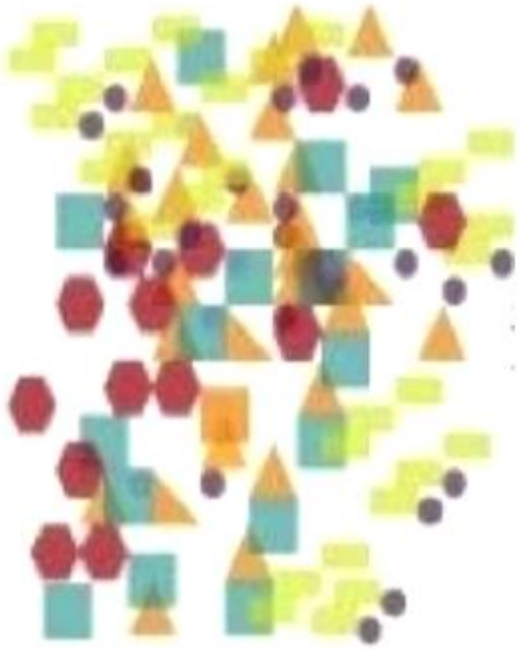
Unsupervised Learning

- You **do not** know the right answer, and there is too much data for you to guess
  - Example: you discover a geyser
  - It erupts at intervals… you sense that the eruption strength relates to the rainfall, but you don't know how
  - You want to predict the characteristics of the next eruption…
  - **How do you do that?**

Unsupervised Learning

1. You collect data about the eruptions (e.g. height, duration, and the amount of rainfall over the previous 3 days)
2. Then you plot (e.g. 3 parameters)
3. Then you ask the system: can you tell me if there are patterns?



Raw Data

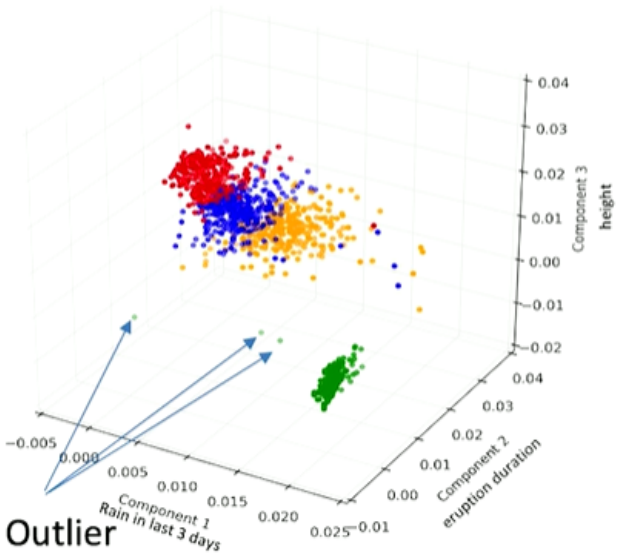
Your graph will group eruptions that have similar characteristics.

- In math, this is simply grouping points that are close to one another

And will spot the outliers

- Those are the abnormal eruptions

4 groups = 4 eruption families

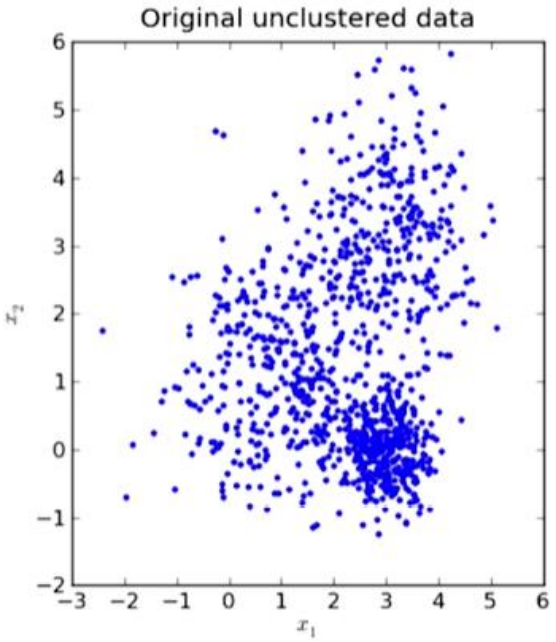


Outlier

The math can take may forms, but a common form is **K-means**

K-means is based on the idea of groups

You choose a group number, then compute the best group membership for each point on the graph



K-means Example: 3 Clusters

Generate 3 random points

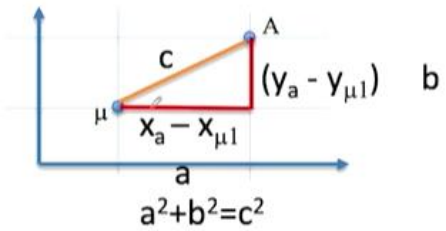
- We call them centroids (μ_1, μ_2, μ_3)
- Each has a position on the graph:
  - μ_1 ($x_{\mu 1}$, $y_{\mu 1}$), μ_2 ($x_{\mu 2}$, $y_{\mu 2}$), μ_3 ($x_{\mu 3}$, $y_{\mu 3}$)

Compute each point distance to each centroid

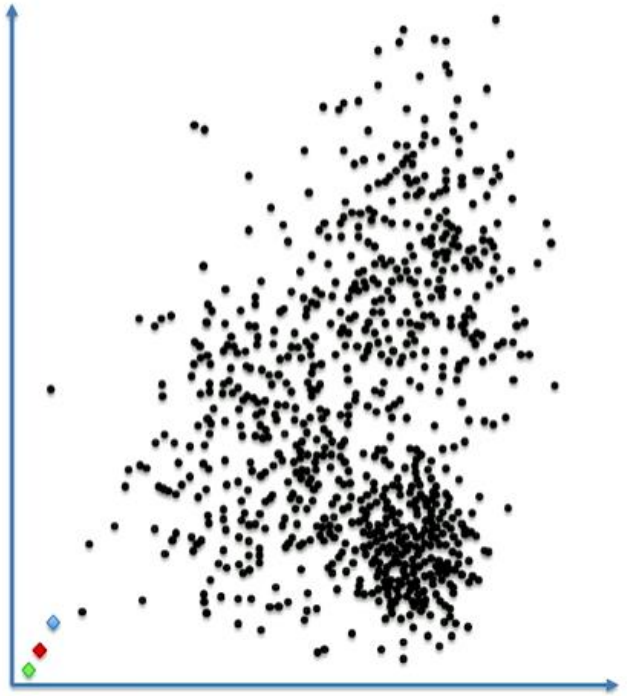
This can be as simple as, for each point A (x_a, y_a), compute:

$$\text{sqrt}[(x_a - x_{\mu 1})^2 + (y_a - y_{\mu 1})^2]$$

$$\text{sqrt}[(x_a - x_{\mu 1})^2 + (y_a - y_{\mu 1})^2]$$



$a^2+b^2=c^2$



K-means Example: 3 Clusters

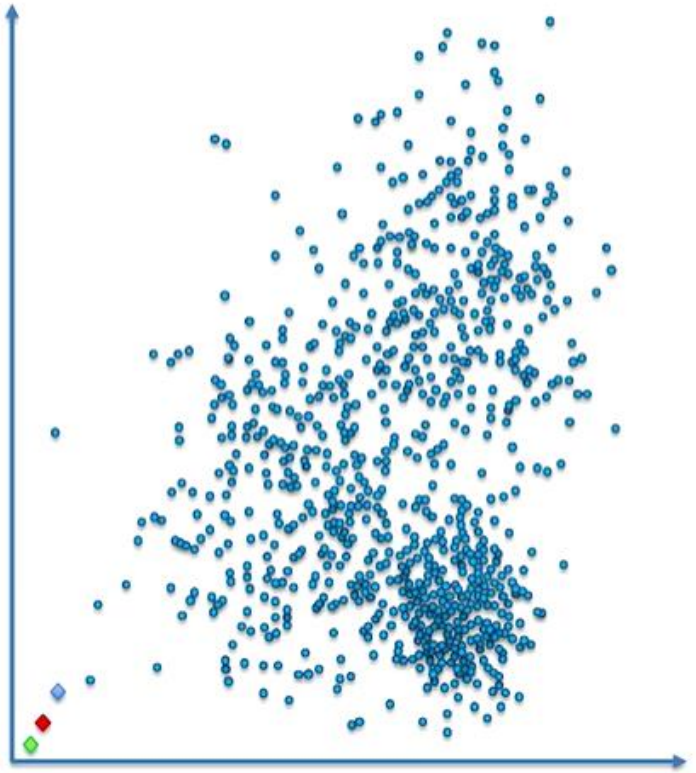
Compute each point distance to each centroid

Then associate each point to the closest centroid

Then compute the center of the cluster (all points belonging to each centroid)

Here, all belong to blue!

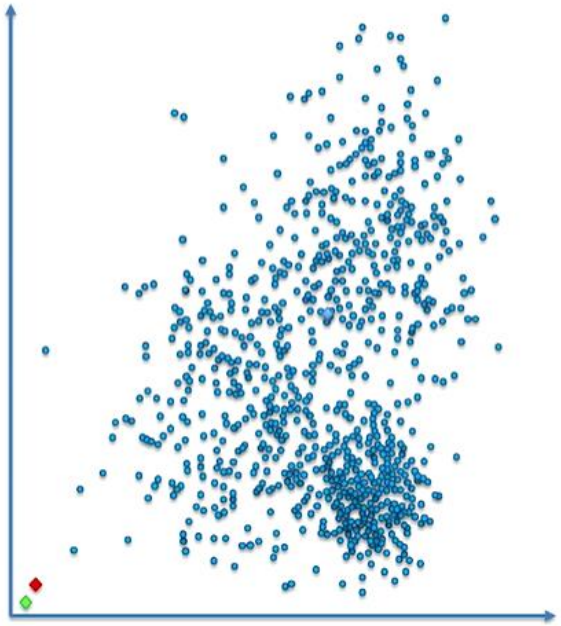
center is $\left(\frac{all\ x}{number\ of\ points}, \frac{all\ y}{number\ of\ points}\right)$



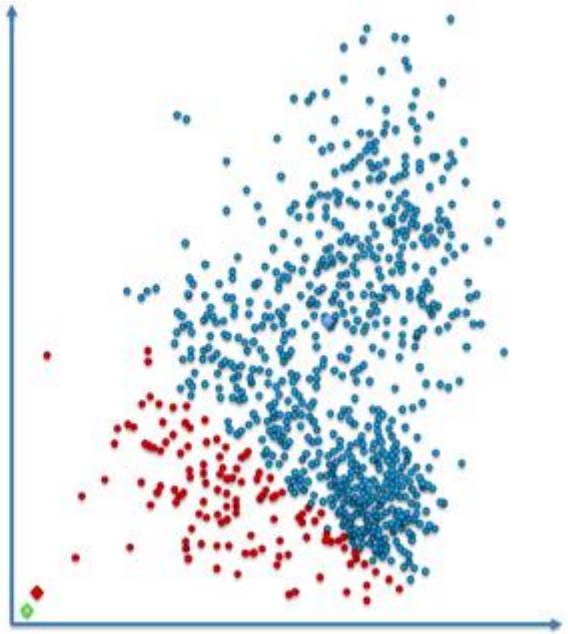
K-means Example: 3 Clusters

Move your centroids there

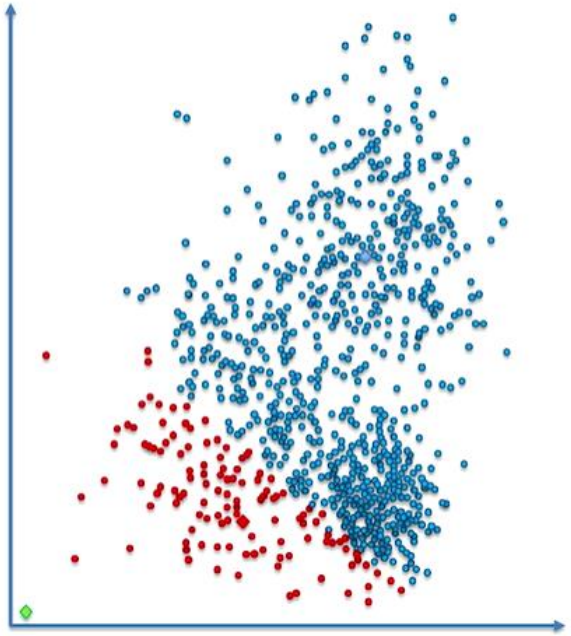
(here **red** and **green** do not move as they have no member)



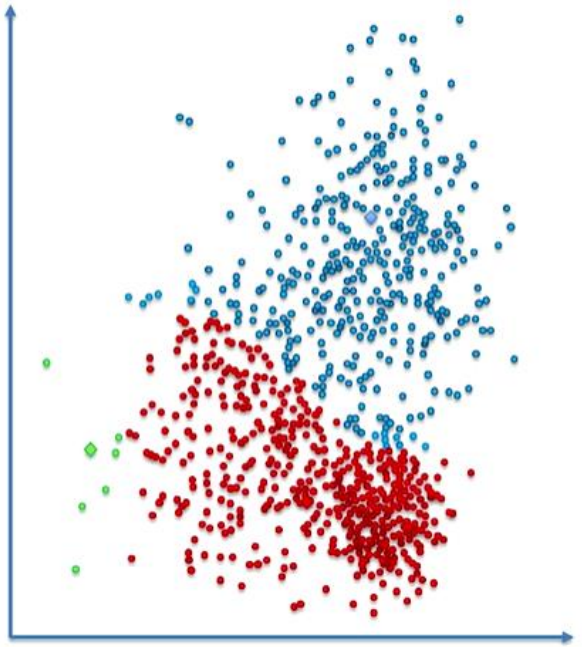
Then repeat (compute each point distance to each centroid, associate to closest centroid)



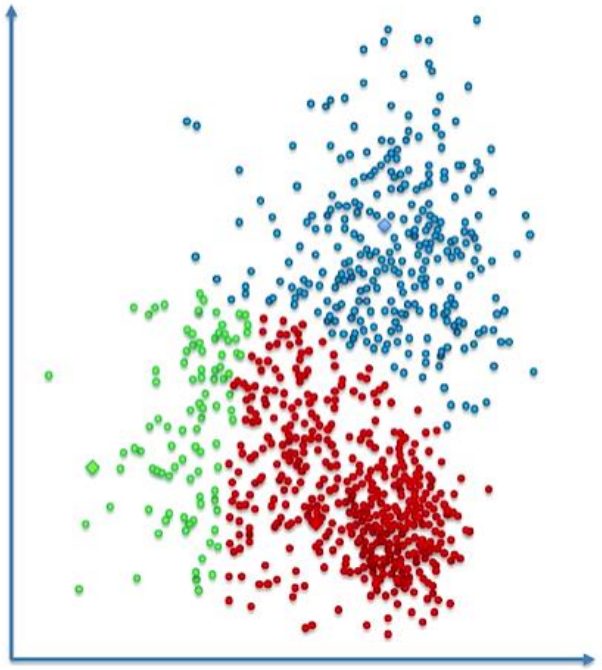
Move again each centroid to the cluster center



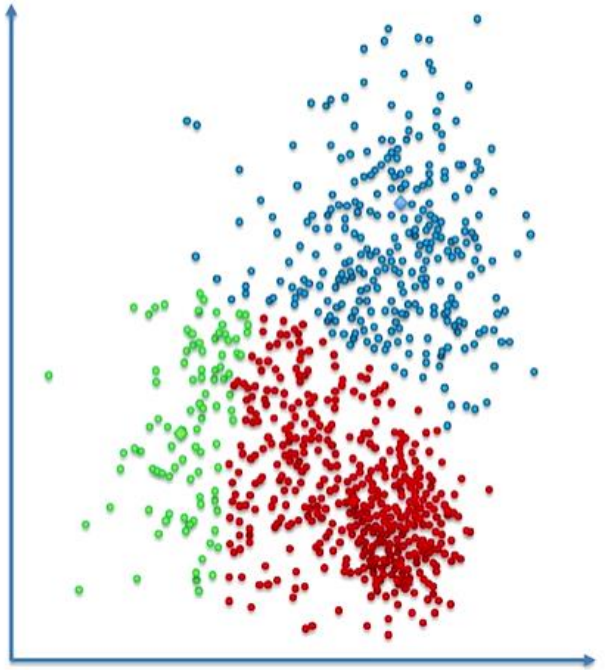
Move again each centroid to the cluster center



Then repeat (compute each point distance to each centroid, associate to closest centroid)



Move again each centroid to the cluster center

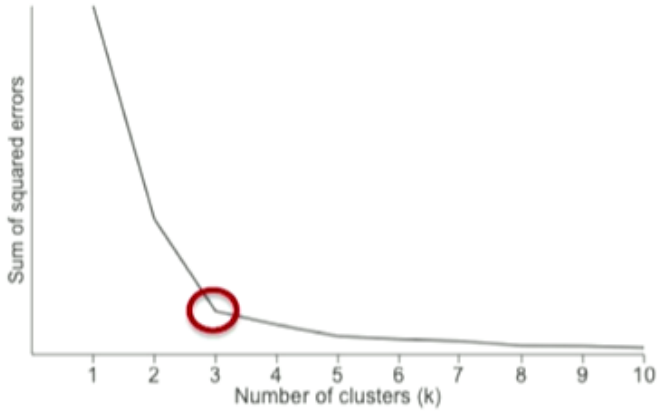


K-means Challenges

A difficulty with K-means is that you need to choose K

If you are unsure, you can use the ***elbow method***:

- Use several K values, and for each K, compute the Sum of Squared Errors (SSE) – this is the distance of each point to its centroid
- When K=1, SSE is high
- When K = number of points, SSE = 0
- You are looking for the number of "Ks" where adding one more K does not add much benefits



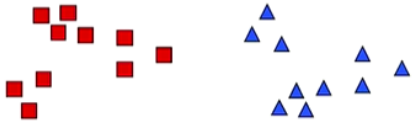
K-means Challenges

"Cluster" is not a well-defined concept – sometimes, the elbow method is not sufficient

How many clusters?

Four Clusters

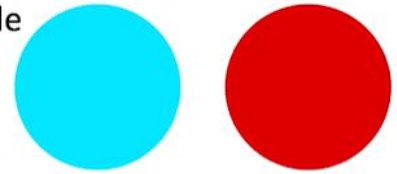
Six Clusters

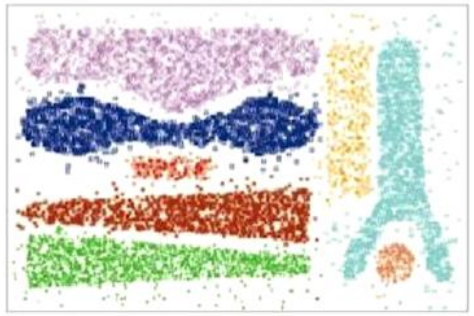


Two Clusters

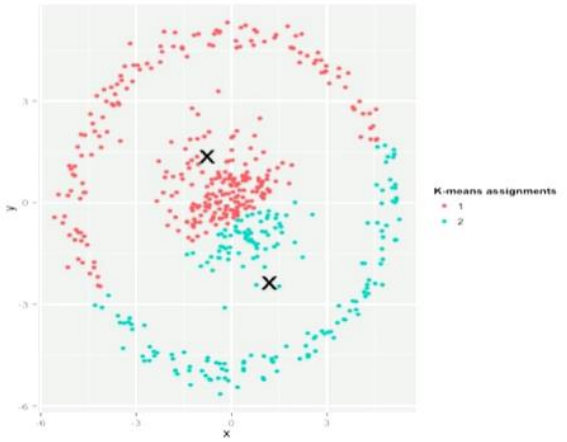
When clusters are well separated, you can usually decide

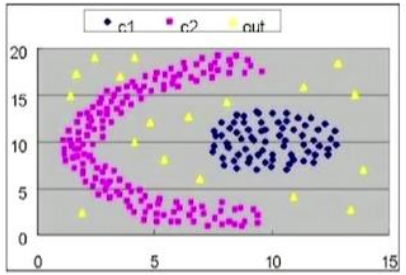
Even when their shape is surprising





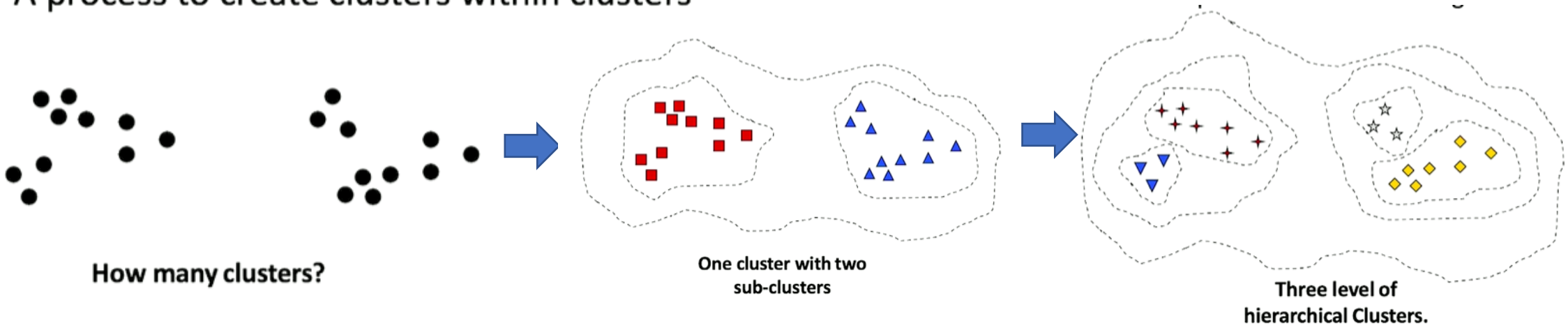
However, K means also fails on certain shapes





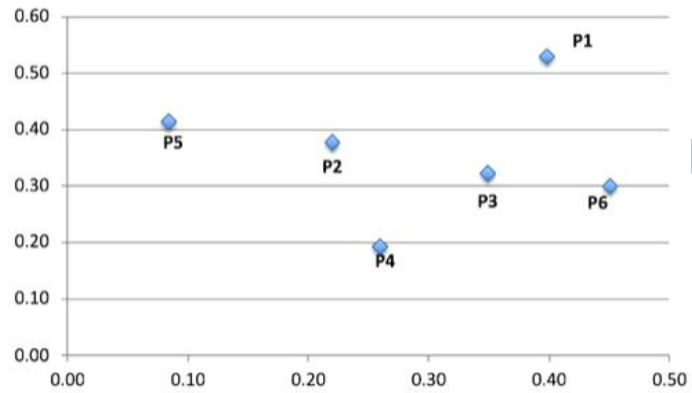
Hierarchical Clustering

A process to create clusters within clusters

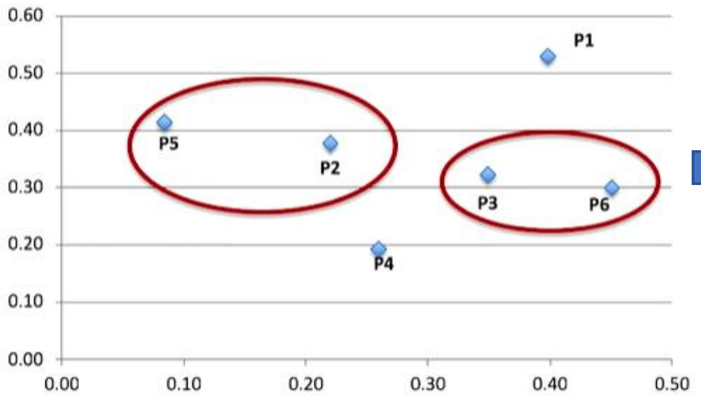


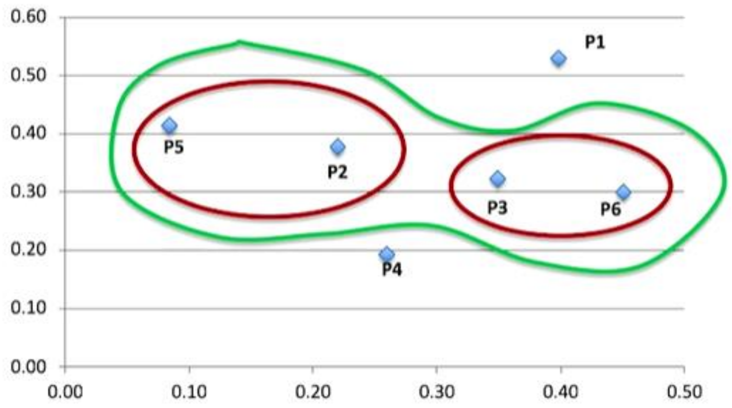
- Works similarly as K-means, but towards the opposite direction
- Create n clusters, then identify neighbor clusters and merge them
- Repeat until there is a single cluster

Hierarchical Clustering Illustration

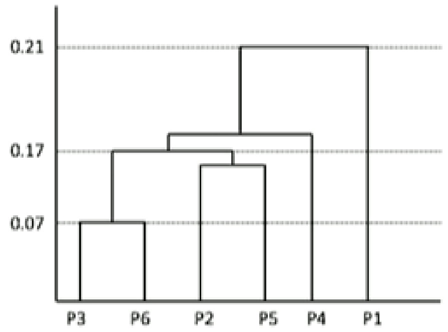


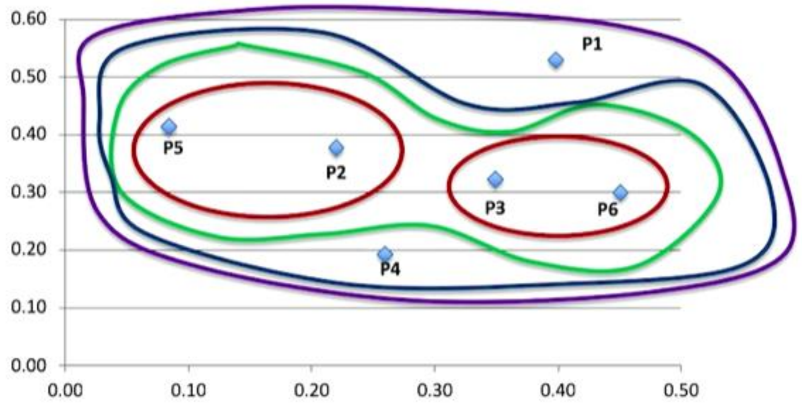
Suppose we found the following 6 centroids





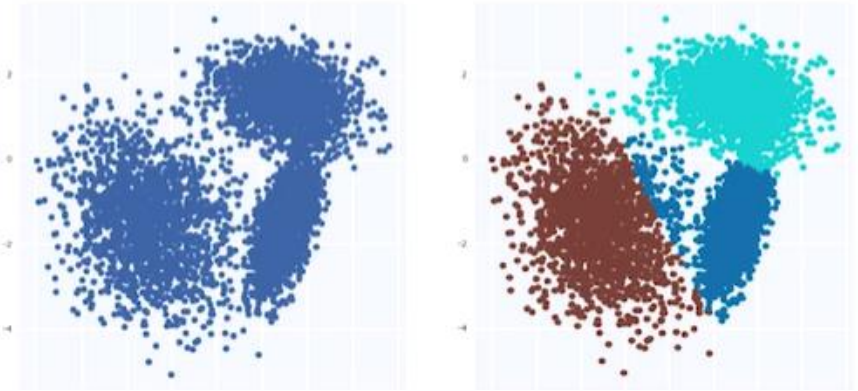
The result can also be represented as a dendrogram, showing the relationships and distances between clusters





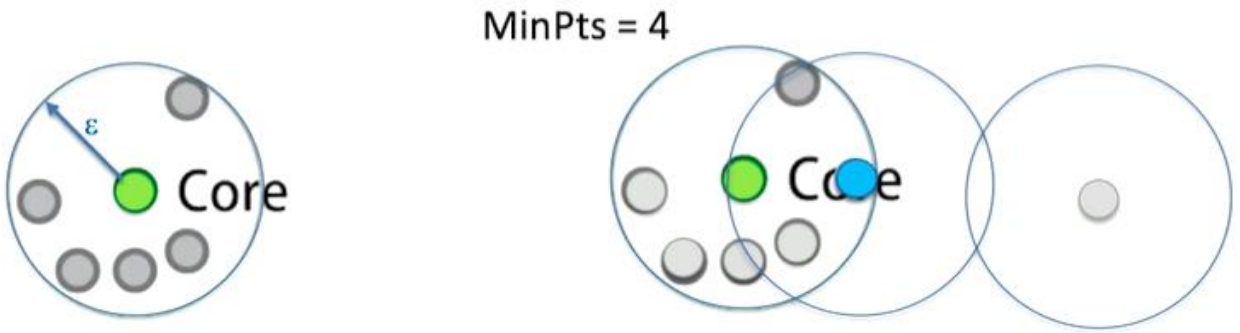
DBSCAN

- Density Based Spatial Clustering of Applications with Noise
  - Add “Hierarchical” to make it HDBSCAN
- Does not need you to input the number of clusters
- Good with “noisy” or “messy” data
- Allows “noise” or “outliers” to exist outside of the clusters



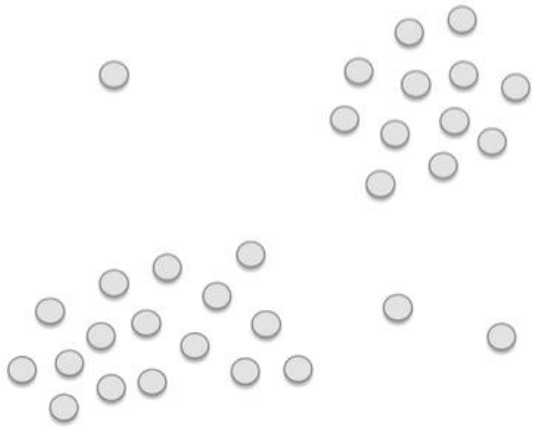
How DBSCAN Works

- You need to define some parameters:
  - How many points (minimum) in a cluster (MinPts)
  - The “density”, i.e. the minimum radius of a cluster (Epsilon, ε)
- DBSCAN understands 3 types of points:
  - Core points have MinPts or more neighbors within a radius of ε
  - Border points have a core neighbor, but less than MinPts in their ε radius
  - Noise points are neither core nor border

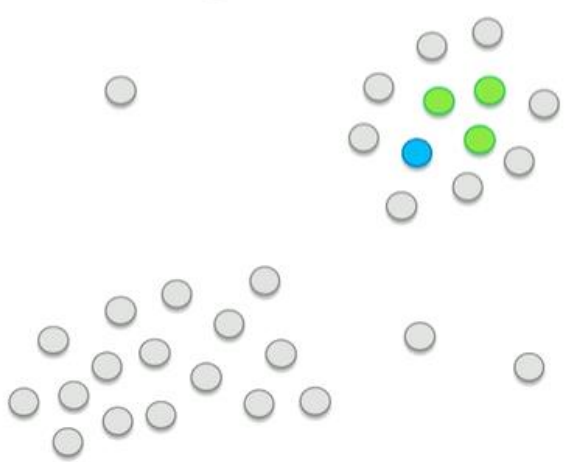


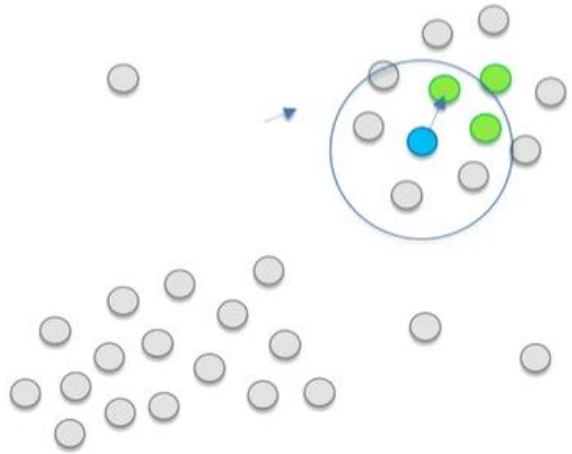
Running DBSCAN

DBSCAN first computes each point's ε, then counts neighbors to determine which points are core

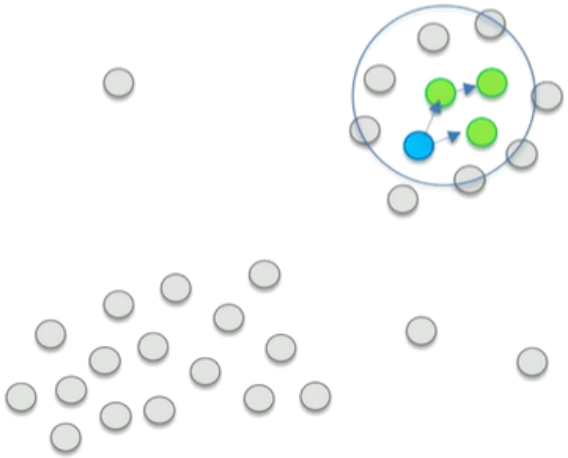


- Then DBSCAN picks a (random) core point, and selects all its core neighbors

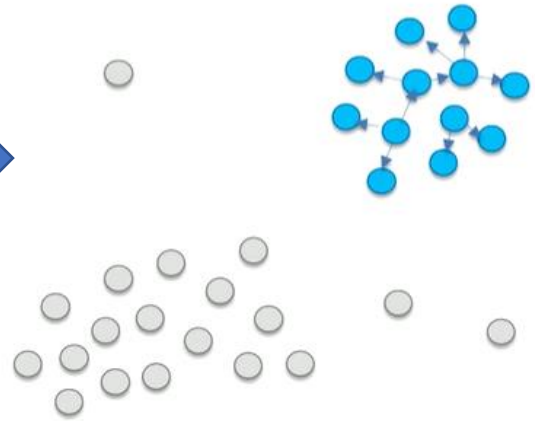




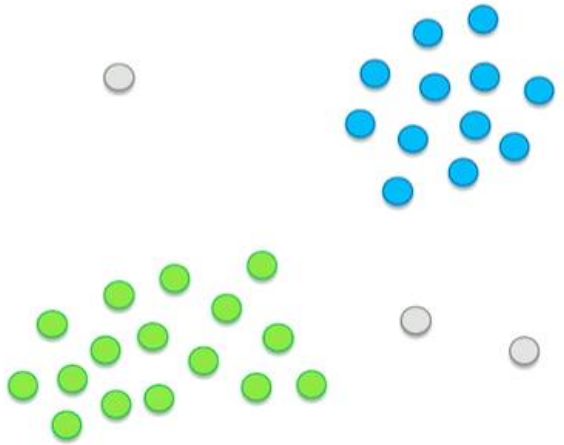
- Repeat until all cores in the cluster are found and linked
- Then find all neighbors (within ε) of all cores



- This is your first cluster



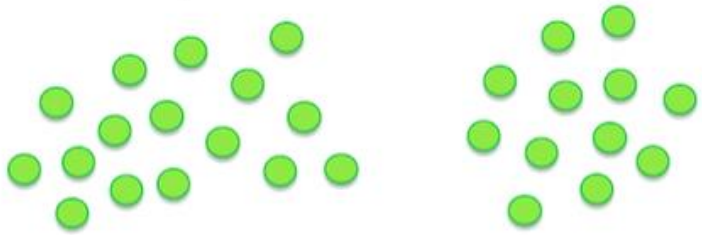
- Repeat until all cores are explored -> all clusters are formed
  - The cluster count has been automatically determined
  - Outliers are allowed outside of clusters

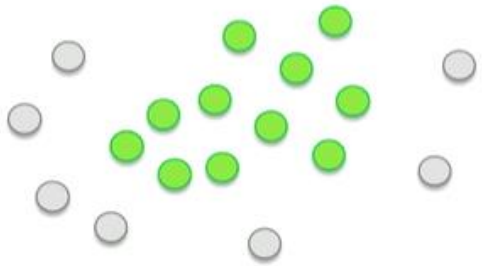


DBSCAN and Beyond

DBSCAN solves some limitations of K-Means, but is sensitive to parameters:

- ε too large -> gigantic cluster
- ε too small -> sparse cluster is seen as noise





- There are many other methods (Gaussian Mixtures Models, Gaussian expectation-minimization, k-harmonic, fuzzy k-means, and many others)
- Start with K-means, think of how you want to graph your data, and explore beyond if it fails…