Intro to Data Mining Lecture 2b

Clustering

Created by Jon Witkowski on 12/29/2023

Supervised vs. Unsupervised Learning

- Much of Data Mining is supervised
  - We use a training set of data to try and model a situation and then see how new data fits in
  - Or we have particular target variable and we wish to determine whether it will appear, increase, decrease etc.
  - We will see supervised methods when we discuss decision trees, neural networks and k-nearest neighbors
- The classical Unsupervised method is clustering
  - Often we have no idea what we are looking for so to get a handle on the data we try to find some structure via clustering
  - In this case we input a number of variables for each record and see if the clustering algorithm can group things together
- Often clustering is used first. Clusters are then named and described to identify variables of interest. Then a supervised method is invoked.

What is Clustering?

- Clustering groups like data together. We want to see what data is similar and how the attributes within these clusters behave.
    - How are similar rows similar to each other?
    - How are they different from other rows?
    - Do the clusters make sense?

Clustering Applications

- As one of the most used machine learning algorithms, clustering is used in a lot of different applications like:
    - Targeted marketing (i.e. identifying demographics for a product)
    - Grouping stocks with similar prices and fluctuations
    - Recommendation systems (i.e. Netflix or Spotify)
    - Summarizing data
    - Finding documents or search results that are similar to each other

Partitional vs. Hierarchical

- Clusters can either be partitional or hierarchical
  - Partitional Clusters are exclusive. Each object (row) in the data is part of 1 and only 1 cluster. There is no overlapping of clusters here.
  - Hierarchical clusters are exclusive at thresholds, but not overall. An object in the data can be in Cluster 2 at a threshold of 0.5, but in Cluster 5 at a threshold of 1. We'll get more into thresholds later in the slides

Types of Clustering

- There are many different types of clustering. In this class, we will only go over 4 of them:
  - K-Means
  - Hierarchical/Agglomerative
  - Minimum Spanning Tree
  - DBSCAN/Density

Quick Terminology

- In clustering, we'll use the words "centroid" and "clutroid" a lot. These are not the same thing.
- A centroid is the theoretical center of a cluster. It does not have to be an existing datapoint within the cluster. It is just the average point that may or may not exist.
- A clustroid is the existing point closest to the cluster's centroid
- Know the difference between these 2 words

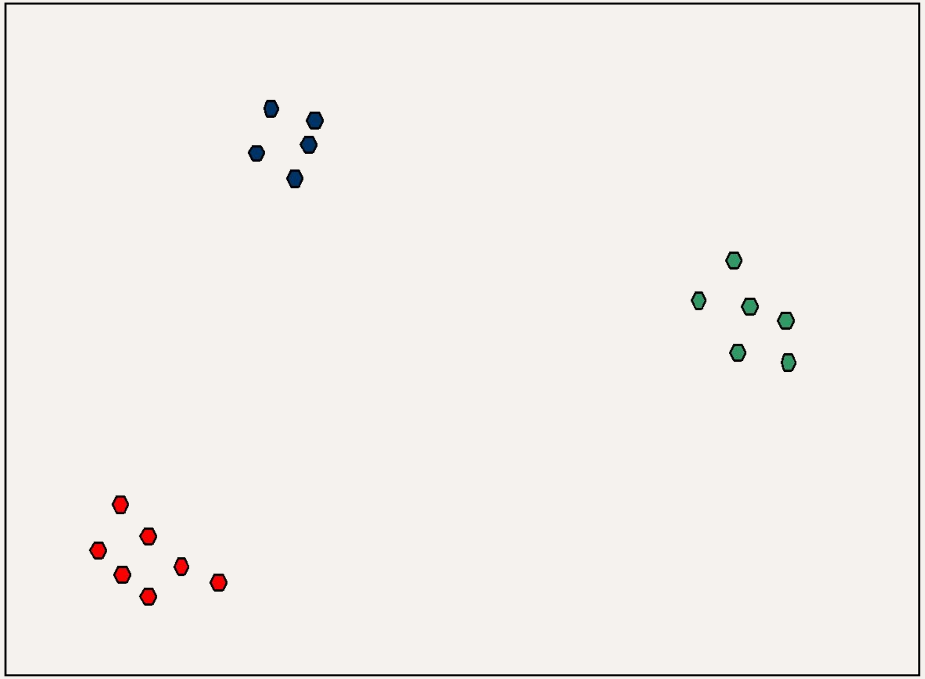
K-Means

- K-means is an iterative partitional clustering algorithm that starts with a number K.
- This number dictates the number of clusters that will be formed

- K means is very simple, having just 4 steps.
  - Select K points to be centroids
  - Assign all points to the nearest centroid to form the clusters.
  - Recalculate the centroids based on the clusters that have been formed
  - If the clusters are the same as before recalculation, you’re done, otherwise repeat Steps 1-3

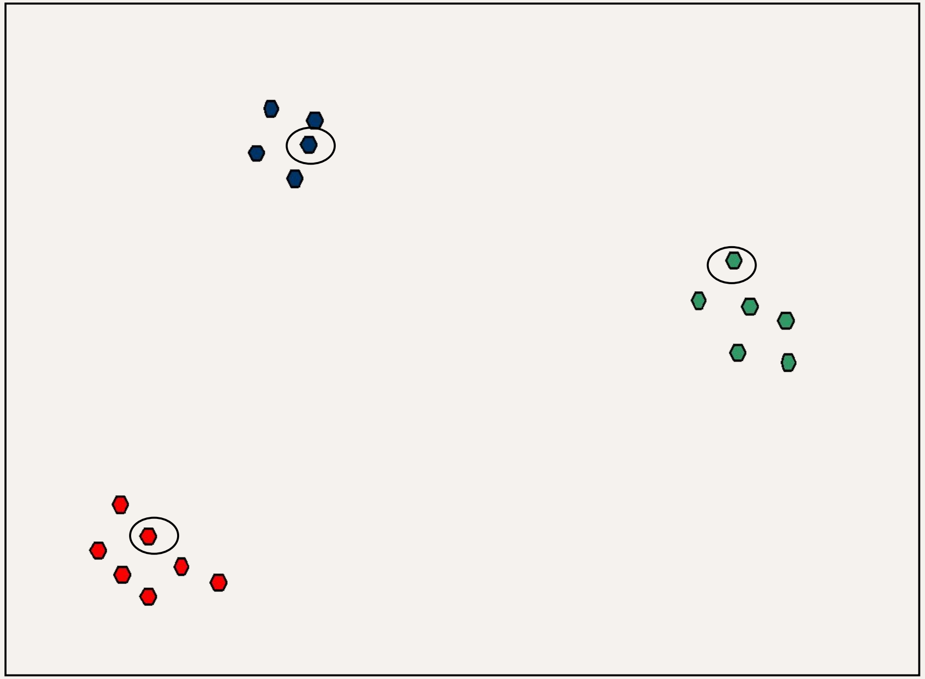
Example

- Here, we have some data points. They haven't been clustered, but it's pretty obvious how they'll cluster, so let's run some examples on this data to show up K-Means works



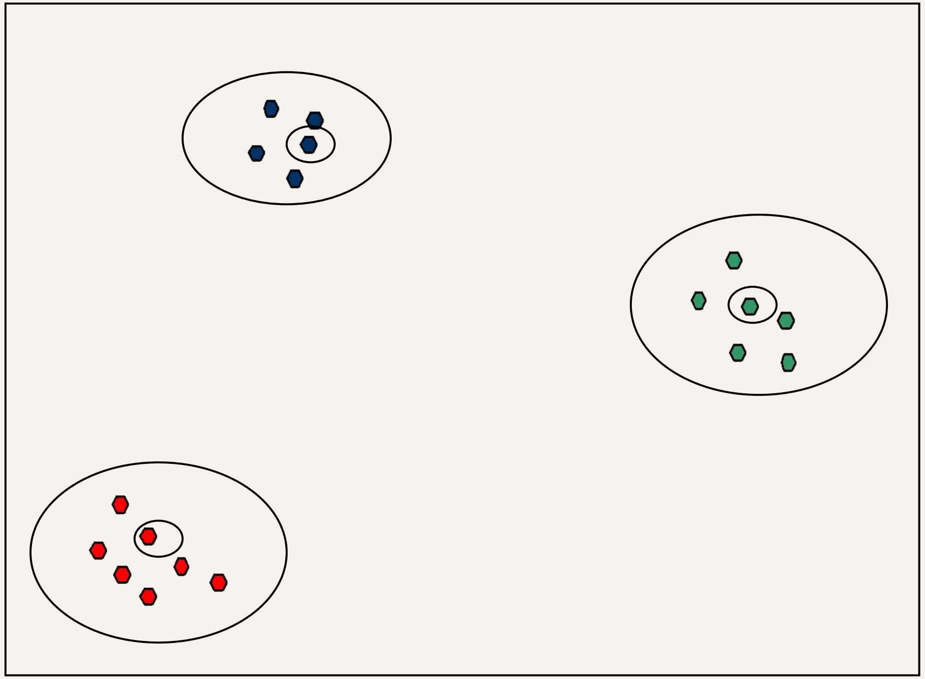
Step 1:

- With K=3, we select our centroids. These centroids are conveniently located so that the obvious clusters will form and we'll be done pretty quickly



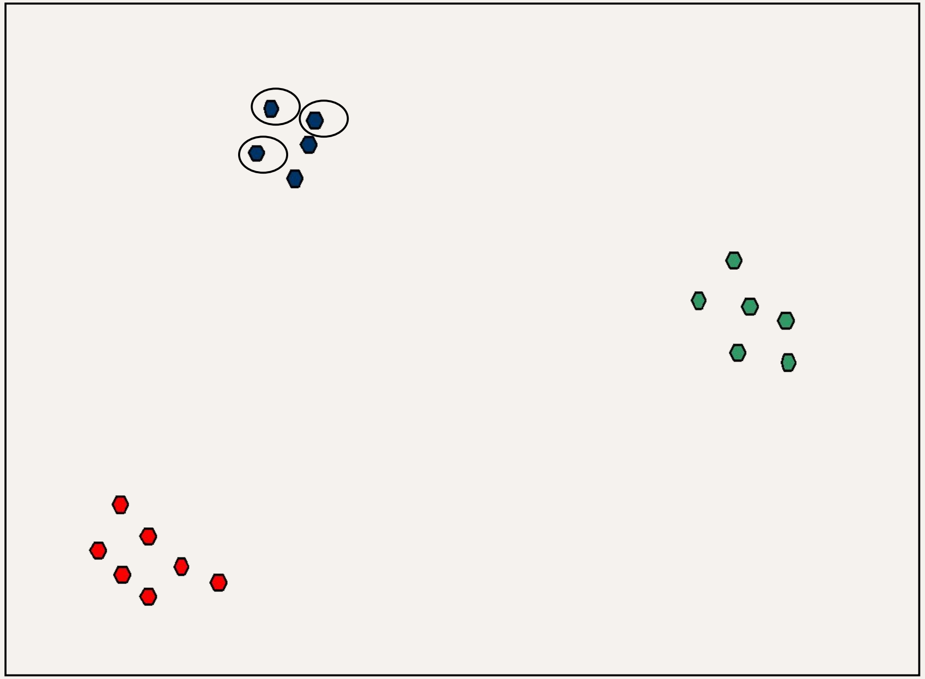
Step 2-4:

- Look at that. We formed the clusters around the centroids and we got exactly the clusters that we expected to get.
- We'll get the same clusters when we recalculate the centroids, so we're done after the end of this iteration.



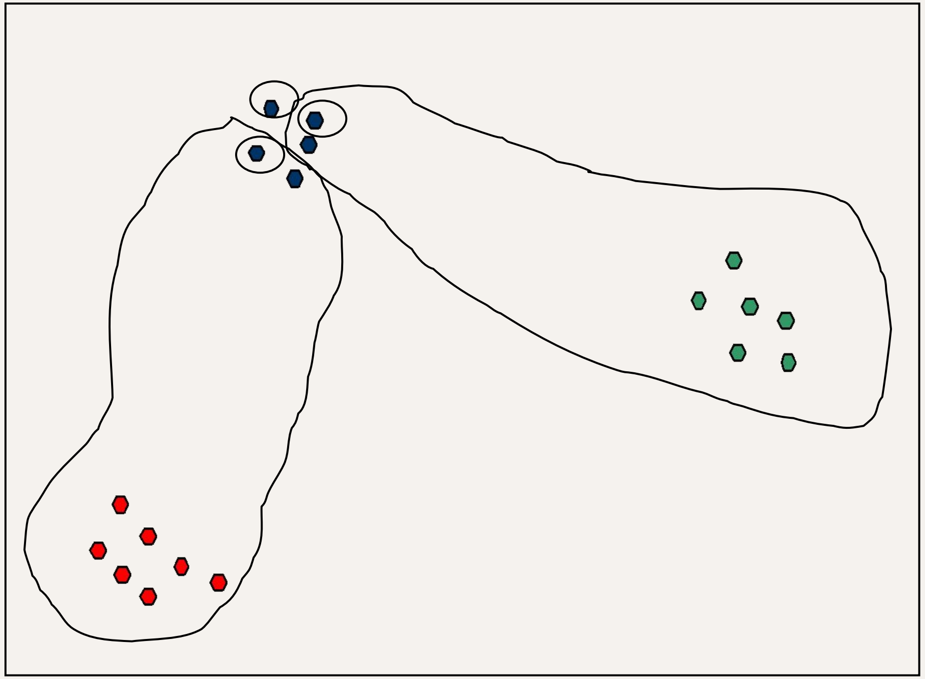
Example 2:

- Look at that! We picked bad centroids. How do you think the points are going to cluster?



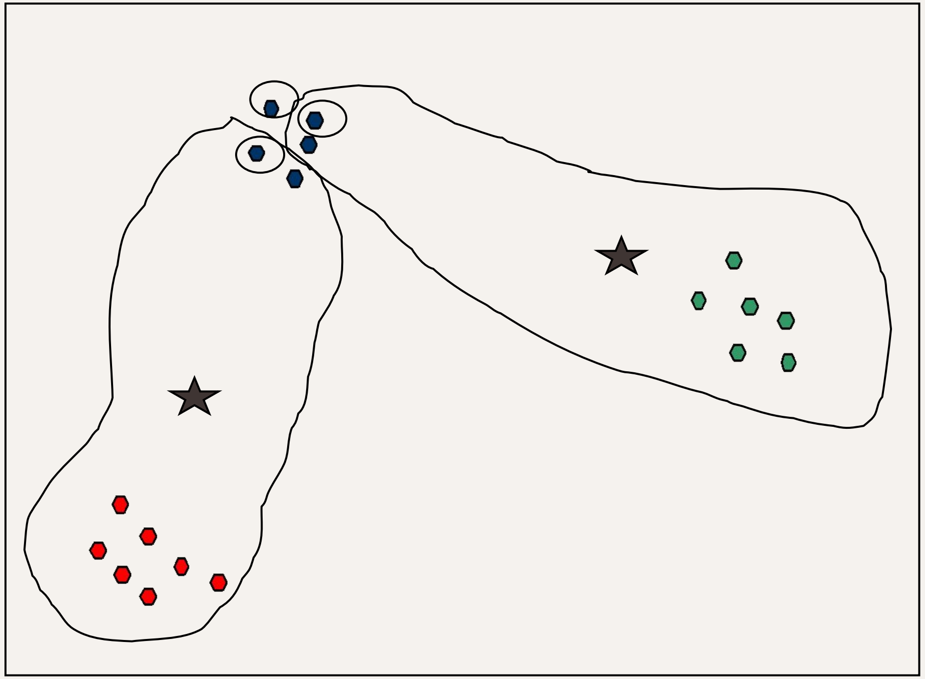
Example 2:

- Not very good clusters here. The next step is to recalculate the centroids based on the points in each cluster



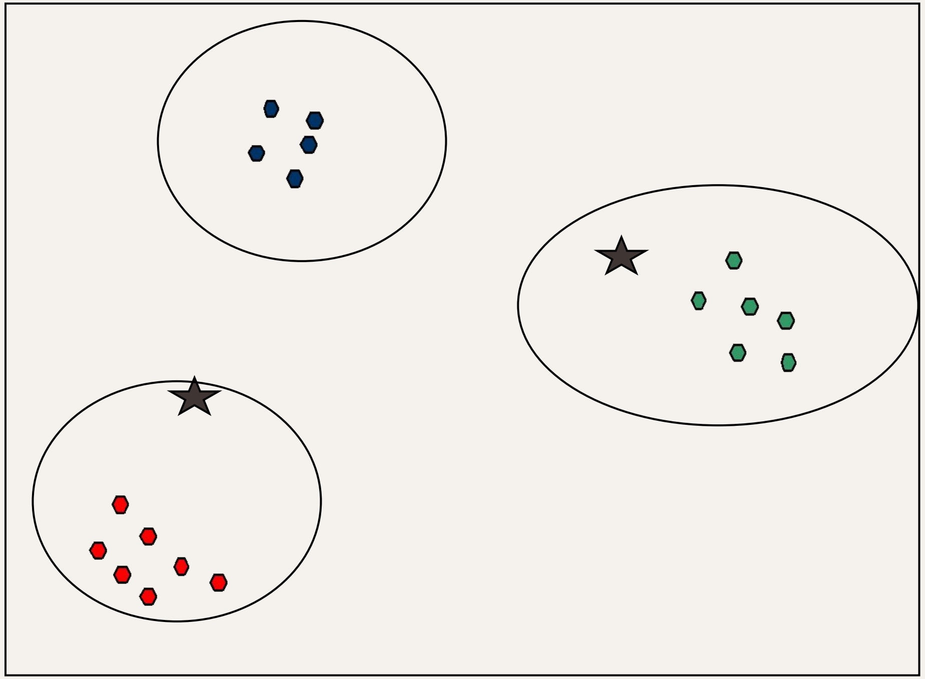
Example 2:

- After recalculating the centroids, our next step is to recalculate the clusters by regrouping the points closest to each centroid



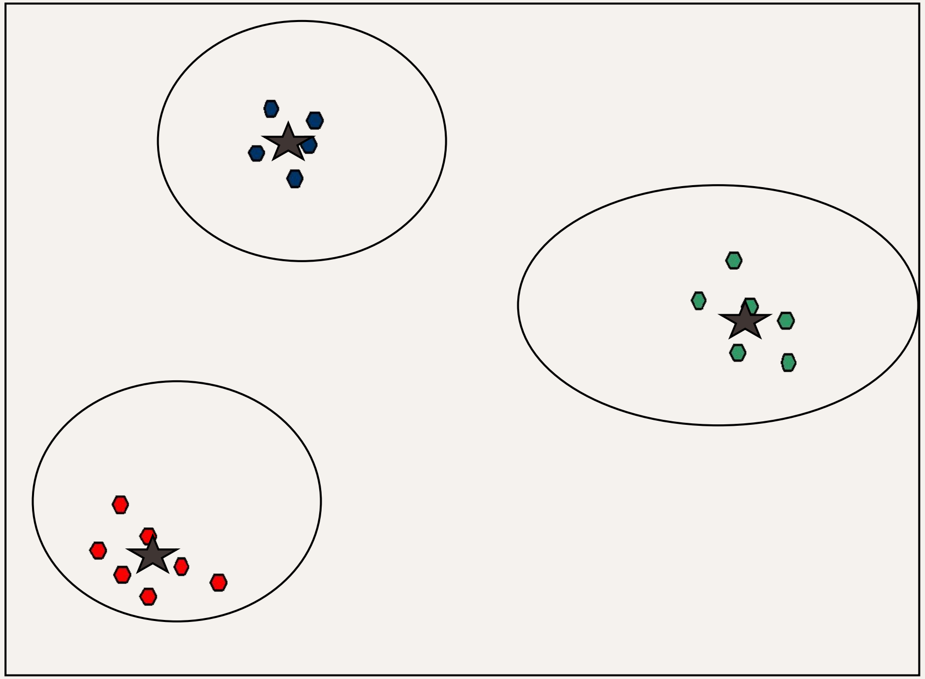
Example 2:

- And we have now gotten the clusters we expect to get. All we need to do now is recalculate the centroids and then clusters and see if we can stop



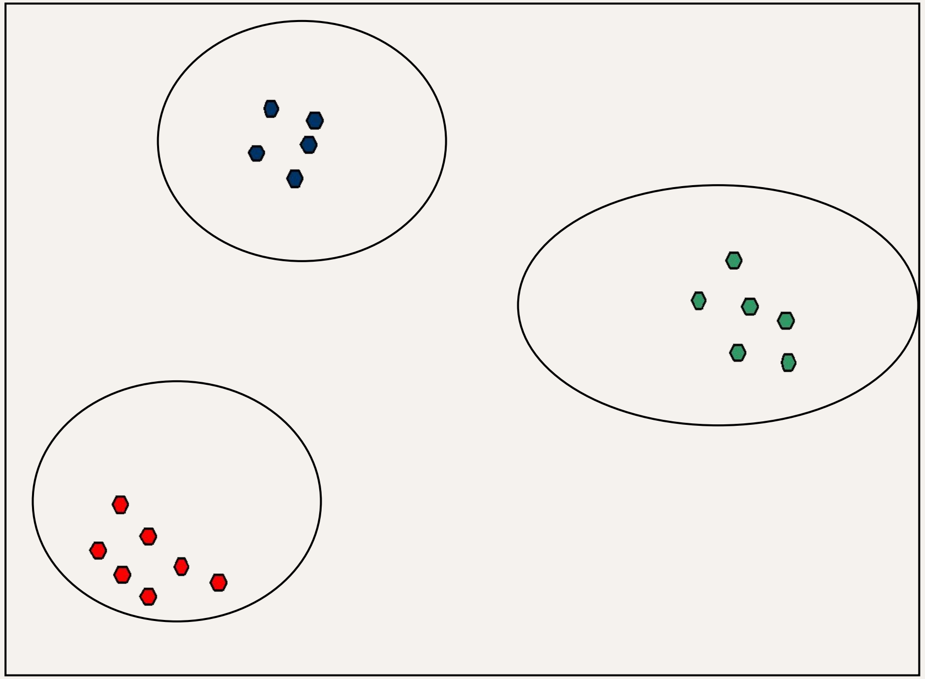
Example 2:

- We've recalculated the centroids and it looks like we're going to get the same clusters this iteration



Example 2:

- The clusters haven't moved, so we're done. This whole process took 2 iterations with good initial centroids and 3 iterations with bad initial centroids



Example 3:

- The last example with suboptimal initial centroids worked out in the end, but that isn't always the case

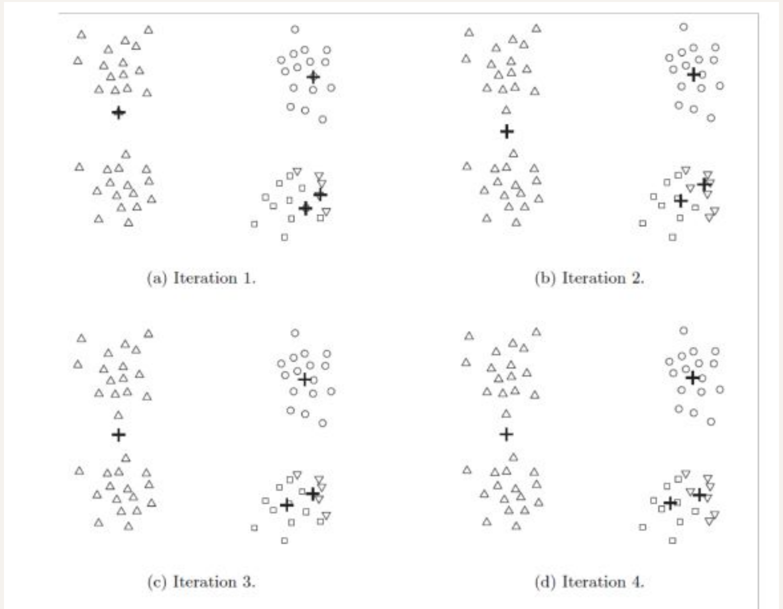


Figure 7.7. Two pairs of clusters with more or fewer than two initial centroids within a pair of clusters.

More on K-Means

- K-Means is the most used clustering algorithm because, as we just saw, it’s fast and easy to do.
- Advantages
    - Fast
    - Easy
- Disadvantages
    - Non-repeatable. Unless you set a seed, running it twice in a row may result in different clusters.
    - You have to choose how many clusters to pick, which isn’t always easy if you don’t know anything about the data

More K-Means Limitations

- K-Means doesn't work well on data of different shapes, sizes, or densities. We'll see DBSCAN mostly doing these better later on

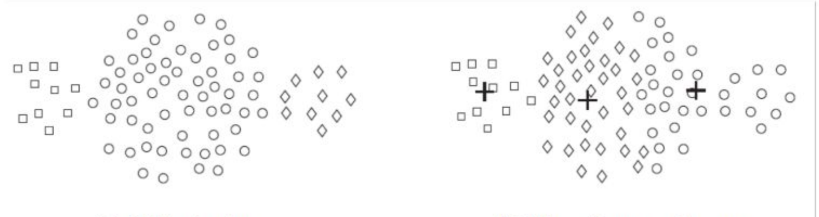


Figure 7.9. K-means with clusters of different size.

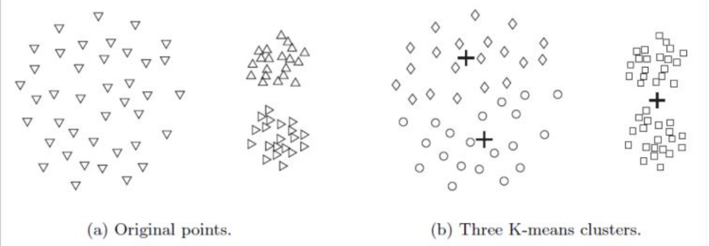


Figure 7.10. K-means with clusters of different density.

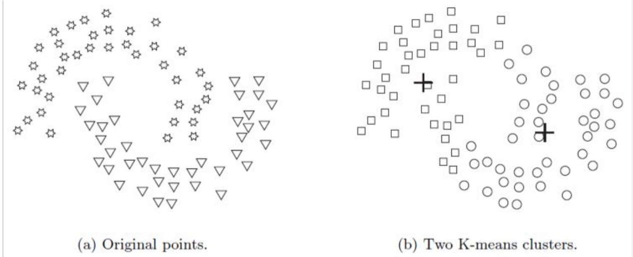
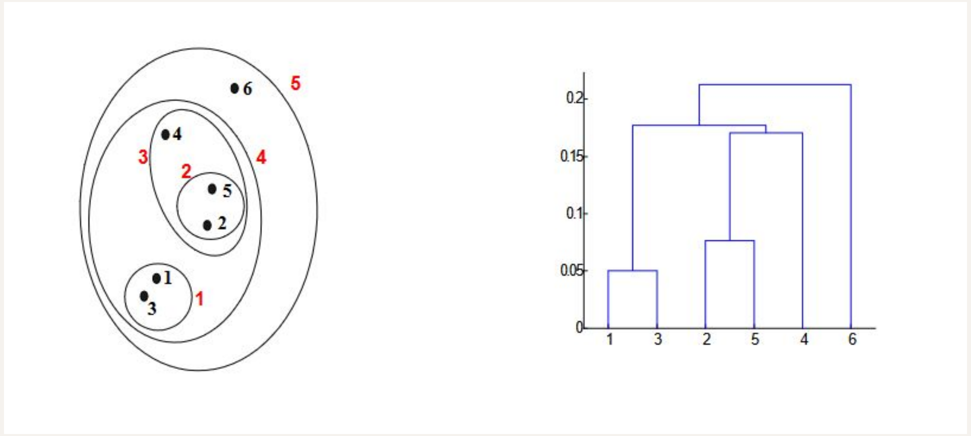


Figure 7.11. K-means with non-globular clusters.

Hierarchical

- Unlike k-means, hierarchical clustering is slow and repeatable. It’s still fairly simple though. It is more susceptible to outlier data though, so make sure you clean your data first.
- Hierarchical creates clusters based on distance which then merge up and eventually become one cluster with all the other ones nested within. These are delimited by the distance threshold you choose to cut off at.
- These can be visualized with nested circles or with a dendrogram. Here’s an example of each



Hierarchical Steps

1. Create a similarity matrix of the data
2. Find the max similarity in the matrix and merge the rows with that similarity
3. Recalculate the similarity matrix. This should now be n-1 x n-1 because we've joined 2 rows together
4. Rinse and repeat until we get to cluster or threshold is reached for distance/similarity

We can alter this to improve performance by merging values instead of recalculating our entire similarity matrix

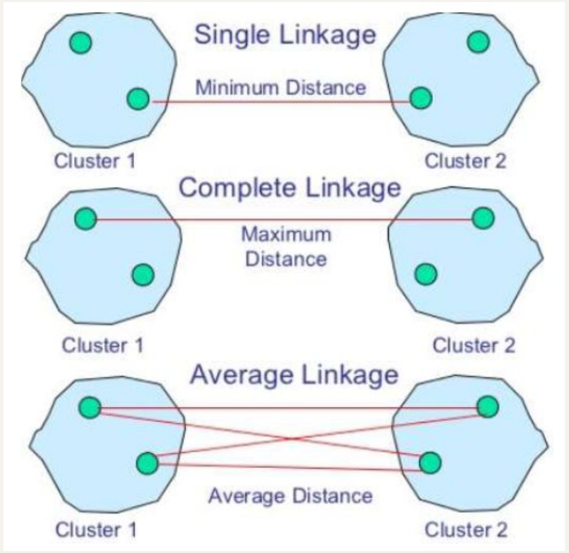
- Suppose we're merging rows 4 and 5. All we really need to change is the similarity from other rows to rows 4 and 5 and adjust the similarity between 4 and 5 since they're 1 cluster now.
- There are different ways we can adjust these values. We call these linkages.

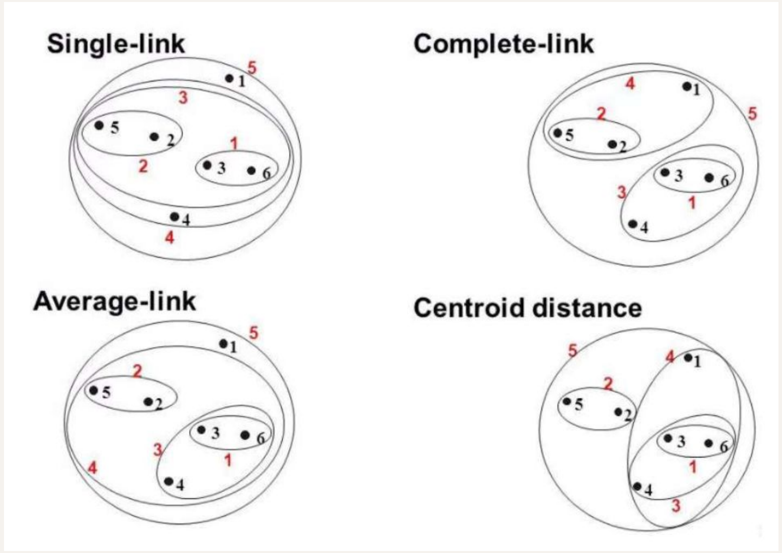
Hierarchical Linkages

- The three main types of linkages we're going to use are:
    - Single
        - Uses the minimum distance between points in clusters to merge
    - Complete
        - Uses the maximum distance between points in clusters to merge
        - This one is confusing at first, so to clarify, we're still picking the closest clusters, we're just picking "closeness" using the greatest distance between points. The minimum maximum distance.
    - Average
        - Uses the average distance between points in clusters to merge
- There is a 4th called ward linkage that sklearn uses by default. It's worth checking out if you're curious, but we won't be going over it in class. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html

Hierarchical Linkages

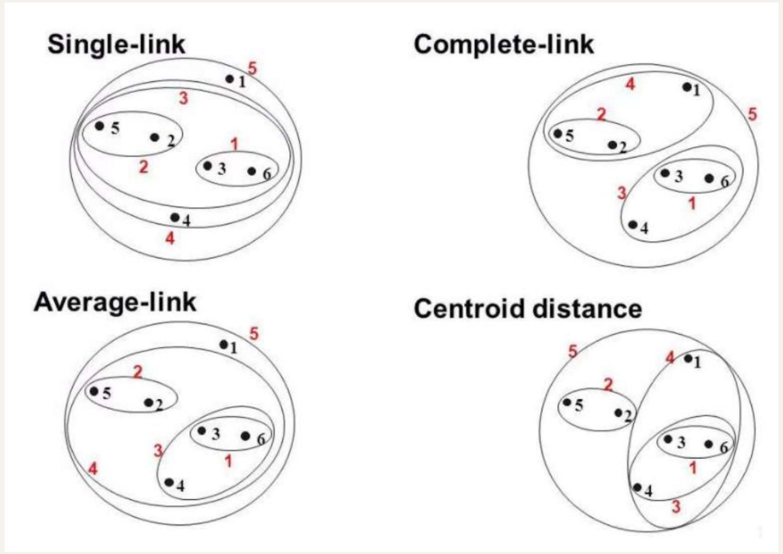
- Here are some cool diagrams of how these linkages work. I think they're good





Exercise

- 4 of you are going to come up to the whiteboard and draw dendrograms for these clusters



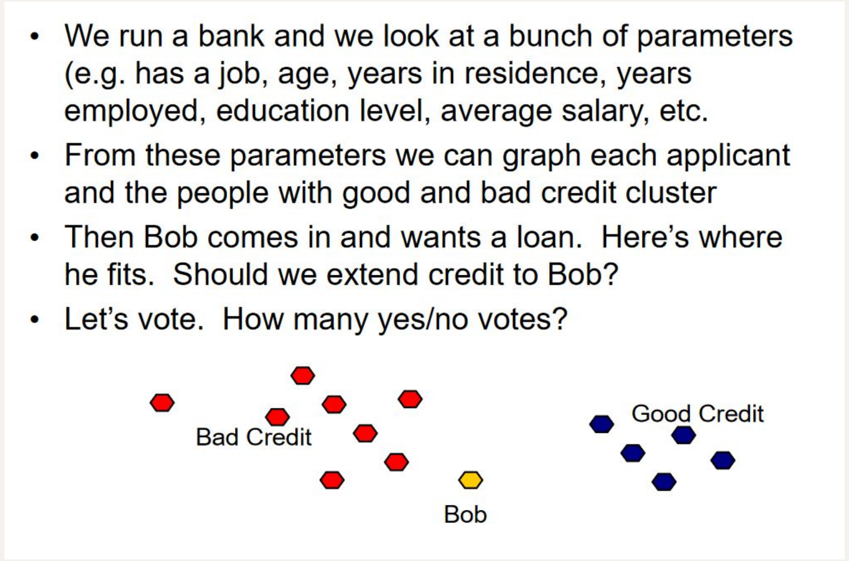
Another Exercise

- Here's the similarity that corresponds to the distance matrix we computed earlier in the class. Let's cluster it with single and complete linkages and draw the dendrograms.

	Susan	Jim	Joe	Jane	Sam	Michelle
Susan	1.000	0.469	0.130	0.242	0.645	0.870
Jim	0.469	1.000	0.518	0.092	0.681	0.434
Joe	0.130	0.518	1.000	0.277	0.342	0.095
Jane	0.242	0.092	0.277	1.000	0.125	0.295
Sam	0.645	0.681	0.342	0.125	1.000	0.610
Michelle	0.870	0.434	0.095	0.295	0.610	1.000

Importance of Linkage

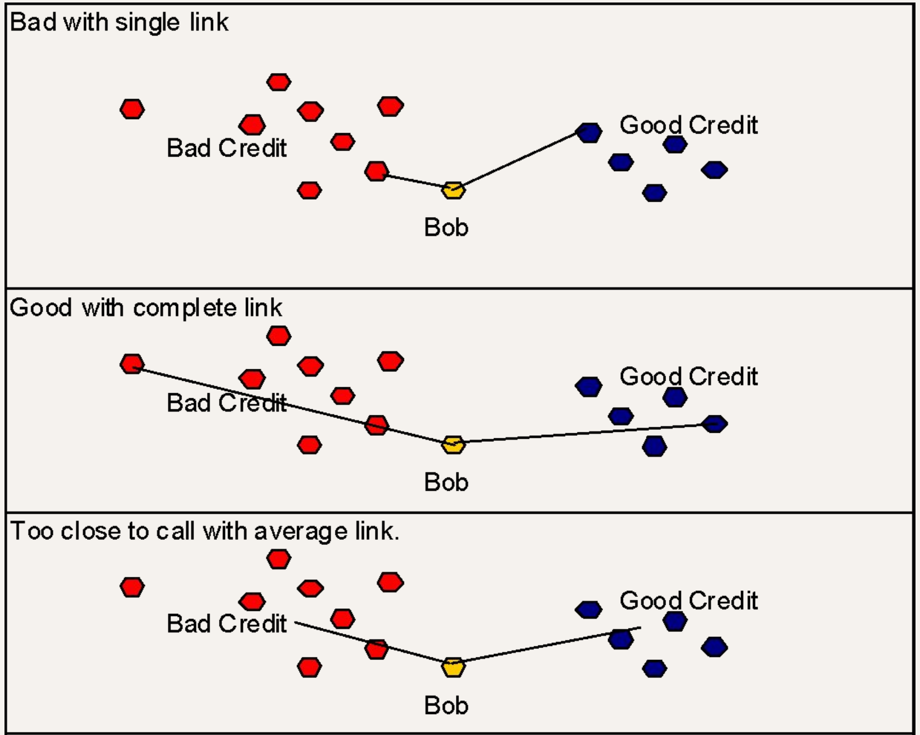
- Since different linkages create clusters using different measures, obviously the clusters are going to be different and you might not always know how to group objects together. Here's a quick example:



- We run a bank and we look at a bunch of parameters (e.g. has a job, age, years in residence, years employed, education level, average salary, etc.
- From these parameters we can graph each applicant and the people with good and bad credit cluster
- Then Bob comes in and wants a loan. Here's where he fits. Should we extend credit to Bob?
- Let's vote. How many yes/no votes?

Bad Credit
Good Credit
Bob

Is Bob a credit risk?



Note in general when using clustering for classification, we should probably not classify cases that are not clearly in one camp or the other

More on Linkage Choice

Single Pro

Good on
non-elliptical data

Complete Pro

Better at dealing
with outliers

Average Pro

Less susceptible to
noise and outliers

Single Con

More susceptible to
outliers

Complete Con

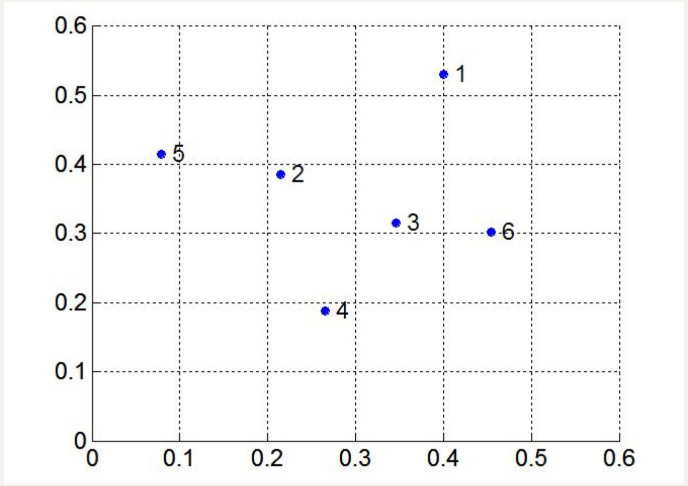
Biased towards
globular data
Has trouble with
larger clusters

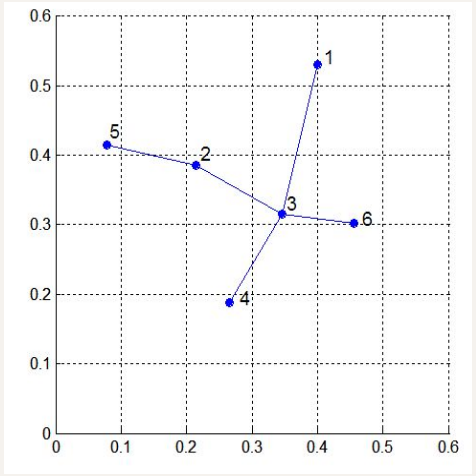
Average Con

Biased towards
globular data

MST

- Build MST (Minimum Spanning Tree)
- Delete K longest paths until you have the required number of clusters
- E.g. 4 clusters below:{(1),(4),(3,6),(5,2)}



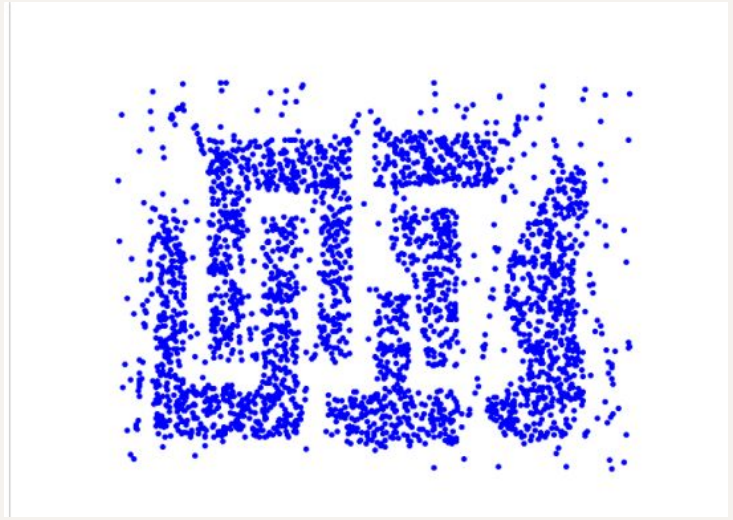


What’s a MST?

- For those of you who do not know what a minimum spanning tree is, we want to take an acyclic path (impossible to loop endlessly) between all points where the sum of all the edges is minimized
- There are several algorithms to do this, including Prim’s Algorithm, Kruskal’s Algorithm, etc. If you need a refresher on calculating minimum spanning trees, you’ll won’t need to do it by hand in this class, but it’s still good to know. Here are some videos to watch on your own time.
    - https://www.youtube.com/watch?v=4ZlRH0eK-qQ&pp=ygUmbWluaW11bSBzcGFubmluZyB0cmVlIHByaW0ncyBhbGdvcml0aG0%3D
    - https://www.youtube.com/watch?v=eB61LXLZVqs&pp=ygUmbWluaW11bSBzcGFubmluZyB0cmVlIHByaW0ncyBhbGdvcml0aG0%3D
    - https://www.youtube.com/watch?v=cplfcGZmX7I&pp=ygUmbWluaW11bSBzcGFubmluZyB0cmVlIHByaW0ncyBhbGdvcml0aG0%3D

Density/DBSCAN

- We saw earlier that K-Means can’t handle data of different densities or non-globular data. Luckily, we have DBSCAN to (mostly) save the day.
- This is used for groups of high density data that are separated by low density regions, as pictured below:



DBSCAN

- DBSCAN stands for Density Based Spatial Clustering of Applications with Noise
- Density refers to the number of points (MinPts) within a radius (Eps)
- A point is considered a core point if it has a certain number of points within that radius of it. These will generally be the points that are found more on the interior of each cluster
- A border point is close to a core point, but isn't one
- A noise point is neither core, nor border

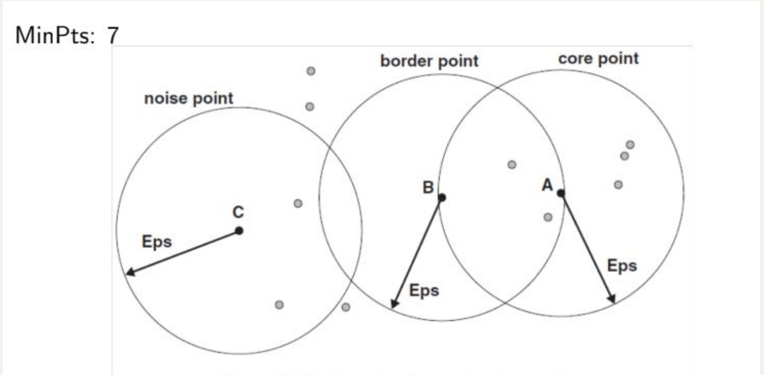


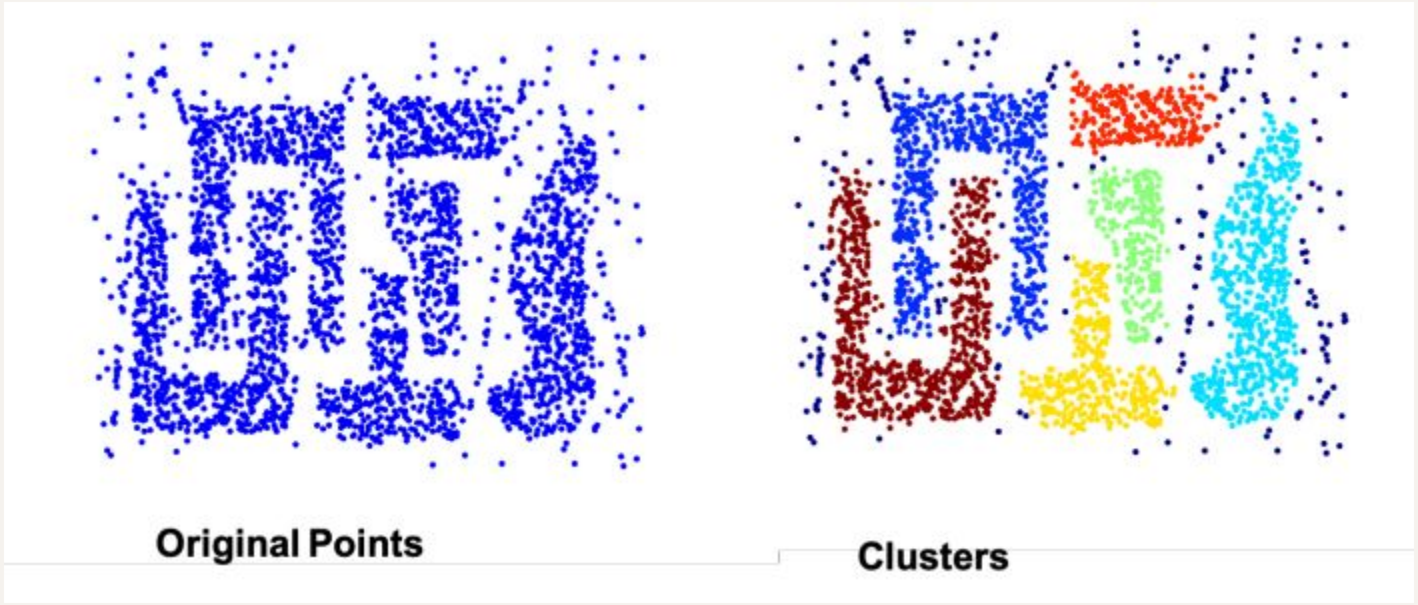
Figure 7.21. Core, border, and noise points.

DBSCAN Steps

- Form clusters using core points, and assign border points to one of its neighboring clusters
  - Label all points as core, border, or noise points.
  - Eliminate noise points.
  - Put an edge between all core points within a distance Eps of each other.
  - Make each group of connected core points into a separate cluster.
  - Assign each border point to one of the clusters of its associated core points

DBSCAN Strengths

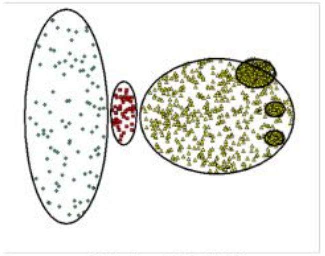
- As we discussed, DBSCAN is good on non-globular data. It’s also resistant to noise and outliers and works well on data of different size and shapes

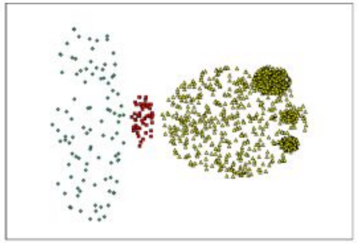


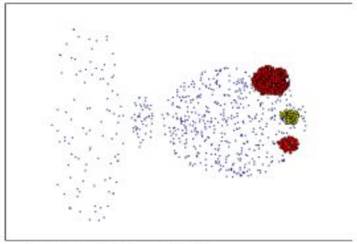
DBSCAN Weaknesses

- Unfortunately, DBSCAN isn't very strong on data with multiple densities and high dimensionality

- Varying densities
- High-dimensional data


Original Points


(MinPts=4, Eps=9.92).


(MinPts=4, Eps=9.75)

Review Quiz

- What type of clustering should we use if we know the data has a lot of outliers?
- True/False: The centroid is the point in the dataset at the middle of each cluster.
- True/False: K-Means will always give you the same clusters when you run with the same dataset
- True/False: Hierarchical will always give you the same clusters when you run with the same dataset
- Which types of clustering work well on globular/non-elliptical data?