McMaster University ENGINEERING

Chemical Engineering 4H03

Introduction to Clustering

Jake Nease
McMaster University

Where are We?

- We have covered some introductory material
  - Visualization
  - Types of data
- We have covered several supervised machine learning methods and tools good for mapping inputs (**X**) to continuous outputs (**Y**)
  - Regression
  - PCA/PCR/PLS
  - ANNs
- We have not yet looked at a modeling method for classification, NOR have we looked at unsupervised learning

Objectives for this Topic

- We would like top introduce one of the oldest and most approachable unsupervised learning methods: **clustering**

- Specifically, we will investigate K-Means clustering
  - What is clustering?
  - Motivating example
  - K-Means clustering
  - Training procedure
  - Coded demo in MATLAB
  - Benefits/drawbacks

- This will lead us to a more holistic conversation around other clustering techniques and why we may use them

Motivation

- Imagine for a moment that you are building a model to predict an output according to some inputs **X**, BUT the general "location" of the inputs affects the model's performance

- Examples:
  - A bank predicting which financial products a customer can afford
  - NETFLIX recommending international films for my wife and Parks/Recreation for me
  - Trying to identify if a person is in an "at-risk" domestic situation
  - Wanting to predict the performance of a chemical or manufactured product, but the properties and performance are very different depending on the product's composition
  - Others?

Clustering

- **Clustering** the process of dividing data into subgroups or "clusters" that exhibit local similarities in data
  - For us, that means they have similar values in columns of **X**
- This can be very powerful if, for example, our data exhibits piecewise-linear correlation
  - Then we can train separate PCA models that are more accurate locally, and then assign new points to the appropriate "clustered model" as they arrive
- I'll gently point out here that most examples are two-dimensional
  - BUT, practically speaking any number of dimensions (including one!) is permissible

Unsupervised Learning

- As mentioned before, clustering is a form of **unsupervised learning**, whereas everything else in 4H to this point has been **supervised learning**

- Supervised learning can compare the output of a model to a known and desired output
  - "Training"

- Unsupervised learning does not have an output to compare to, and must come up with that on its own
  - How many clusters?
  - Where to place the clusters?
  - How to ensure the clusters are behaving as they should?

Unsupervised Learning

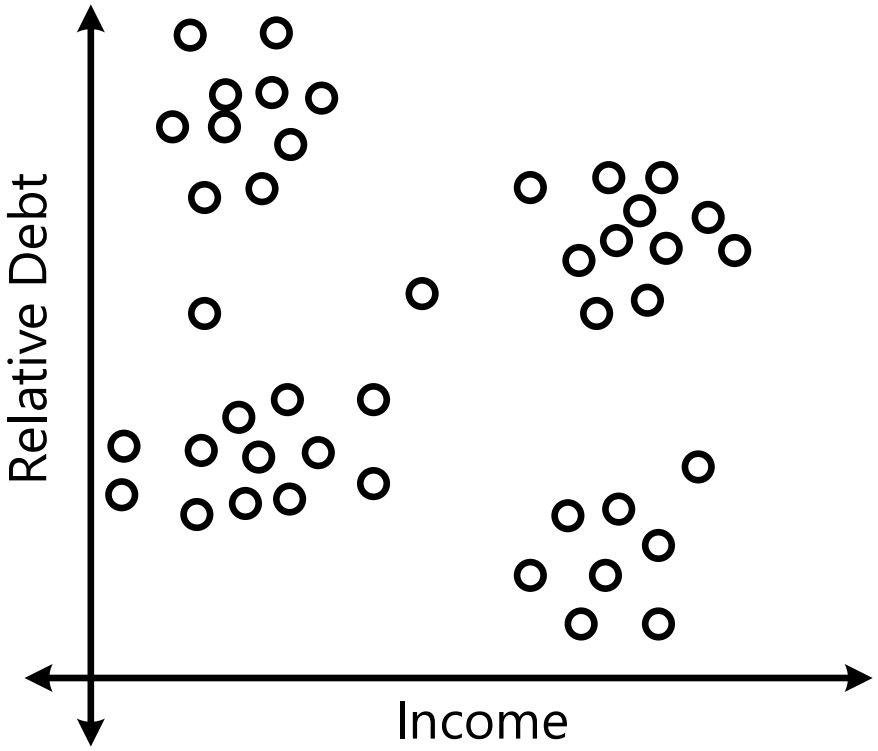
- As mentioned before, clustering is a form of **unsupervised learning**, whereas everything else in 4H to this point has been **supervised learning**

- Supervised learning can compare the output of a model to a known and desired output
  - "Training"

- Unsupervised learning does not have an output to compare to, and must come up with that on its own
  - How many clusters?
  - Where to place the clusters?
  - How to ensure the clusters are behaving as they should?

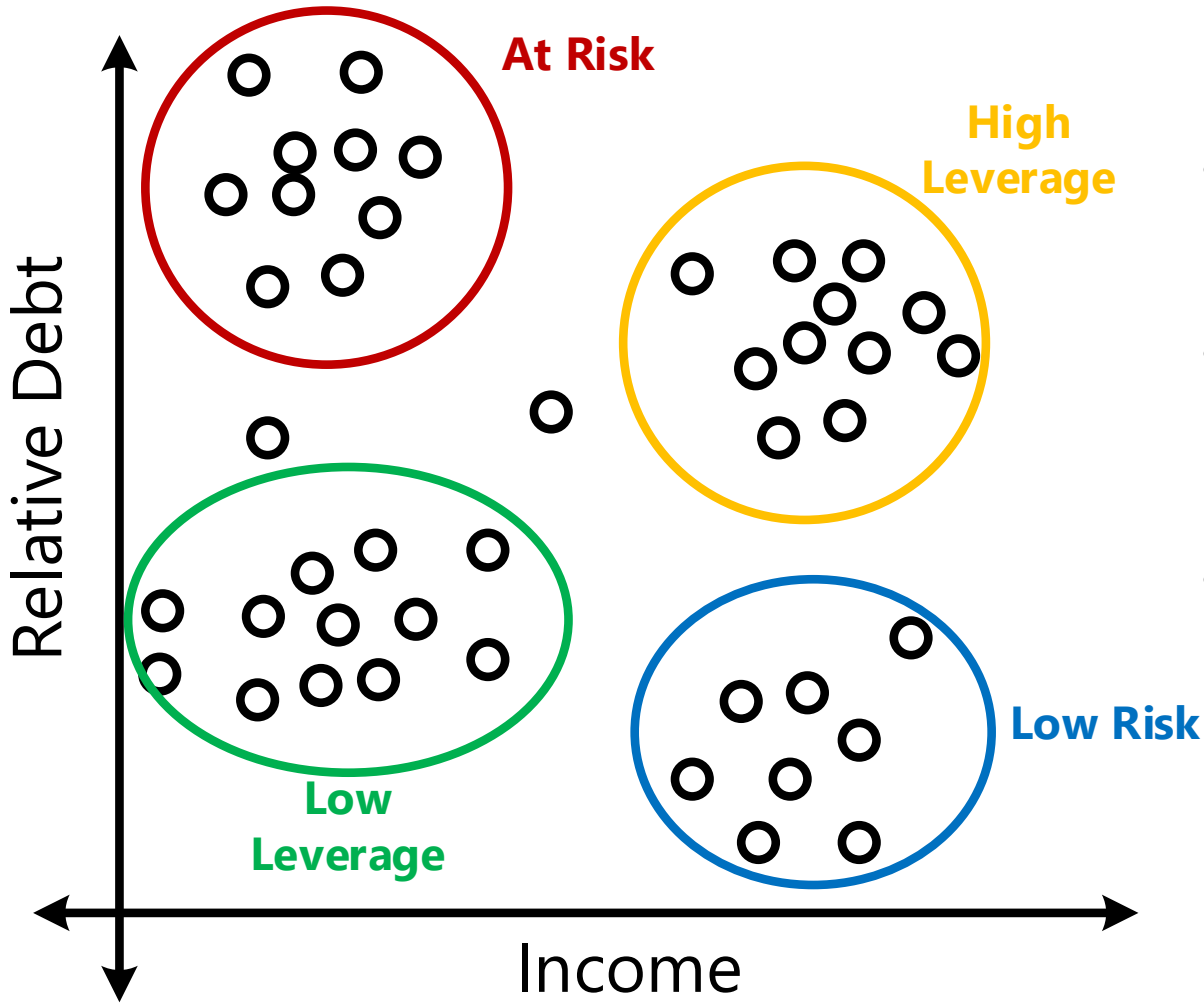
A Motivating Example

- A bank is looking at the **debt** and **income** levels of customers. With this knowledge, they may have several objectives:
  - Ensuring minimal risk to the bank for giving loans
  - Identifying at-risk customers
  - Suggesting the right product based on financial stability



A Motivating Example

- You might visually separate the data this way



- What about those middle points?
- Could we use more than four clusters in this case?
- I will also gently mention that this data is a good candidate for support-vector machines

A Motivating Example

- Some logical next questions:
  - How do we find those “ellipses”?
  - What do we use as an initial guess?
  - What is the terminology?

- I’ll mention here that there are different forms of clustering:
  - **Partitional Clustering** (divides data into non-overlapping groups) and is the **topic of focus for 4H**
  - **Hierarchical Clustering** (builds a decision-tree style hierarchy to cluster based on the relationships in the data)
  - **Density-Based Clustering** (determines the number and locations of clusters based on maximizing the density of each cluster and separating them by low density regions)

McMaster University ENGINEERING

Fitting Clusters

Now I want granola. Thanks, marketing.

What Makes a Good Cluster?

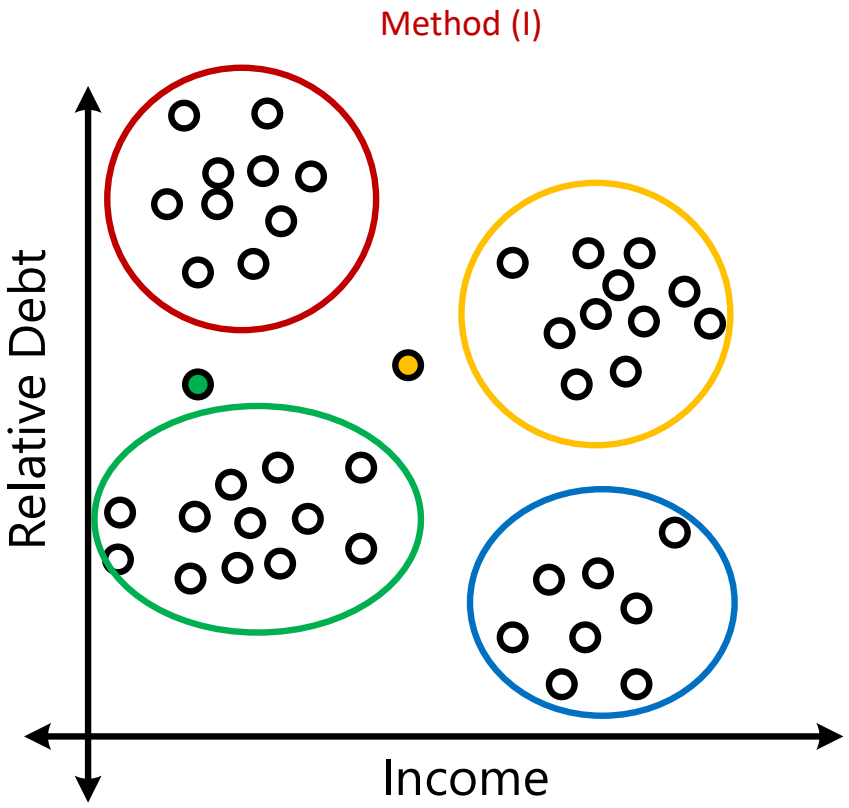
- For a method to successfully define clusters, it must follow two rules:

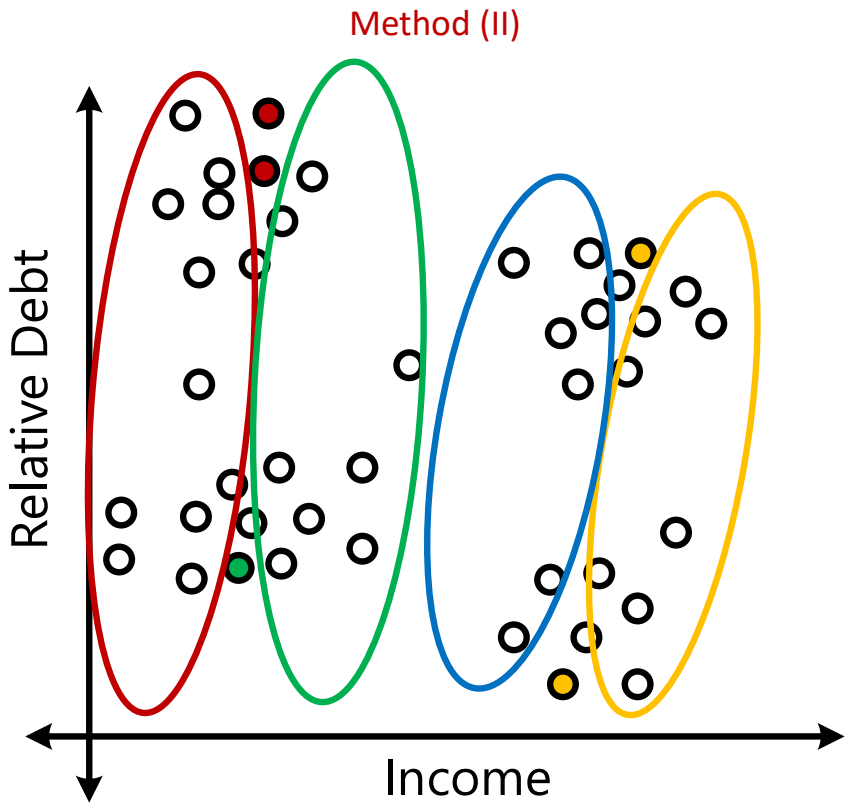
1. The data in a given cluster must be similar to each other
   - This corresponds to having similar values in the columns of **X**
   - Visually, it means the data are gathered in local subspaces of **X**

2. The data that belong to different clusters must be as different as possible
   - A good clustering method delineates between subspaces as much as it focuses on identifying them in the first place

Back to the Example

- Consider the data from before with two clusters sets:
  - What are all the reasons method (II) is inferior to (I)





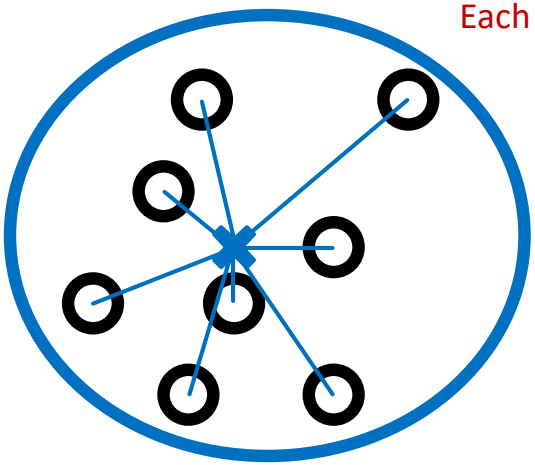
Measuring Cluster Performance

- Our example contains only two columns in **X**
  - So, it is pretty easy to visually verify the clusters
- When we have many more columns (especially >3D), we require mathematical representations of cluster quality
- Inertia calculates the intra-cluster distance between points as Euclidian distances
  - Used to measure how “similar” the points in a cluster are
- The Dunn Index measures the distance between clusters
  - Used to measure how “different” any cluster is from another

Measuring Cluster Performance

- **Inertia** of cluster c_k is nothing more than the cumulative distance from all points in a cluster to the cluster centroid

$$\mathcal{I}_k = \sum_{i \in c_k} \|\boldsymbol{d}_i\| = \sum_{i \in c_k} \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|$$



- $\mathcal{I}_k$ is the inertia for cluster c_k
- $\boldsymbol{x}_i$ is observation i
- $i \in c_k$ means that point i belongs to cluster c_k
- $\boldsymbol{\mu}_k$ is the centroid (dimensional average) of cluster c_k

Objective Function of K-Means

- The **objective** of K-means clustering is to minimize the cumulative squared inertia by assigning points $\boldsymbol{x}_i$ to cluster c_k for $k = 1 \dots K$ clusters:
- In general, we say that we want to partition the data according to a set of K clusters $C = \{c_1, c_2, \dots, c_K\}$ to minimize:

$$\min_C \phi = \sum_{k=1}^{K} \sum_{i \in c_k} \|\boldsymbol{d}_i\|^2 = \sum_{k=1}^{K} \sum_{i \in c_k} \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|^2$$

Yup, it's basically minimizing the sum of squared errors again :3

- C is the set of clusters
- $\boldsymbol{x}_i$ is observation i
- $i \in c_k$ means that point i belongs to cluster c_k
- $\boldsymbol{\mu}_k$ is the centroid (average) of cluster c_k

A Couple of Notes

- Each observation $\boldsymbol{x}_i$ has the same number of columns according to our data $\mathbf{X}$
  - Thus, the "location" of $\boldsymbol{x}_i$ can be imagined as a "location" in N dimensions, most easily visualized by two dimensions
- Each centroid $\boldsymbol{\mu}_k$ also has the dimension as $\boldsymbol{x}_i$ and can be thought of as the center of mass of the data cloud belonging to c_k if all points "weigh" the same
- The centroid of a cloud of data c_k is thus:

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i \in c_k} \boldsymbol{x}_i$$

- In the above expression, N_k is the number of points in cluster c_k
- Thus, the vector $\boldsymbol{d}_i$ is the same dimension as $\boldsymbol{x}_i$ and $\boldsymbol{\mu}_k$, and represents a vector drawn from the centroid $\boldsymbol{\mu}_k$ to $\boldsymbol{x}_i$

The K-Means Procedure

INITIALIZATION

- Let $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots \boldsymbol{x}_i, \boldsymbol{x}_{i+1}, \cdots \boldsymbol{x}_N\}$ be a set of CENTERED/SCALED data
- Commit to K clusters (this can be chosen adaptively later)
- Let $\mathrm{M}^{(0)} = \left\{\boldsymbol{\mu}_1^{(0)}, \boldsymbol{\mu}_2^{(0)}, \cdots \boldsymbol{\mu}_k^{(0)}, \boldsymbol{\mu}_{k+1}^{(0)}, \cdots \boldsymbol{\mu}_K^{(0)}\right\}$ be a set of cluster centers for iteration $j = 0$
- Randomly assign vectors in X as the initial guesses for M

ALGORITHM

1. Compute the distance $\boldsymbol{d}_{i,k}$ from each point $\boldsymbol{x}_i$ to each center $\boldsymbol{\mu}_k^{(j)}$
2. Assign each point $\boldsymbol{x}_i$ to $c_k^{(j)}$ by selecting the k such that $\boldsymbol{d}_{i,k}$ is minimized for all i. Assign this to value $\boldsymbol{d}_i$ and let $i \in c_k^{(j)}$
   - IF no points x_i were assigned a different $c_k^{(j)}$ than $c_k^{(j-1)}$, STOP. The current set of clusters is locally optimal
   - ELSE, proceed to (3)
3. Calculate the new center of mass of each cluster according to the points $\boldsymbol{x}_i$ contained in c_k : $\boldsymbol{\mu}_k^{(\boldsymbol{j+1})} = \frac{1}{N_k}\sum_{i \in c_k} \boldsymbol{x}_i$
4. Update the iteration counter $j = j + 1$ and return to (1)

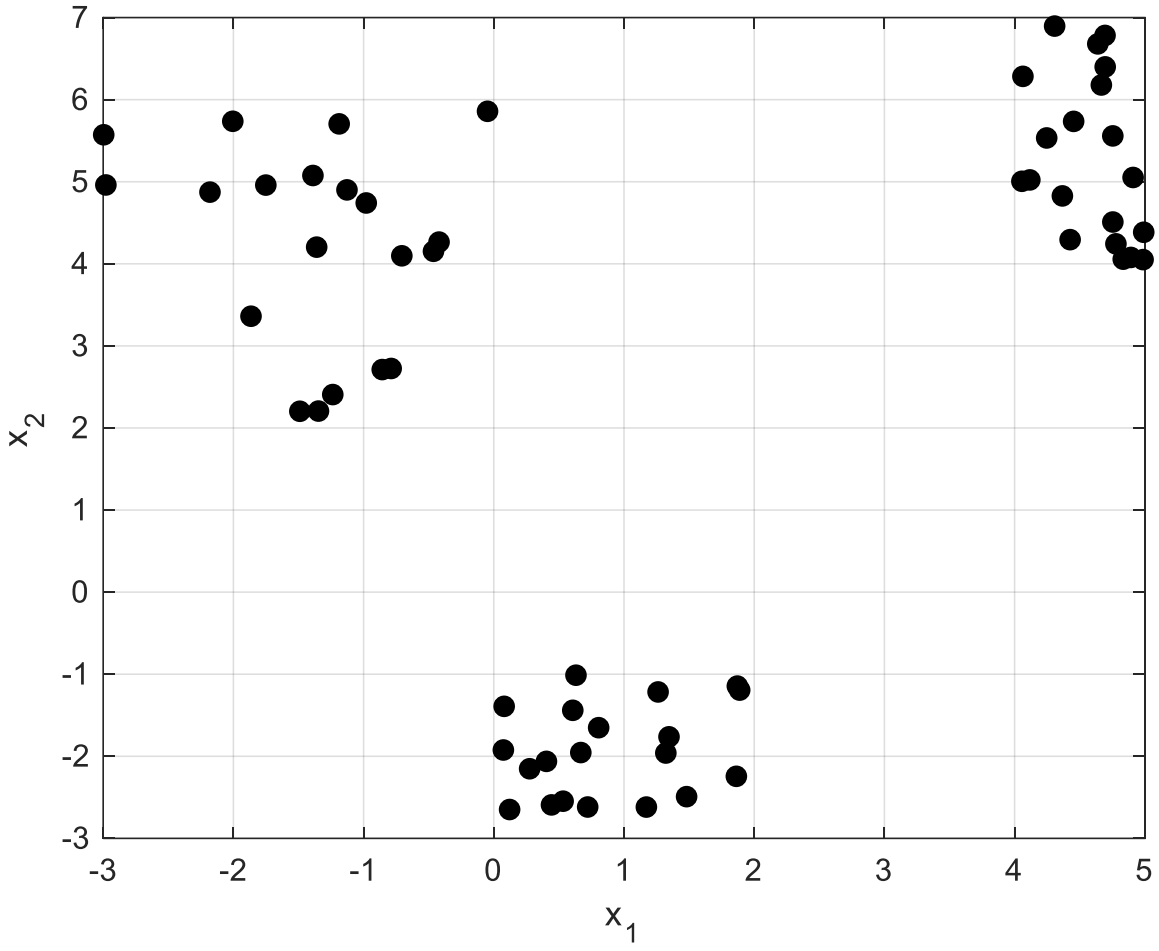
WHEW. In English, Please?

ALGORITHM

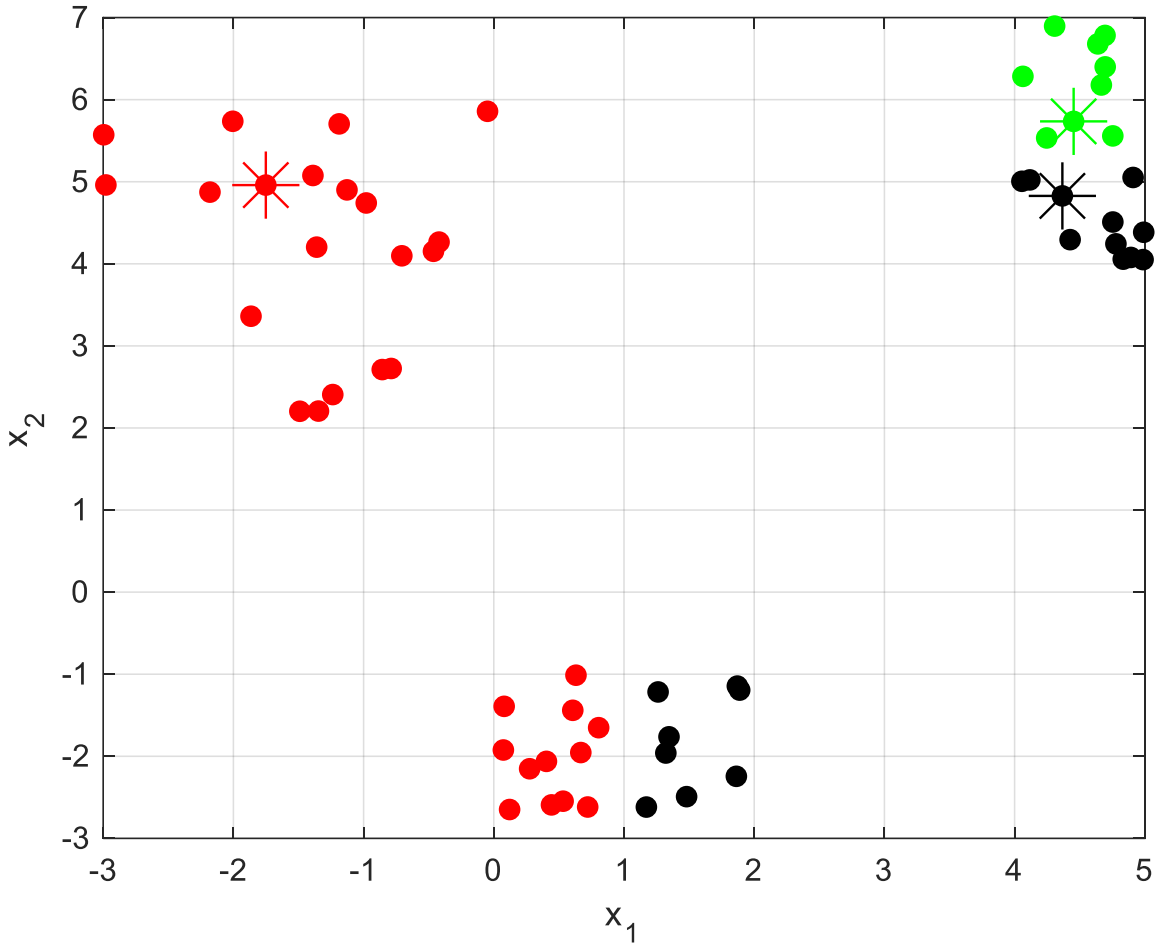
1. Find the distance from each point to each cluster center
2. Whichever center the point is closest to, consider the point as part of that cluster
   - IF the same points are in all clusters, they will stay that way forever, so stop
   - ELSE, proceed to (3)
3. Chances are some points changed clusters. If this is the case, we compute a new cluster center of mass (noting that it does NOT have to be the same as a specific point)
4. Then do the whole dang thing again

Example in `MATLAB`

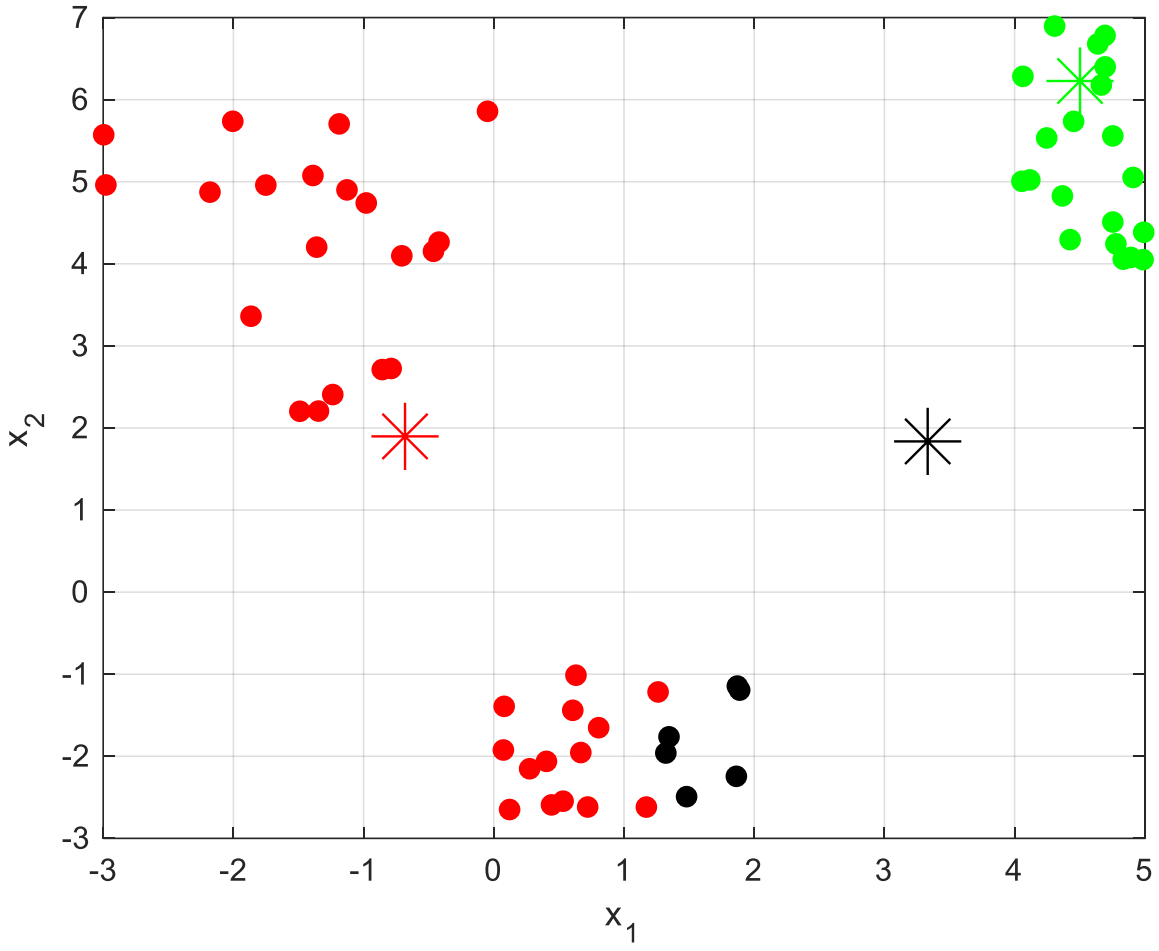
Graphically



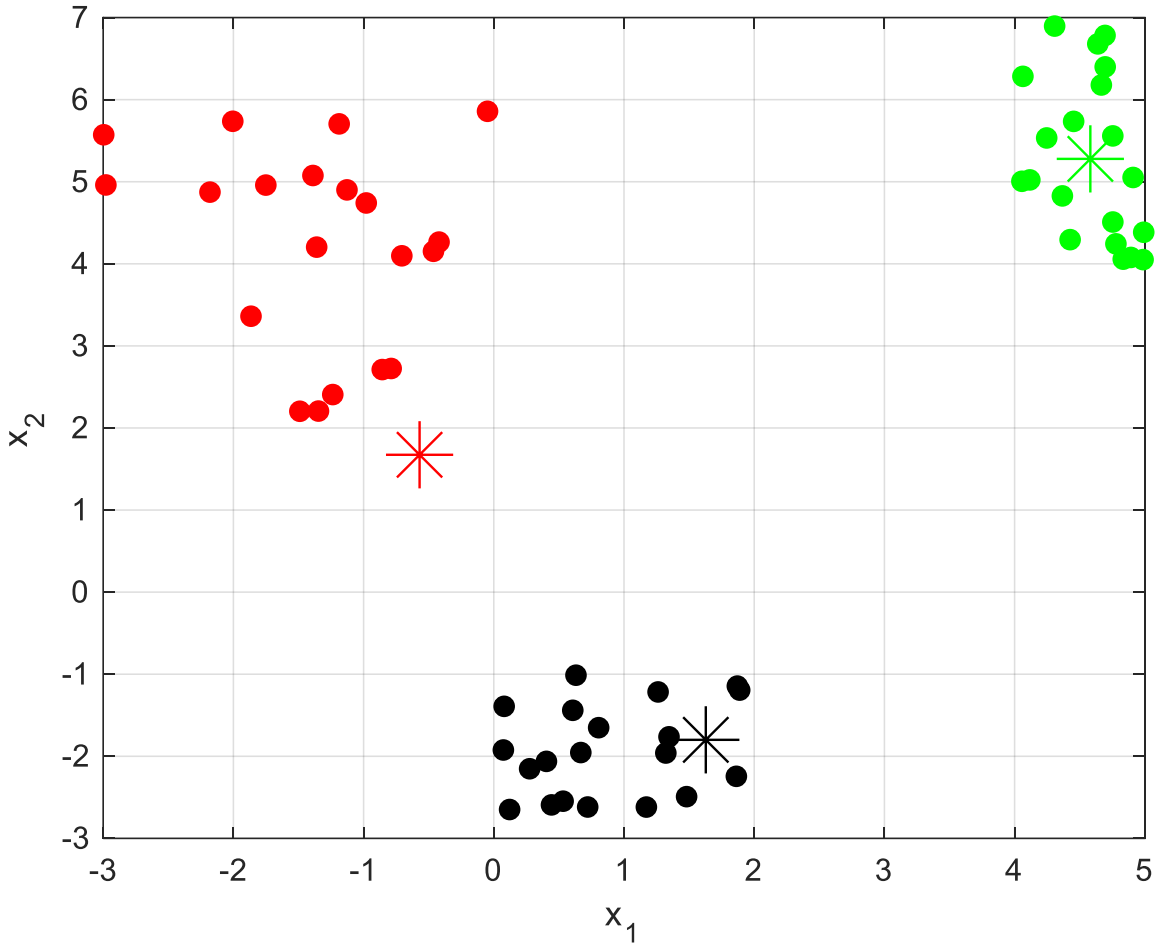
Graphically



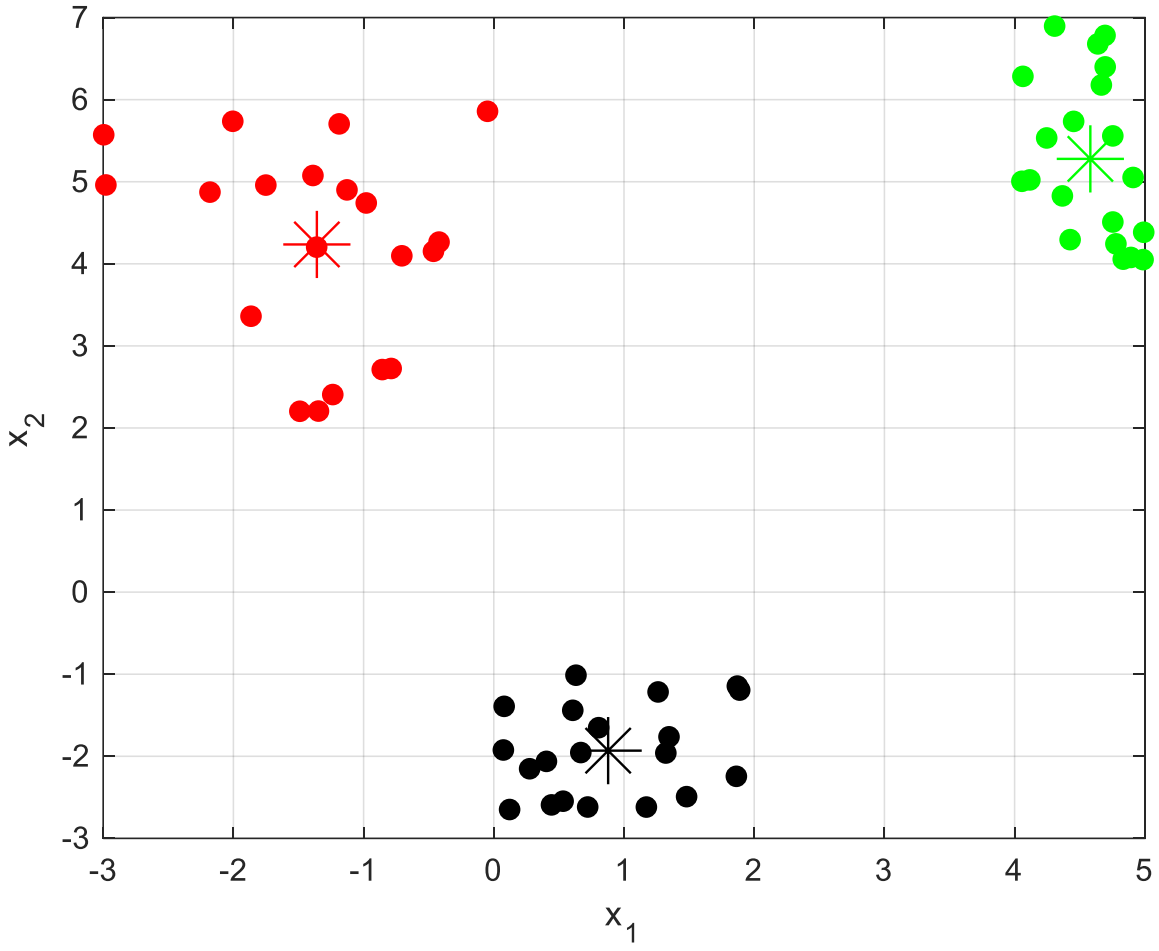
Graphically



Graphically



Graphically



McMaster University ENGINEERING

Benefits/Drawbacks of K-Means

Don’t feel bad. No one is perfect.

K-Means Clustering Boons/Busts

- Advantages of K-Means
  - Relatively easy to use
  - Gives us a (locally) optimal solution to the least-squares problem
  - Effective if data have high degree of separation
  - Relatively efficient computationally

- Disadvantages of K-Means
  - Need to know number of clusters beforehand
  - Highly overlapping data may be misclassified
  - Works only for continuous data (requires $\boldsymbol{\mu}_k$)
  - Requires centering/scaling to avoid biasing distances
  - Data sets of different sizes/shapes can also lead to bad clustering
  - Randomly choosing initial clusters can lead to bad results

How Many Clusters?

- Since K-Means requires us to know the number of clusters ahead of time, it is often not immediately obvious how many clusters to use
  - It is pretty easy to visually verify in 2D, but (as we know) many data sets have many columns in **X**

- A strategy for determining the appropriate number of clusters is to track the **total inertia** $\mathcal{I}_K$ of the converged clusters as a function of number of clusters K

$$\mathcal{I}_K = \sum_{k=1}^{K} \mathcal{I}_k$$

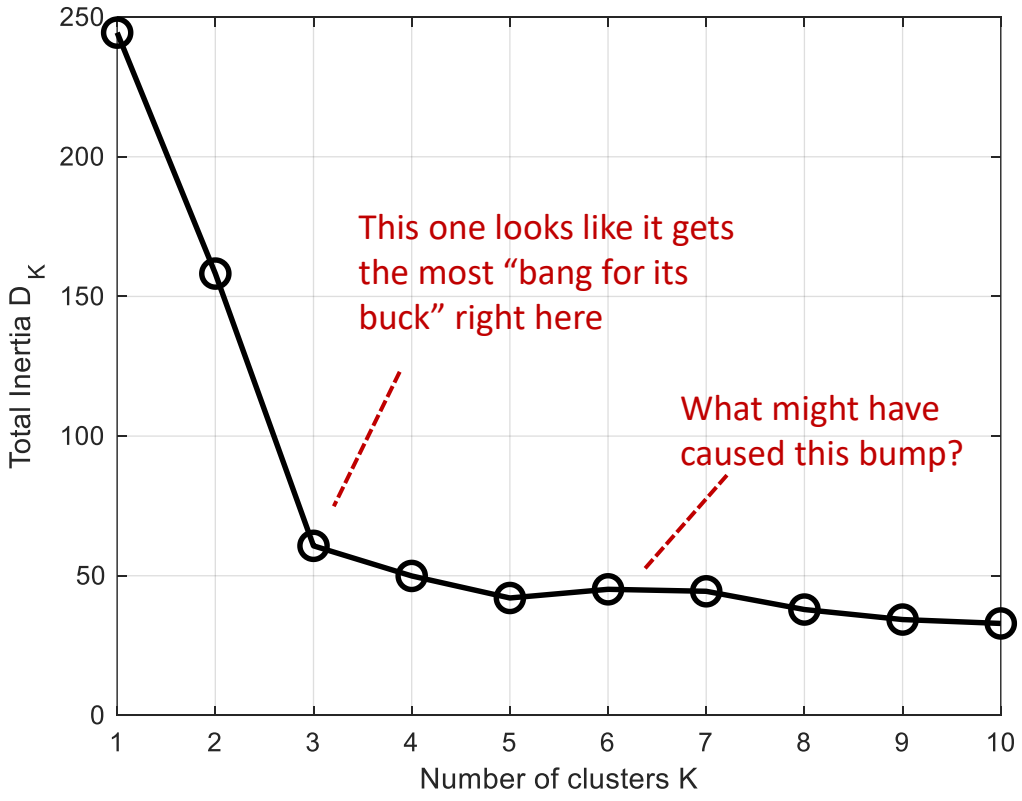
- The total inertia $\mathcal{I}_K$ is computed once the K clusters have CONVERGED

The Elbow Plot

- One can visualize the total inertia $\mathcal{I}_K$ as a function of K
- The below plot is for our example (can be much more bendy but three is clearly the right number here)

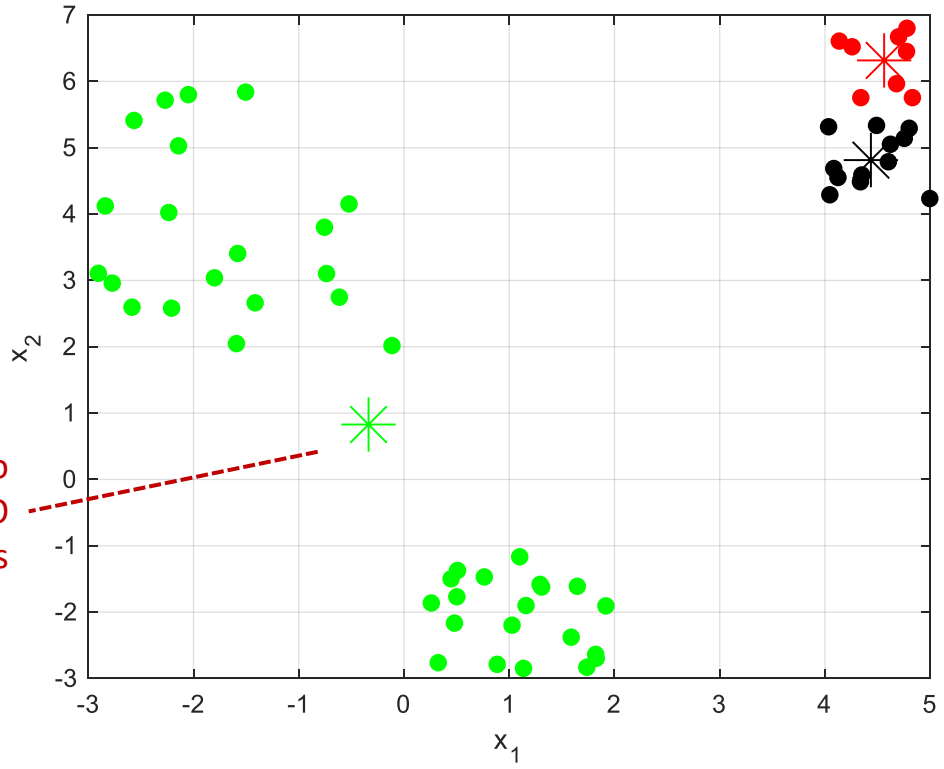
- Rule of thumb: when the slope of this plot reaches it's "elbow," stop using additional clusters

- Generally, this is represented by a sudden change in slope that **continues linearly**



Random Initialization

- Sometimes random initialization will lead to bad clusters
  - Can be caused from bad initial placement of cluster centers



This cluster seems to be explaining TWO distinct clusters

This is clearly not the best use of these clusters

Better Initialization: K++

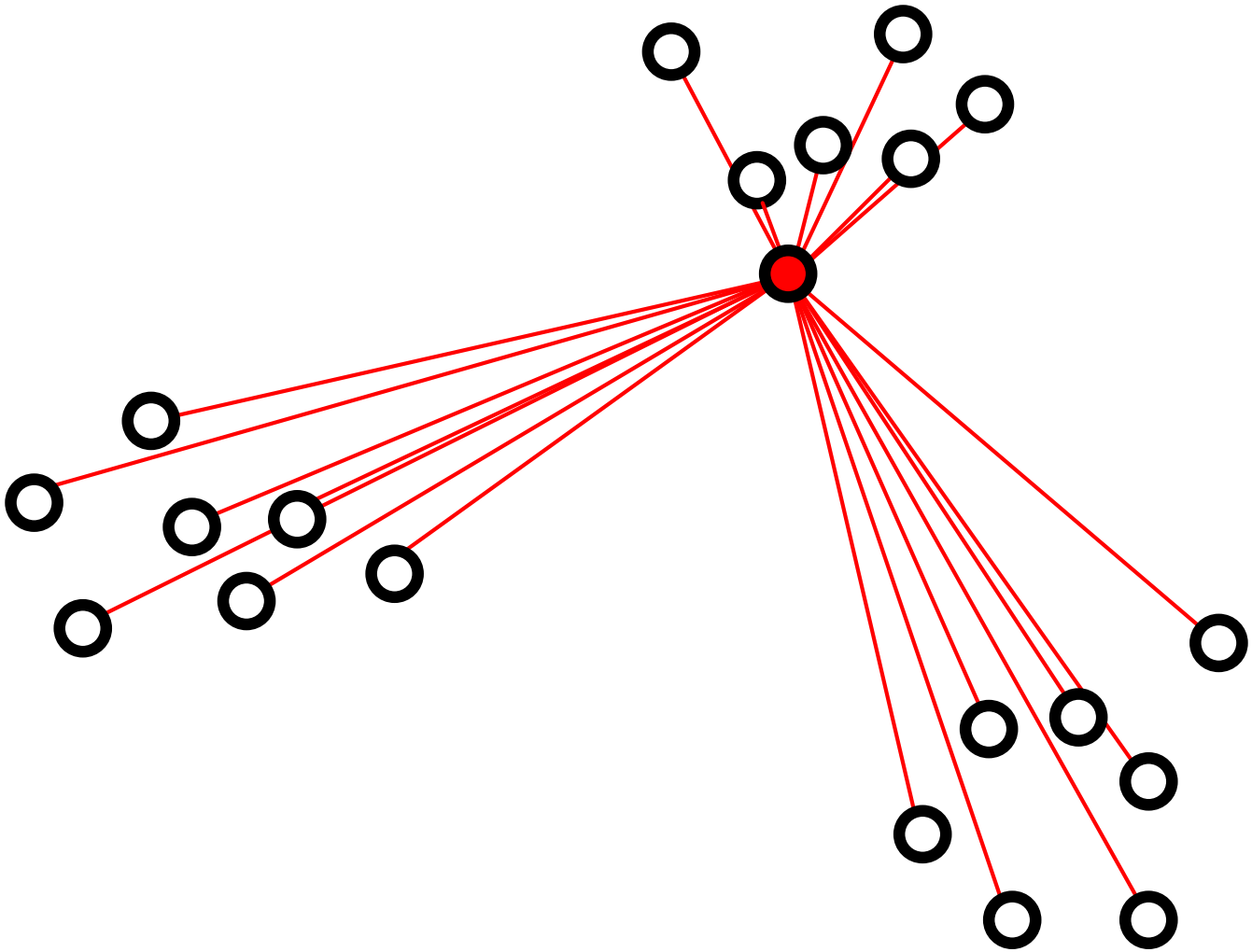
- K++ Initialization attempts to start the clusters as far apart as possible (initialization of M)
  - Intended to prevent accidental segregation of data that should not be separated
  - Works very well if the clusters are already well separated
  - More computationally expensive

Algorithm to select initial $\mathrm{M}^{(0)} = \left\{\boldsymbol{\mu}_1^{(0)}, \boldsymbol{\mu}_2^{(0)}, \cdots \boldsymbol{\mu}_k^{(0)}, \boldsymbol{\mu}_{k+1}^{(0)}, \cdots \boldsymbol{\mu}_K^{(0)}\right\}$

1. Select a random point $\boldsymbol{x}_i$ as $\boldsymbol{\mu}_1^{(0)}$, let $\mathrm{M} = \left\{\boldsymbol{\mu}_1^{(0)}\right\}$ and set $k = 1$
2. Compute all $d_{i,k}$ for all $\boldsymbol{x}_i$ to all $\boldsymbol{\mu}_k^{(0)}$ and assign each point $\boldsymbol{x}_i$ to a cluster $c_{k'}$ recording $\boldsymbol{d}_i$ as the inertia of $\boldsymbol{x}_i$ to the nearest cluster center $\boldsymbol{\mu}_k^{(0)}$
3. Assign the point $\boldsymbol{x}_i$ with highest resulting d_i (furthest point from any cluster) as $\boldsymbol{\mu}_{k+1}^{(0)}$ and let $\mathrm{M} = \{\mathrm{M}, \boldsymbol{\mu}_{k+1}^{(0)}\}$
4. IF $k + 1 = K$
   - **TRUE**: let M be the initial set of cluster centers
   - **FALSE**: let $k = k + 1$ and return to (2)

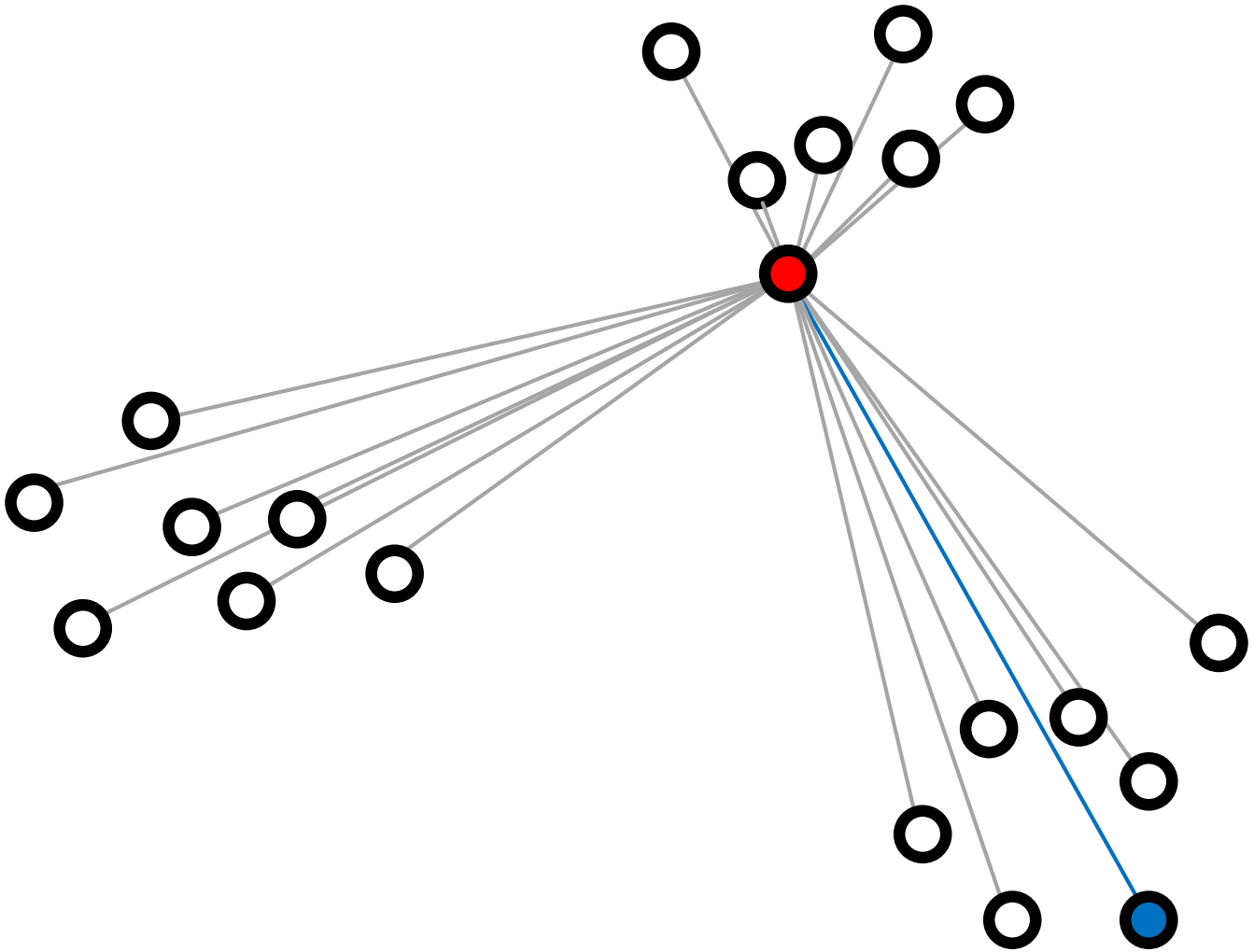
K++ Initialization Graphically

- First center chosen at random



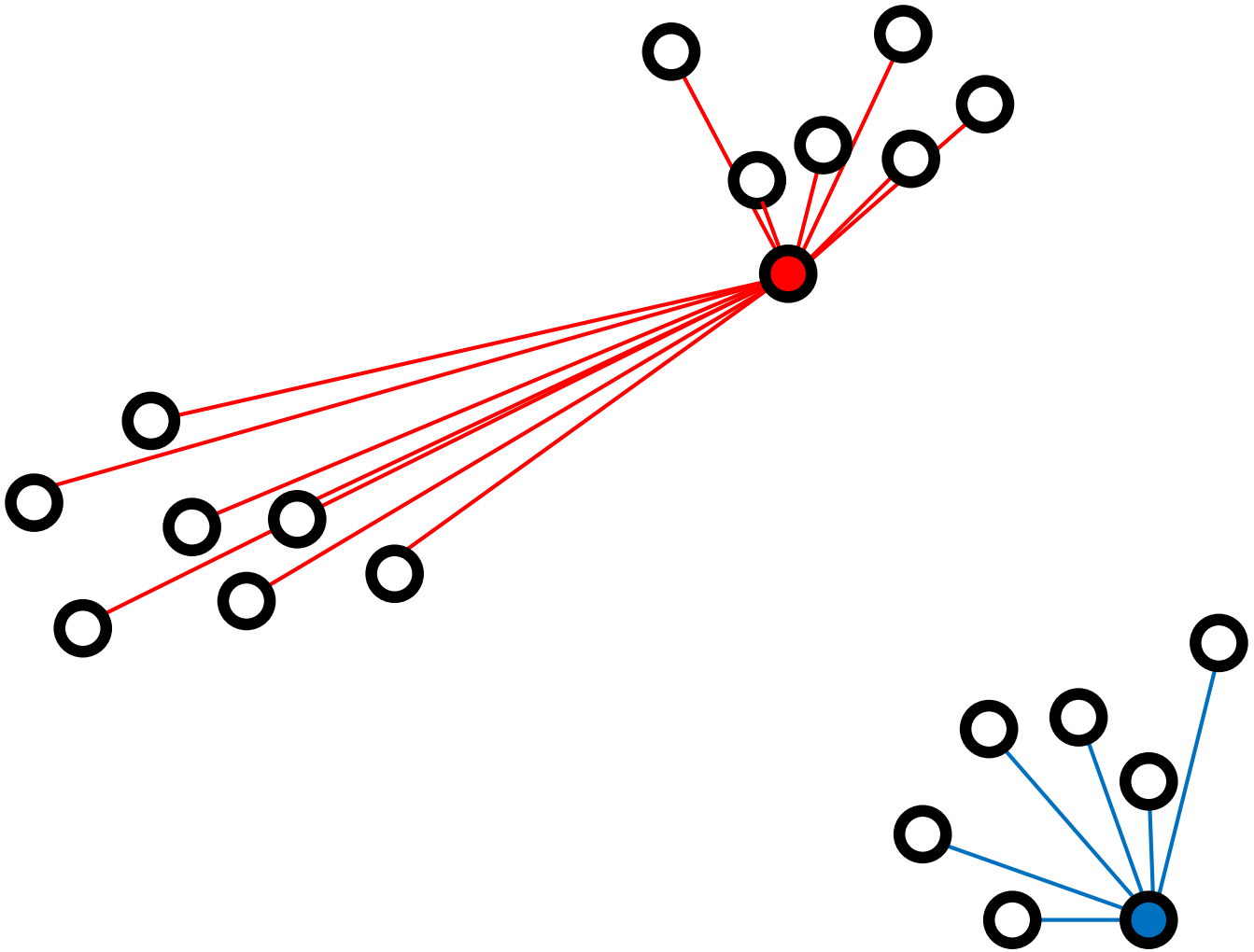
K++ Initialization Graphically

- Identify longest distance and assign as new center



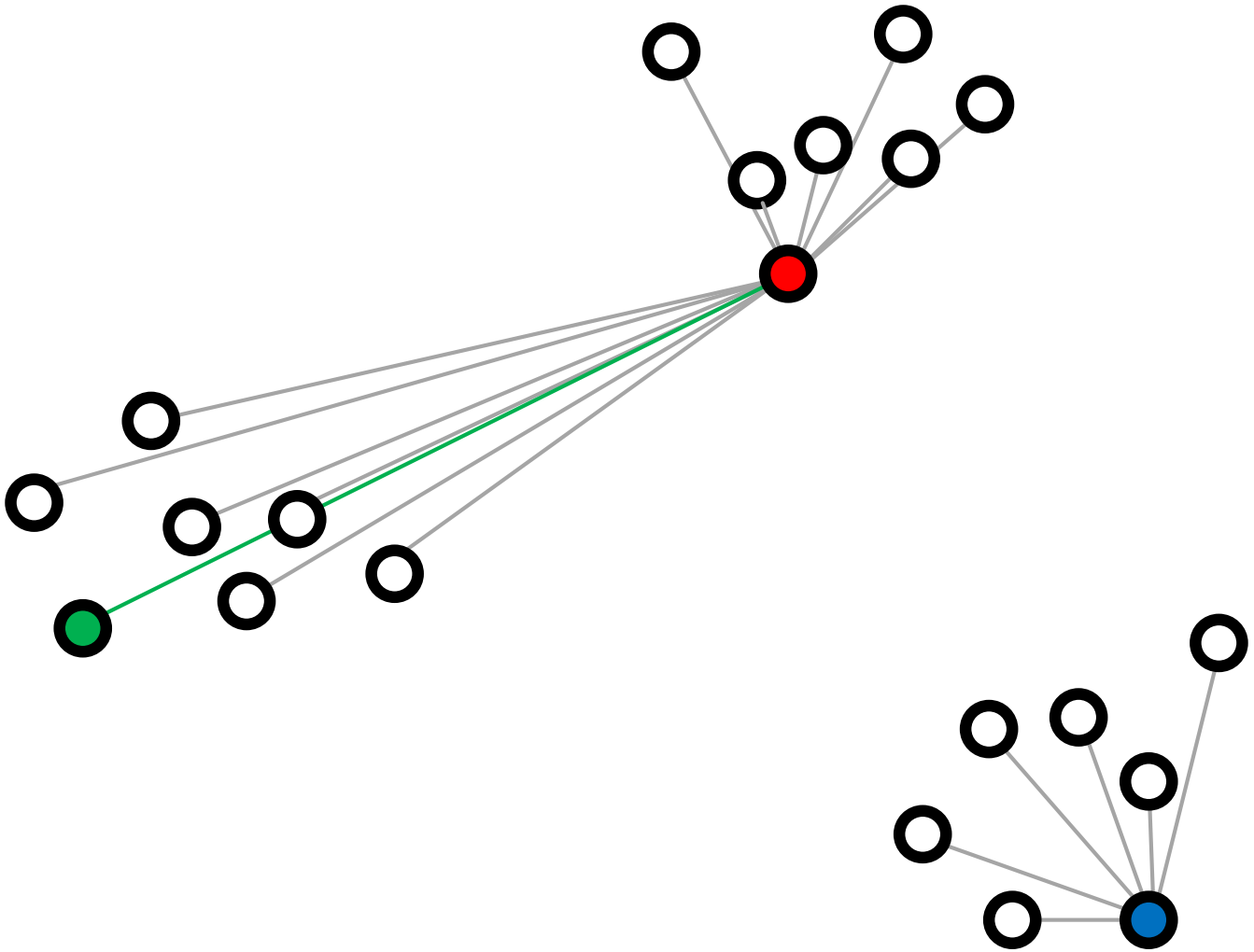
K++ Initialization Graphically

- Assign to closest center (again)



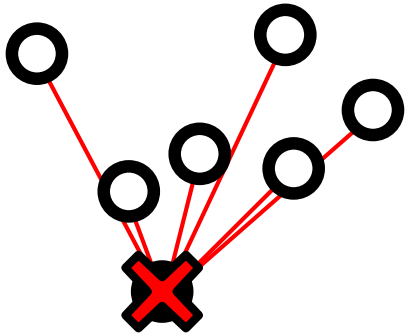
K++ Initialization Graphically

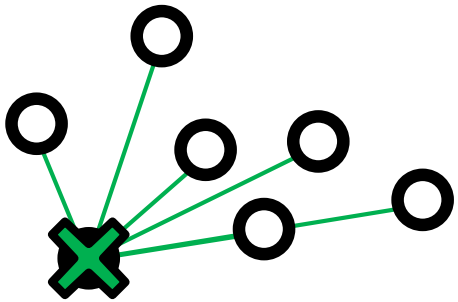
- Identify longest distance and assign as new center

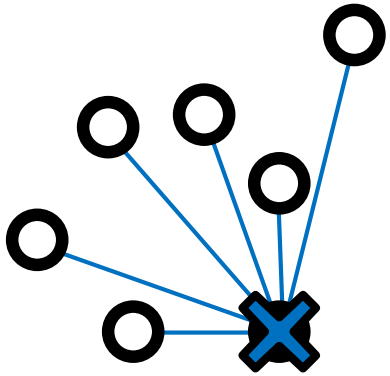


K++ Initialization Graphically

- Begin K-Means procedure with K cluster centers







Measuring Cluster Quality

- The "quality" of clusters can be used to see how compact a set of clusters is

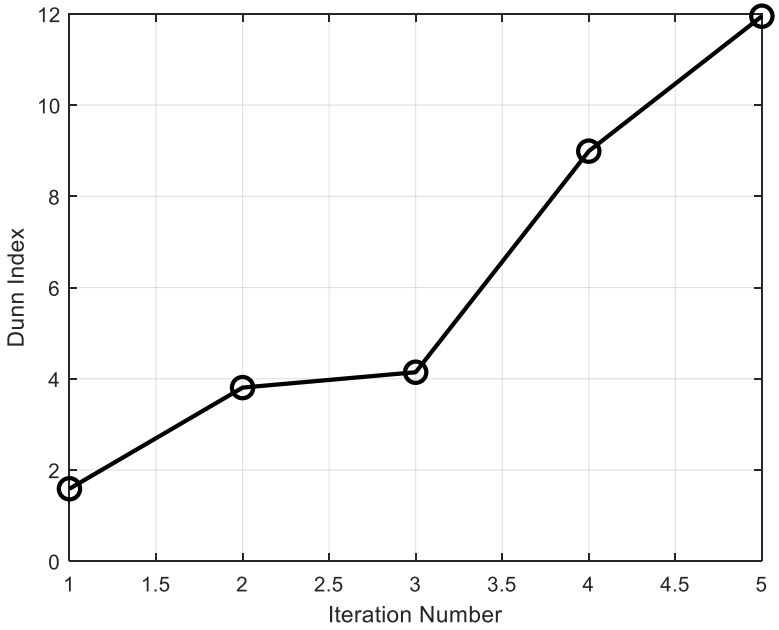
- The metric used to measure this is known as the Dunn Index
  - Measures the ratio between the **closest two clusters** in the set of clusters versus the **maximum single inertial distance** contained by one cluster
  - Our objective is to maximize the Dunn Index $\mathcal{D}$:
  - $\mathcal{D} = \frac{\min(\text{inter-cluster inertia})}{\max(\text{intra-cluster inertia})}$

Nothing more than the distance between all cluster centers

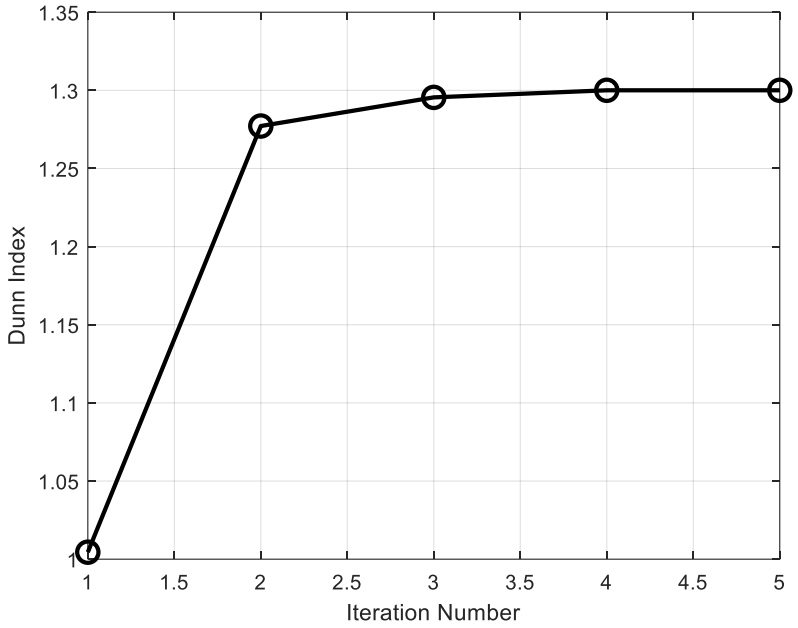
$$\mathcal{D} = \frac{\min\left(\left\|\mu_k - \mu_j\right\| \forall\, k, j \neq k\right)}{\max(d_i \,\forall\, i)}$$

For Our Data Set

Using K = 3 Clusters



Using K = 5 Clusters



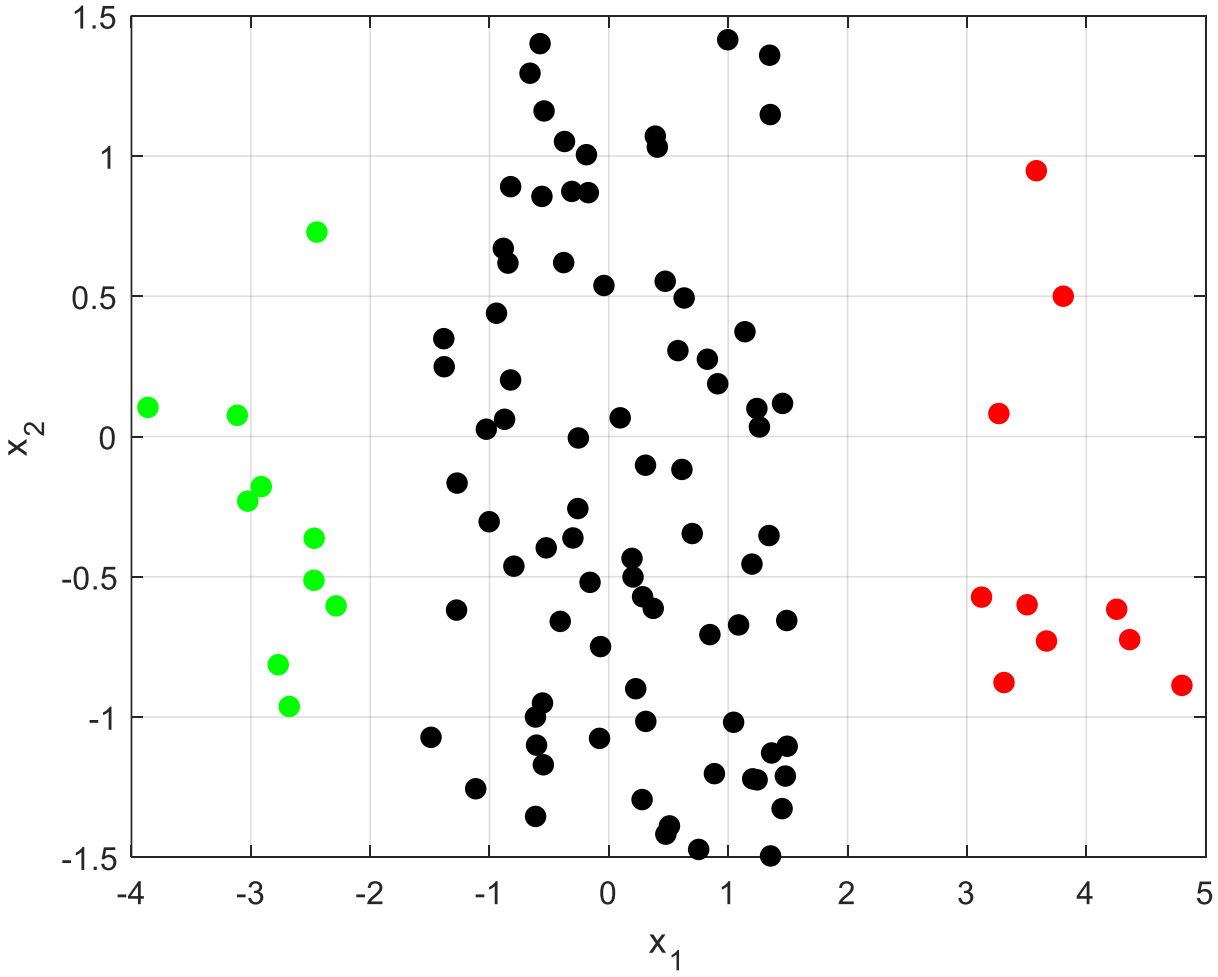
- The Dunn Index gives us another measure of how tightly packed our clusters are AND how far apart they are
  - Useful for helping choose the number of clusters
  - Useful for arguing that a cluster network is supported by the data

Differing Cluster Sizes

- Another K-Means challenge is when the size of clusters is different
  - Since we are minimizing the **cumulative inertia**, it is often optimal to assign large clusters to *several* cluster centers
  - Can also cause issues when more clusters are chosen, as large (broad) data sets often result in substantial inertia improvements when more clusters are used to describe the same "cloud"

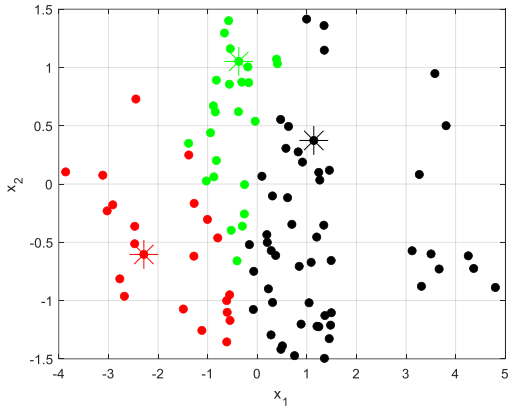
- We don't really have a good solution to this for this algorithm, although using the Dunn Index can really help
  - Identifies when clusters are forming close to each other, which may mean segregating an otherwise broadly grouped set of points

Differing Cluster Sizes

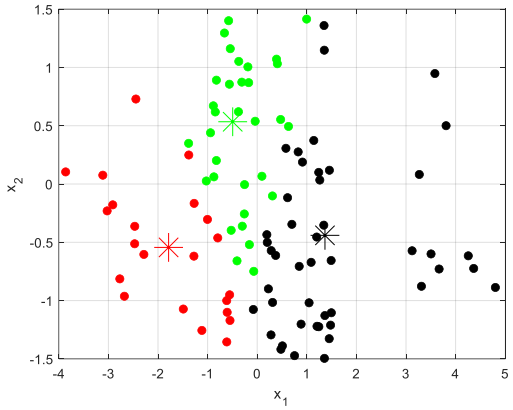


Differing Cluster Sizes

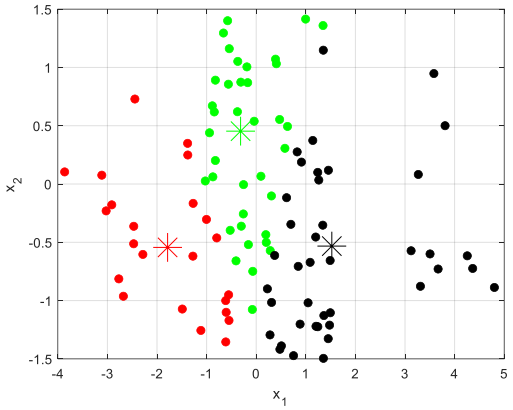
iteration 1

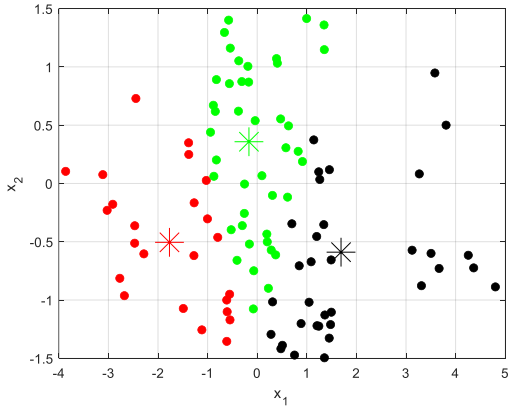


iteration 2

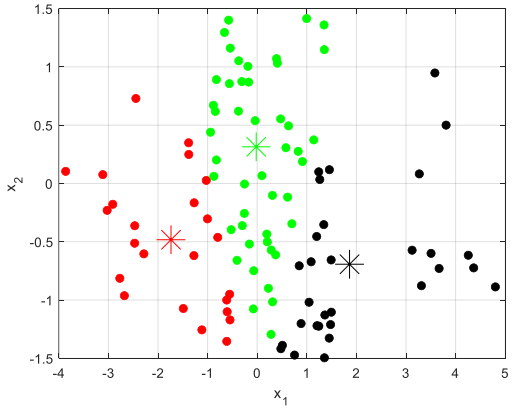


iteration 3

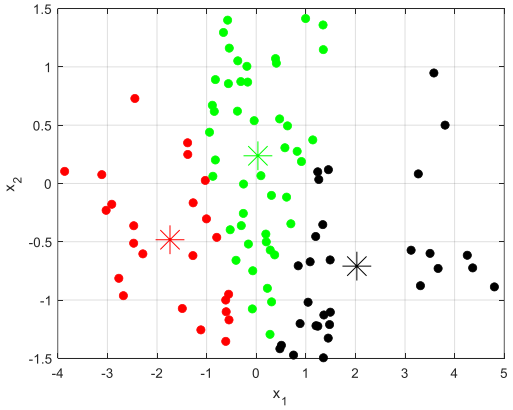




iteration 4

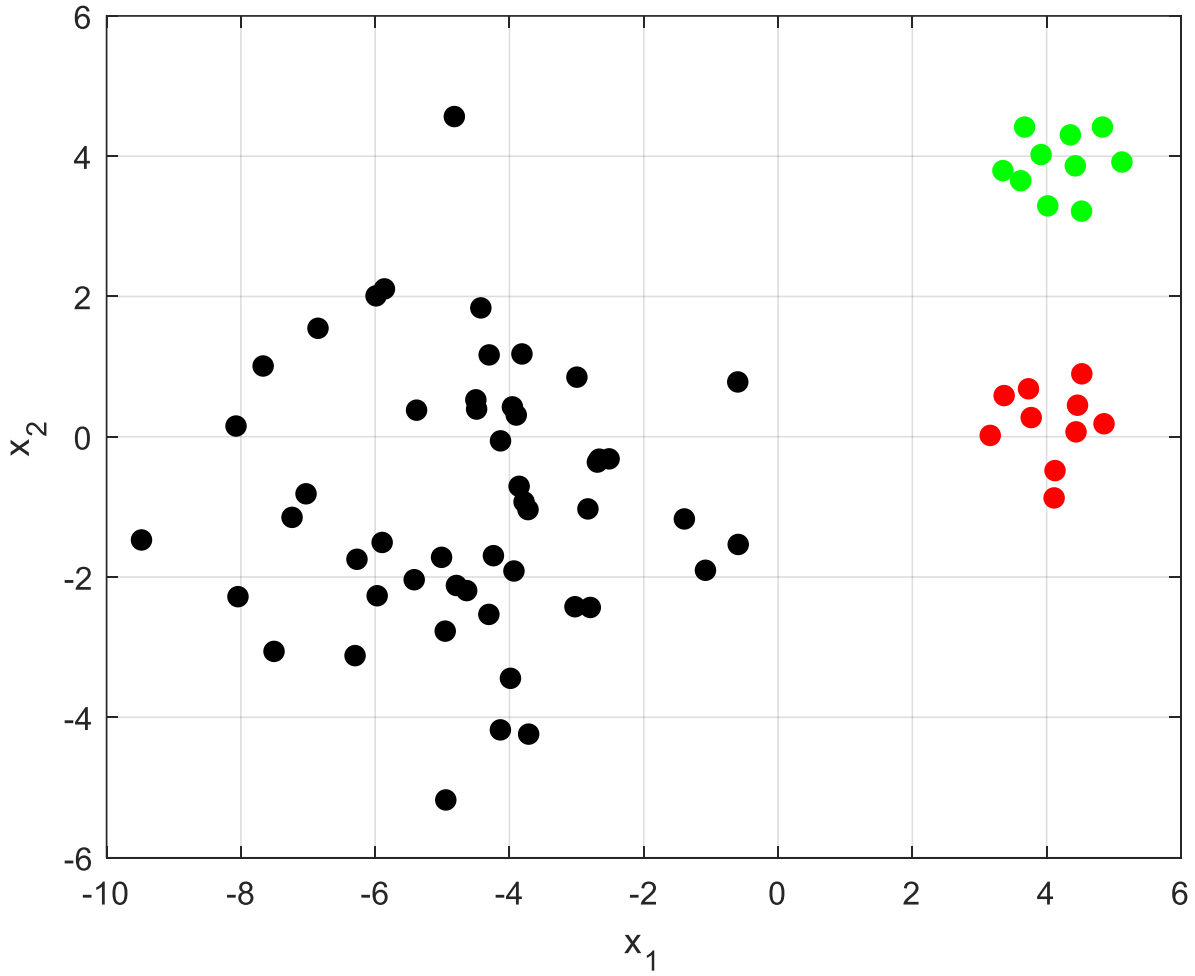


iteration 5



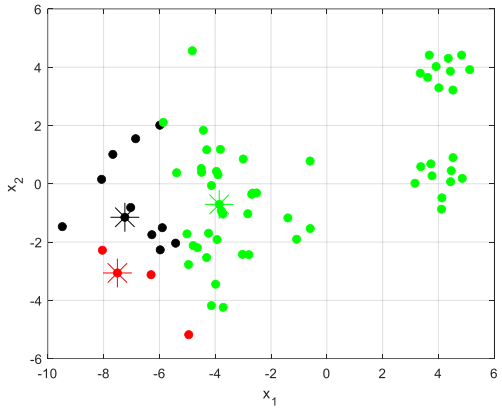
iteration 6

Differing Cluster Sizes

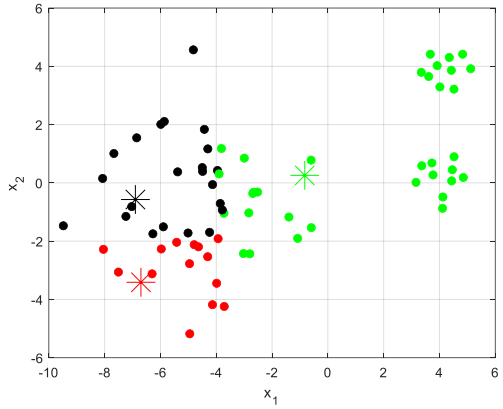


Differing Cluster Sizes

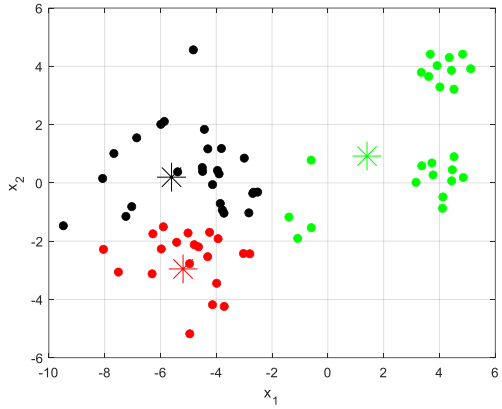
iteration 1

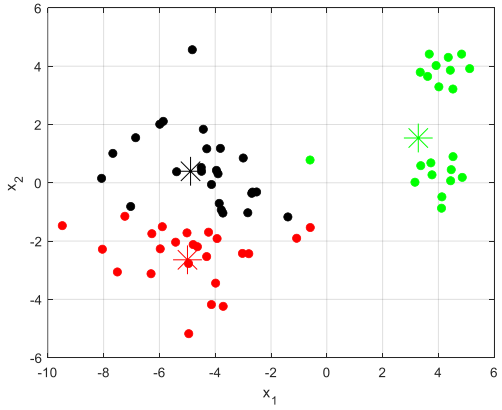


iteration 2

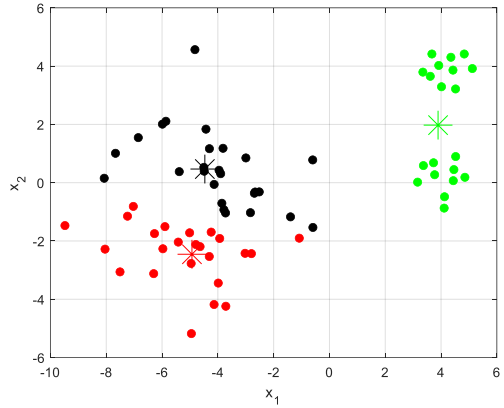


iteration 3

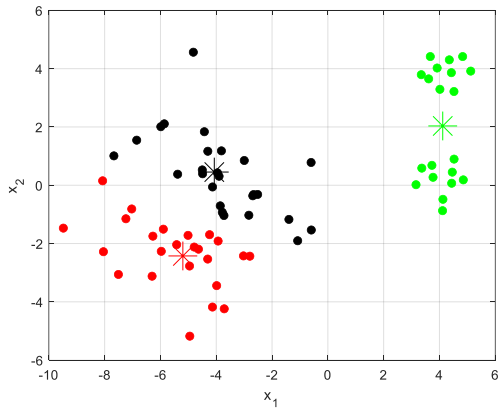




iteration 4



iteration 5

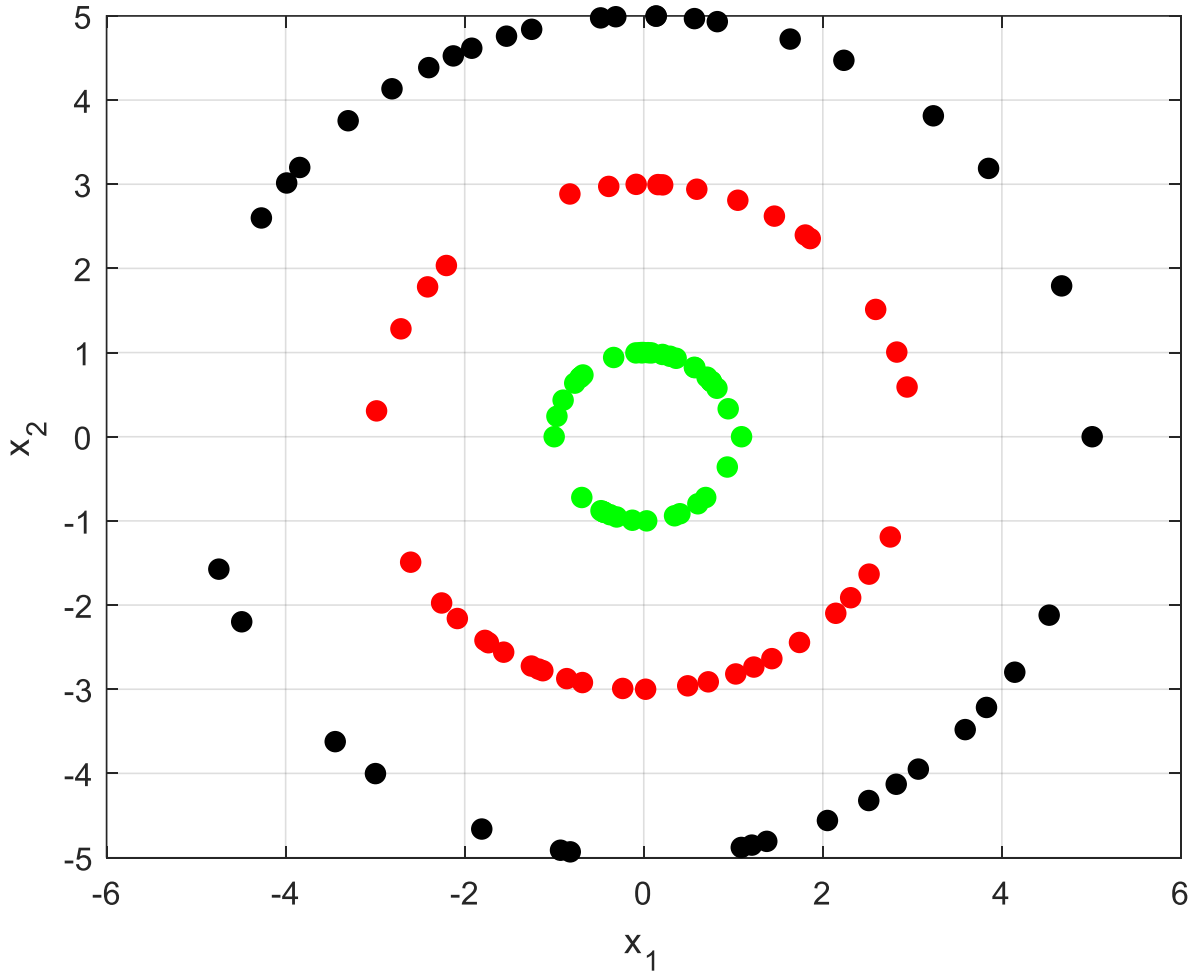


iteration 6

Nonlinear Partitions

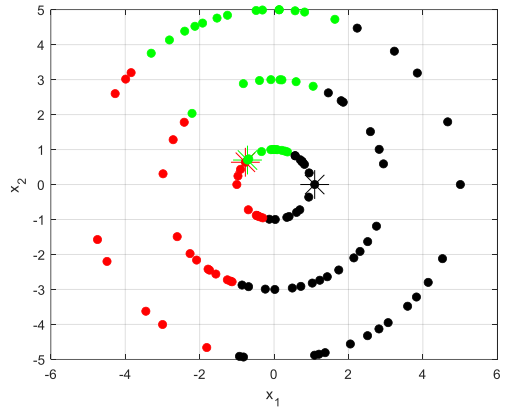
- K-Means is unable to partition data that differentiates in a **nonlinear** fashion
- Typically, this means that the data are not grouped "spatially" in the coordinate space to which they belong
  - In other words, it might be inappropriate to measure inertia between points as the Euclidian distance or "as the crow flies"
- The commonly used example for this type of data are the target rings or half-moons

Nonlinear Partitions

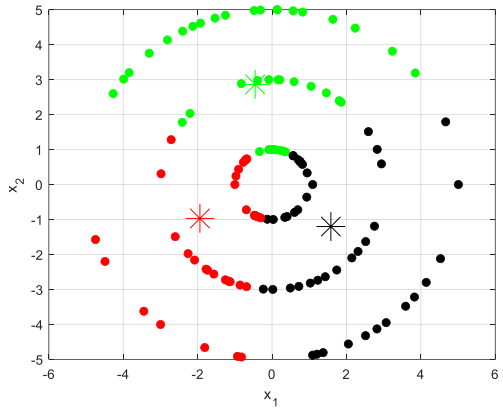


Nonlinear Partitions

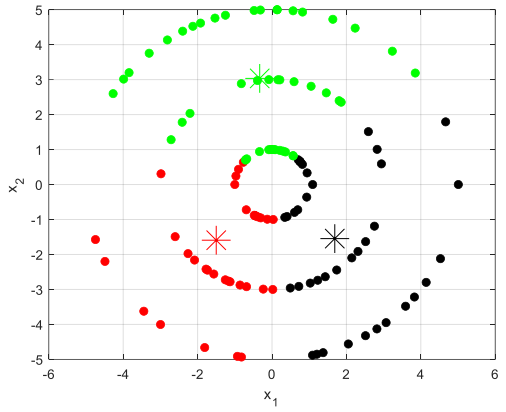
iteration 1

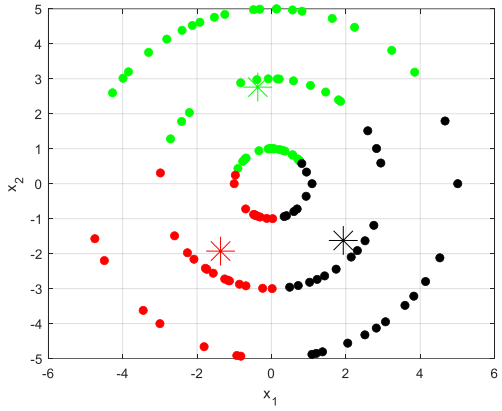


iteration 2

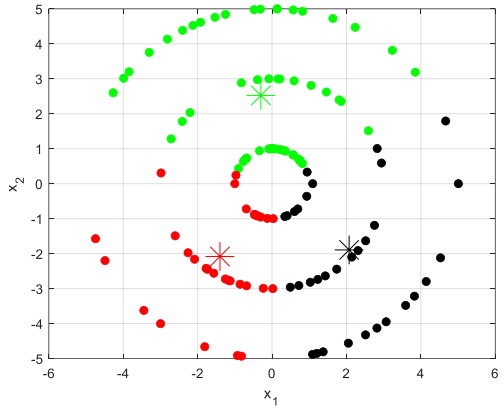


iteration 3

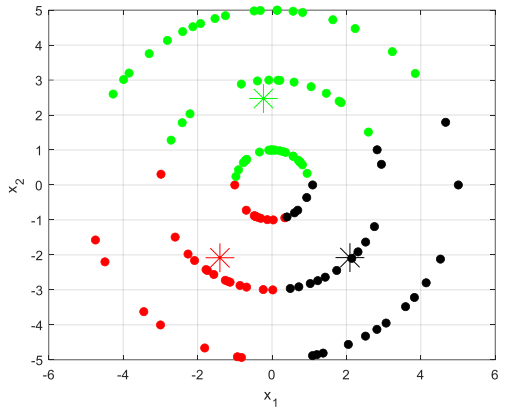




iteration 4



iteration 5



iteration 6

Nonlinear Partitions

- Since K-Means is an **unsupervised learning** method, it is very difficult to know if the data is misclassified based on shape

- For data **known** to be highly nonlinear, consider support vector machines with kernel functions to project the nonlinear data into a linear space
  - This is not UNSUPERVISED because we need to know which group each point belongs to (supervised)
  - We MIGHT cover this in 4H if interested, else check out:
  - The Wiki page
  - The Scikit formulation page
  - This idiot's guide to SVM from MIT (their name, not mine)

Conclusions

- K-Means Clustering is our first (and only) exposure to unsupervised learning methods
- Very handy for identifying sub-classes in a data set
  - Locally optimal separation based on “inertia” of data
- Comes with some nice tools to help us fit when graphics are not an option
  - Elbow plot
  - K++ initialization
  - Dunn Index
- Not good for all types of data
  - Different sizes of data that should belong to each cluster
  - Different variances of data that should be within same cluster
  - Nonlinear shapes of data that do not conform to inertia optimization