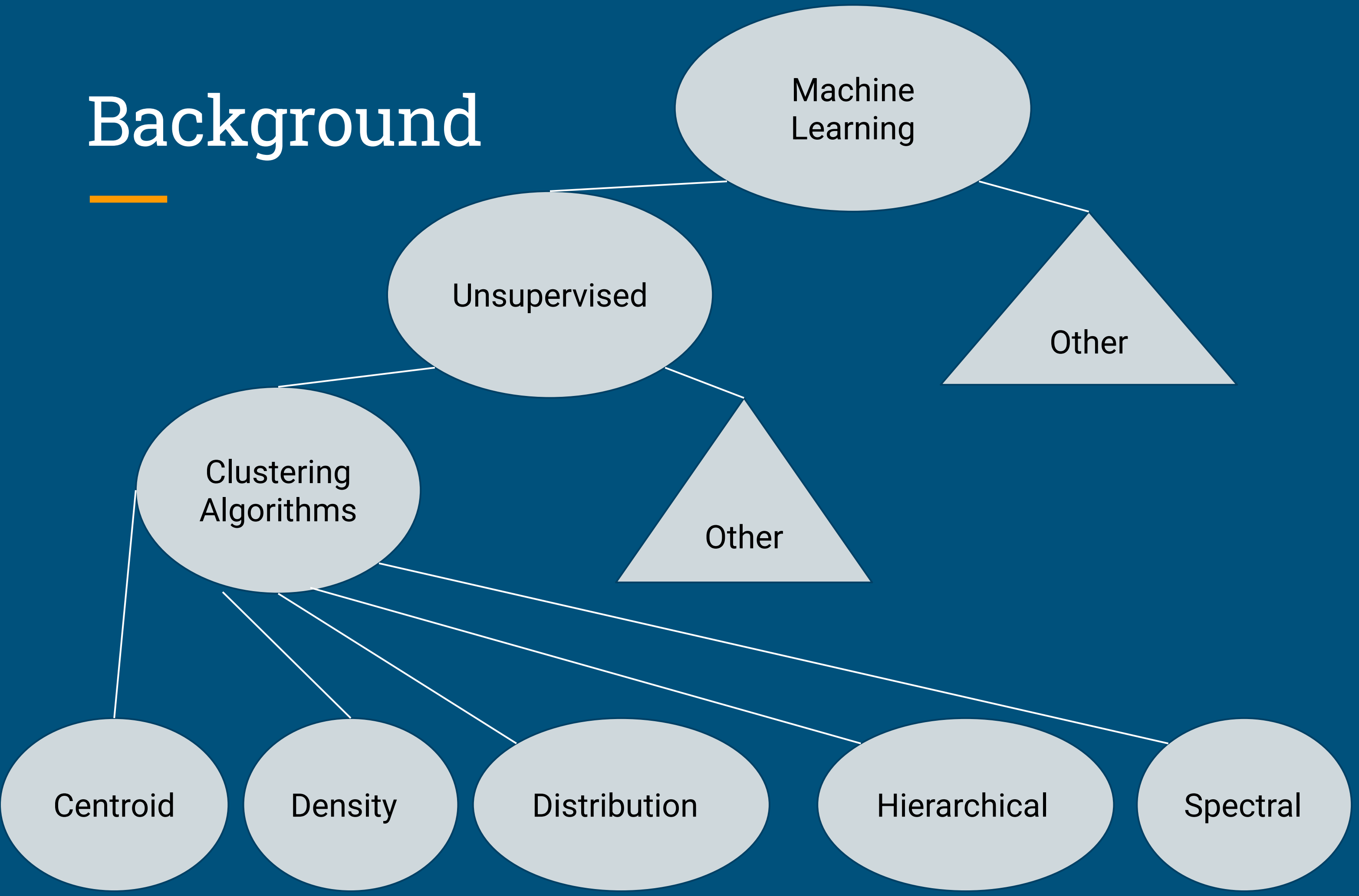
DBSCAN Theory and Applications Introduction

Eduardo Lopez

Background


Machine
Learning
Unsupervised
Other
Clustering
Algorithms
Other
Centroid
Density
Distribution
Hierarchical
Spectral

K-Means(Centroid Based)

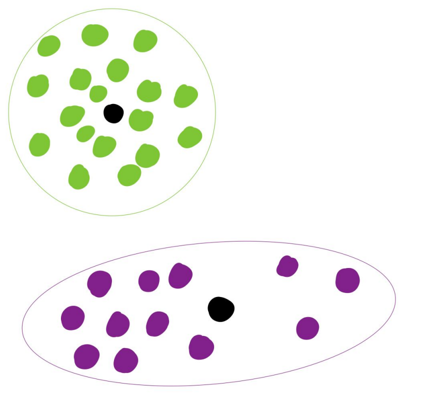
- K-means
  - Dev: 1950s - 1960s by a group of researchers
  - Purpose: clustering data with K number
  - of clusters
  - Required parameter: n_clusters=#

Definitions:

- Centroid: the mean/center of the distributed cluster
  - On each iteration, the centroid is assigned to minimize the variance/distribution of each cluster
- Distance Metric:
  - Function the calculates the distance between two points (e.g Squared Euclidean Distance, Manhattan, etc.)

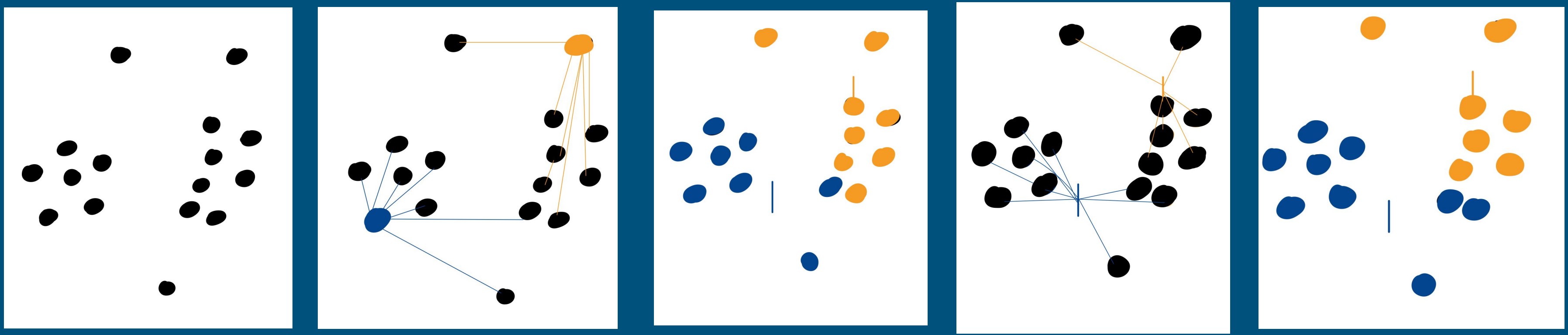
1. Initialize **cluster centroids** $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n$ randomly.
2. Repeat until convergence: {

   For every i, set
   $$c^{(i)} := \arg\min_j ||x^{(i)} - \mu_j||^2.$$
   For each j, set
   $$\mu_j := \frac{\sum_{i=1}^m 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)} = j\}}.$$
   }

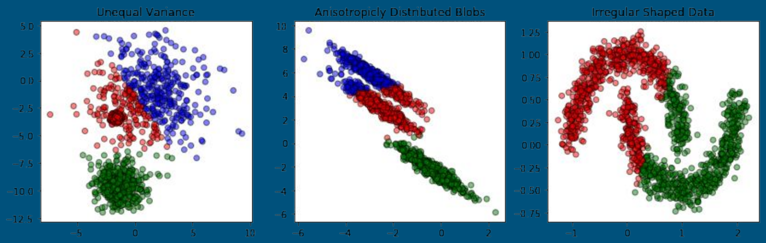


$$d(x_i, x_{i'}) = \sum_{j=1}^p (x_{ij} - x_{i'j})^2 = ||x_i - x_{i'}||^2$$

Simulation(k=2)



Drawbacks!!



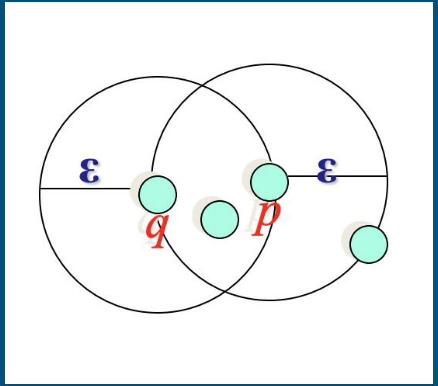
- The number of clusters have to be set by the user
    - Dataset may have hundreds, thousands of clusters, many features!!
- Doesn’t perform well with complex clusters/noisy data

Solution??

(Density Based) Spatial Clustering with App. Noise

- DBSCAN:
  - Dev: 1996 by a group of researchers
  - Purpose: clustering data by discovery
  - Required parameters: eps=#, min_samples=#

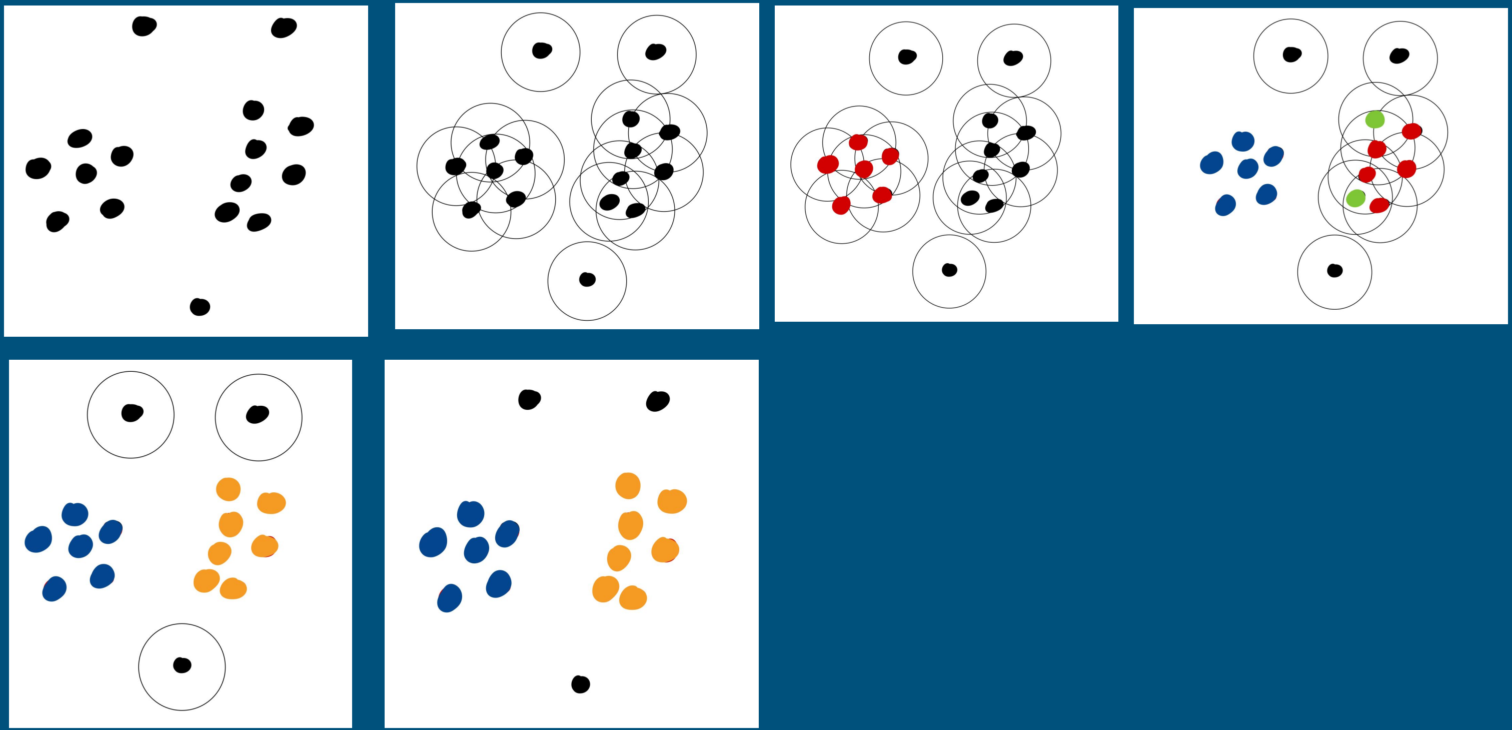
min_samples=3



Definitions:

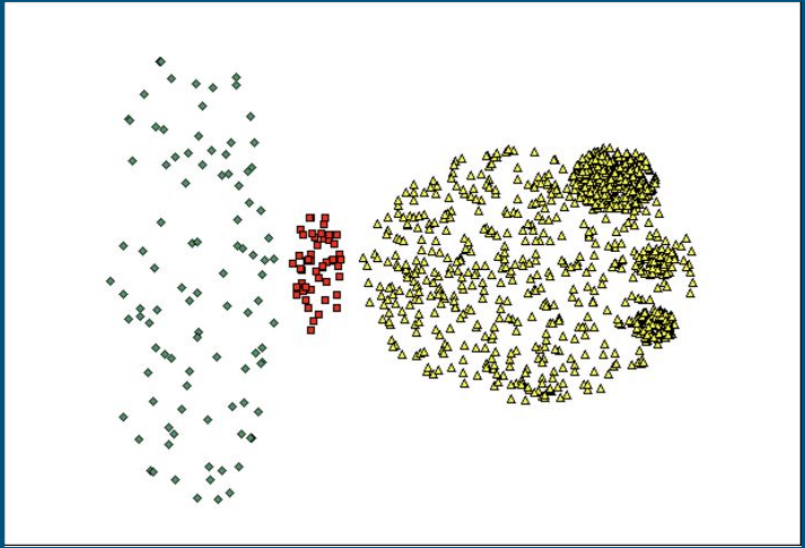
- Core point: a point that meets the following criteria
  - Given eps, it has at least min_samples of neighbors in its neighbor(within radius of eps)
- Border point:
  - Has fewer than min_samples but is in the neighborhood of another core point
- Noise point:
  - Is neither a core point nor a border point

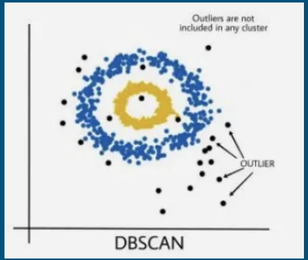
Simulation(min_samples=3)



Drawbacks!

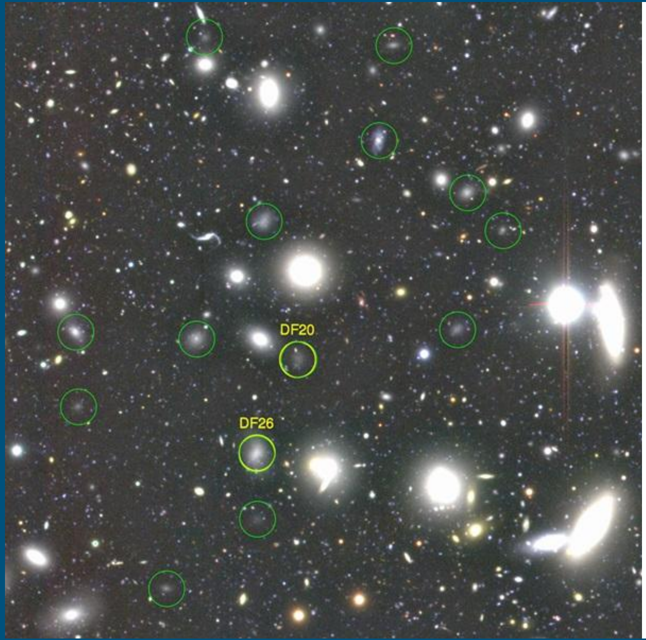
- Cannot handle varying densities within cluster
- Sensitive parameters may result in difficult training

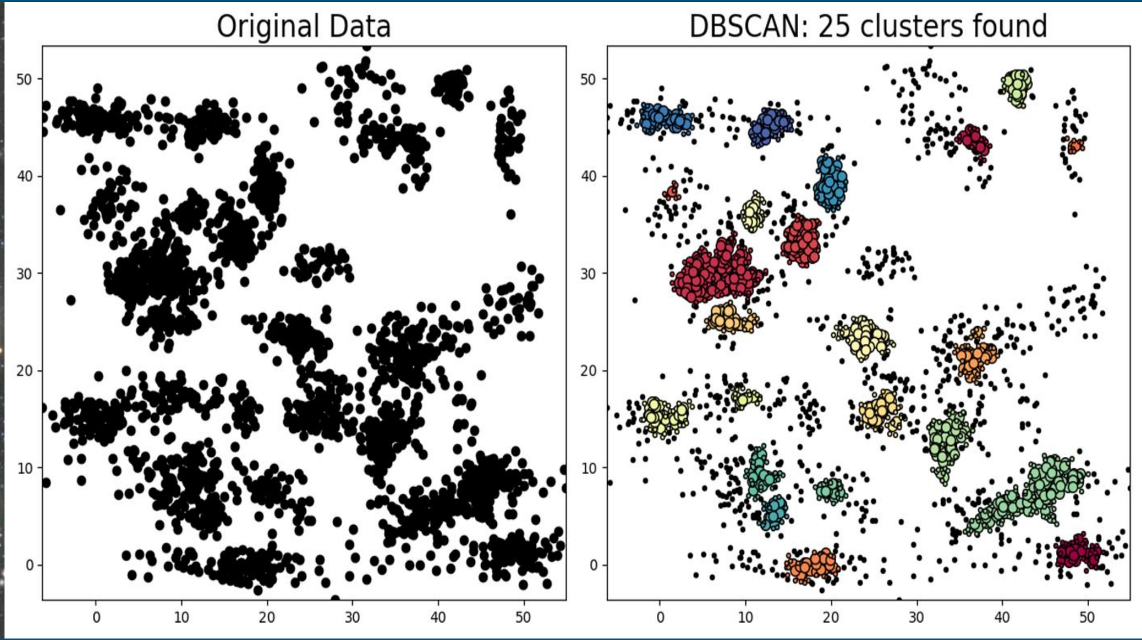




Astronomy Applications

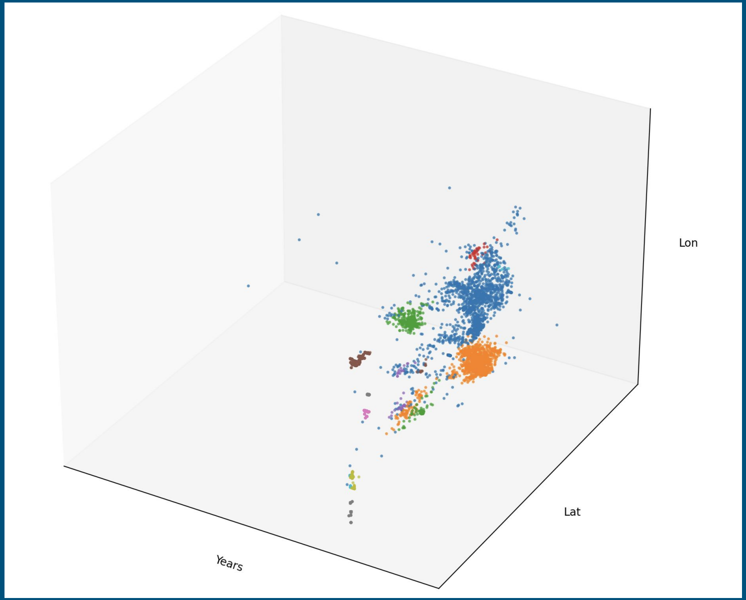
Great for discovering star clusters, galaxy groups, gamma-spectrum observations





My Own Application

Meteorite hotspots identification



- Discovering meteorite hotspot clusters on geographical data
- Membership testing can be done post training
- Using SVM soft scoring Kernel Density Estimator, membership testing is possible using cluster densities
- Gaussian RBF kernel is more simple!

$$k(x, y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$$

- argmax[Pr(X is on cluster i) = density_i * weight_i]

Pseudocode

```
DBSCAN(D, eps, MinPts)
   C = 0
   for each unvisited point P in dataset D
      mark P as visited
      NeighborPts = regionQuery(P, eps)
      if sizeof(NeighborPts) < MinPts
         mark P as NOISE
      else
         C = next cluster
         expandCluster(P, NeighborPts, C, eps, MinPts)

expandCluster(P, NeighborPts, C, eps, MinPts)
   add P to cluster C
   for each point P' in NeighborPts
      if P' is not visited
         mark P' as visited
         NeighborPts' = regionQuery(P', eps)
         if sizeof(NeighborPts') >= MinPts
            NeighborPts = NeighborPts joined with NeighborPts'
      if P' is not yet member of any cluster
         add P' to cluster C

regionQuery(P, eps)
   return all points within P's eps-neighborhood (including P)
```

Time Complexity
(Polynomial time problem):

Average Case: *O(n log n)*
Worst Case: *O(n^2)*

Space Complexity:
O(n)

Scikit-learn Implementation

```python
# importing packages
import numpy as np
from sklearn import DBSCAN
from sklearn.preprocessing import StandardScaler

# preprocessing the dataset before implementation
dataset = np.array(dataset).astype(float)

# optional scaling depending on dataset to prevent bias
scaler = StandardScaler()

# scaling dataset
scaled_dataset = scaler.fit_transform(dataset)

# initializing DBSCAN algorithm and setting parameters
model = DBSCAN(eps=0.5, min_samples=5)

# training model to the dataset
output_dataset = model.fit(scaled_dataset)

# attributes (output after training)
labels = output_data.labels_
core_points_indices = output_data.core_sample_indices_
seen_features = output_data.n_features_in_
```

Resources

Scikit-learn:

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

Textbook:

https://hastie.su.domains/ElemStatLearn/download.html

Papers:

https://file.biolab.si/papers/1996-DBSCAN-KDD.pdf