Clustering

k-Means

Learning Objectives

- What's **Unsupervised Learning**?
  - How does it compare to supervised learning? Why do this?
- What's **Clustering**?
  - How do you do it?
- What's the ***k*-Means Algorithm**?
  - How does it work? What are centroids? How does the algorithm know when it's done?
  - How do you choose ***k***?
- Morning Assignment
  - **Implement** the k-means algorithm from scratch and **test** it on the classic Fisher's Iris dataset

Overview

Supervised vs. Unsupervised Learning

Clustering

- Intuition
- Definition

k-Means Algorithm

- Pseudocode
- Centroid Initialization
- Stopping Criteria
- Step-through
- Evaluation
- Problems
- Choosing k

Supervised vs. Unsupervised Learning

Supervised

- Have a target/label that we model
- Models look like functions that take in data and create prediction
- Have an error metric that we can use to compare models

Unsupervised

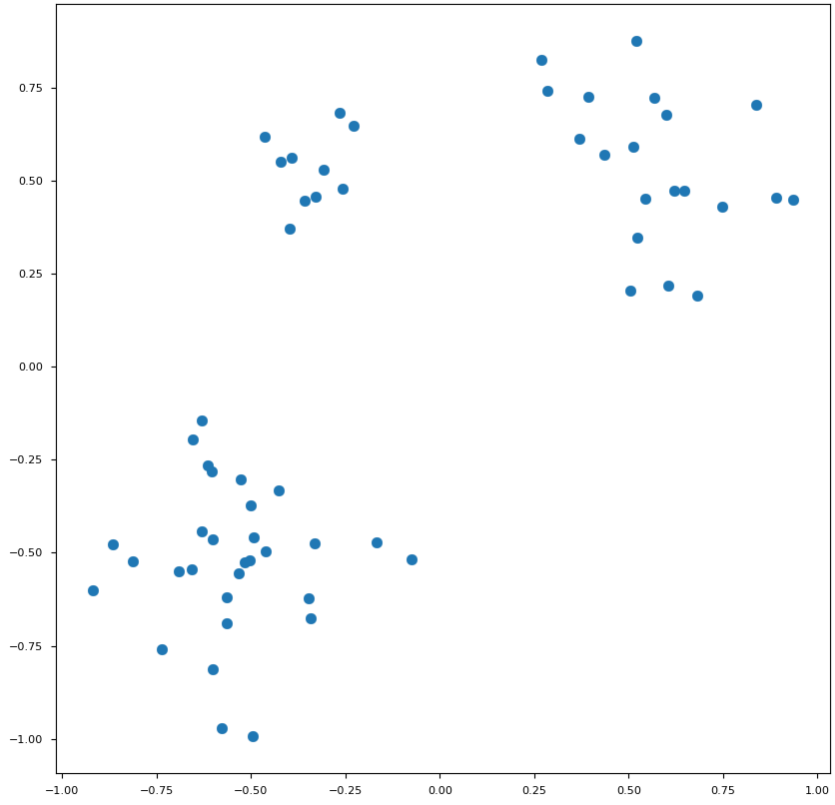
- No labels $\rightarrow$ No target!
- No stark error metric to compare models with
- It's easy to be wrong, but it's hard to prove you're right
- Trying to **uncover/discover *(hidden)* structure** in our data

Unsupervised Learning

- No response variable y
  - Just based on predictors X_1, ..., X_p
- A fuzzy endeavor...
  - No cross-validation
    - to choose "best model" in usual sense
    - to know how well you're doing
- Unsupervised learning provides
  - Exploratory Data Analysis (EDA) to look at/uncover feature structure
  - Anomaly detection to provide data quality control (QC)
  - Dimensionality reduction to simplify large feature spaces (e.g., PCA)

Overview

What's a "Cluster"?



- How many clusters do you see?
- What makes something a cluster?
- What makes something not a cluster?

Overview

Defining "Cluster"

- A partition of the dataset
  - Not necessarily crisp
- A strong internal similarity
  - Small intra/within cluster distance
- A strong external dissimilarity
  - Large extra cluster distance

Overview

k-Means Algorithm

The algorithm in all its glory:

1. Initialize centroids
2. While stopping condition not met:
    - 2.1 Find closest centroid to each point
    - 2.2 Update centroids to the average of all the points closest to them

This training algorithm may look pretty simple... and that's because it is

Overview

Centroid Initialization

- The simplest way to do this is to randomly choose k points from your data and make their locations your initial centroid locations
  - A.k.a., the *Random Choice* centroid initialization
- Another straightforward method is to randomly assign a label (numbered 1-k) to each data point, and start the initialize the i^{th} centroid to the average of the points with the i^{th} label (in each dimension)
  - All centroids start close to the "center" of the feature space
  - A.k.a., the *Random Assignment* centroid initialization

k-Means++, a 3^{rd} centroid initialization method

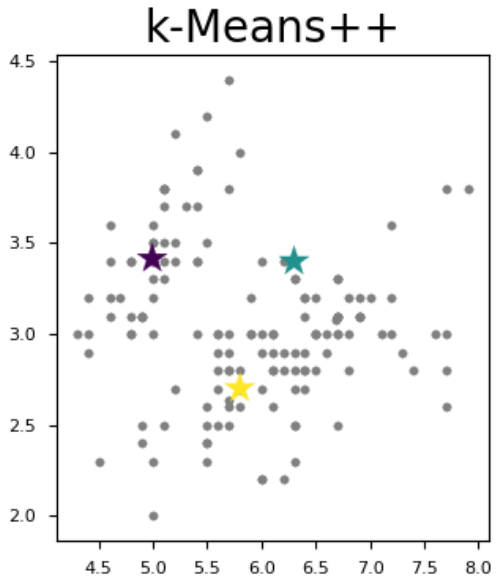
A more advanced centroid initialization method, known as k-Means++, chooses **well spread** initial centroids

$\rightarrow$ sklearn: init='k-means++', set as default

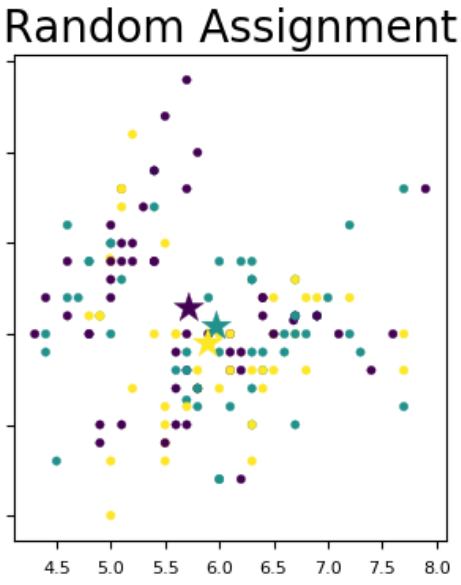
k-Means++ follows the procedure:

1. Choose the first centroid to be the location of a data point chosen at random
2. For each remaining centroid, choose the location of a data point with probability proportional to its squared distance from the point's closest existing centroid
   - Points further from existing centroids have higher probability of being chosen as the next centroid

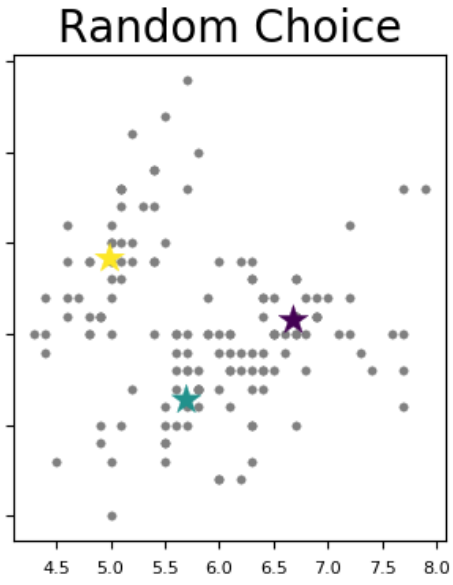
Initialization - Visual Comparison



More even spread to start with



All start close to the center



Who the eff knows... could be anything!

Overview

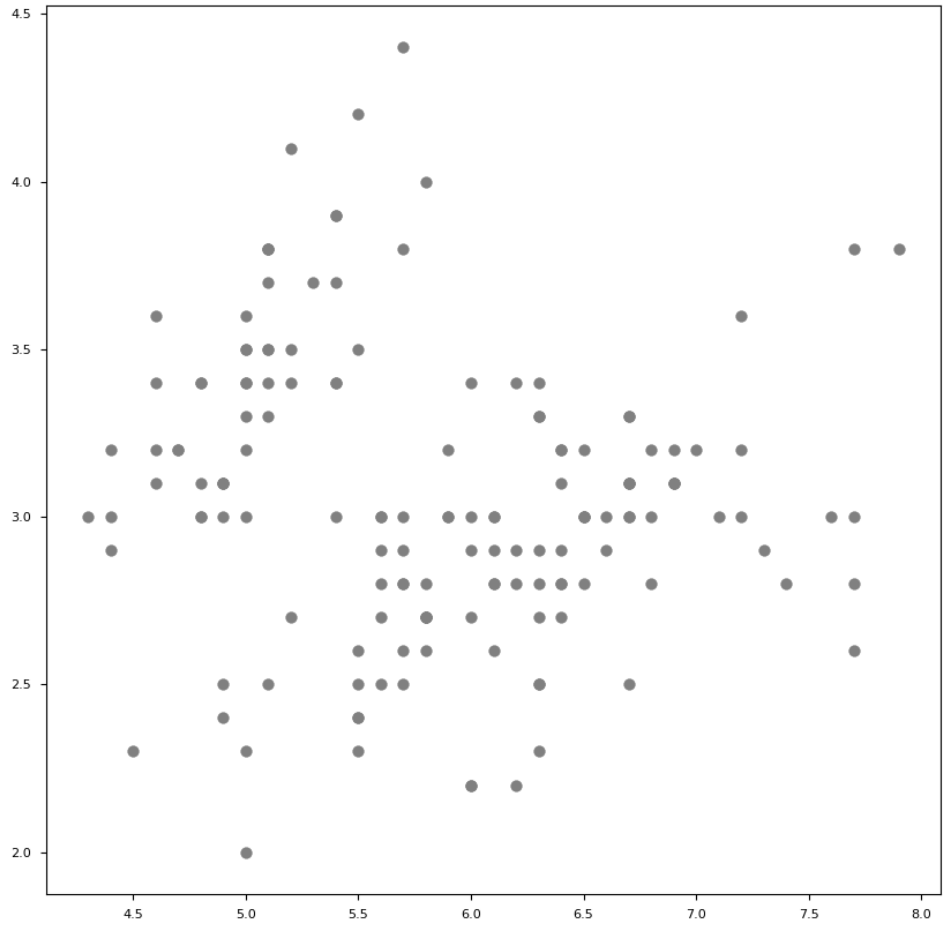
Stopping Criteria

We can update...

1. For a pre-specified number of iterations
   $\rightarrow$ sklearn: *max_iter*=1000
2. Until the centroids don't change at all
   - May take a ton of iterations
3. Until the centroids don't move very much
   $\rightarrow$ sklearn: *tol*=0.0001, for tolerance of "how much"
   - Takes fewer iterations

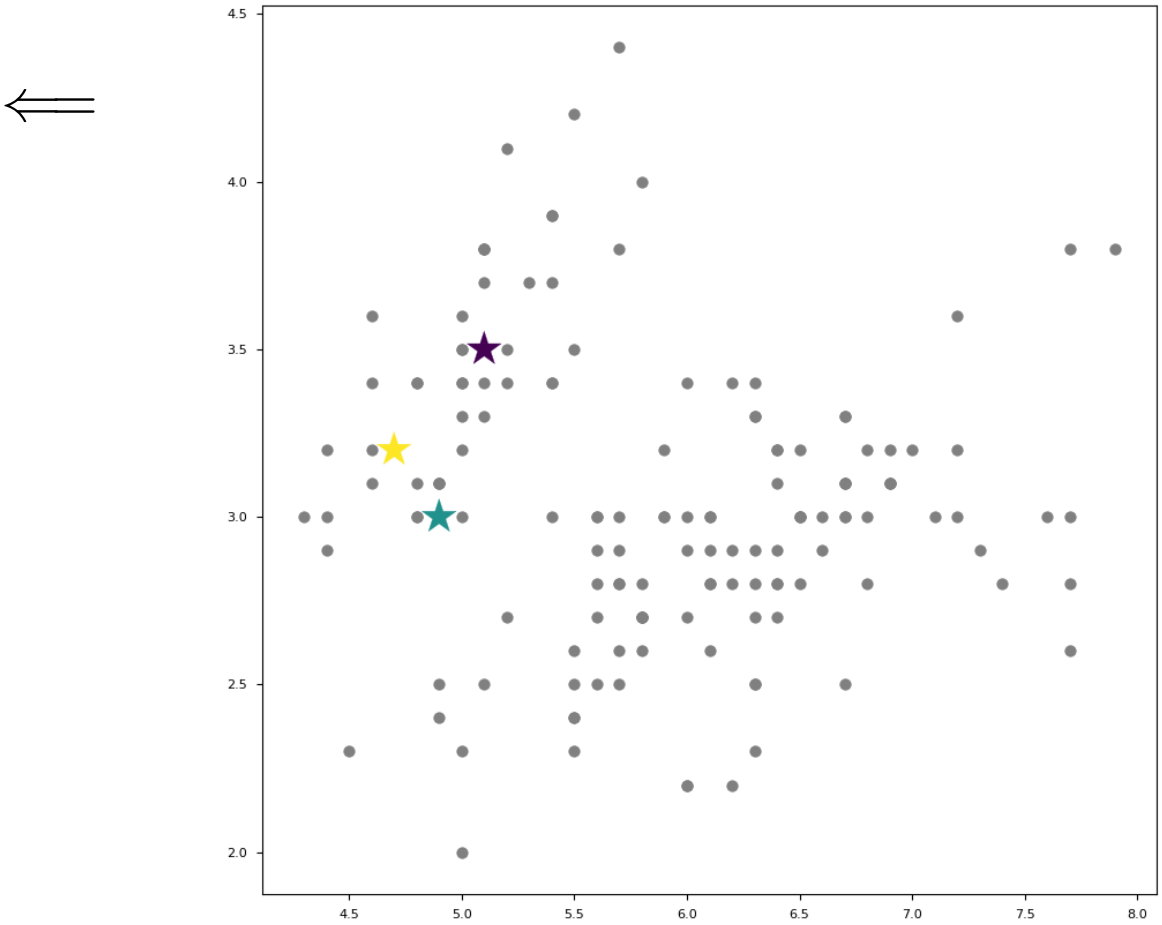
Overview

Step-by-step Execution: DATA!!



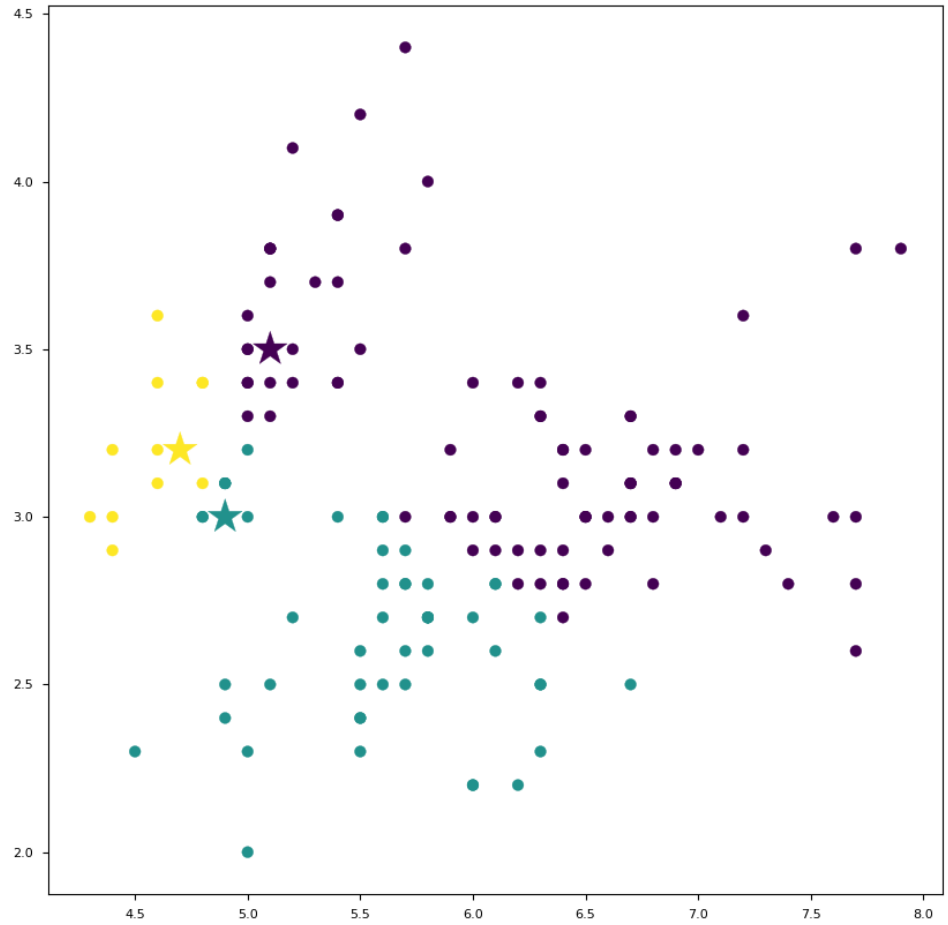
Step-by-step Execution: Initialize

1. **Initialize** centroids
2. While not stopping condition:
   - 2.1 Assign points to centroid
   - 2.2 Update centroids to new average location



Step-by-step Execution: Iteration 1 - Assign

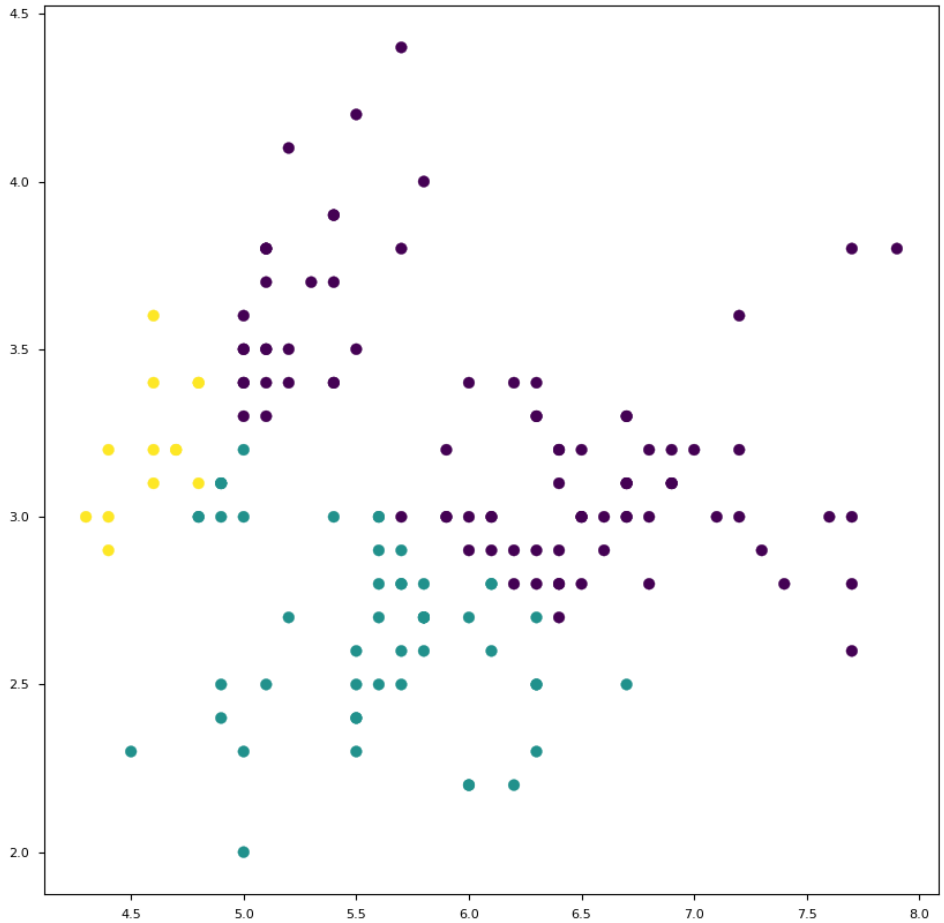
1. Initialize centroids
2. While not stopping condition:
   - 2.1 **Assign points** to centroid
   - 2.2 Update centroids to new average location



Step-by-step Execution: Iteration 1 - Post-Assign

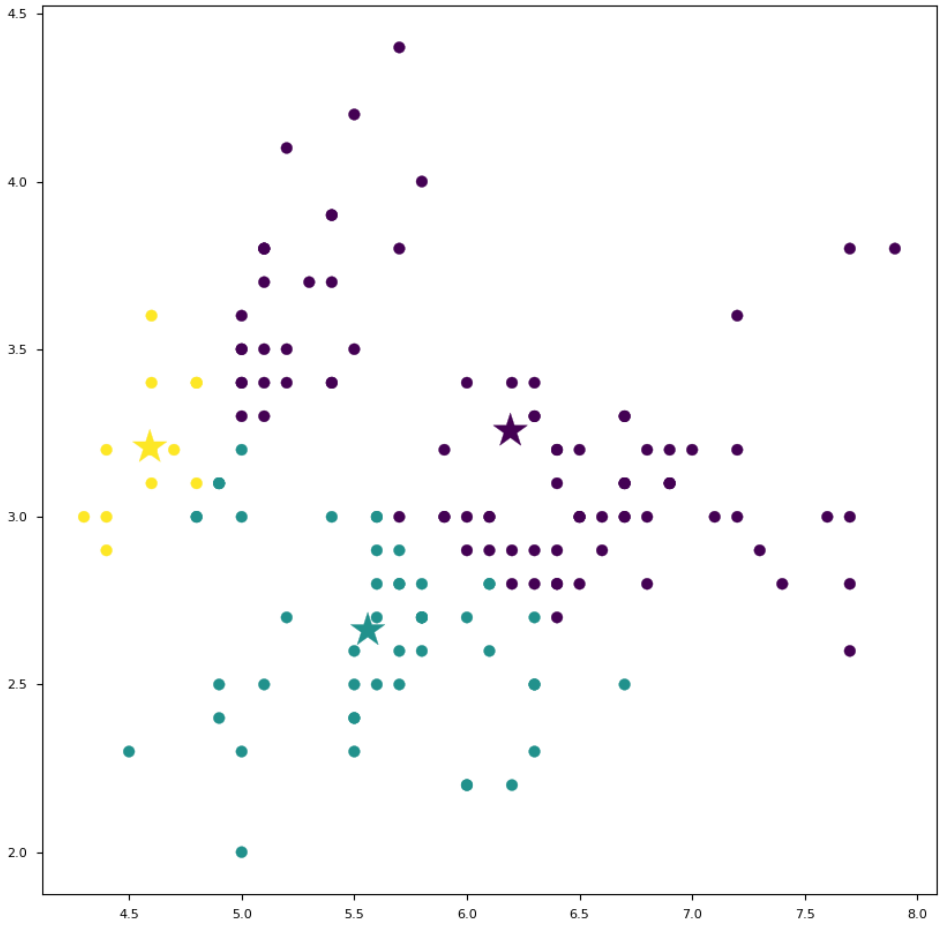
1. Initialize centroids
2. While not stopping condition:
   - 2.1 Assign points to centroid
   - 2.2 Update centroids to new average location

$\Longleftarrow$



Step-by-step Execution: Iteration 1 - Update

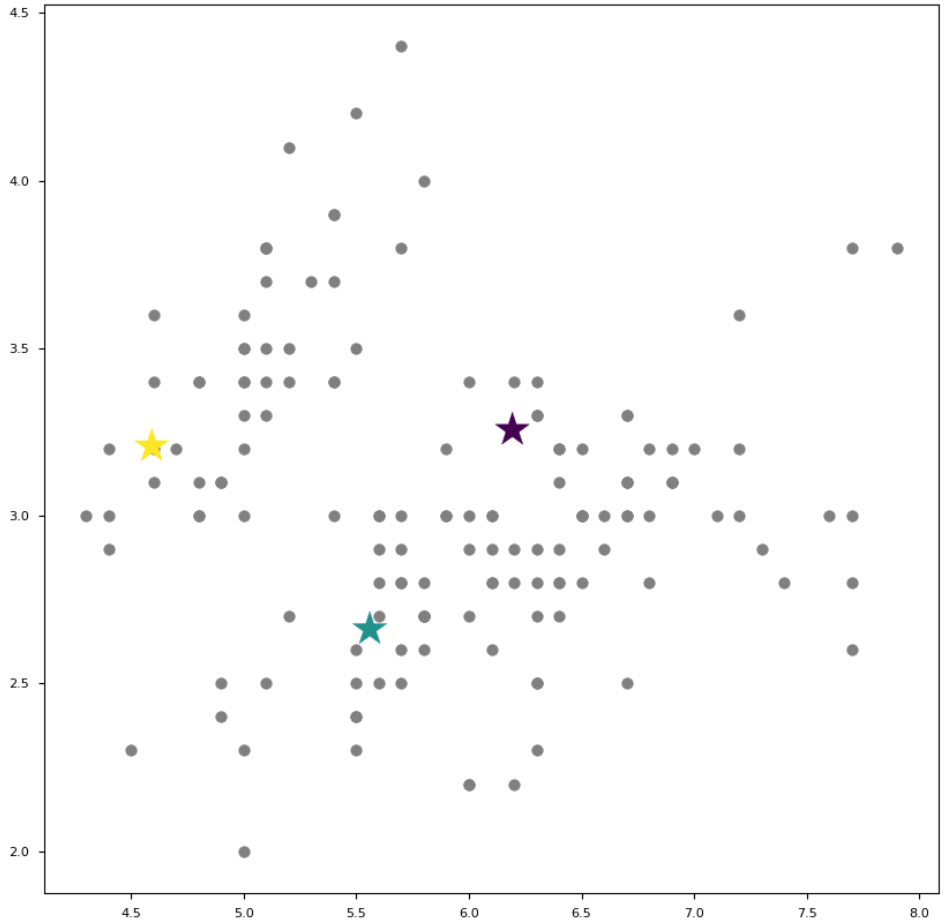
1. Initialize centroids
2. While not stopping condition:
   - 2.1 Assign points to centroid
   - 2.2 **Update** centroids to new average location



Step-by-step Execution: Iteration 1 - Post-Update

1. Initialize centroids
2. While not stopping condition:
   - 2.1 Assign points to centroid
   - 2.2 Update centroids to new average location

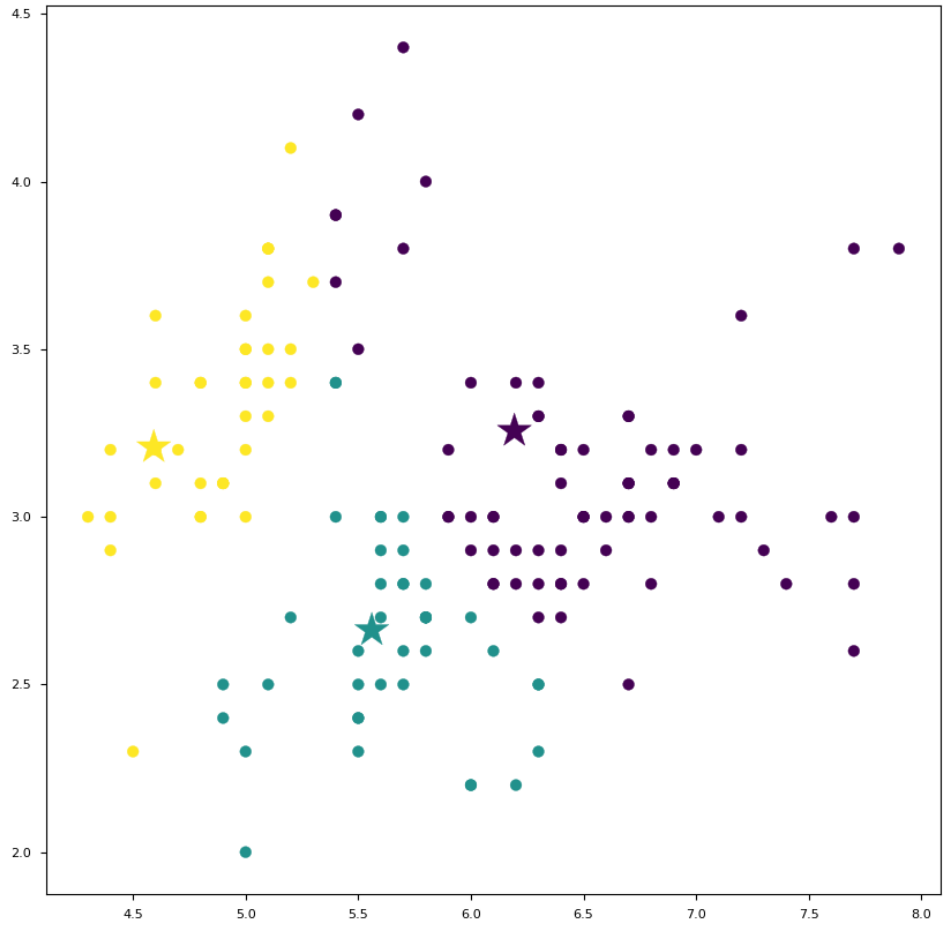
$\Longleftarrow$



Step-by-step Execution: Iteration 2 - Assign

1. Initialize centroids
2. While not stopping condition:
   - 2.1 **Assign** points to centroid
   - 2.2 Update centroids to new average location

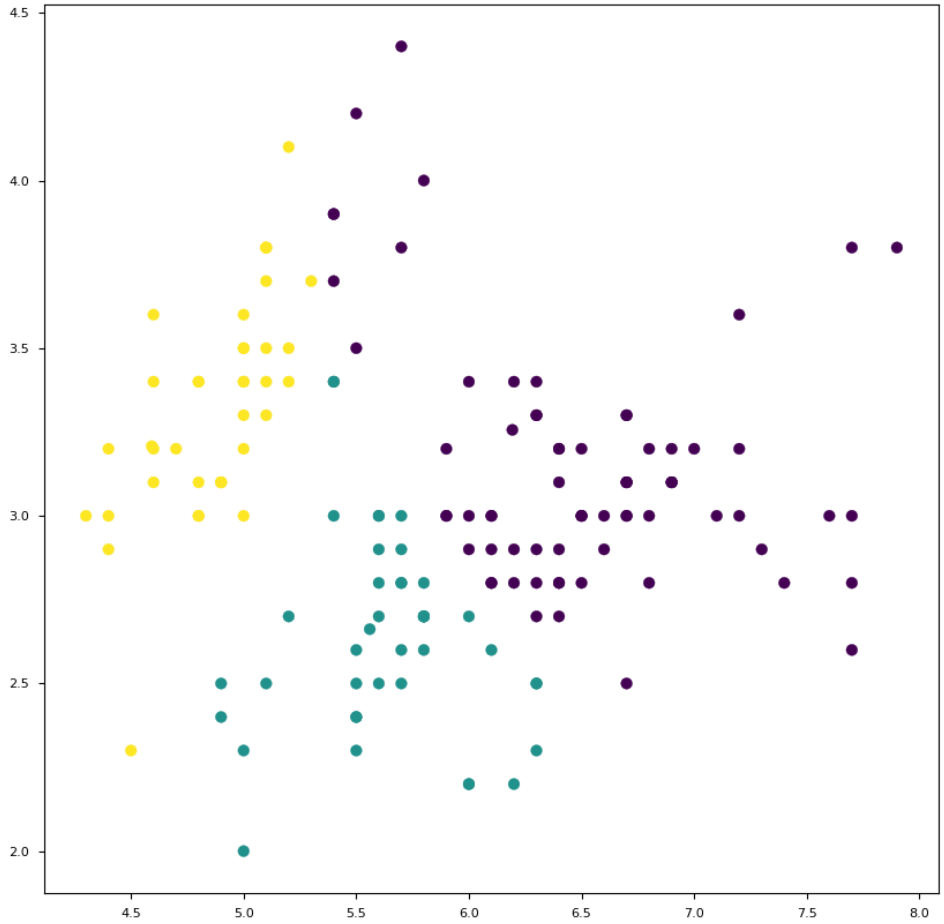
$\Longleftarrow$



Step-by-step Execution: Iteration 2 - Post-Assign

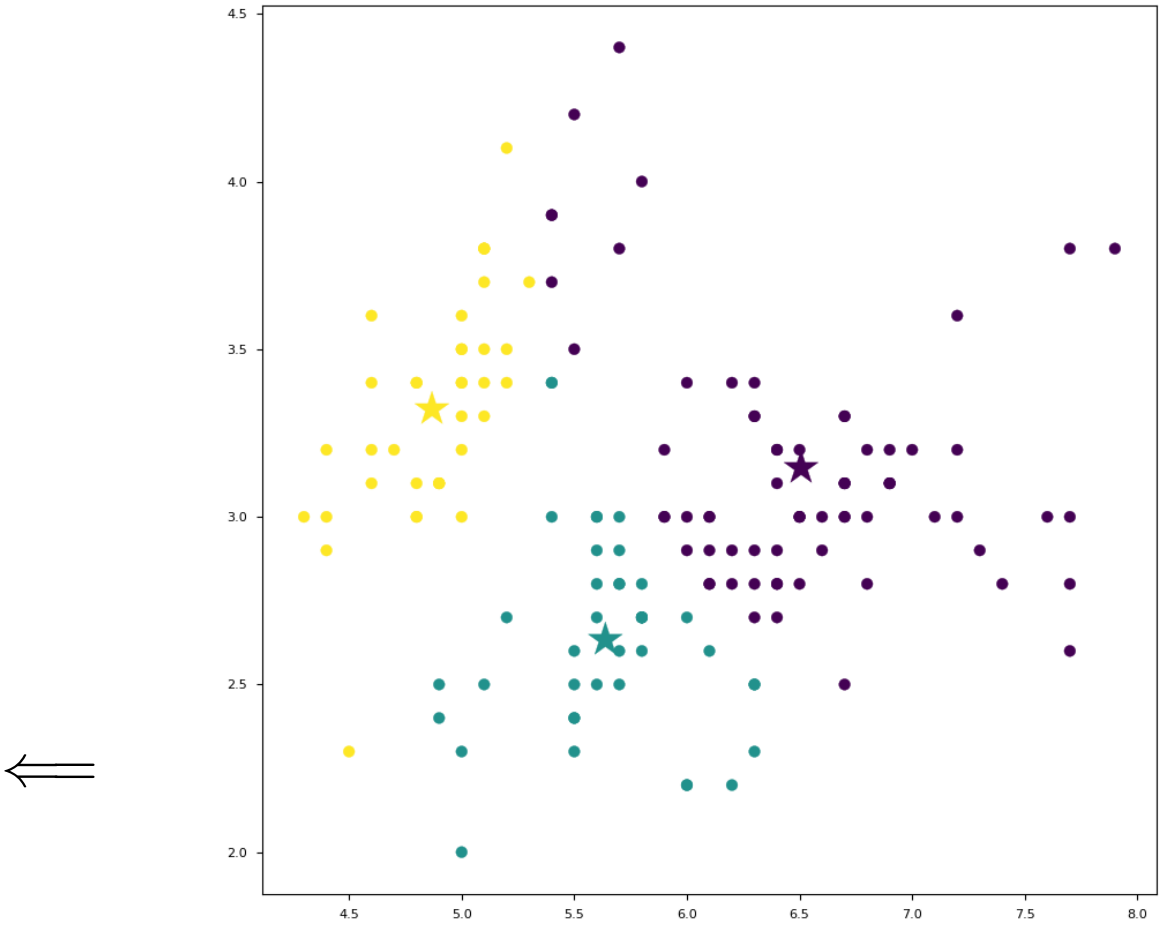
1. Initialize centroids
2. While not stopping condition:
   - 2.1 Assign points to centroid
   - 2.2 Update centroids to new average location

$\Longleftarrow$



Step-by-step Execution: Iteration 2 - Update

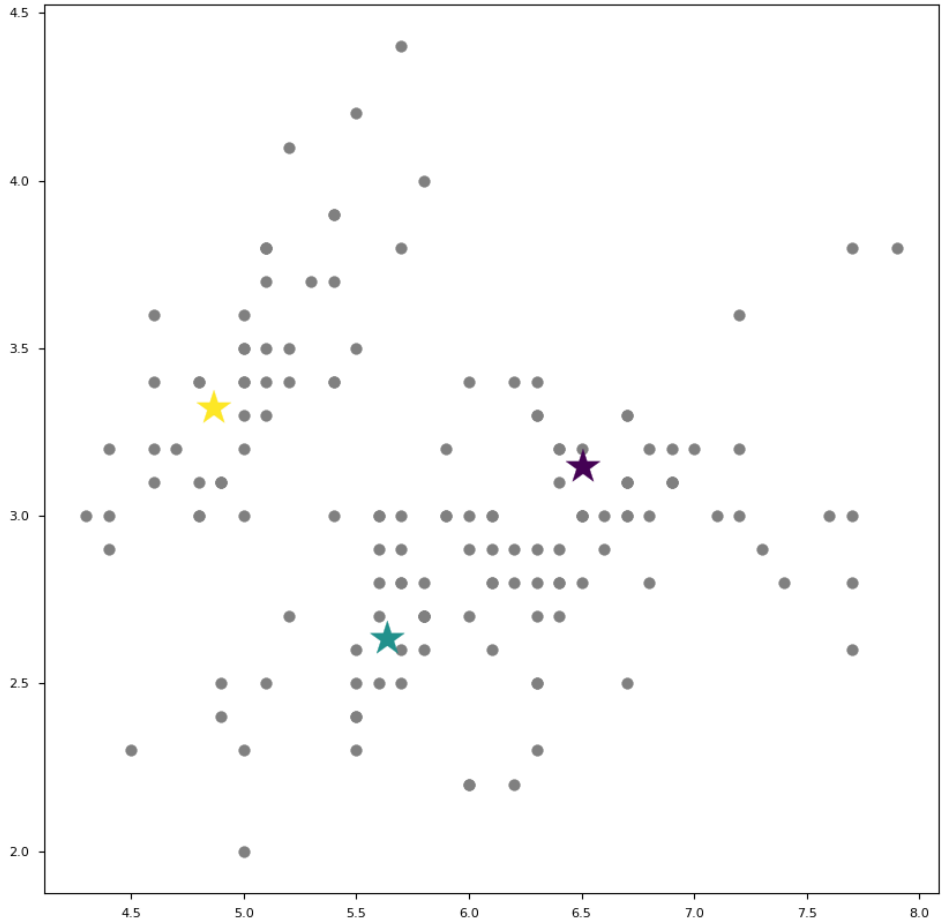
1. Initialize centroids
2. While not stopping condition:
   - 2.1 Assign points to centroid
   - 2.2 **Update** centroids to new average location



Step-by-step Execution: Iteration 2 - Post-Update

1. Initialize centroids
2. While not stopping condition:
   - 2.1 Assign points to centroid
   - 2.2 Update centroids to new average location

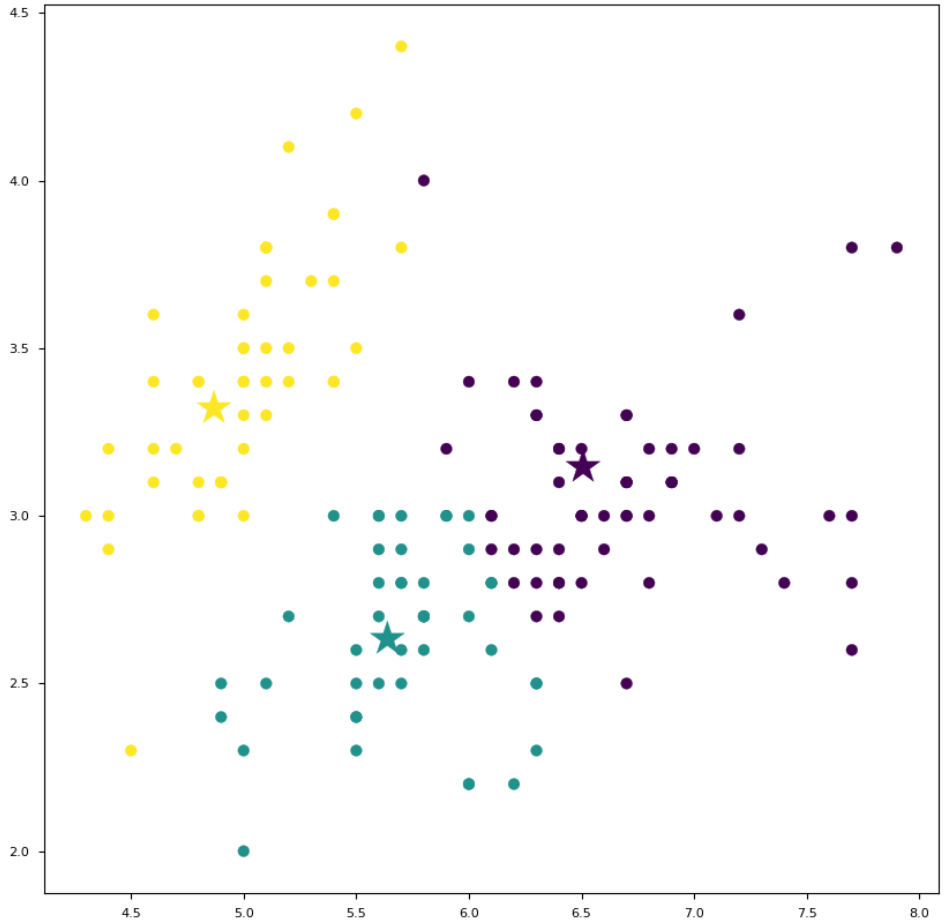
$\Longleftarrow$



Step-by-step Execution: Iteration 3 - Assign

1. Initialize centroids
2. While not stopping condition:
   - 2.1 **Assign** points to centroid
   - 2.2 Update centroids to new average location

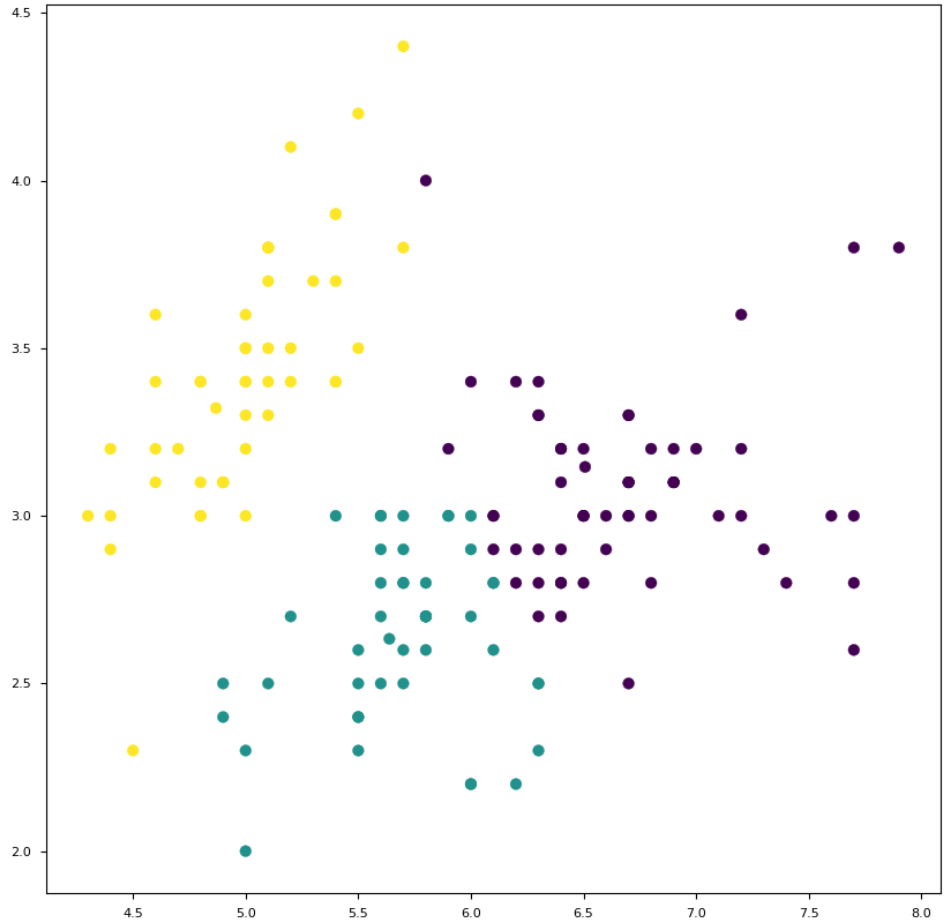
$\Longleftarrow$



Step-by-step Execution: Iteration 3 - Post-Assign

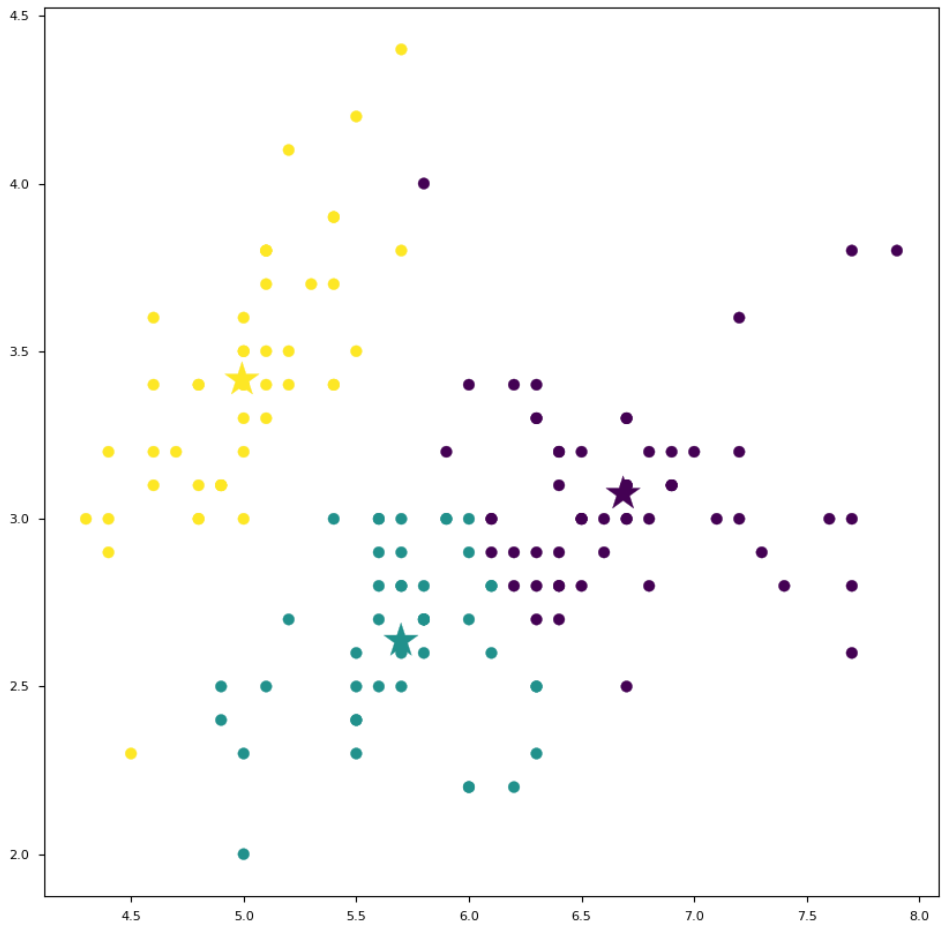
1. Initialize centroids
2. While not stopping condition:
   - 2.1 Assign points to centroid
   - 2.2 Update centroids to new average location

$\Longleftarrow$



Step-by-step Execution: Iteration 3 - Update

1. Initialize centroids
2. While not stopping condition:
   - 2.1 Assign points to centroid
   - 2.2 **Update** centroids to new average location



Overview

Evaluating k-Means

- How can we quantify how "good" our clustering is?
- A good measure should quantify how similar things are in a cluster
- The metric that we will use is called intra-cluster or **Within-Cluster Variance (WCV)**:

$$WCV = \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i_1, i_2 \in C_k} \sum_{j=1}^{p} (x_{i_1 j} - x_{i_2 j})^2$$

Overview

Problems

- Centroids that are "discovered" will likely be different depending on initialization
  $\longrightarrow$ Run algorithm more than once and choose the run that yields the **smallest** within-cluster variance
- k-Means is highly dependent on distance as a metric
  $\longrightarrow$ Have to think about the curse of dimensionality
  $\longrightarrow$ Normalize features before clustering

Overview

Choosing k

Unsupervised

Choosing k is HARD!!!
It usually takes some work and you're never quite sure if you're "right"

There are a number of ways you can go about choosing k:

- Domain knowledge
- Elbow method
- Silhouette score
- GAP Statistic

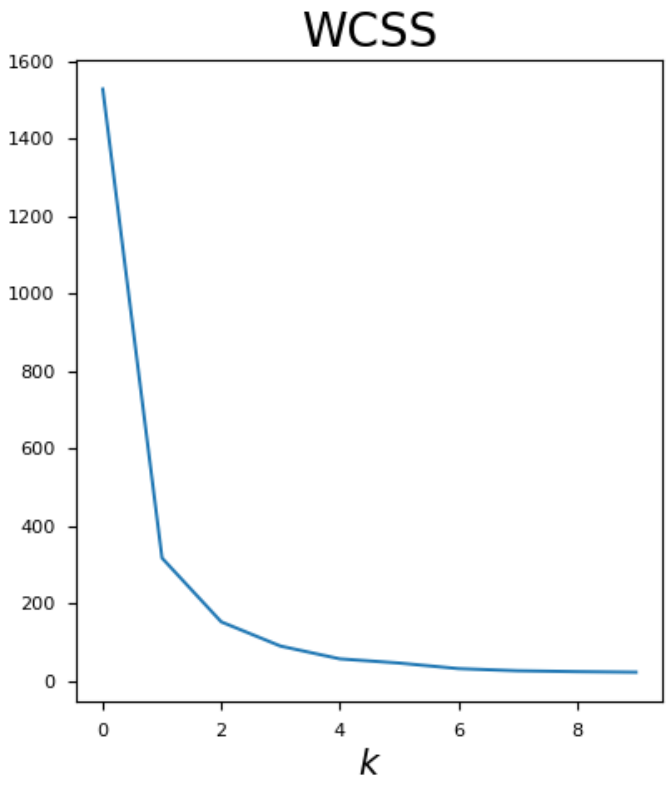
Choosing k: Elbow Method

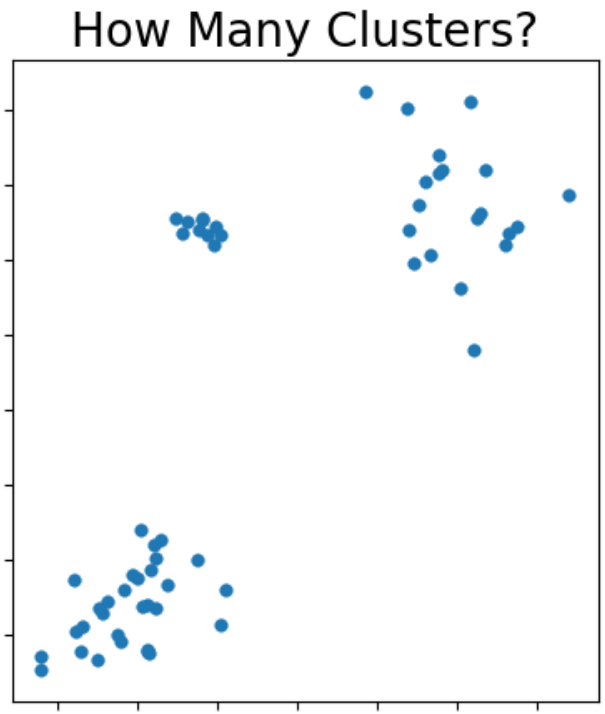
- Looks at the total amount of within-cluster sum of squares (WCSS) across all the clusters for different values of k

$$WCSS = \sum_{k=1}^{K} \sum_{i_1, i_2 \in C_k} \sum_{j=1}^{p} (x_{i_1 j} - x_{i_2 j})^2$$

- Chooses the k such that adding one more cluster doesn't decrease the WCSS by much more. Leads us to look for an elbow in the k vs. $WCSS$ plot

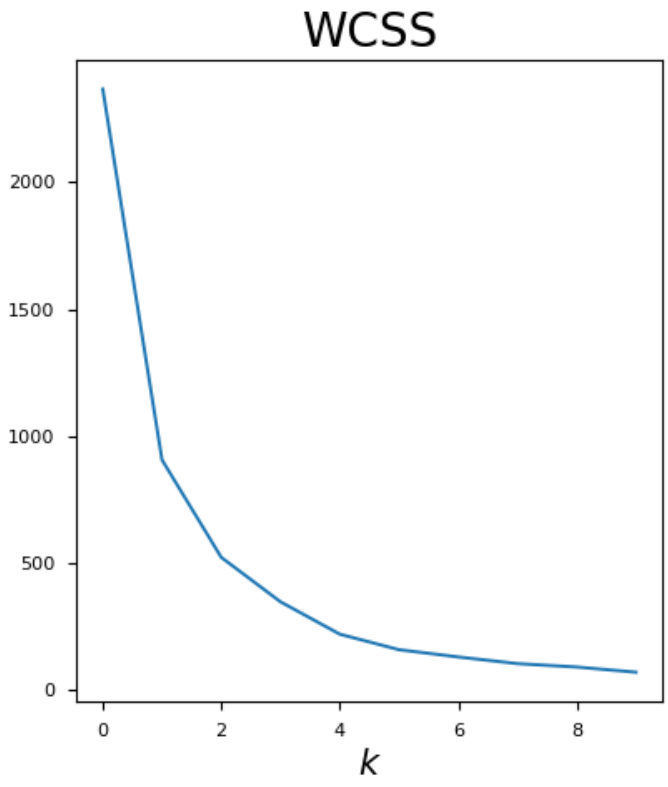
Choosing k: Elbow Method

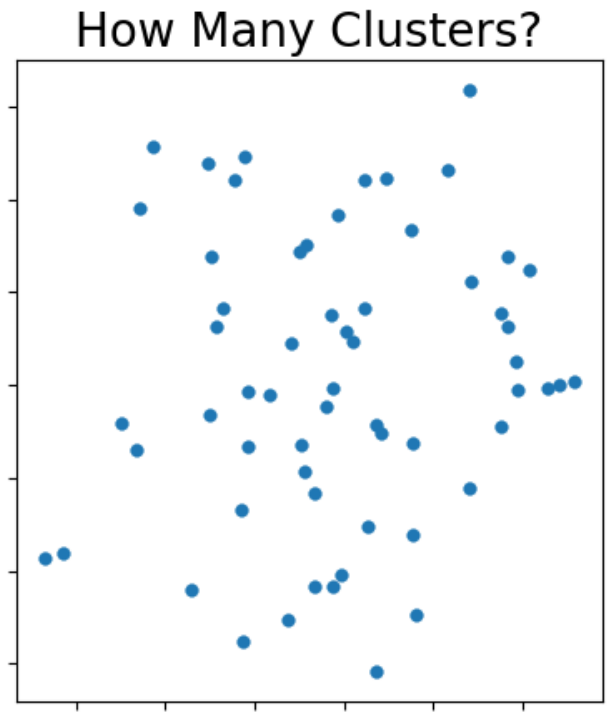




Question: Do you think the elbow will always be so obvious?

Choosing k: Elbow Method - Not Always So Clear





Question: How is this related to the curse of dimensionality?