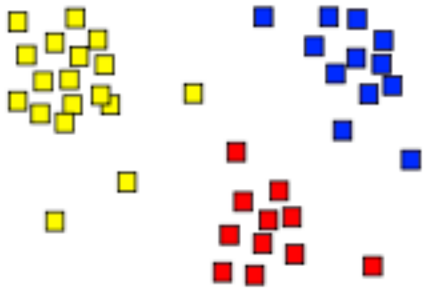
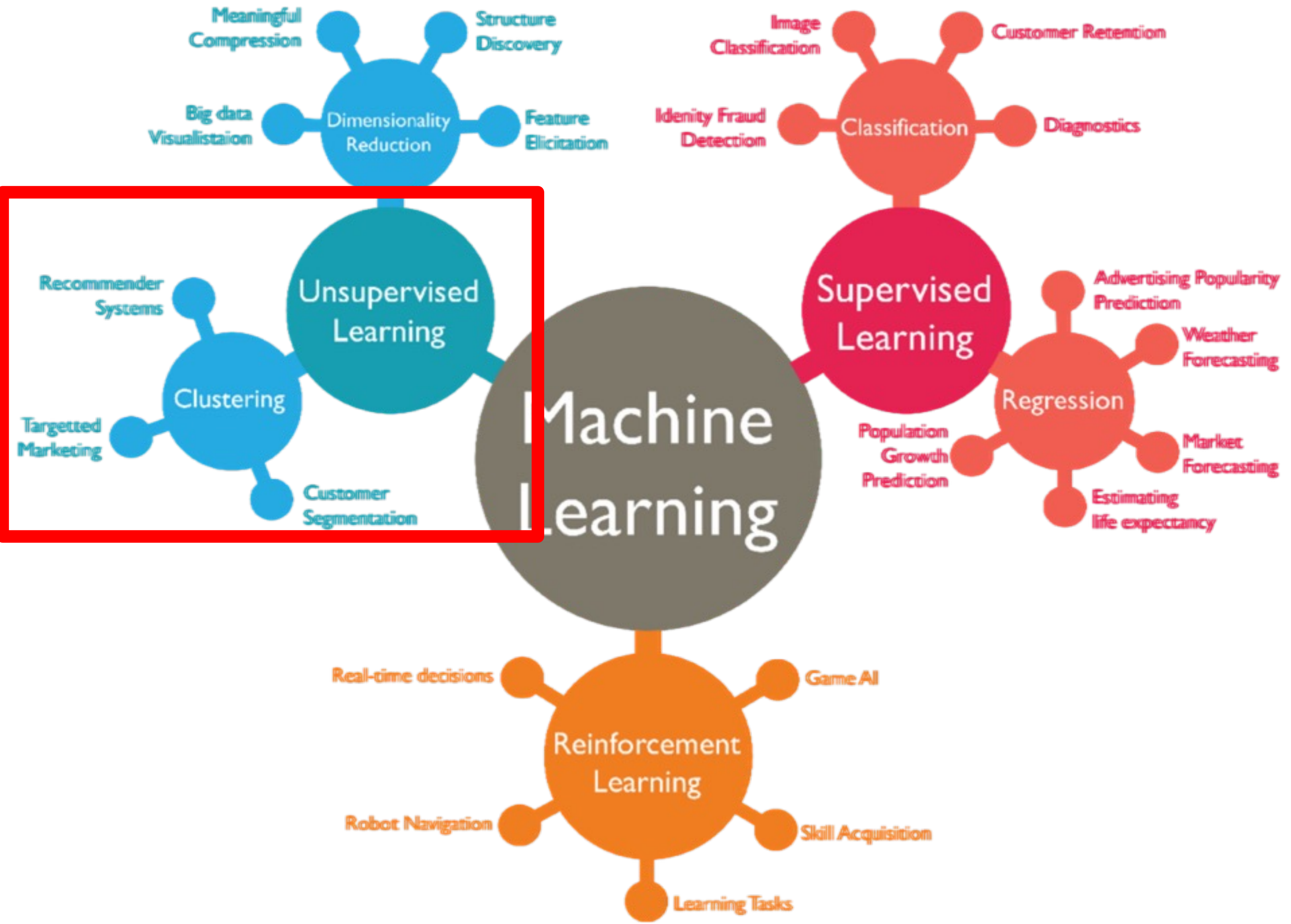


Unsupervised Learning: Clustering

Introduction and Simple K-means

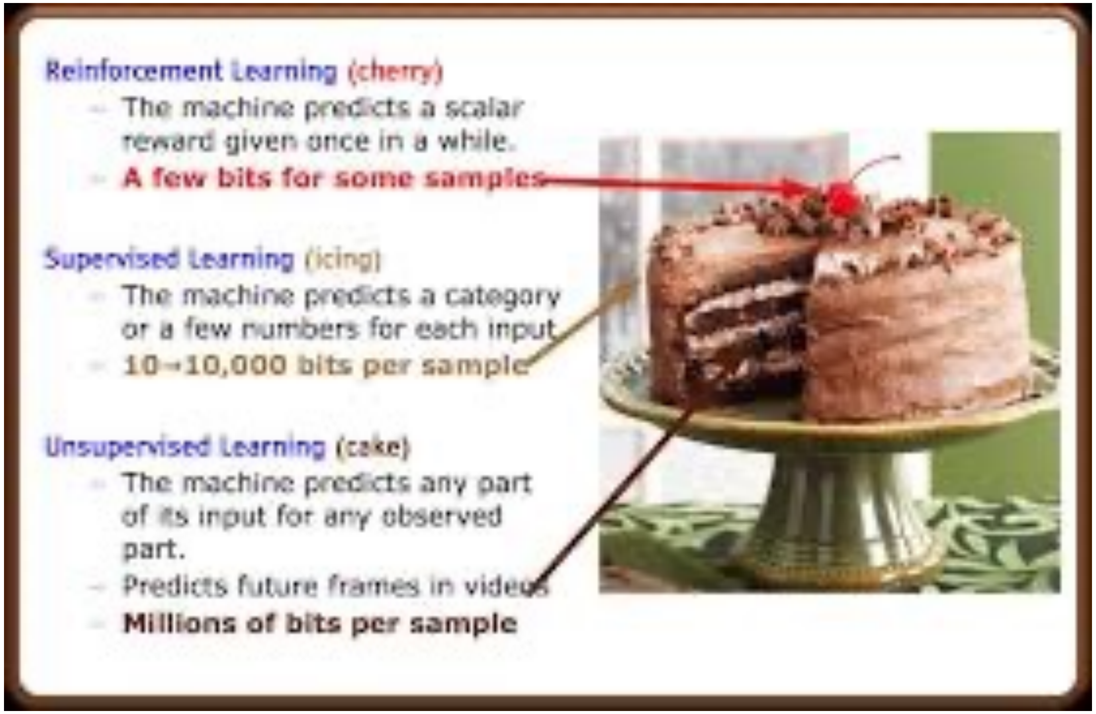
Some material adapted from slides by Andrew Moore, CMU


Machine Learning
Unsupervised Learning
Dimensionality Reduction
Meaningful Compression
Structure Discovery
Big data Visualistaion
Feature Elicitation
Clustering
Recommender Systems
Targetted Marketing
Customer Segmentation
Supervised Learning
Classification
Image Classification
Customer Retention
Idenity Fraud Detection
Diagnostics
Regression
Advertising Popularity Prediction
Weather Forecasting
Market Forecasting
Estimating life expectancy
Population Growth Prediction
Reinforcement Learning
Real-time decisions
Game AI
Robot Navigation
Skill Acquisition
Learning Tasks

Yann LeCun on Unsupervised Learning

"Most of human and animal learning is *unsupervised learning*. If intelligence was a cake, unsupervised learning would be the cake, *supervised learning* would be the icing on the cake, and *reinforcement learning* would be the cherry on the cake. … We know how to make the icing and the cherry, but we don't know how to make the cake. We need to solve the unsupervised learning problem before we can even think of getting to true AI."*

--

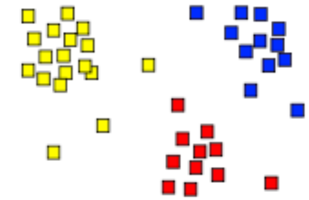


* Yann LeCun (*Head of Facebook AI, NYU CS Prof.)* on AlphaGo's success and AI, 2016

Unsupervised Learning

- Supervised learning used labeled data pairs (x, y) to learn a function $f : X \rightarrow y$
- 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 based on their **similarity**
- It can be used for **knowledge discovery**

Clustering algorithms



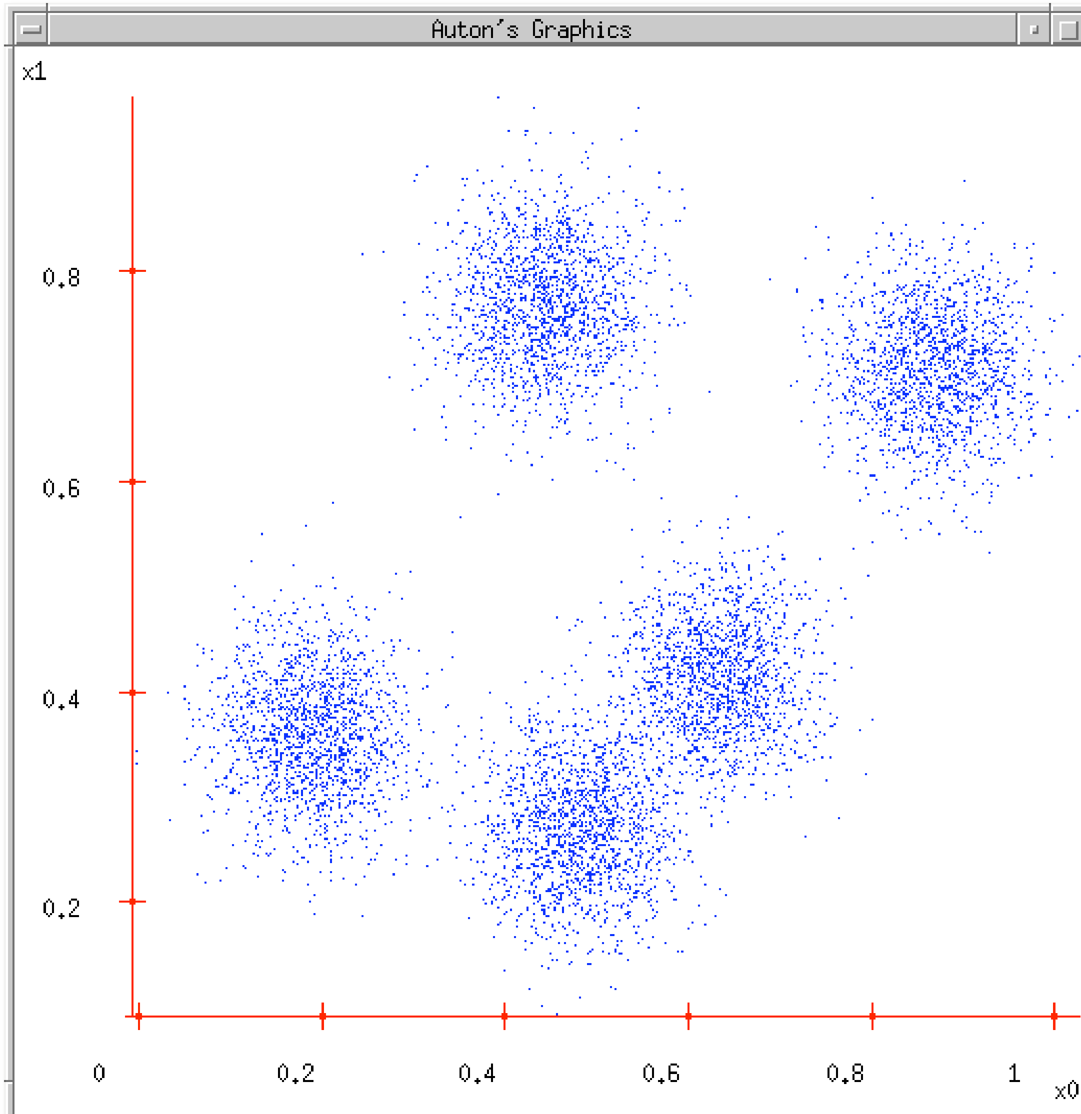
- Many clustering algorithms
- Clustering typically done using a **distance measure** defined between instances or points
- Distance defined by instance **feature space**, so it works with numeric features
  - Requires encoding of categorial values; may benefit from normalization
- We'll look at three popular approaches
  1. Centroid-based clustering (e.g., Kmeans)
  2. Hierarchical clustering
  3. DBSCAN

Clustering Data

Given a collection of points (x,y), group them into one or more clusters based on their distance from one another

How many clusters are there?

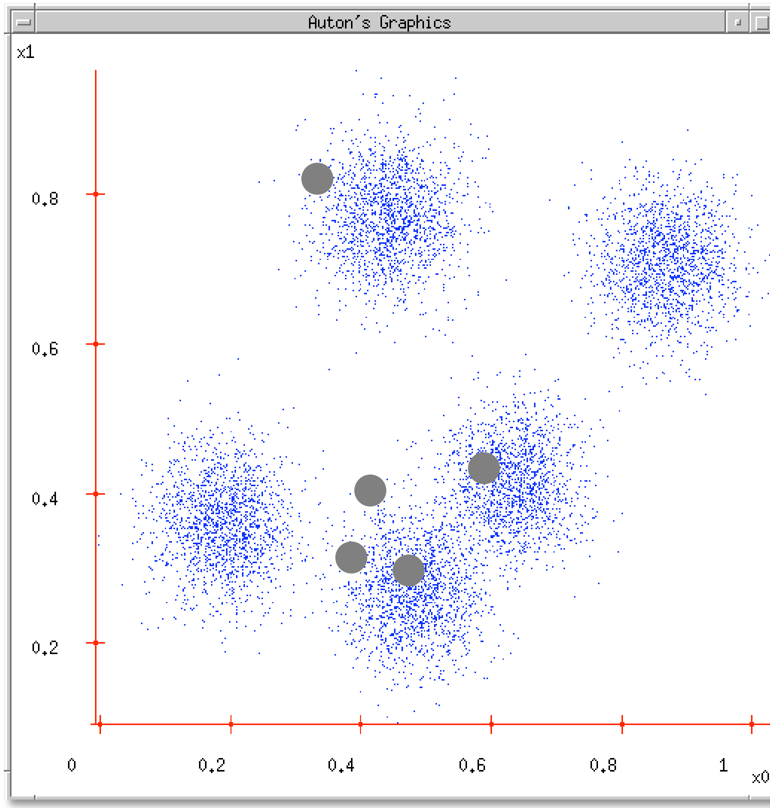
How can we find them



(1) K-Means Clustering

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

k = 5



k-MEANS CLUSTERING

1. k centerpoints are randomly initialized.
2. Observations are assigned to the closest centerpoint.
3. Centerpoints are moved to the center of their members.
4. Repeat steps 2 and 3 until no observation changes membership in step 2.

Chris Albon

distance, centroids

- Distance between points (X_0,Y_0,Z_0) and (X_1,Y_1,Z_1) is just $\mathrm{sqrt}((X_0 - X_1)^2 + (Y_0 - Y_1)^2 + (Z_0 - Z_1)^2)$
- In numpy: distance between two points

```
>>> import numpy as np
>>> p1 = np.array([0,-2,0,1]) ; p2 = np.array([0,1,2,1]))
>>> np.linalg.norm(p1 - p2)
3.605551275463989
```

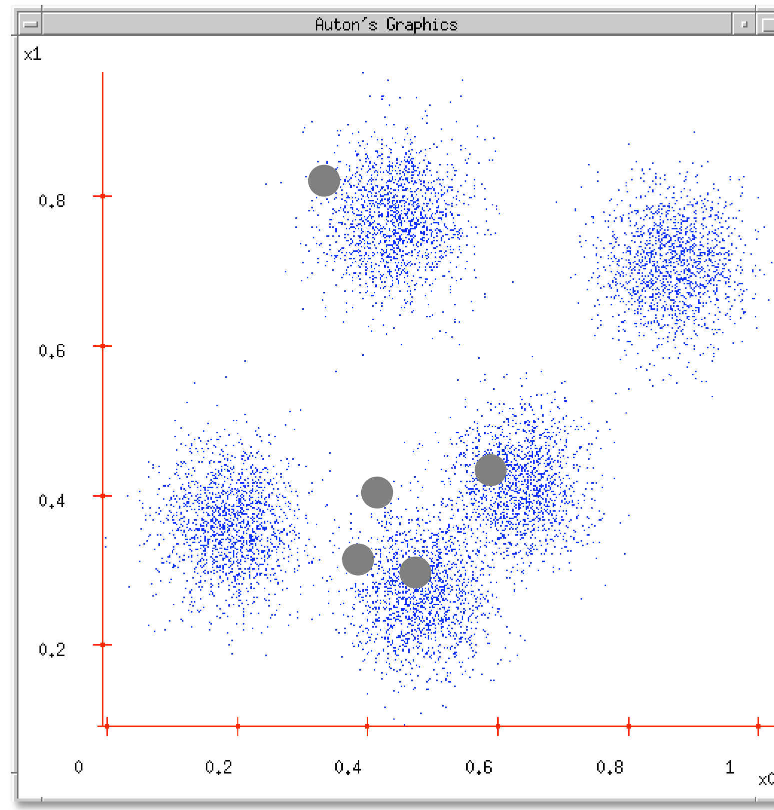
- Computing centroid of set of points easy

```
>>> points = np.array([[1,2,3], [2,1,1], [3,1,0]])    # 3D points
>>> centroid = np.mean(points, axis=0)              # mean across columns
>>> centroid
array([2.0, 1.33, 1.33])
```

(1) K-Means Clustering

- Randomly choose k cluster center locations, aka **centroids**
- Loop until convergence
  - assign one 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

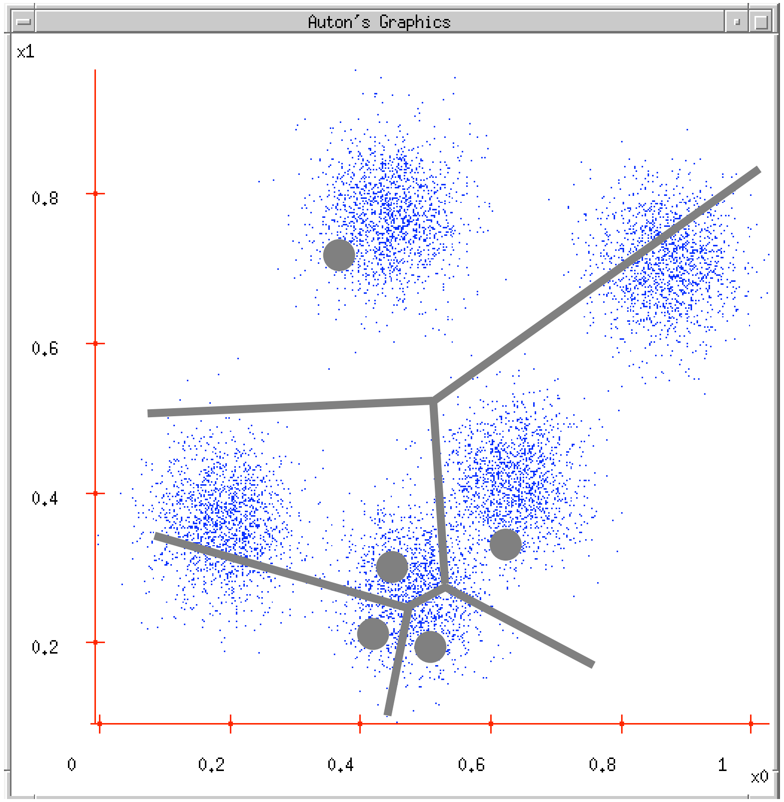
k = 5



K-Means Clustering

K-Means (k, data)

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

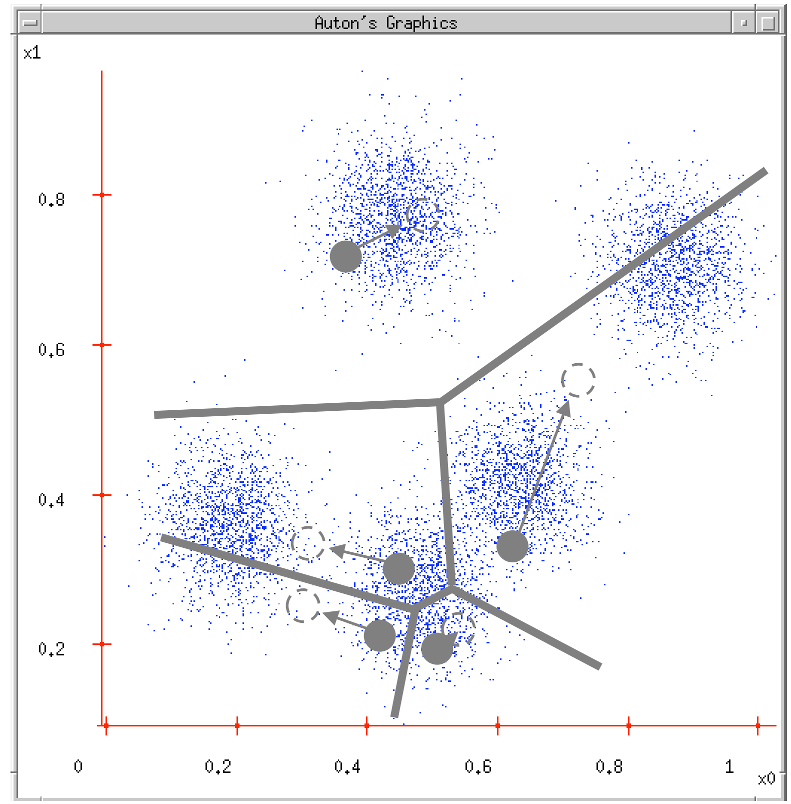


[veroni diagram](): add lines for regions of points closest to each centroid

K-Means Clustering

K-Means (k, data)

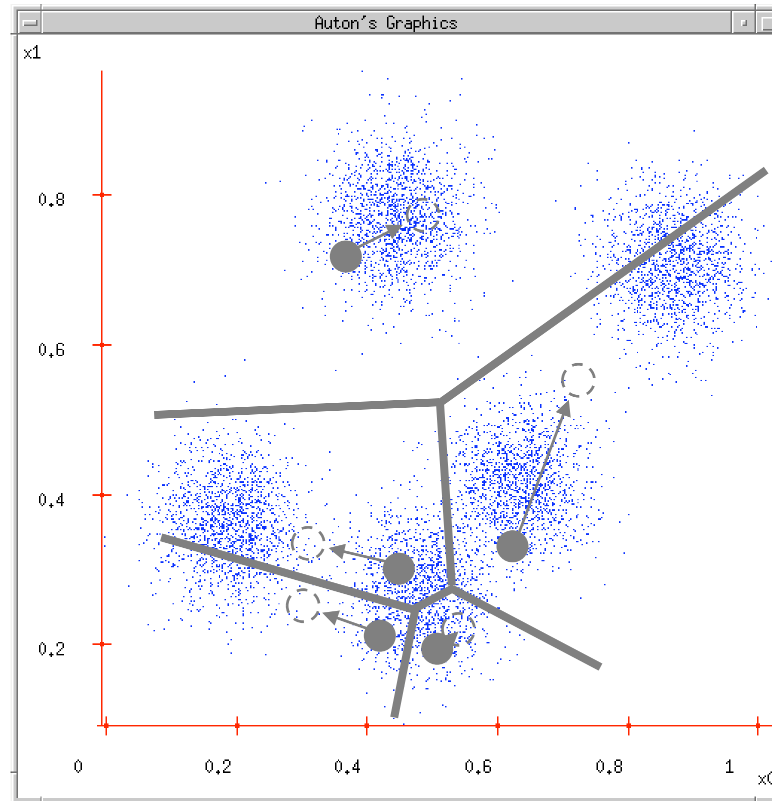
- Randomly choose k cluster center locations (centroids)
- **Loop until convergence**
  - Assign each point to the cluster of closest centroid
  - **Re-estimate cluster centroids based on data assigned to each**
- Convergence: no point is assigned to a different cluster



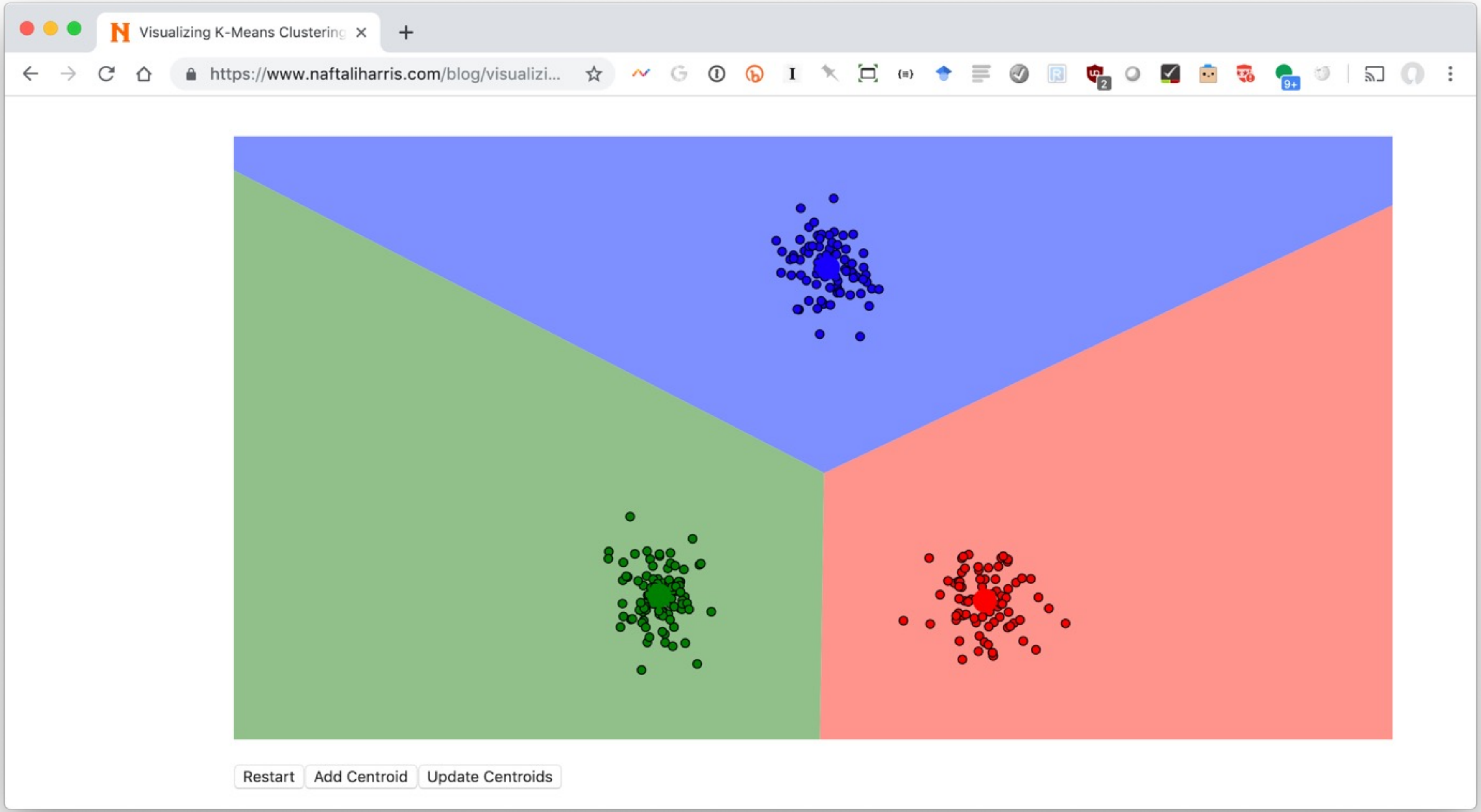
K-Means Clustering

K-Means (k, data)

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



Visualizing k-means



Visualizing k-means

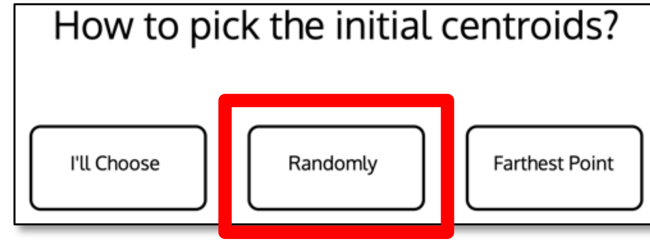
Visualizing k-means

Interactively experiment with K-means clustering

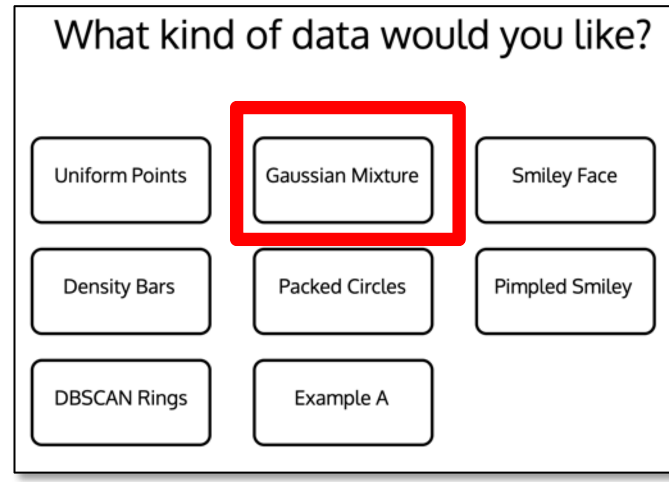
1. Three ways to assign positions of initial centroids
2. Eight ways to generate data points to be clustered
3. You choose the value of k when adding centroids
4. Then walk through the iterations of the k-means algorithm

It can also demonstrate the DBSCAN clustering algorithm

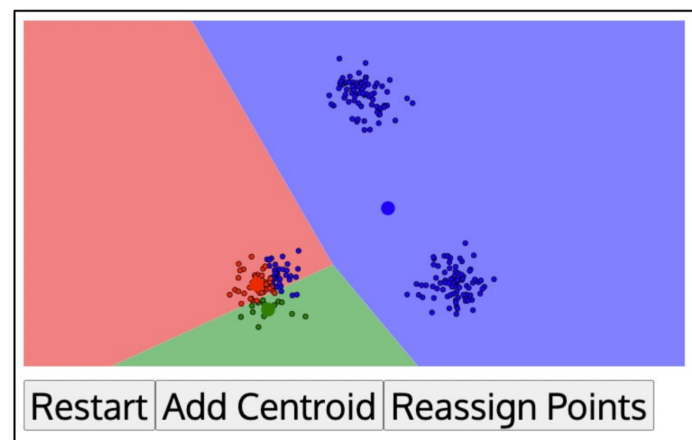
1



2

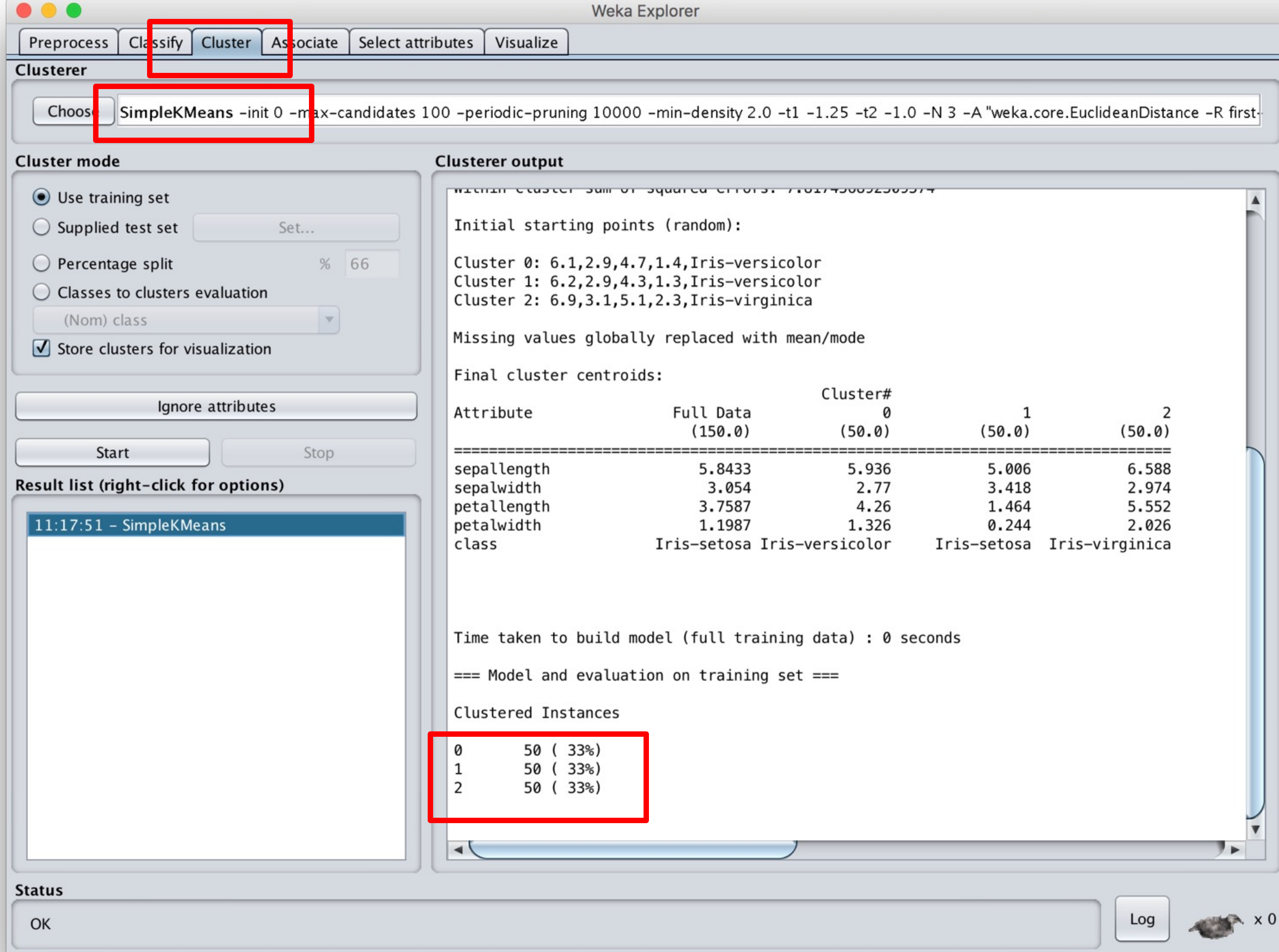


3

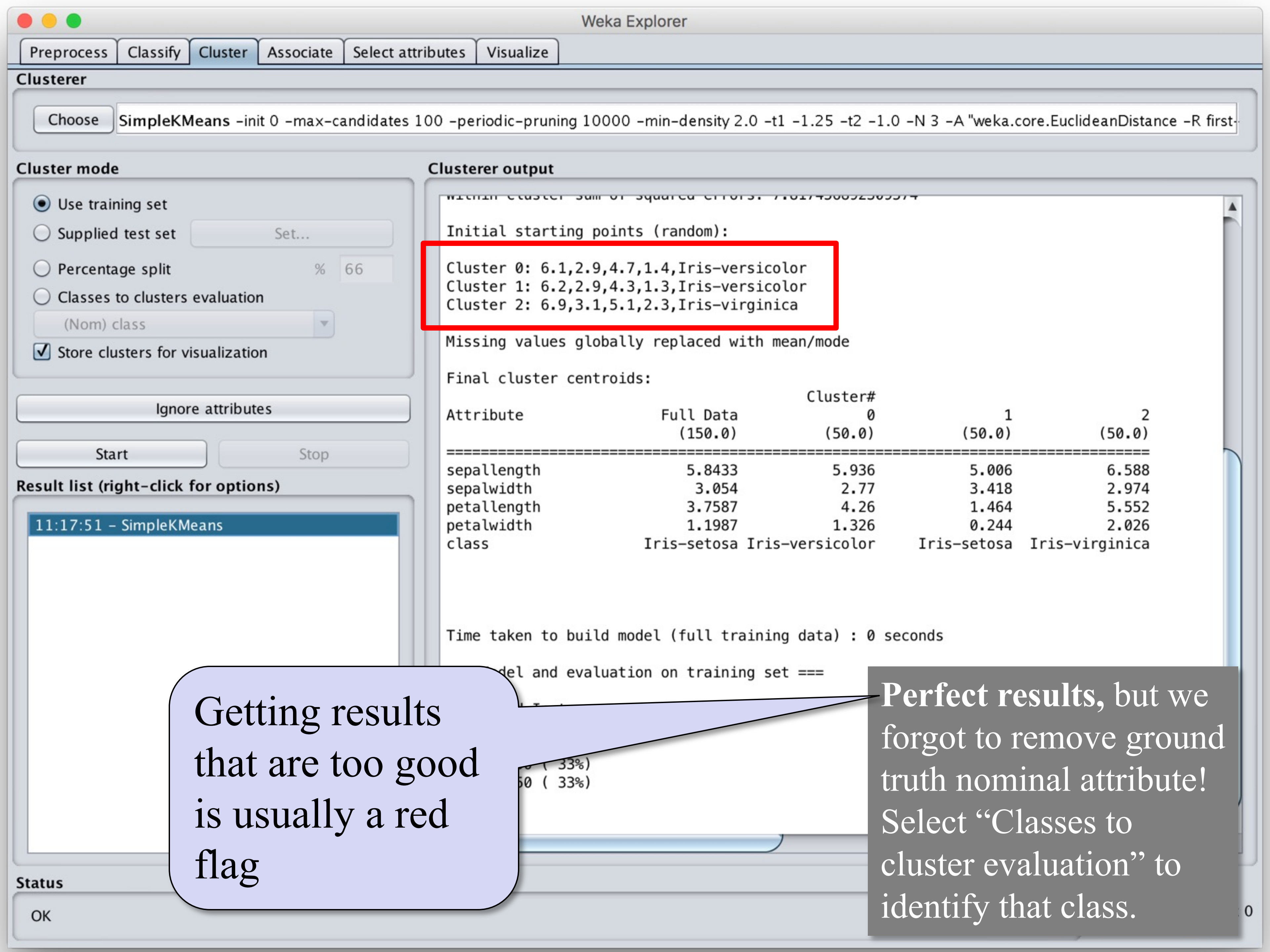


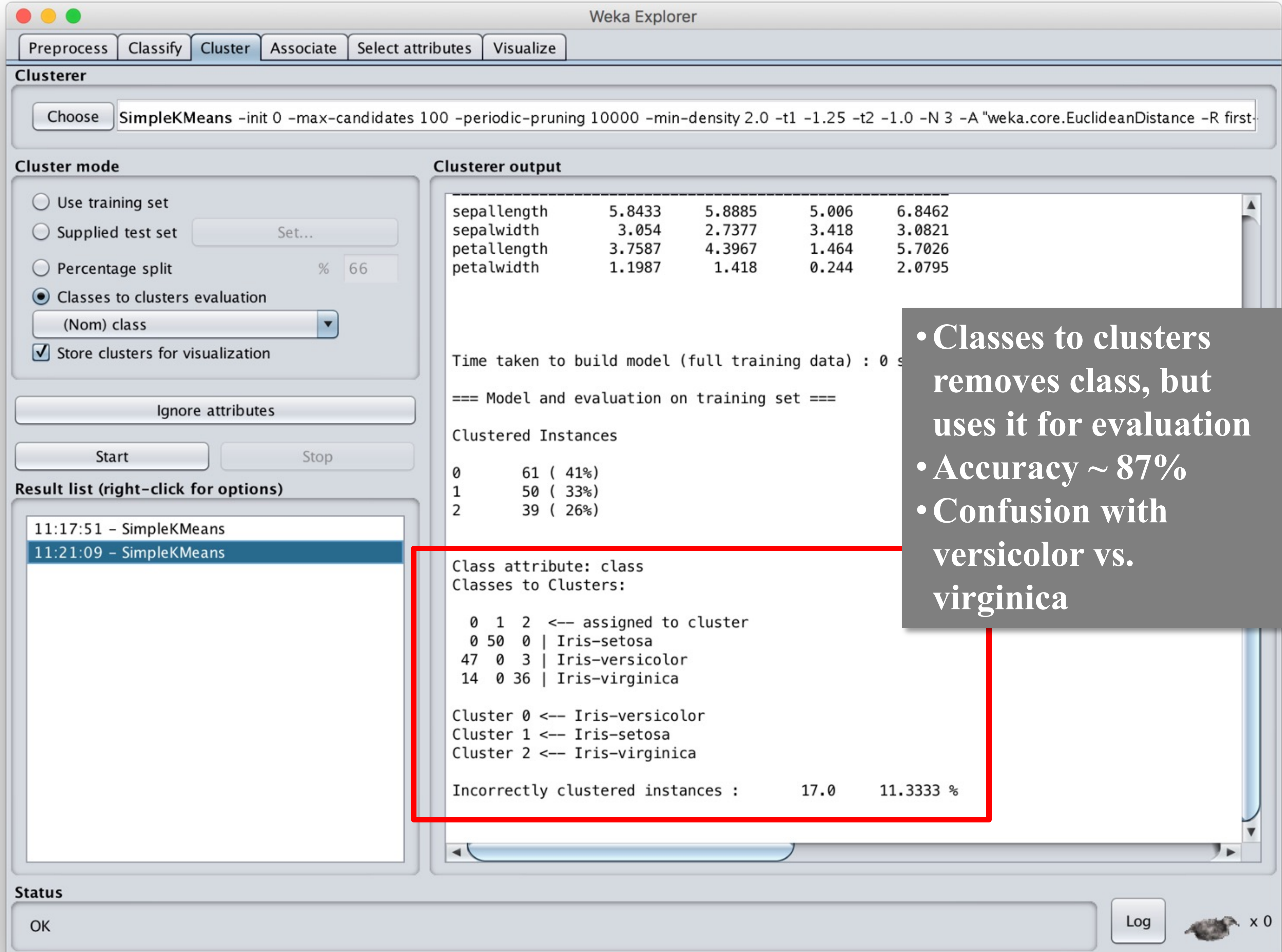
Clustering the Iris Data

- Let’s try using unsupervised clustering on the Iris Data
- First on Weka
- Then using scikit learn on Colab



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
Status
OK
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
del and evaluation on training set ===
Getting results that are too good is usually a red flag
Perfect results, but we forgot to remove ground truth nominal attribute! Select "Classes to cluster evaluation" to identify that class.



- Classes to clusters removes class, but uses it for evaluation
- Accuracy ~ 87%
- Confusion with versicolor vs. virginica

2.3. Clustering

Clustering of unlabeled data can be performed with the module `sklearn.cluster`.

Each clustering algorithm comes in two variants: a class, that implements the `fit` method to learn the clusters on train data, and a function, that, given train data, returns an array of integer labels corresponding to the different clusters. For the class, the labels over the training data can be found in the `labels_` attribute.

Input data

One important thing to note is that the algorithms implemented in this module can take different kinds of matrix as input. All the methods accept standard data matrices of shape `[n_samples, n_features]`. These can be obtained from the classes in the `sklearn.feature_extraction` module. For `AffinityPropagation`, `SpectralClustering` and `DBSCAN` one can also input similarity matrices of shape `[n_samples, n_samples]`. These can be obtained from the functions in the `sklearn.metrics.pairwise` module.

2.3.1. Overview of clustering methods

07_IrisClustering.ipynb - Colab

https://colab.research.google.c...

07_IrisClustering.ipynb

File Edit View Insert Runtime Tools Help

COMMENT SHARE

CODE TEXT CELL CELL RAM Disk EDITING

sklearn K-means clustering on Fisher's Iris dataset

```
[14] %matplotlib inline

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

```
[15] # load training data
     iris = sklearn.datasets.load_iris()
```

The Iris dataset has 150 instances:

- X floats for Sepal Length, Sepal Width, Petal Length and Petal Width *y: integer (0,1,2) representing species (Setosa, Versicolour, Virginica)

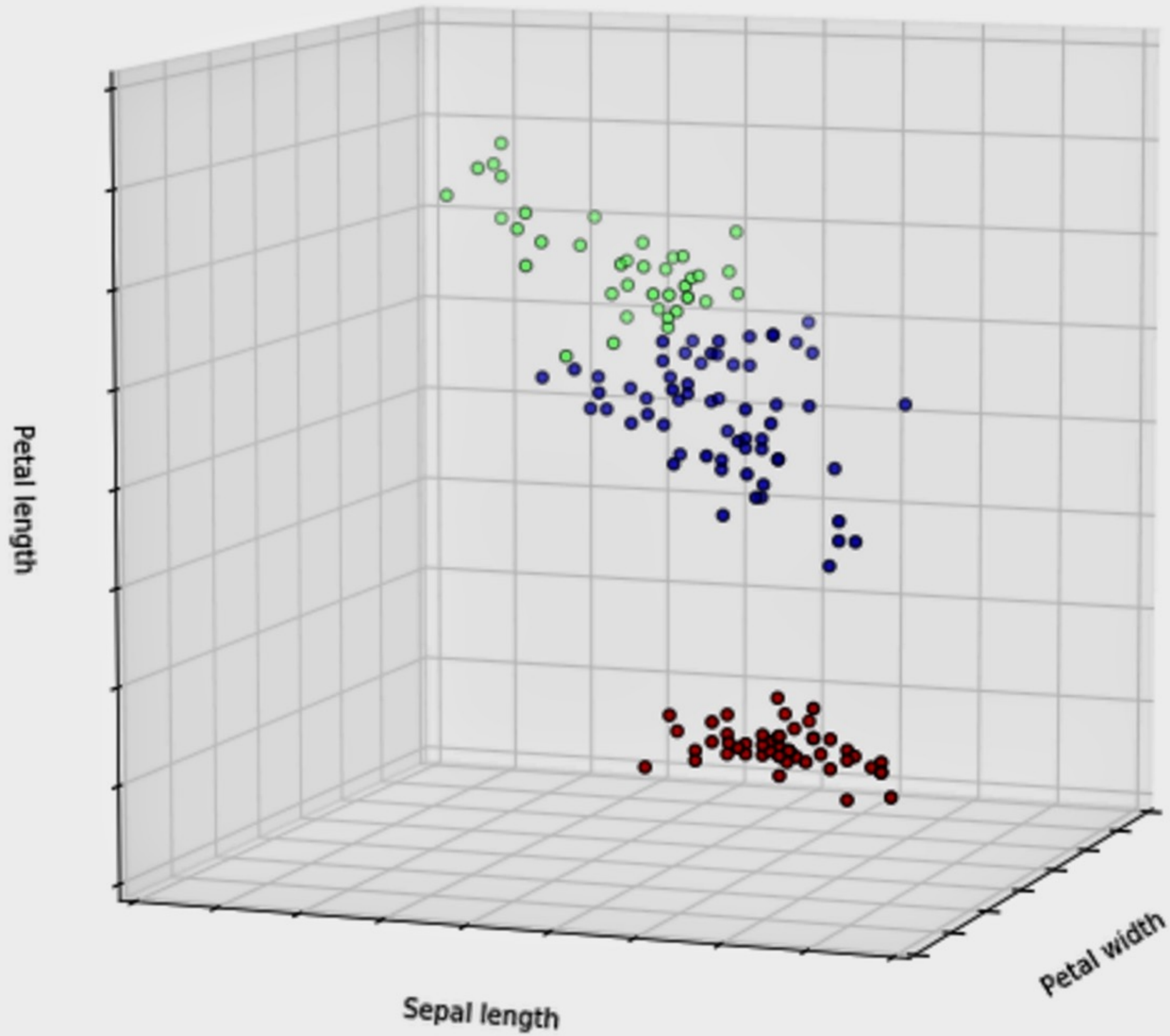
```
[20] X = iris.data
     y = iris.target
```

```
[25] # show first three rows of training instance data and the target classes
     print(X[:3])
     print(y[:3])
```

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]]
[0 0 0]
```

```
[27] # show all values for ground truth class (0,1,2)
     print(y)
```

```
[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
```

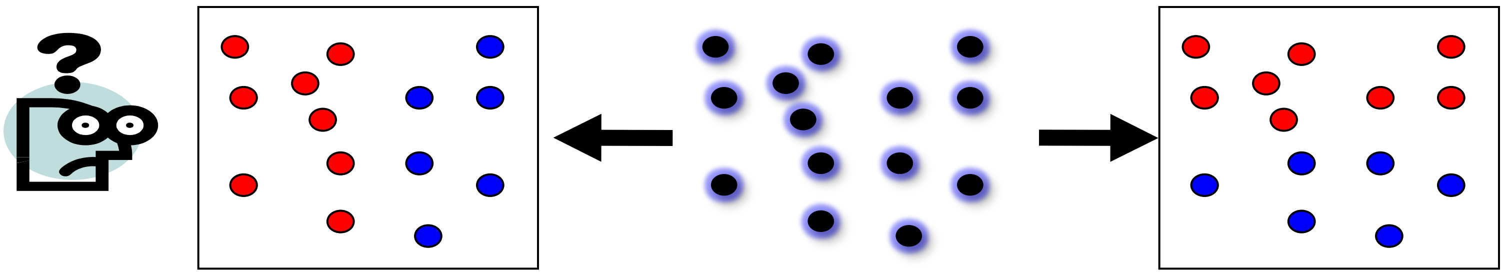

Petal length
Sepal length
Petal width

Problems with K-Means

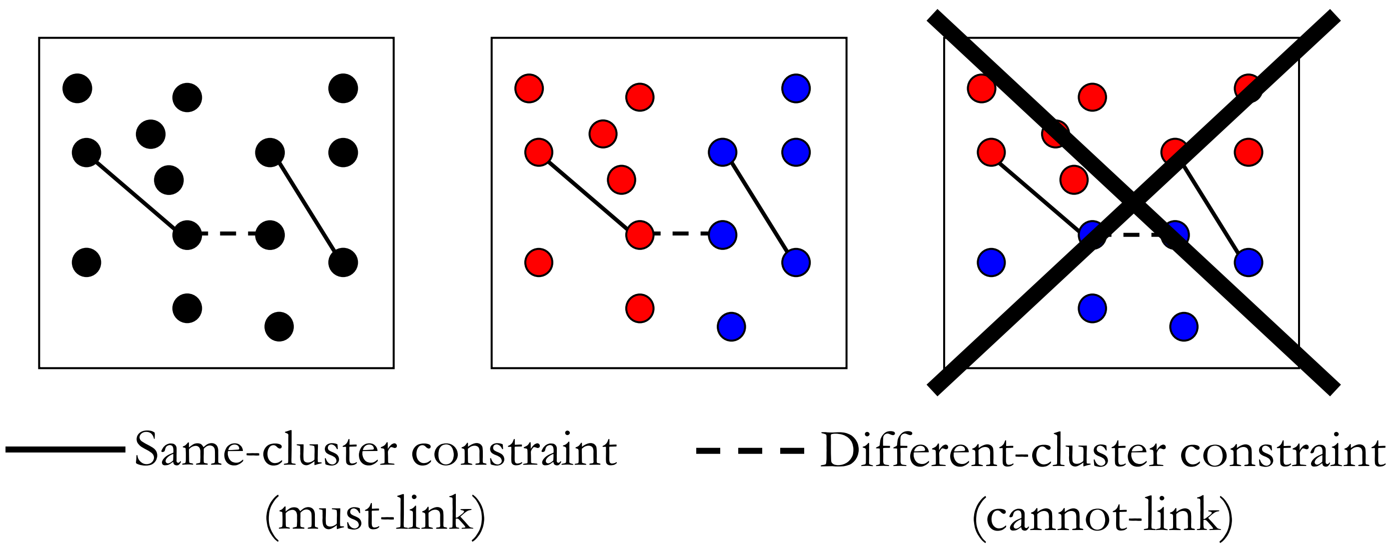
- Only works for numeric data (typically reals)
- ***Very*** sensitive to the initial points
  - **fix:** Do many runs, each with different initial centroids
  - **fix:** Seed centroids with non-random method, e.g., **farthest-first** sampling
- Sensitive to outliers
  - **fix:** identify and remove outliers
- **Must manually choose k**
  - **E.g.: find three**
  - Learn optimal k using some performance measure

Problems with K-Means

- How do you tell it which clustering you want?



- Constrained clustering technique provides hints



K-means Clustering Summary

- Clustering useful & effective for many tasks
- K-means clustering one of simplest & fastest techniques, but
  - Requires knowing how many clusters is right
  - Doesn’t handle outliers well
- There are many other clustering options
  - E.g., DBSCAN, Hierarchical clustering, …