Unsupervised Learning: Clustering

Some material adapted from slides by Andrew Moore, CMU

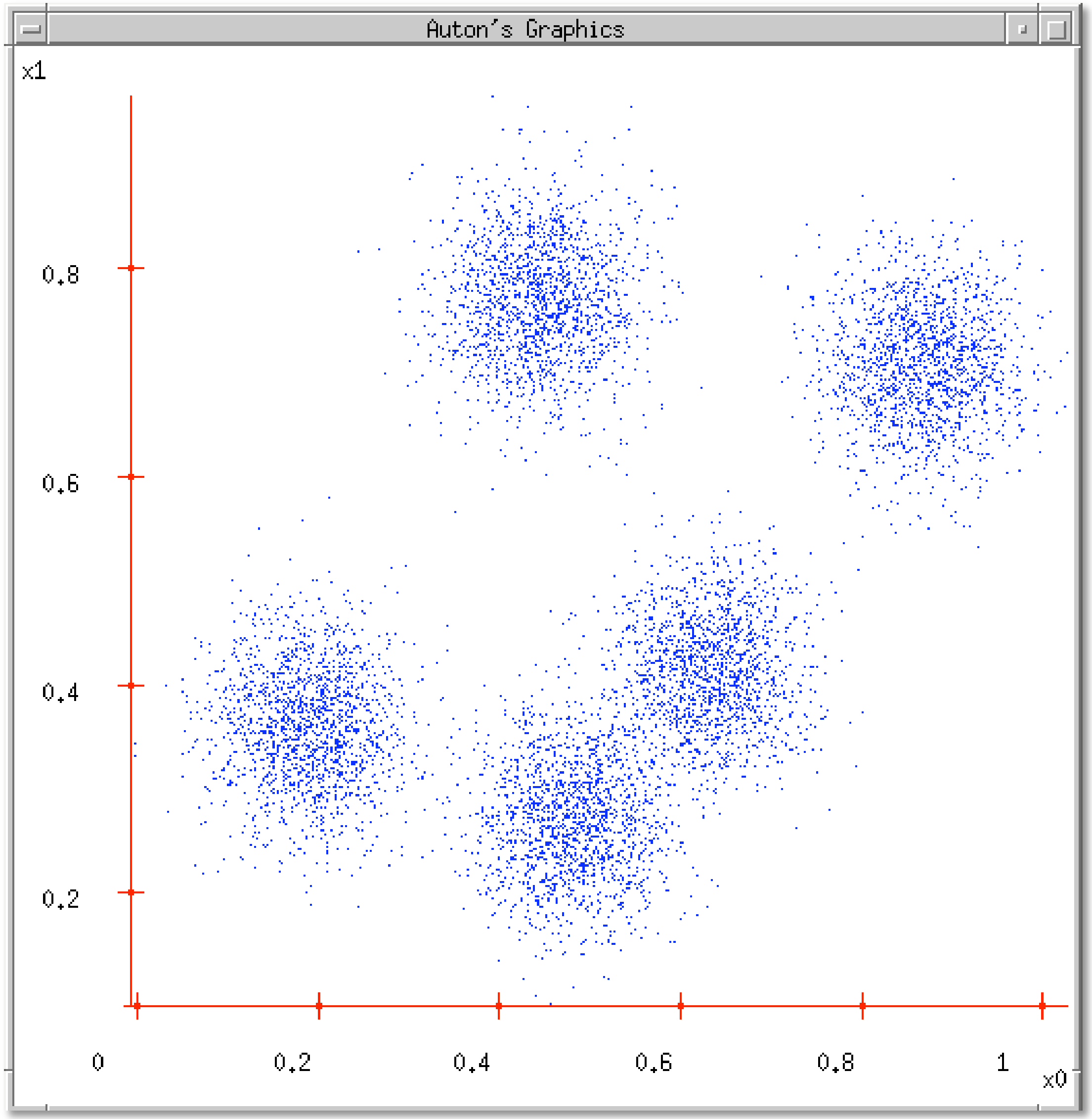
Unsupervised Learning

- 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 is 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 by instance **feature space**
- **Agglomerative** approach works bottom up:
  - Treat each instance as a cluster
  - Merge two closest clusters
  - Repeat until a 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

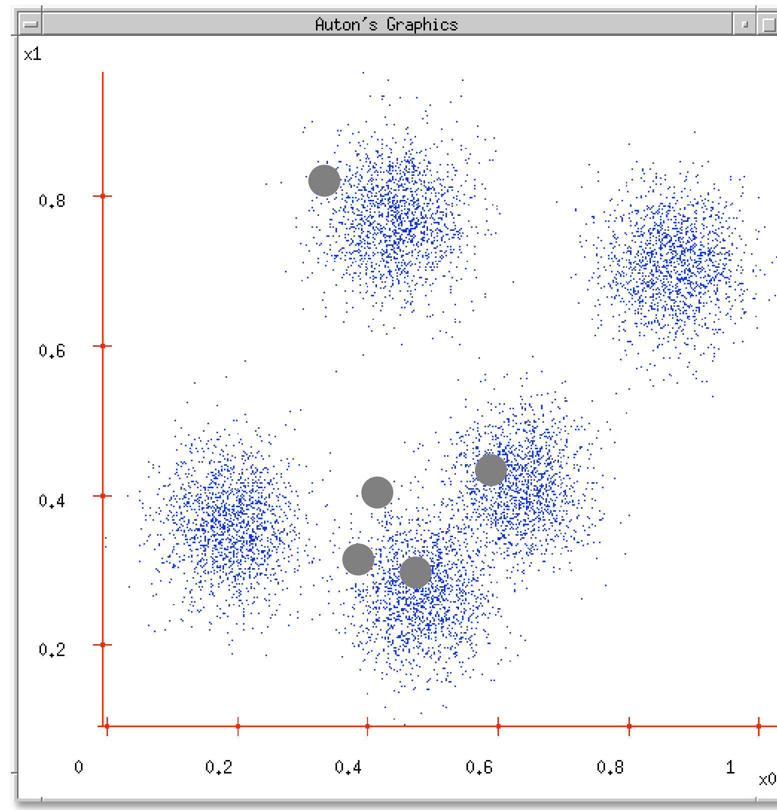
Clustering Data



K-Means Clustering

- Randomly choose k cluster center locations, aka **centroids**
- Loop until convergence
  - assign a point to cluster of the closest centroid
  - re-estimate cluster centroids based on its data assigned
- Convergence: no point is assigned to a different cluster

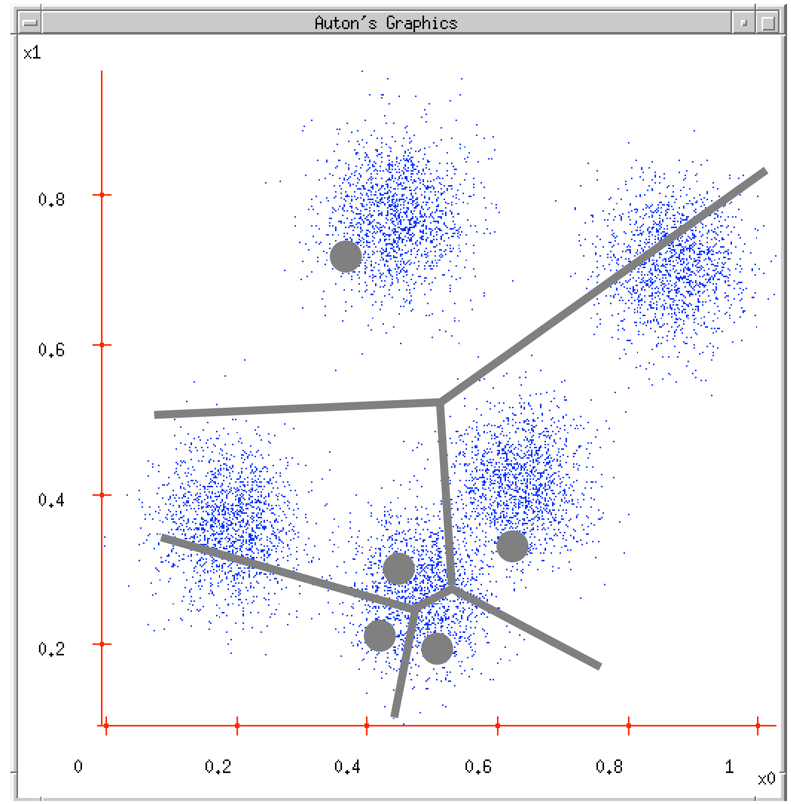
$k = 5$



K-Means Clustering

K-Means (k , data)

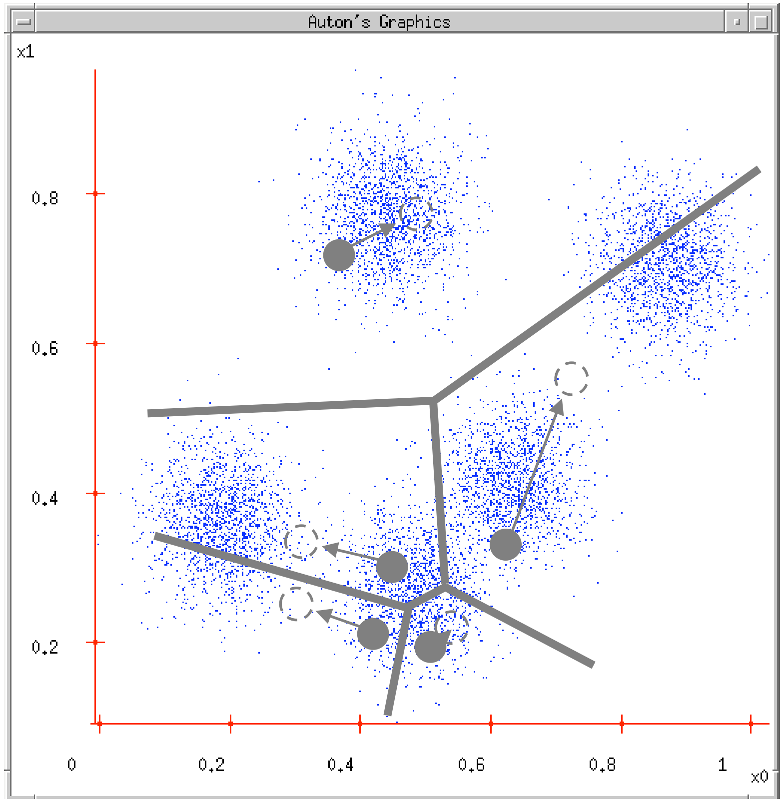
- Randomly choose k cluster center locations (centroids)
- Loop until convergence
  - Assign each point to the cluster of the closest centroid.
  - Re-estimate the cluster centroids based on the data assigned to each
- Convergence: no point is assigned to a different cluster



K-Means Clustering

K-Means (k , data)

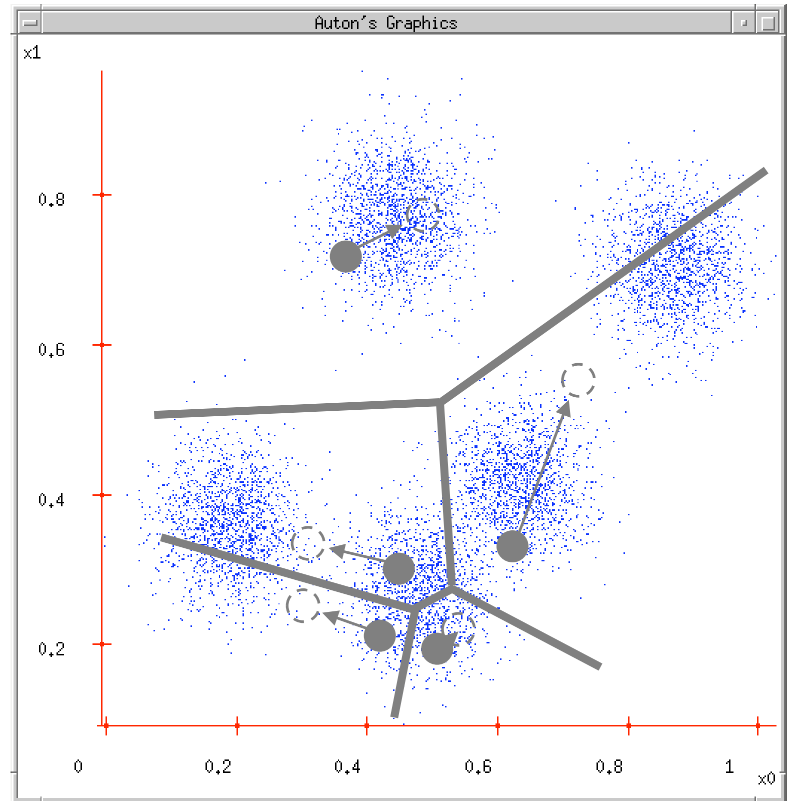
- Randomly choose k cluster center locations (centroids)
- Loop until convergence
  - Assign each point to the cluster of the closest centroid
  - Re-estimate the cluster centroids based on the data assigned to each
- Convergence: no point is assigned to a different cluster

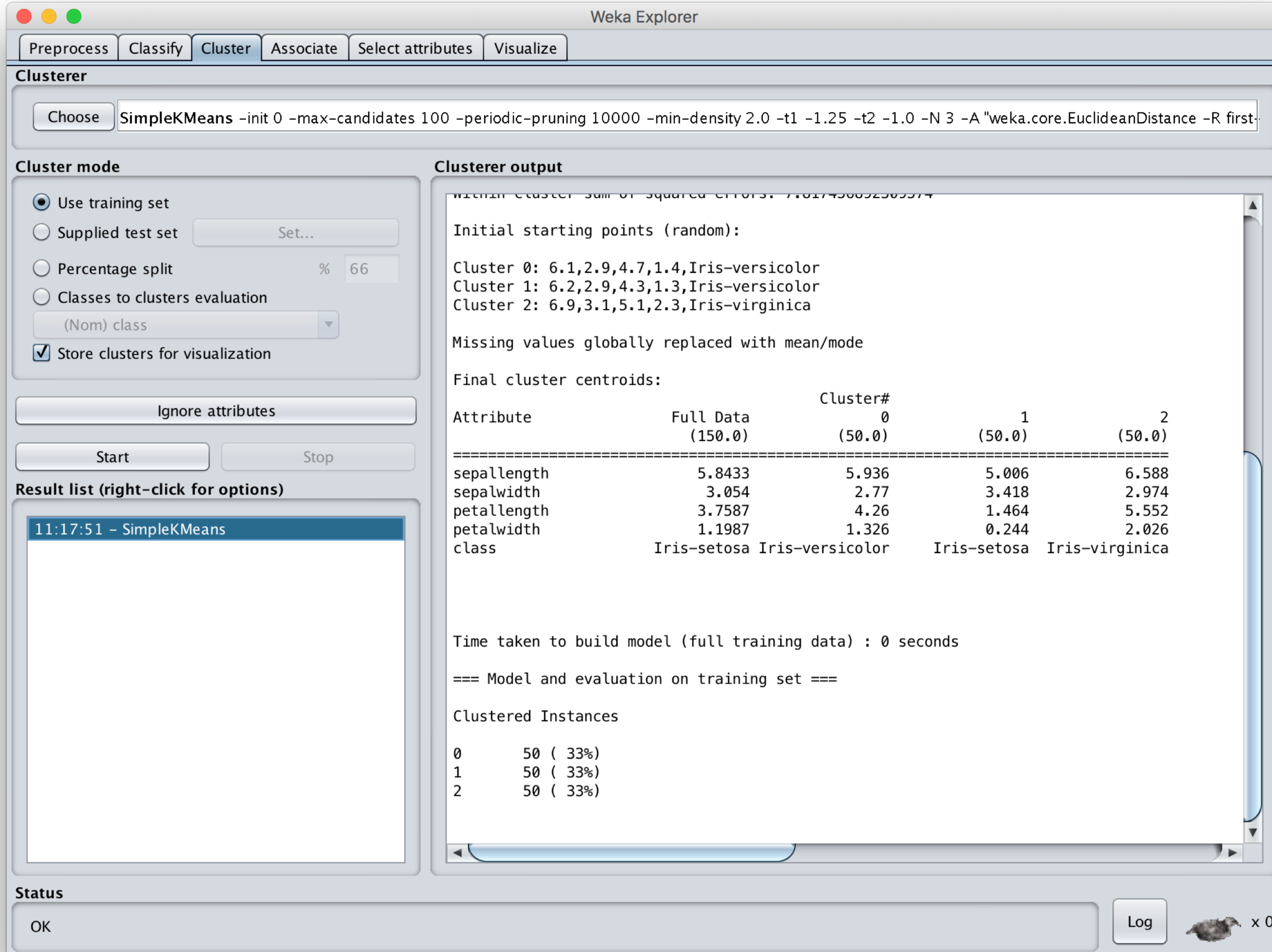


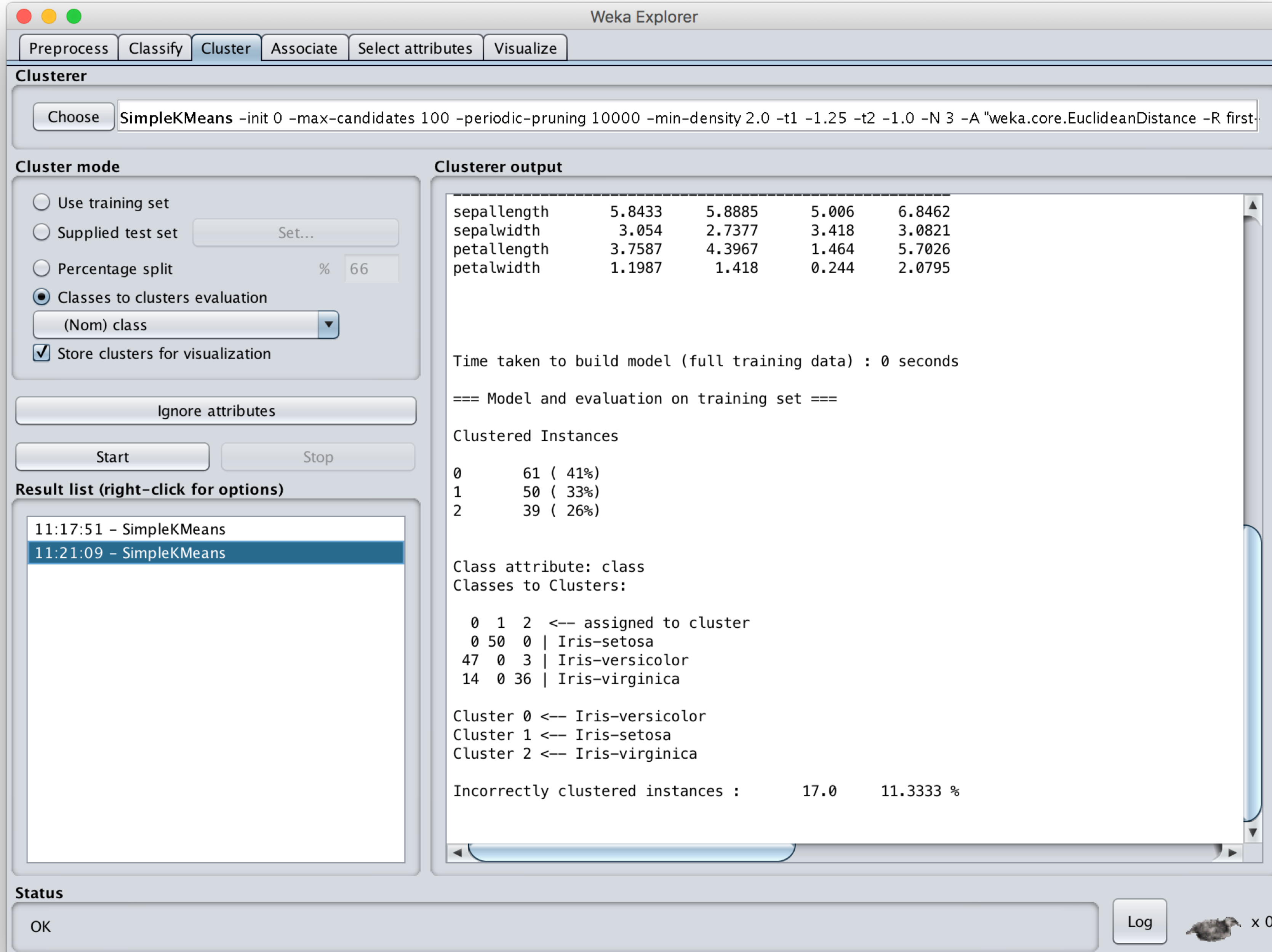
K-Means Clustering

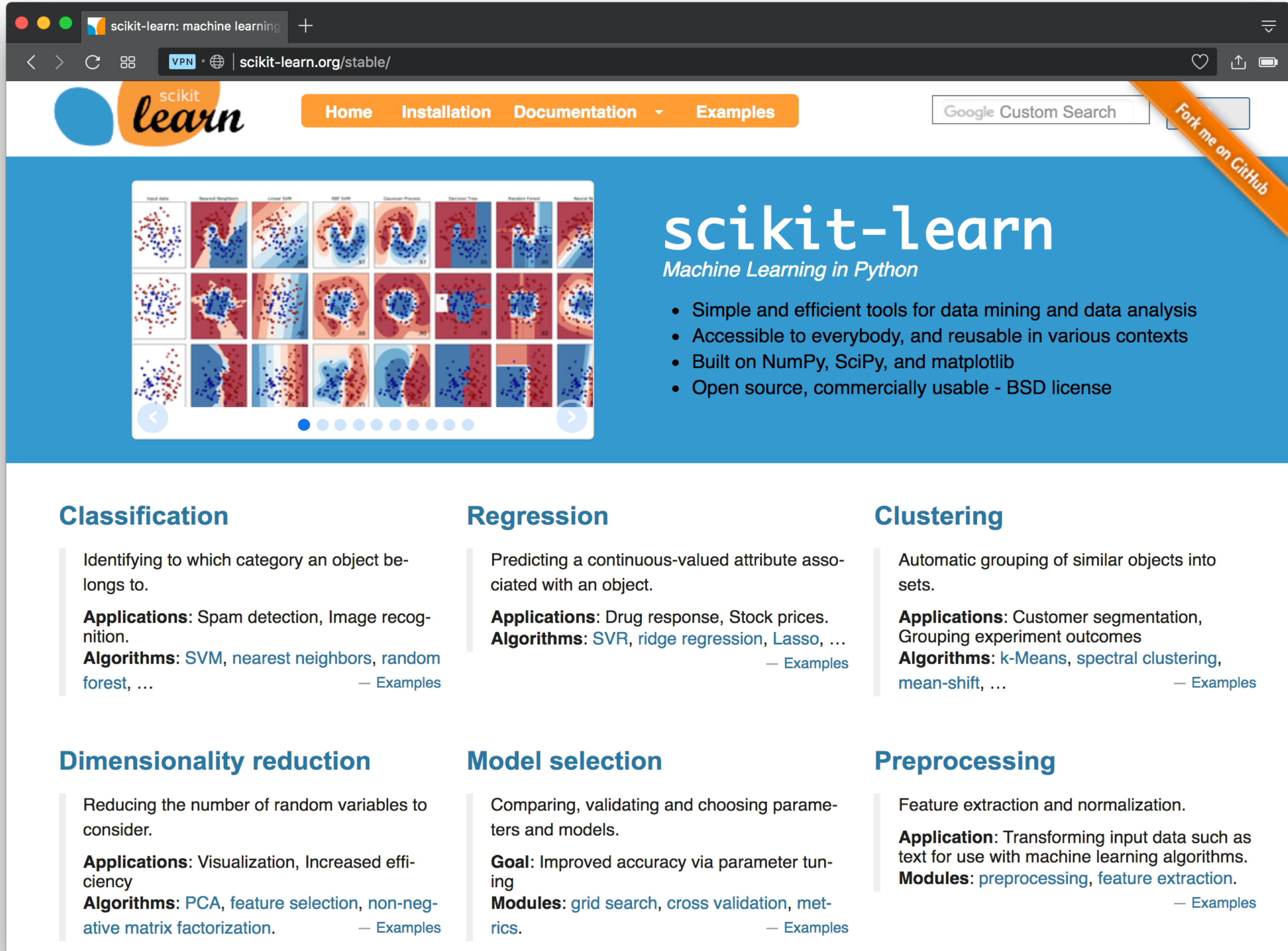
K-Means (k , data)

- 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




Weka Explorer
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 "weka.core.EuclideanDistance -R first-
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
% 66
Classes to clusters evaluation
(Nom) class
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
11:17:51 - SimpleKMeans
Clusterer output
Initial starting points (random):
Cluster 0: 6.1,2.9,4.7,1.4,Iris-versicolor
Cluster 1: 6.2,2.9,4.3,1.3,Iris-versicolor
Cluster 2: 6.9,3.1,5.1,2.3,Iris-virginica
Missing values globally replaced with mean/mode
Final cluster centroids:
Cluster#
Attribute Full Data 0 1 2
(150.0) (50.0) (50.0) (50.0)
sepallength 5.8433 5.936 5.006 6.588
sepalwidth 3.054 2.77 3.418 2.974
petallength 3.7587 4.26 1.464 5.552
petalwidth 1.1987 1.326 0.244 2.026
class Iris-setosa Iris-versicolor Iris-setosa Iris-virginica
Time taken to build model (full training data) : 0 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 50 (33%)
1 50 (33%)
2 50 (33%)
Status
OK
Log
x 0


Weka Explorer
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 "weka.core.EuclideanDistance -R first-
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
% 66
Classes to clusters evaluation
(Nom) class
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
11:17:51 - SimpleKMeans
11:21:09 - SimpleKMeans
Clusterer output
sepallength 5.8433 5.8885 5.006 6.8462
sepalwidth 3.054 2.7377 3.418 3.0821
petallength 3.7587 4.3967 1.464 5.7026
petalwidth 1.1987 1.418 0.244 2.0795
Time taken to build model (full training data) : 0 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 61 (41%)
1 50 (33%)
2 39 (26%)
Class attribute: class
Classes to Clusters:
0 1 2 <-- assigned to cluster
0 50 0 | Iris-setosa
47 0 3 | Iris-versicolor
14 0 36 | Iris-virginica
Cluster 0 <-- Iris-versicolor
Cluster 1 <-- Iris-setosa
Cluster 2 <-- Iris-virginica
Incorrectly clustered instances : 17.0 11.3333 %
Status
OK
Log
x 0


scikit-learn: machine learning
scikit-learn.org/stable/
scikit learn
Home
Installation
Documentation
Examples
Google Custom Search
Fork me on GitHub
scikit-learn
Machine Learning in Python
Simple and efficient tools for data mining and data analysis
Accessible to everybody, and reusable in various contexts
Built on NumPy, SciPy, and matplotlib
Open source, commercially usable - BSD license
Classification
Identifying to which category an object belongs to.
Applications: Spam detection, Image recognition.
Algorithms: SVM, nearest neighbors, random forest, …
— Examples
Regression
Predicting a continuous-valued attribute associated with an object.
Applications: Drug response, Stock prices.
Algorithms: SVR, ridge regression, Lasso, …
— Examples
Clustering
Automatic grouping of similar objects into sets.
Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, …
— Examples
Dimensionality reduction
Reducing the number of random variables to consider.
Applications: Visualization, Increased efficiency
Algorithms: PCA, feature selection, non-negative matrix factorization.
— Examples
Model selection
Comparing, validating and choosing parameters and models.
Goal: Improved accuracy via parameter tuning
Modules: grid search, cross validation, metrics.
— Examples
Preprocessing
Feature extraction and normalization.
Application: Transforming input data such as text for use with machine learning algorithms.
Modules: preprocessing, feature extraction.
— Examples

sklearn.cluster.KMeans

class sklearn.cluster.**KMeans**(*n_clusters=8, init='k-means++', n_init=10, max_iter=300, tol=0.0001, precompute_distances='auto', verbose=0, random_state=None, copy_x=True, n_jobs=1, algorithm='auto'*) [source]

K-Means clustering

Read more in the User Guide.

Parameters: **n_clusters** : int, optional, default: 8

The number of clusters to form as well as the number of centroids to generate.

init : {'k-means++', 'random' or an ndarray}

Method for initialization, defaults to 'k-means++':

'k-means++' : selects initial cluster centers for k-mean clustering in a smart way to speed up convergence. See section Notes in k_init for more details.

'random': choose k observations (rows) at random from data for the initial centroids.

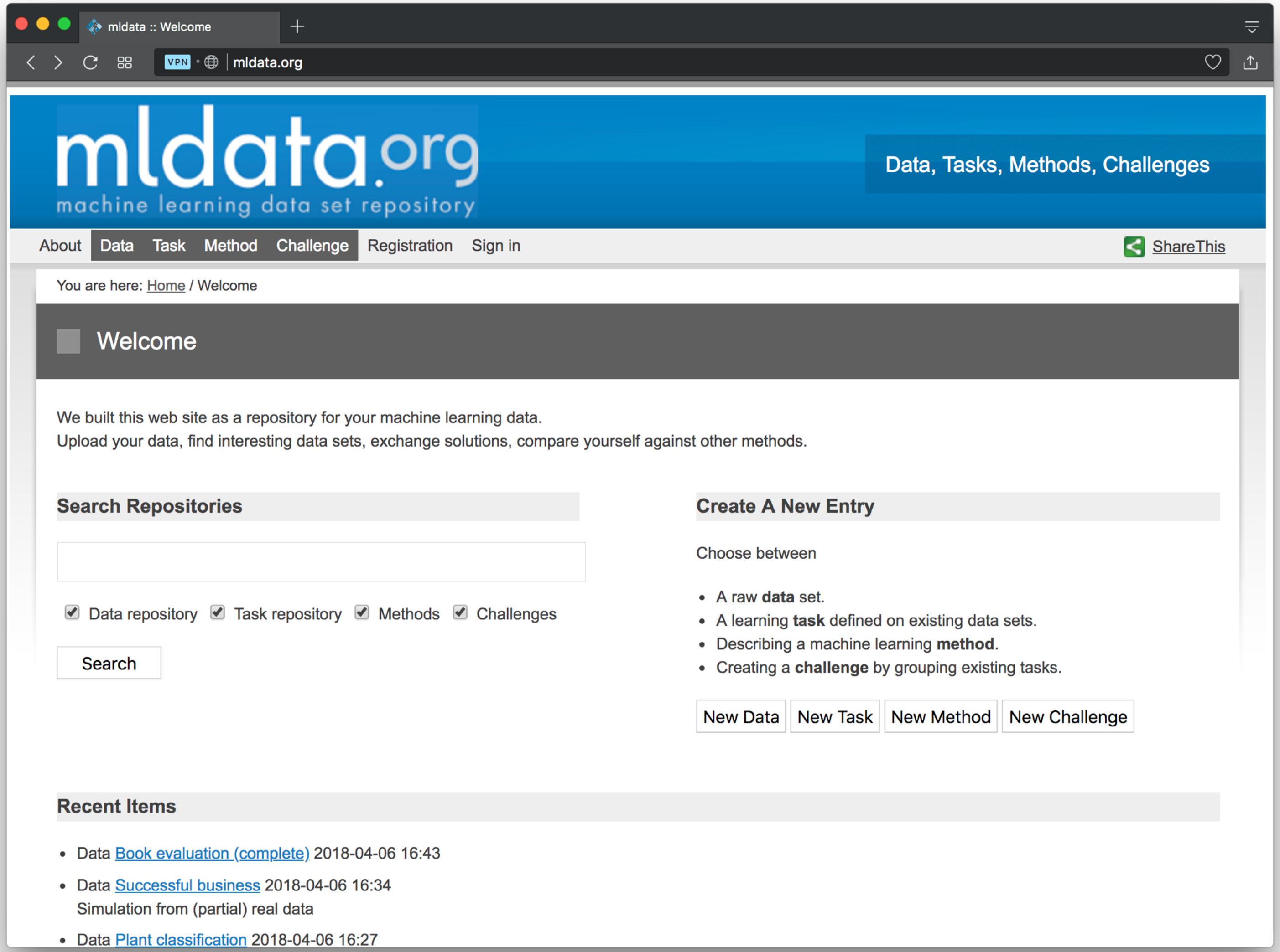
If an ndarray is passed, it should be of shape (n_clusters, n_features) and gives the initial centers.

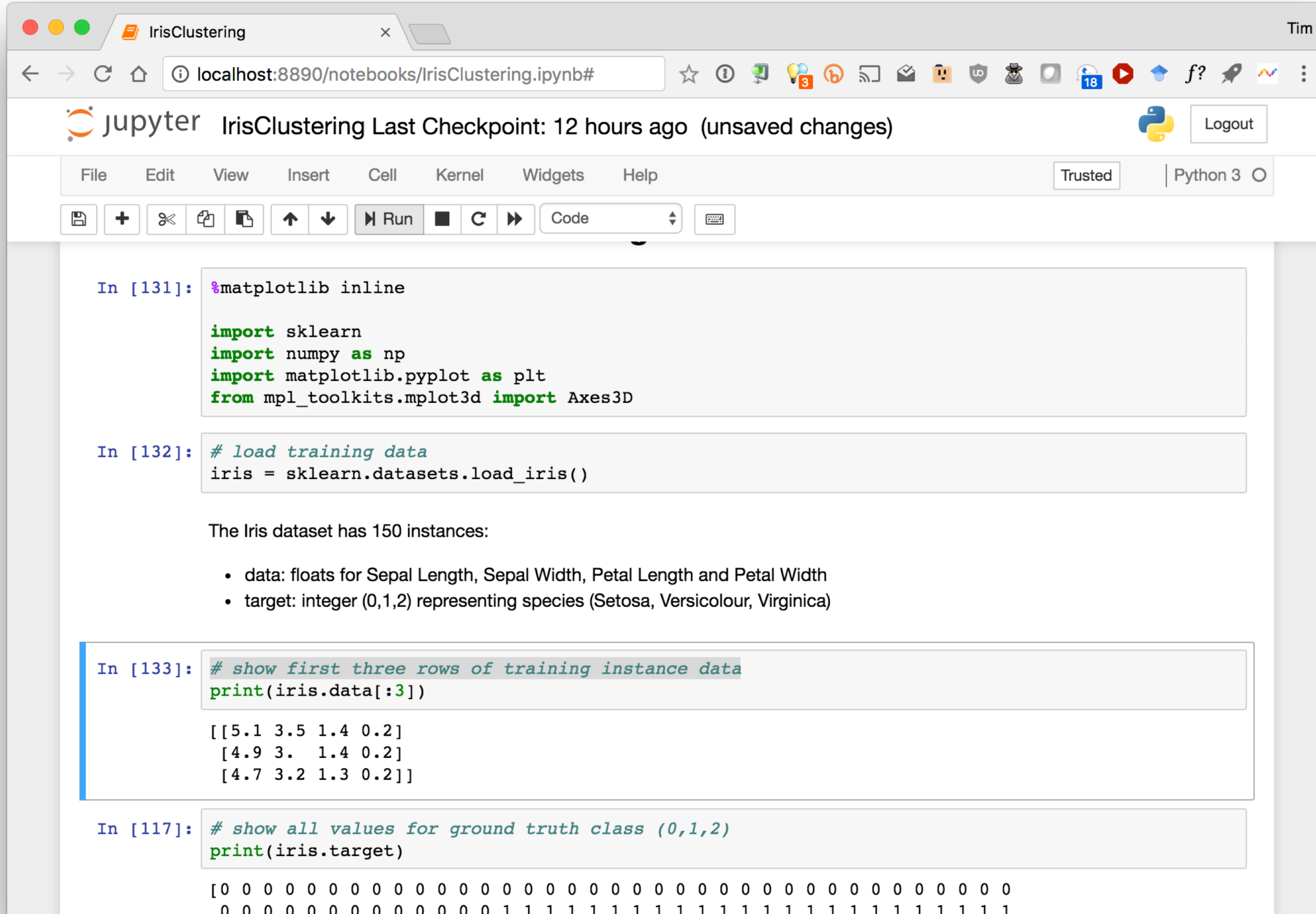
n_init : int, default: 10

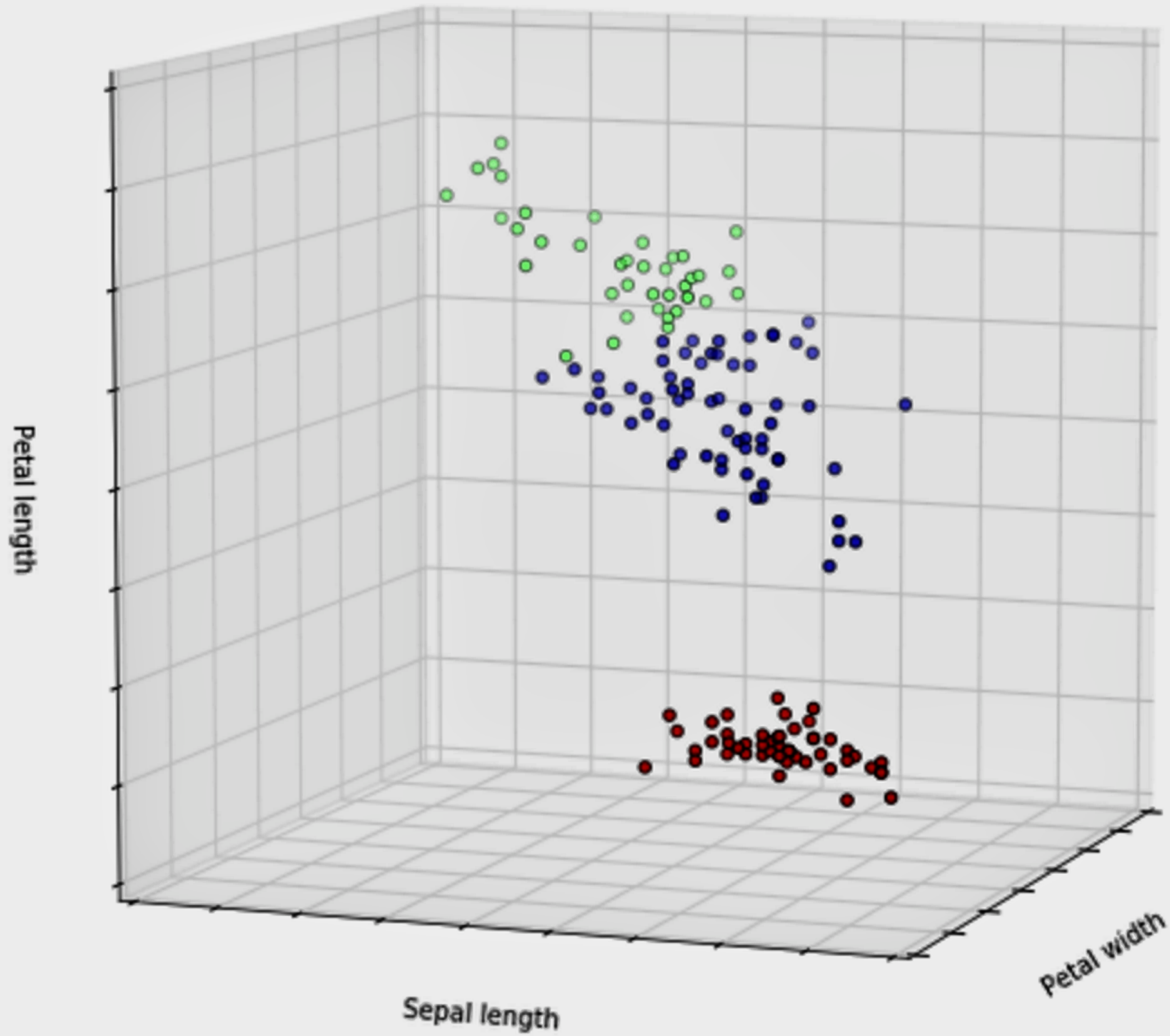
Number of time the k-means algorithm will be run with different centroid seeds. The final results will be the best output of n_init consecutive runs in terms of inertia.

max_iter : int, default: 300

- sklearn has a few test datasets, including IRIS
- Can load data directly from mldata.org & from CSV files
- mldata.org has ~900 datasets for machine learning




IrisClustering
localhost:8890/notebooks/IrisClustering.ipynb#
Tim
jupyter
IrisClustering Last Checkpoint: 12 hours ago (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code
In [131]:
%matplotlib inline
import sklearn
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
In [132]:
# load training data
iris = sklearn.datasets.load_iris()
The Iris dataset has 150 instances:
data: floats for Sepal Length, Sepal Width, Petal Length and Petal Width
target: integer (0,1,2) representing species (Setosa, Versicolour, Virginica)
In [133]:
# show first three rows of training instance data
print(iris.data[:3])
[[5.1 3.5 1.4 0.2]
[4.9 3. 1.4 0.2]
[4.7 3.2 1.3 0.2]]
In [117]:
# show all values for ground truth class (0,1,2)
print(iris.target)
[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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

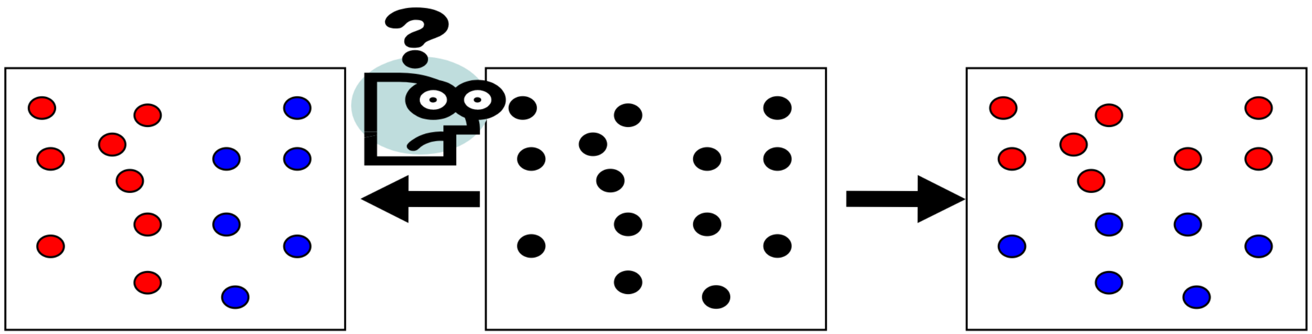

Petal length
Sepal length
Petal width

Problems with K-Means

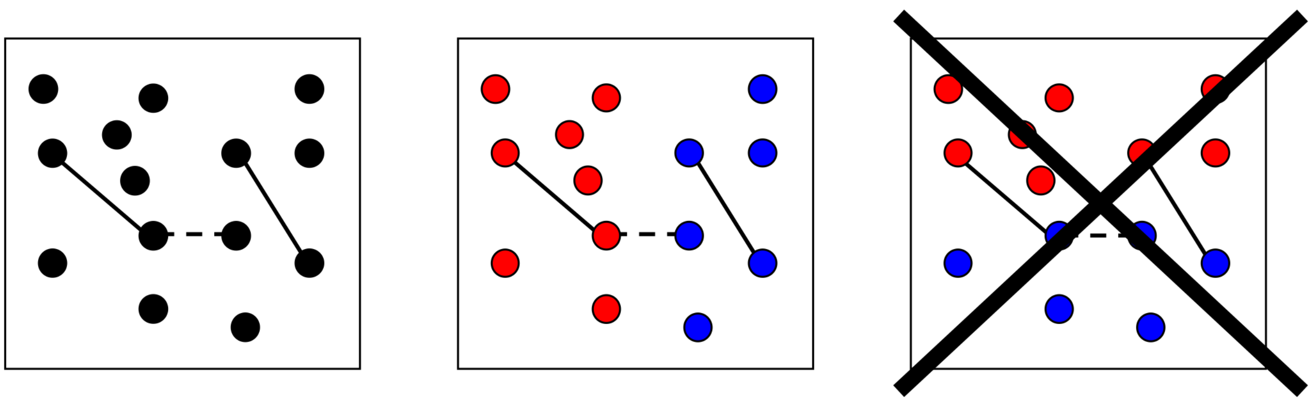
- Only works for numeric data (typically reals)
- ***Very*** sensitive to the initial points
  - Do many runs of k-Means, each with different initial centroids
  - Seed centroids using better method than random (e.g., Farthest-first sampling)
- Must manually choose k
  - Learn optimal k for clustering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

Problems with K-Means

- How do you tell it which clustering you want?



- Constrained clustering technique



Same-cluster constraint (must-link)

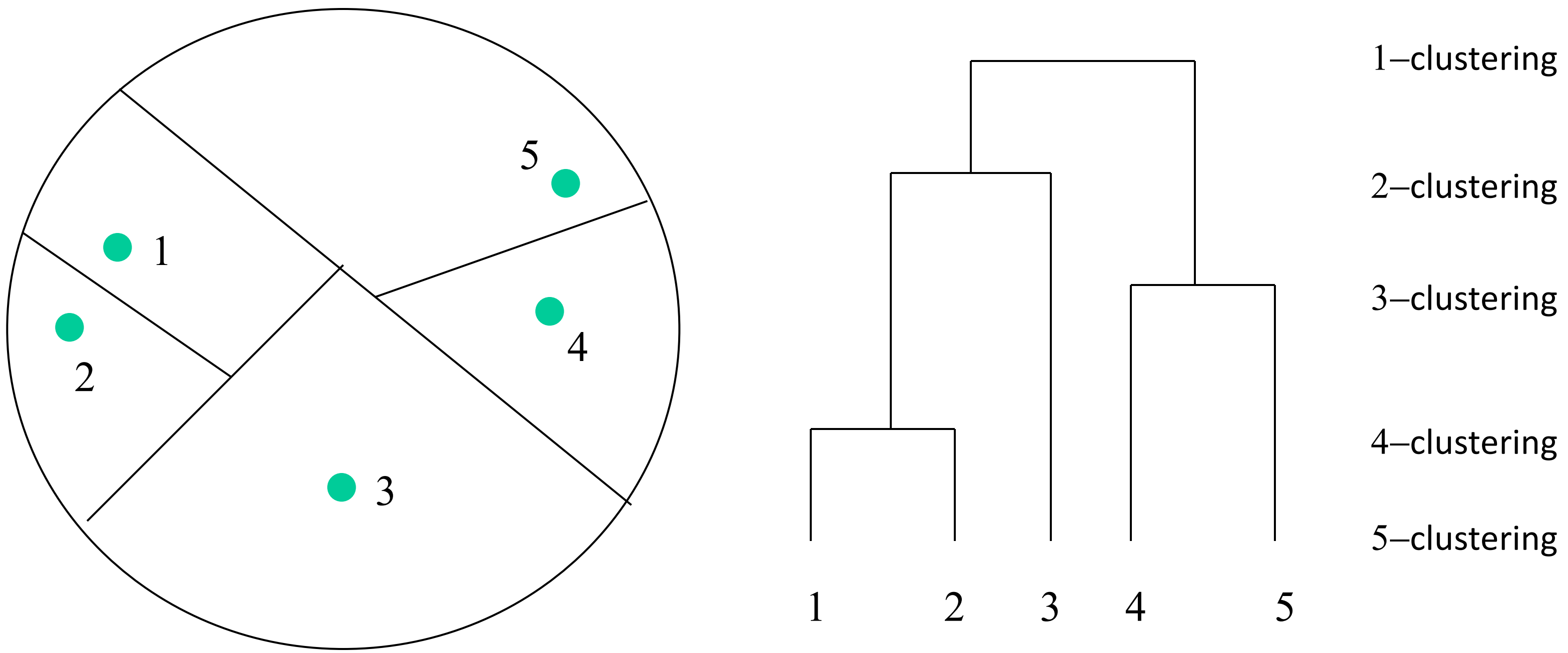
Different-cluster constraint (cannot-link)

Hierarchical clustering

- **Agglomerative**
  - **bottom up** approach: elements start as individual clusters & clusters are merged as one moves up the hierarchy
- **Divisive**
  - **top down** approach: elements start as a single cluster & clusters are split as one moves down the hierarchy

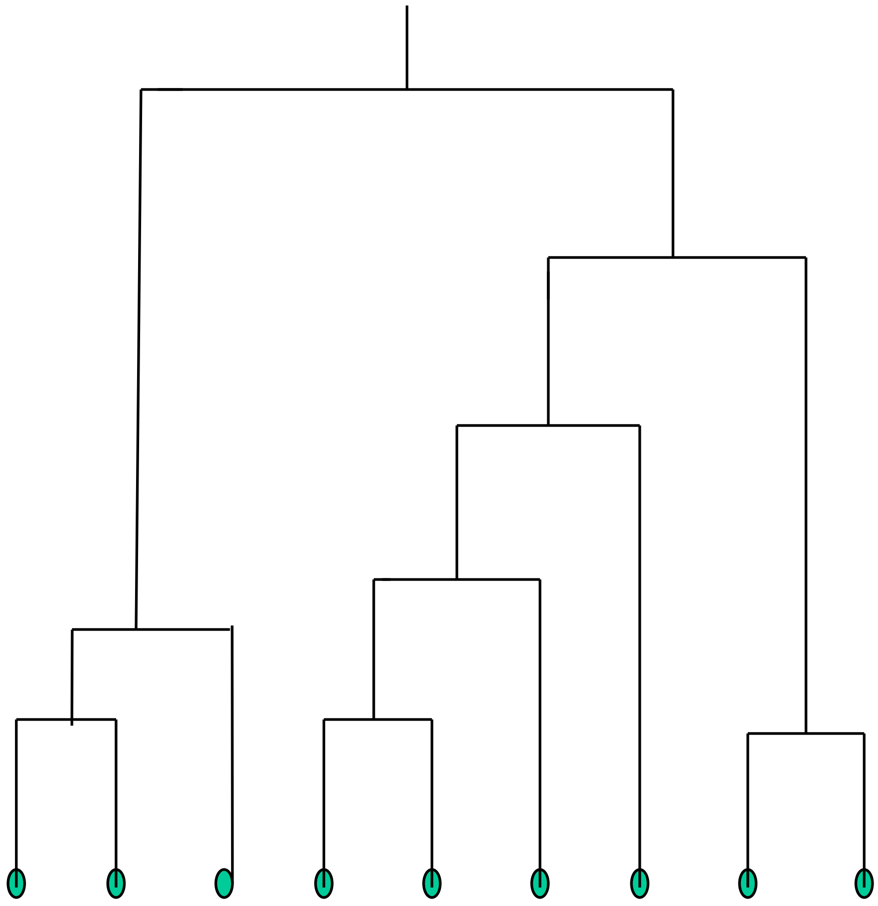
Hierarchical Clustering

Recursive partitioning/merging of a data set



Dendogram

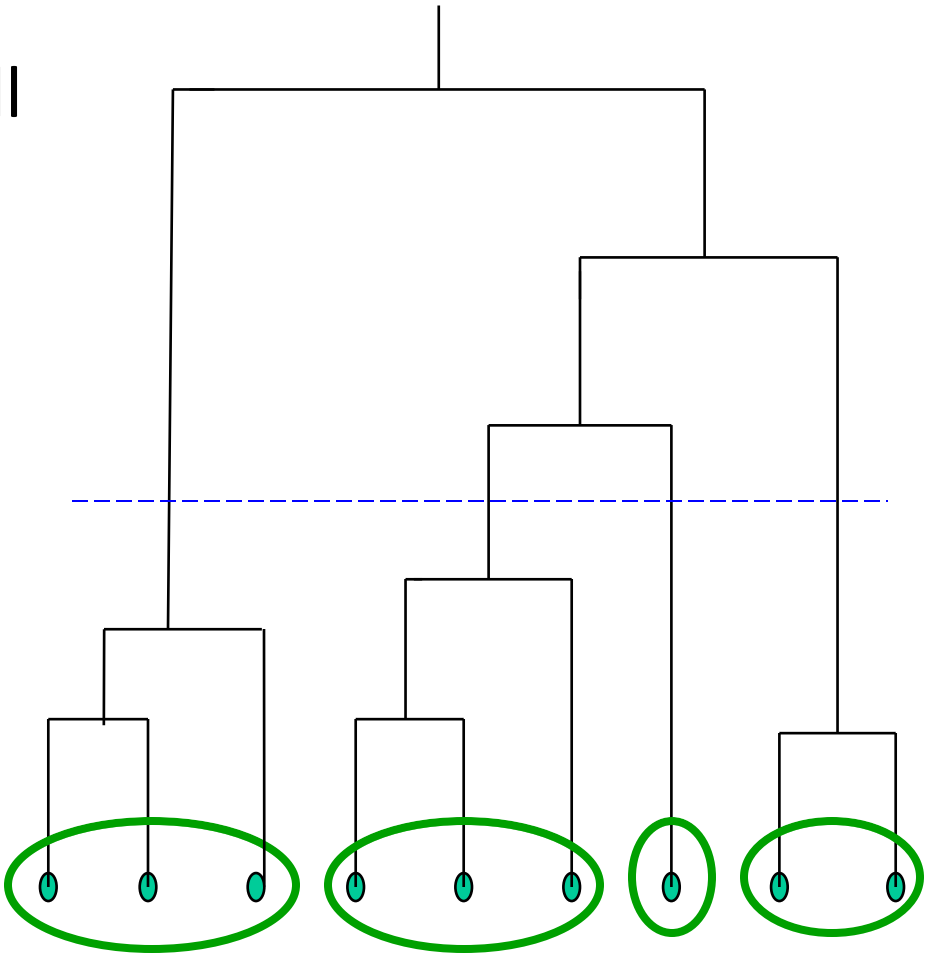
- Tree structure representing all data partitionings
- Constructed as clustering proceeds



Nine items

Dendogram

- Tree structure representing all data partitionings
- Constructed as clustering proceeds
- Get a K-clustering by looking at **connected** components at any given level
- Frequently binary dendo-grams, but n-ary ones easy to obtain with minor algorithm changes



Four clusters

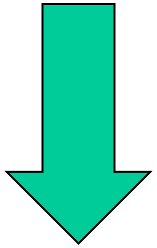
Hierarchical clustering advantages

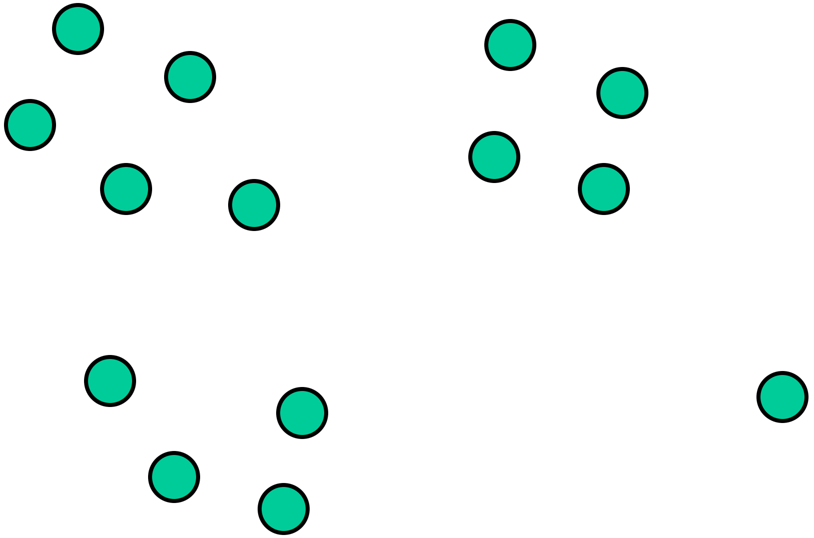
- Need not specify number of clusters
- Good for data visualization
  - See how data points interact at many levels
  - Can view data at multiple granularity levels
  - Understand how all points interact
- Specifies all of the K clusterings/partitions

Divisive hierarchical clustering

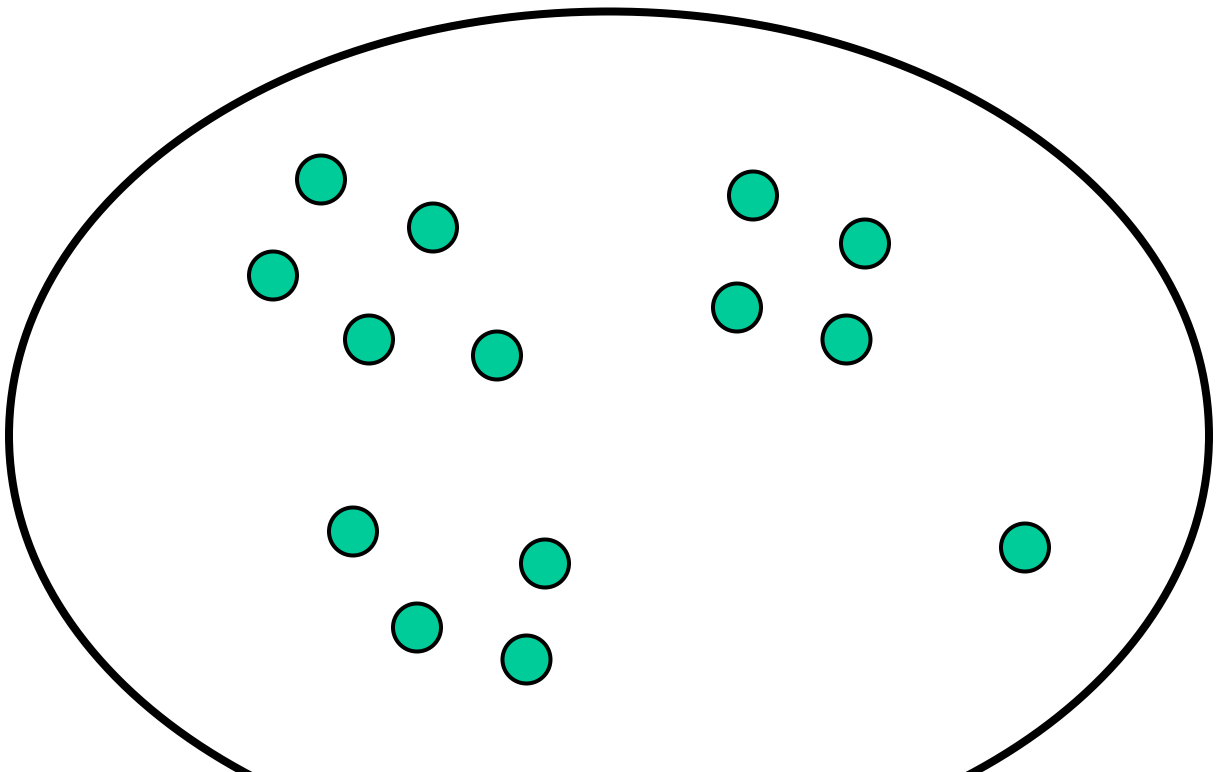
- Top-down
- Finding best partitioning of data generally exponential in time
- Common approach:
  - Let **C** be a set of clusters
  - Initialize **C** to be a one-clustering of data
  - While there exists a cluster *c* in **C**
    - remove *c* from **C**
    - partition *c* into 2 clusters (c_1 and c_2 *)* using a flat clustering algorithm (e.g., k-means
    - Add to c_1 and c_2 **C**
- Bisecting k-means

Divisive clustering



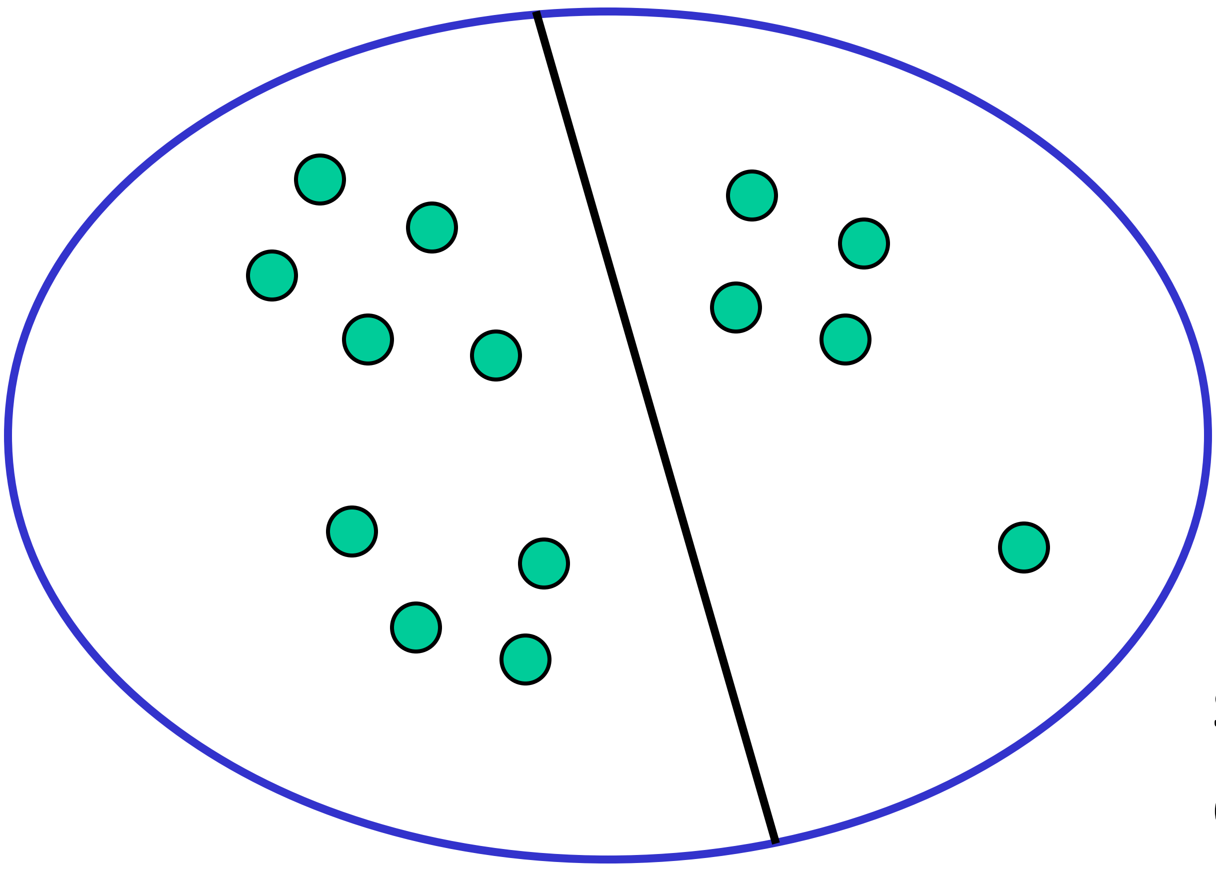


Divisive clustering



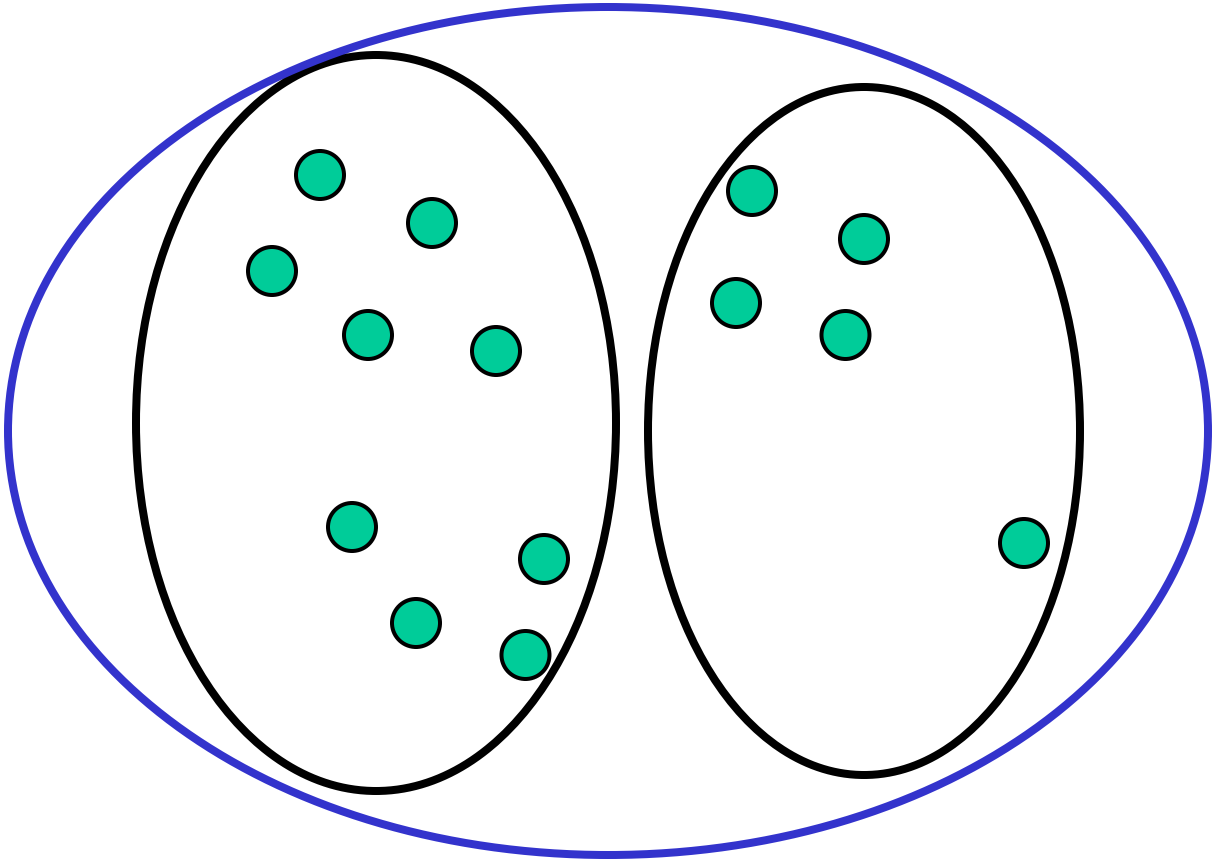
start with one cluster

Divisive clustering

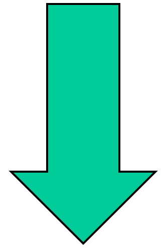


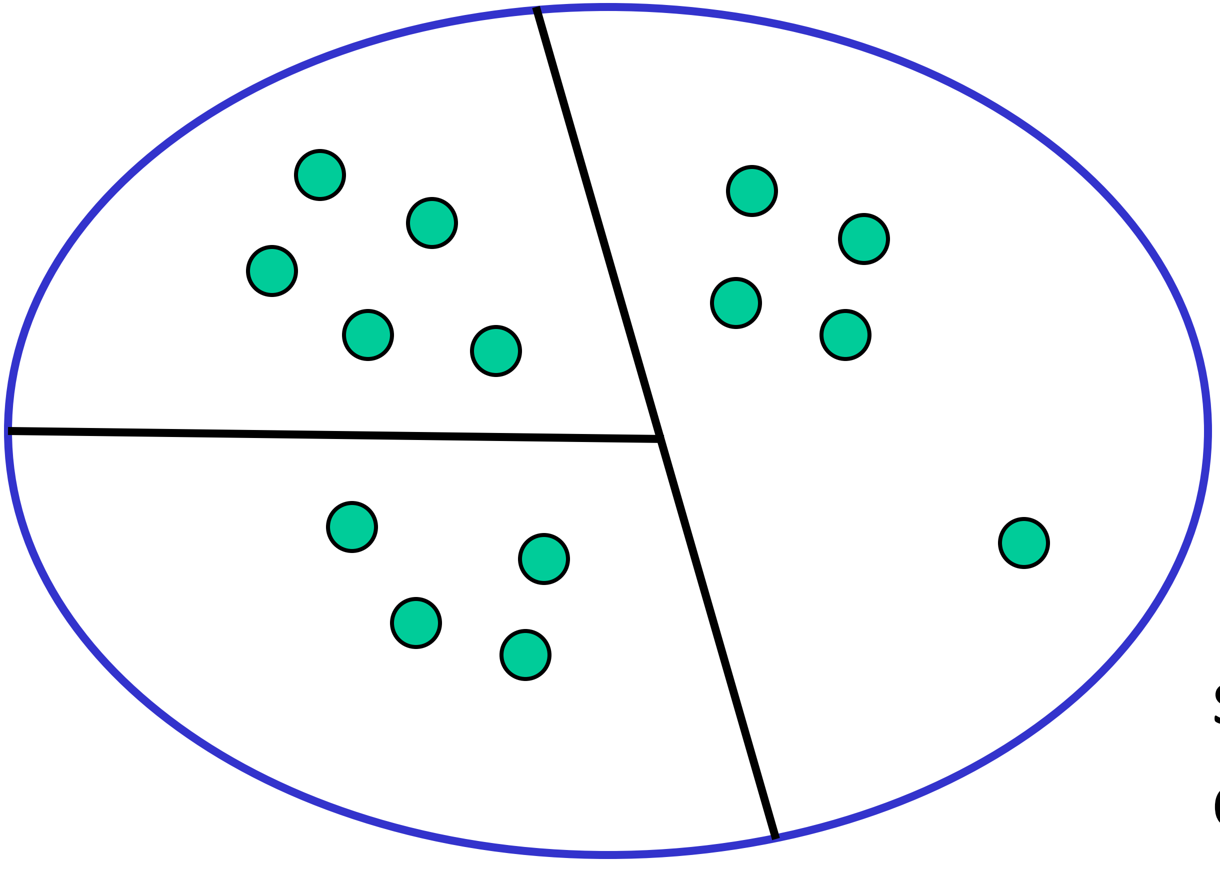
split using flat clustering, e.g., Kmeans

Divisive clustering



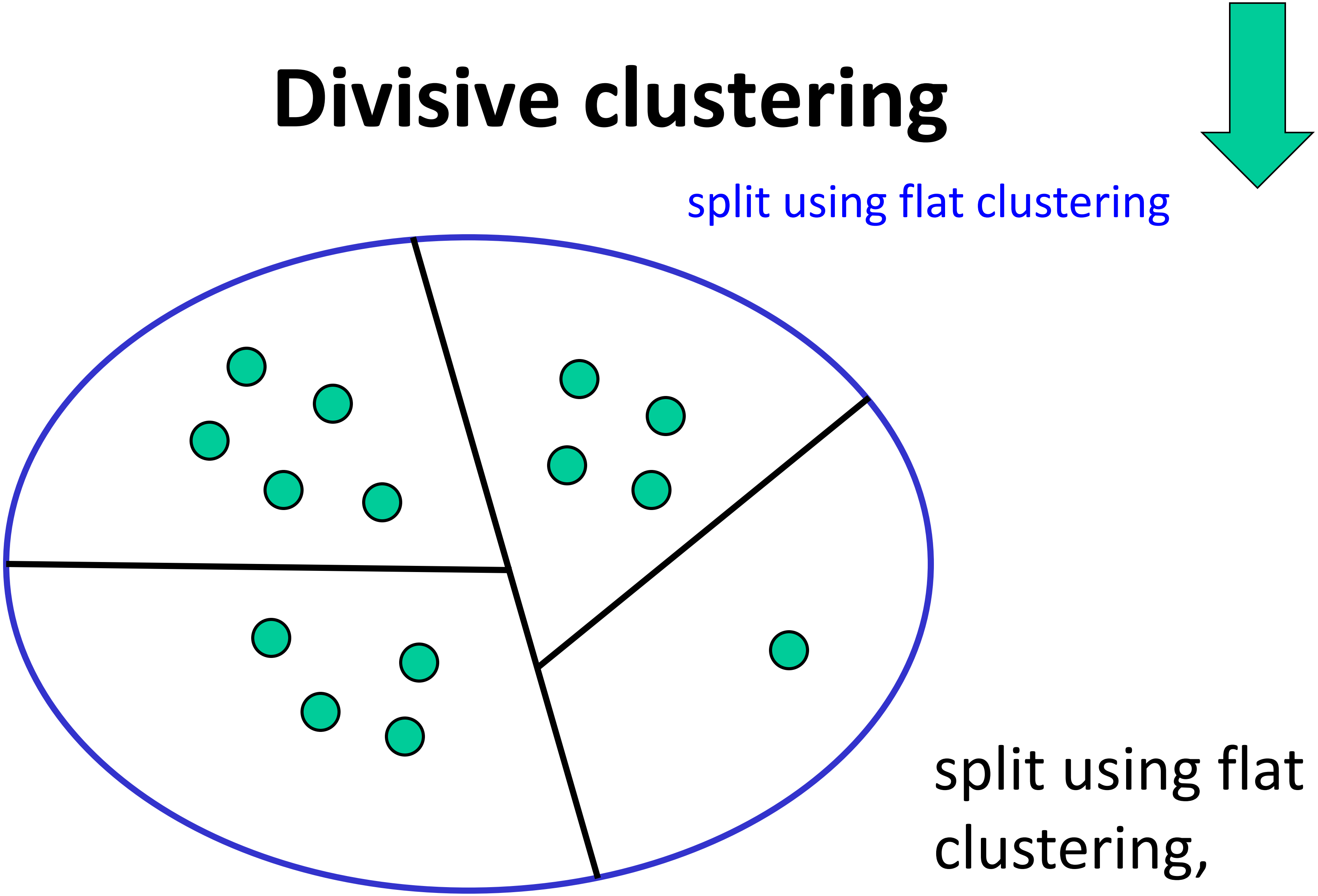
Divisive clustering





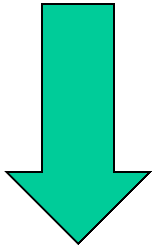
split using flat clustering, e.g., Kmeans

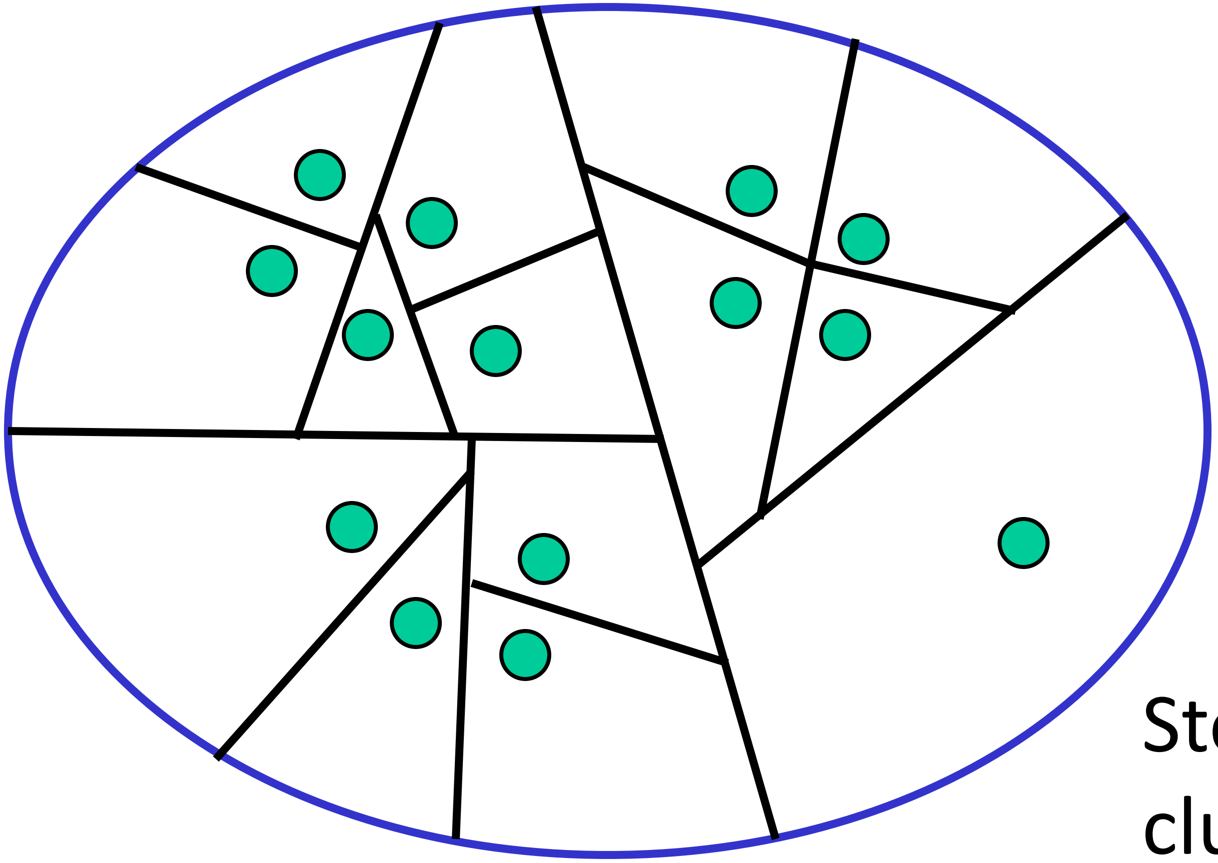
Divisive clustering



split using flat clustering, e.g., Kmeans

Divisive clustering





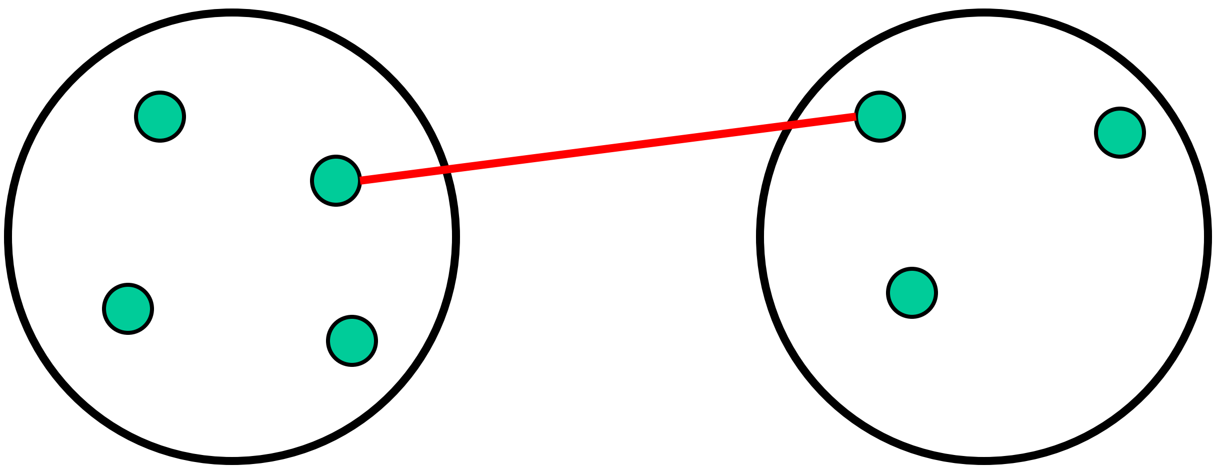
Stop when clusters reach some constraint

Hierarchical Agglomerative Clustering

- Let **C** be a set of clusters
- Initialize **C** to all points/docs as separate clusters
- While **C** contains more than one cluster
  - find c_1 and c_2 in **C** that are **closest together**
  - remove c_1 and c_2 from **C**
  - merge c_1 and c_2 and add resulting cluster to **C**
- Merging history 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)

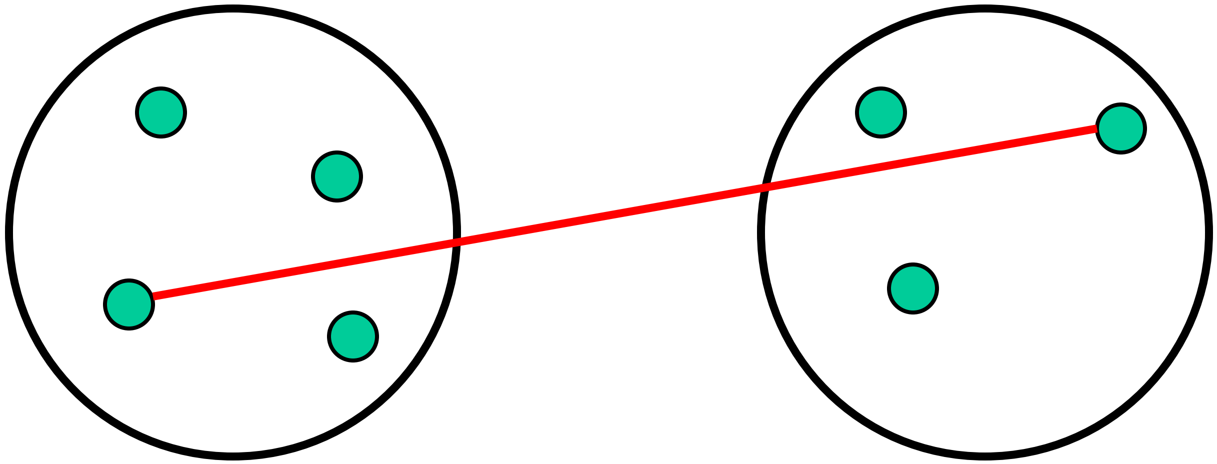


$$\max_{l \in L, r \in R} sim(l, r)$$

Distance between clusters

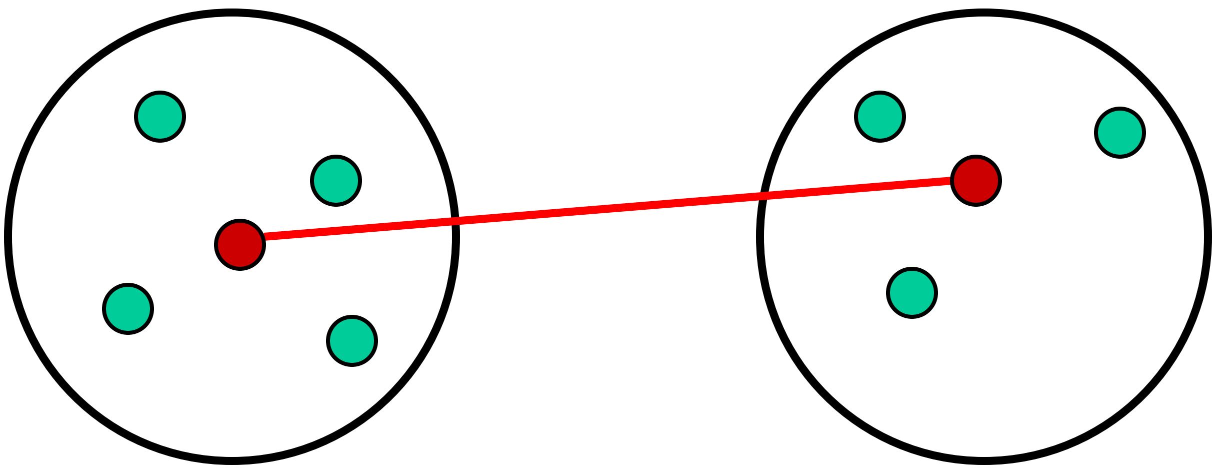
Complete-link: Similarity of the "furthest" points, the *least* similar

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



Distance between clusters

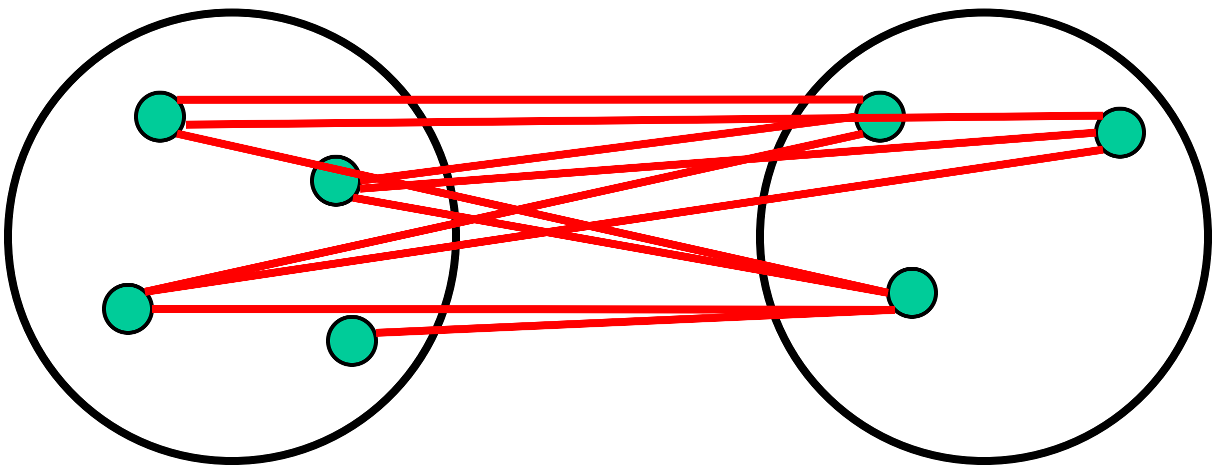
Centroid: Clusters whose centroids (centers of gravity) are the most similar



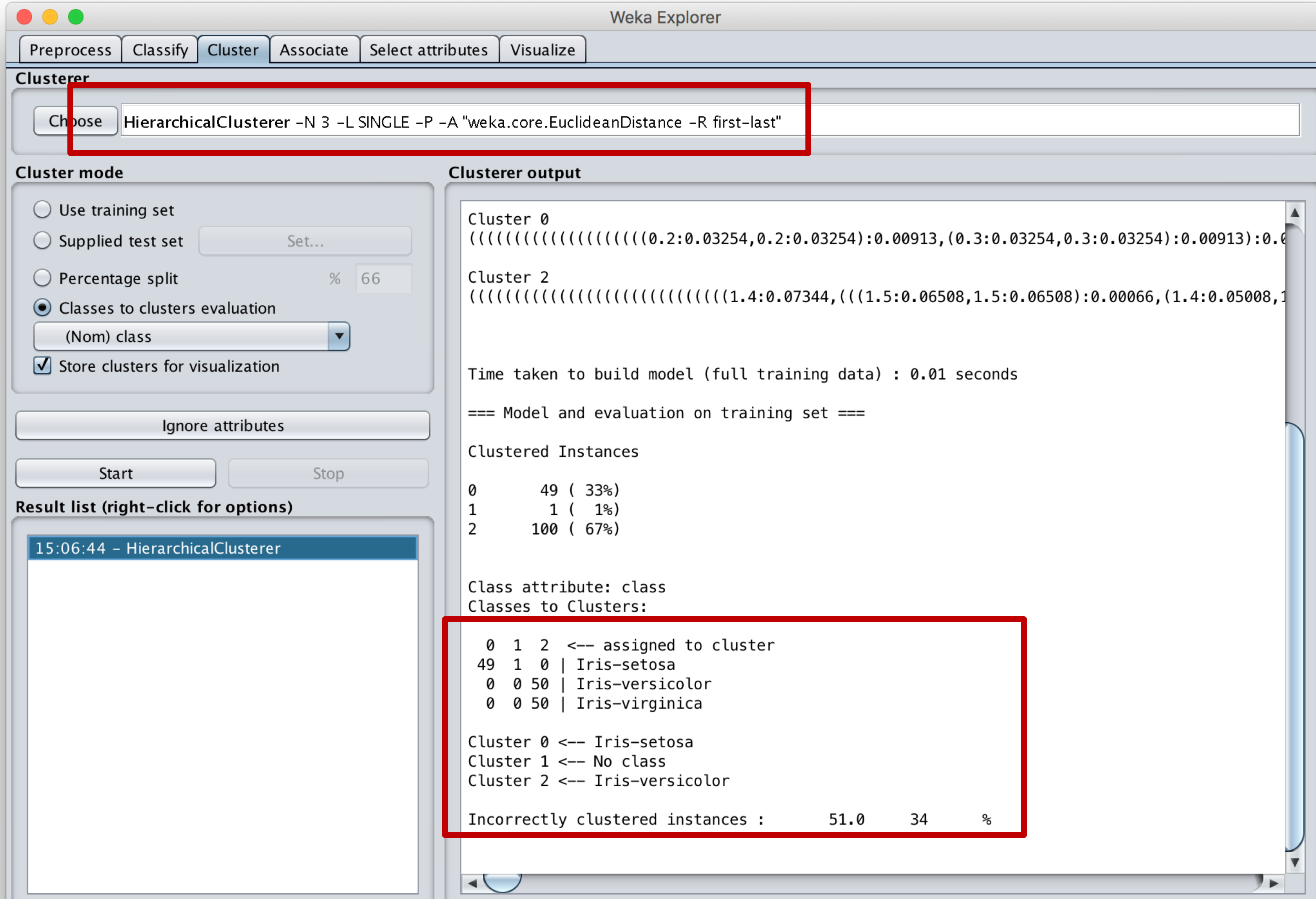
$$\left\| \mu(L) - \mu(R) \right\|^2$$

Distance between clusters

Average-link: Average similarity between all pairs of elements



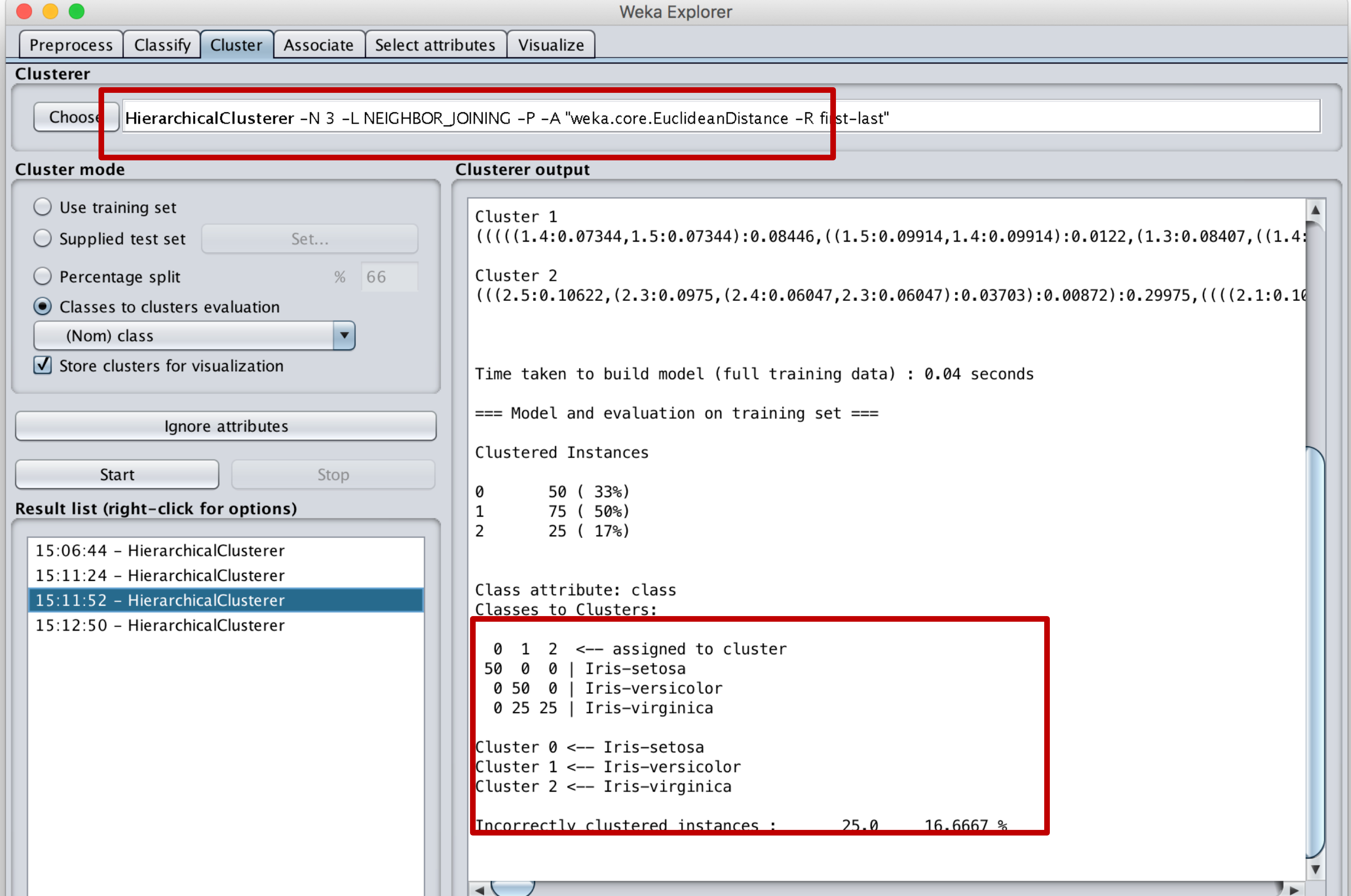
$$\frac{1}{|L| \cdot |R|} \sum_{x \in L, y \in R} \|x - y\|^2$$



Using default **LINK** cluster distance measure gives bad results

Knowing when to stop

- A general issue for hierarchical clustering is knowing when to stop merging/splitting a cluster
- We may have a problem specific desired range of clusters (e.g., 3-6)
- There are some general metrics for assessing the quality of a cluster
- There are also domain specific heuristics for cluster quality



Using **WARD** cluster distance measure improves results