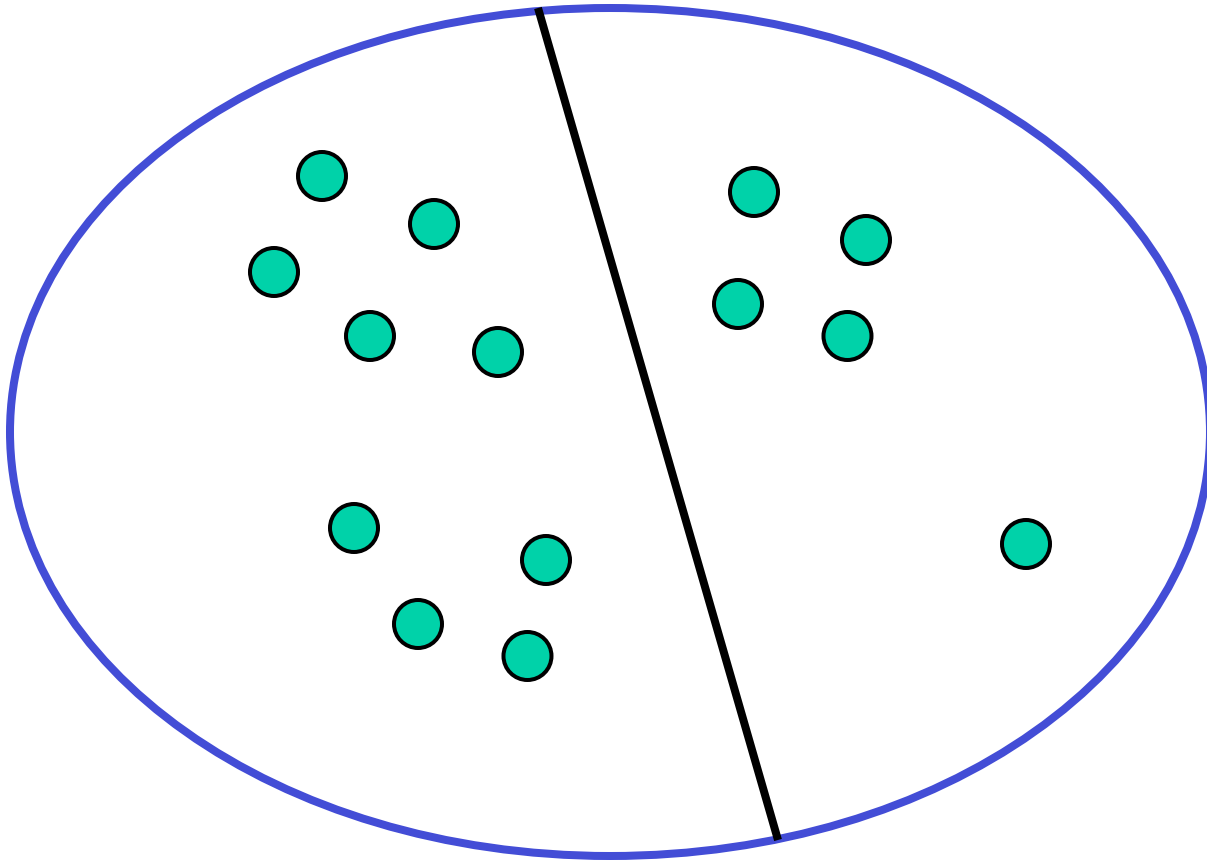


Divisive clustering

split using flat clustering, e.g.
Kmeans

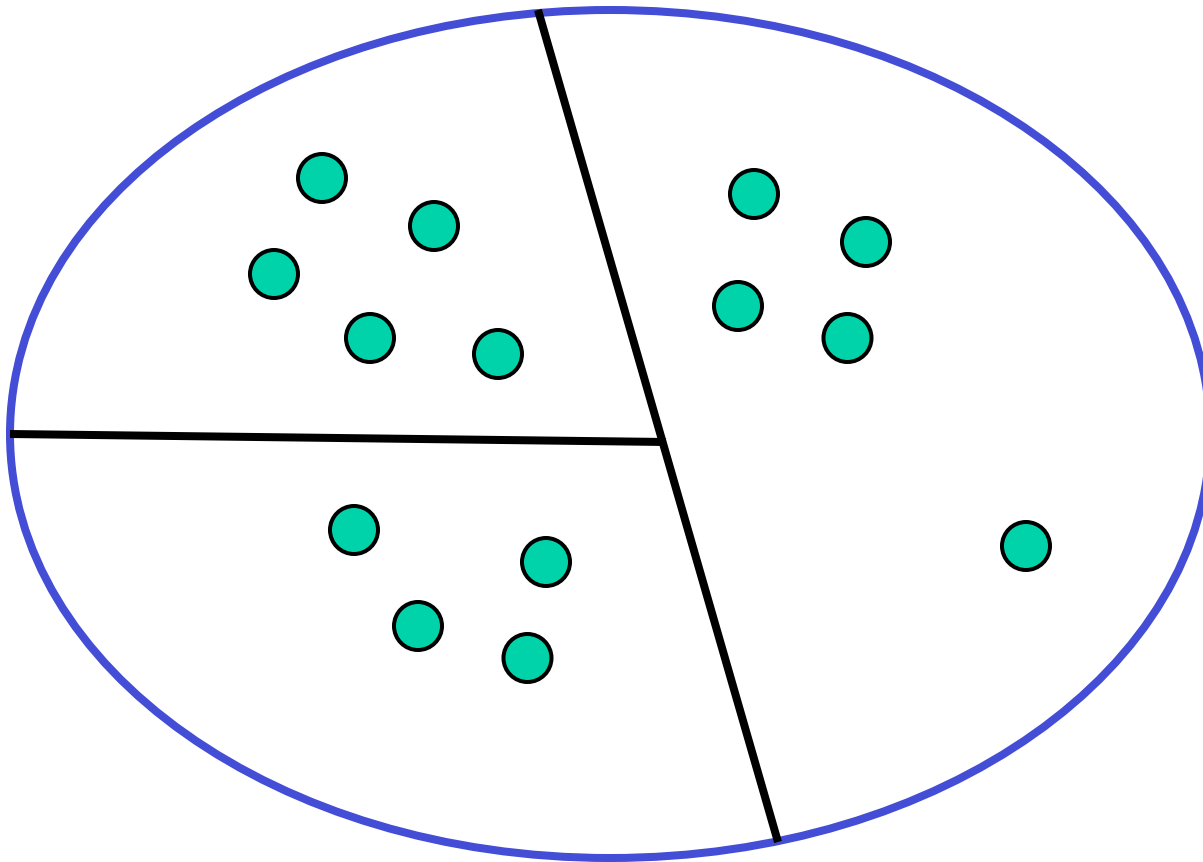


Divisive clustering



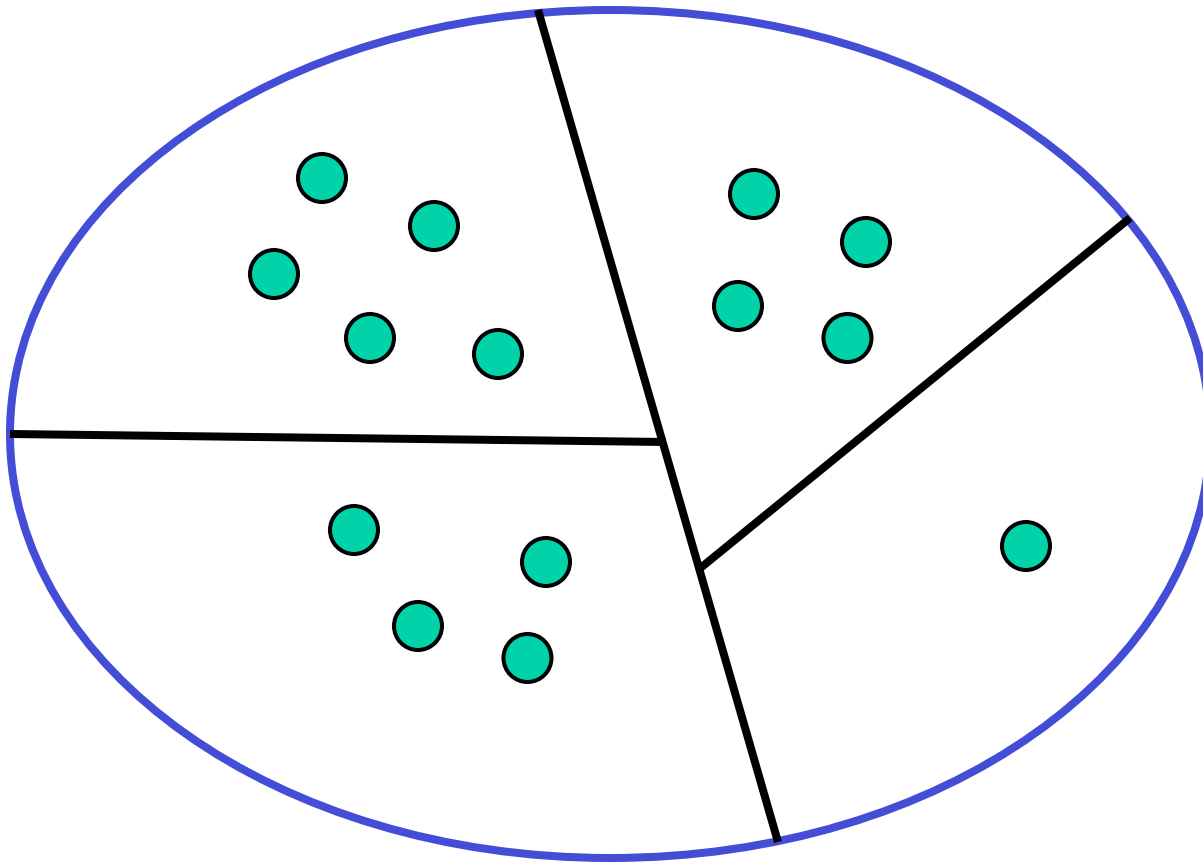
Divisive clustering

split using flat clustering

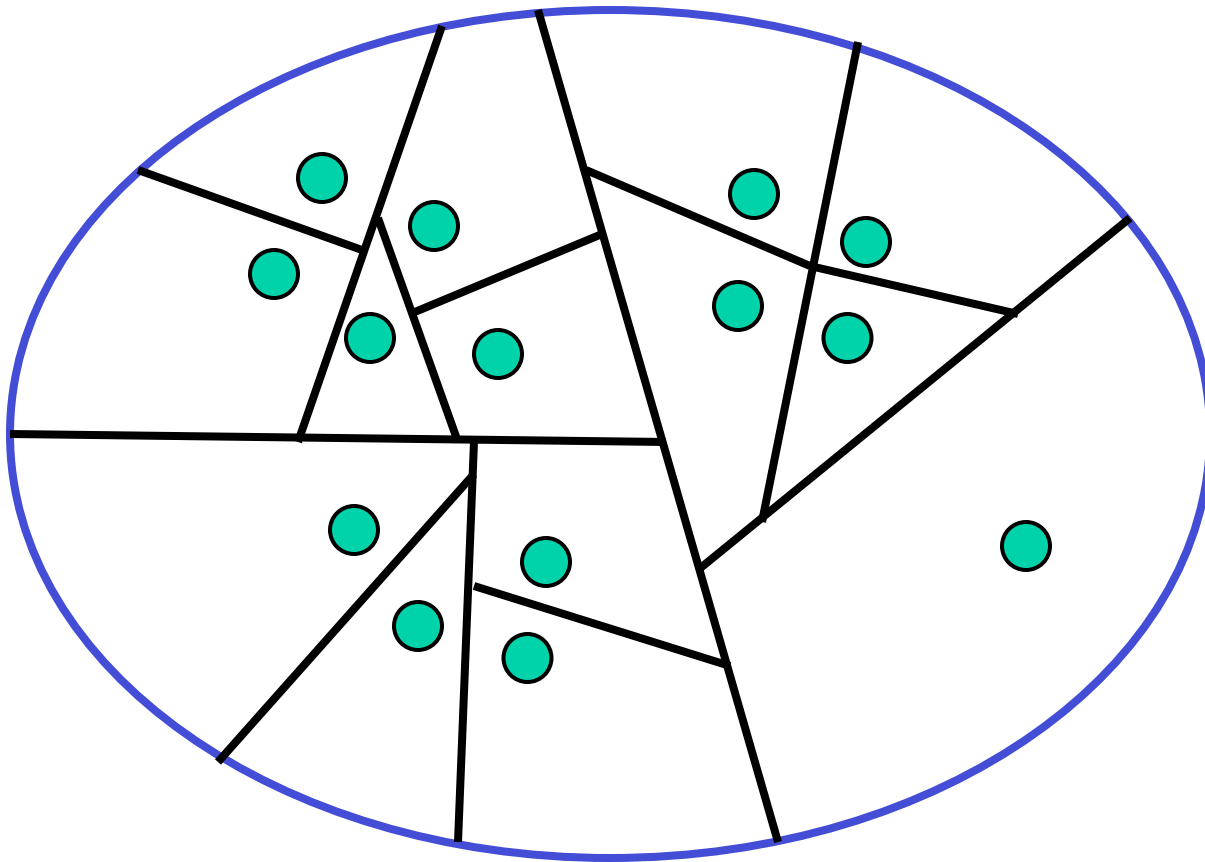


Divisive clustering

split using flat clustering



Divisive clustering



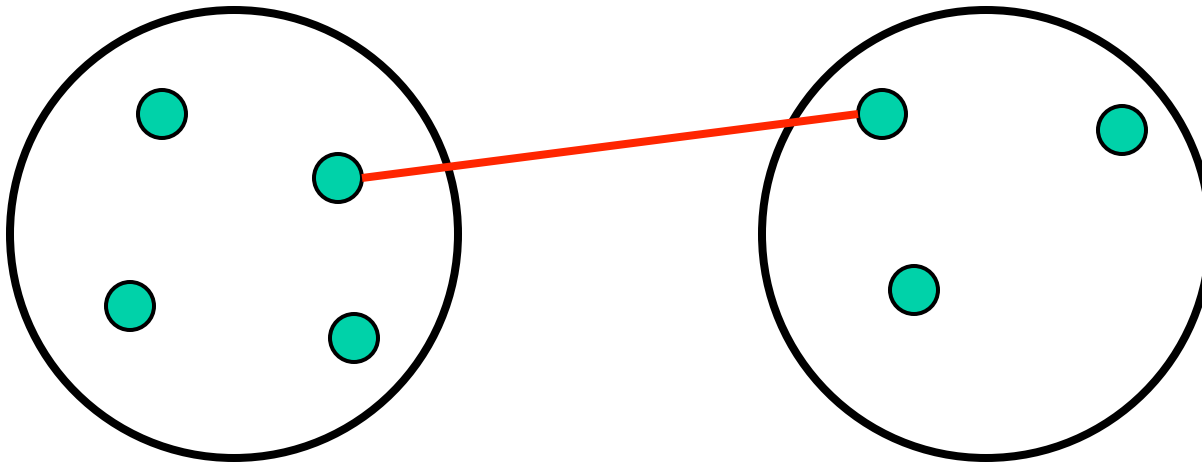
Hierarchical Agglomerative Clustering

- Let \mathbf{C} be a set of clusters
- Initialize \mathbf{C} to all points/docs as separate clusters
- While \mathbf{C} contains more than one cluster
 - find c_1 and c_2 in \mathbf{C} that are **closest together**
 - remove c_1 and c_2 from \mathbf{C}
 - merge c_1 and c_2 and add resulting cluster to \mathbf{C}
- History of merging forms a binary tree or hierarchy
- **Q: How to measure distance between clusters?**

Distance between clusters

Single-link

- Similarity of the *most* similar (single-link)



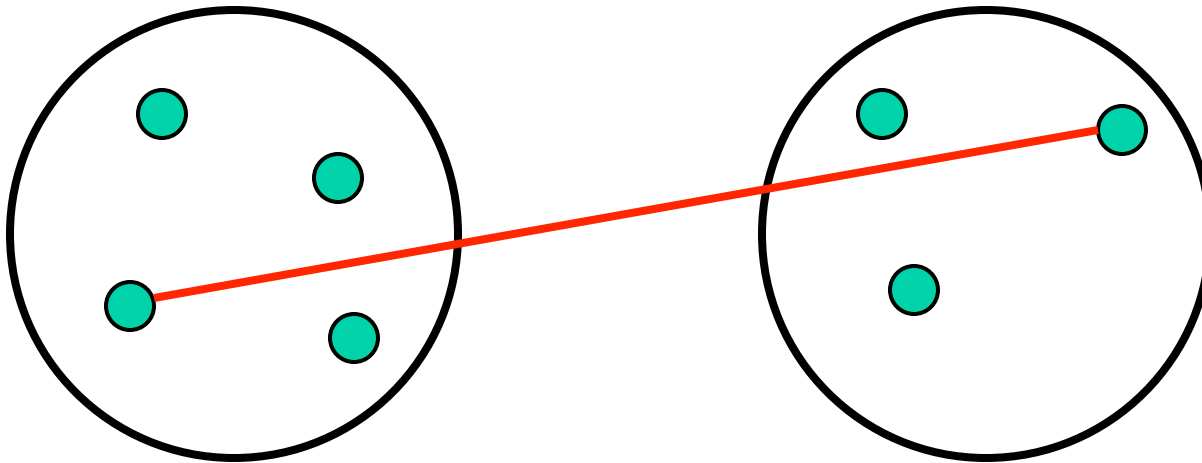
$$\max_{l \in L, r \in R} \text{sim}(l, r)$$

Distance between clusters

Complete-link

- Similarity of the “furthest” points, the *least* similar

$$\min_{l \in L, r \in R} \text{sim}(l, r)$$



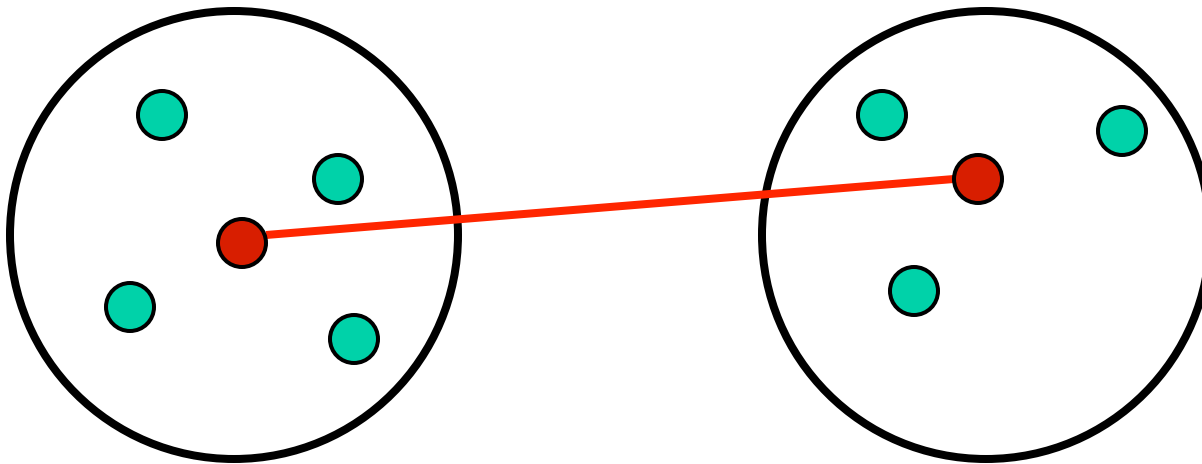
Why are these “local” methods used?

efficiency

Distance between clusters

- **Centroid**

- Clusters whose centroids (centers of gravity) are the most similar

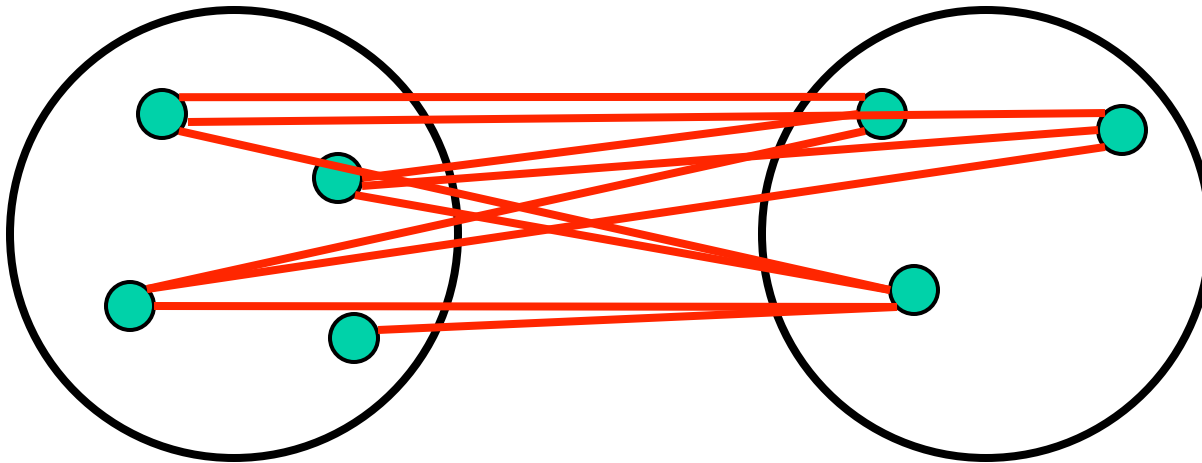


$$\|\mu(L) - \mu(R)\|^2$$

Distance between clusters

- **Average-link**

- Average similarity between all pairs of elements



$$\frac{1}{|L| \cdot |R|} \sum_{x \in L, y \in R} \|x - y\|^2$$