Clustering

Unsupervised Learning

- Unsupervised learning is a machine learning approach in which models do not have any supervisor to guide them.
- Models themselves find the hidden patterns and insights from the provided data.
- It mainly handles the unlabeled data.
- It is similar to how a human learns. It involves thinking by experiences, which moves it closer to real AI.

Clustering Basic Idea

- Say you are given a data set where each observed example has a set of features, but has no labels.
- Labels are an essential ingredient to a supervised algorithm to predict labels given features.
- So we can't run supervised learning. What can we do?
- One of the most straightforward tasks we can perform on a data set without labels is to find groups of data in our dataset which are **similar** to one another -- what we call **clusters**.

Clustering

- Clustering is the process of dividing the entire data into groups (also known as clusters) based on the patterns in the data.
- **Clustering is an unsupervised learning problem**. Labeled data is not available or needed.
- When we have a **target variable to predict** based on a given set of predictors or independent variables, such problems are called **supervised learning problems**.
- Problems without any fixed target variable, are known as **unsupervised learning problems**. In these problems, we only have the independent variables and no target/dependent variable.
- In clustering, we do not have a target to predict. We look at the data and then try to club similar observations and form different groups. Hence **clustering is an unsupervised learning problem**.

Clustering Use Cases

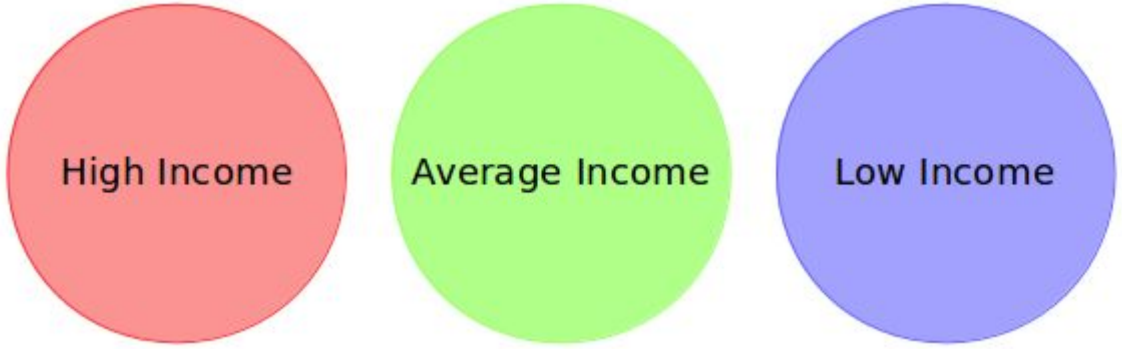
- Customer Segmentation
  - One of the most common applications of clustering across banking, telecom, e-commerce, sports, advertising, sales, etc.
- Document Clustering
  - Helps us group similar documents.
- Image Segmentation
  - Here, we try to club similar pixels in the image together. We can apply clustering to create clusters having similar pixels in the same group.
- Recommendation Engines
  - Let’s say you want to recommend songs to your friends. You can look at the songs liked by that person and then use clustering to find similar songs and finally recommend the most similar songs.

Clustering for Portfolio Creation & Risk Mitigation

- Use cluster analysis to find similarities among companies that we might otherwise not be able to detect.
  - Correlations are time period dependent, so not the only way to find similarities
  - Similarity => lack of diversification
- Further Reading
  - https://www.investopedia.com/terms/c/cluster_analysis.asp
  - https://www.mlq.ai/stock-market-clustering-with-k-means/
  - https://www.alliancebernstein.com/library/Cluster-Analysis-Managing-Risks-You-Didnt-Know-You-Had.htm
  - https://www.cs.princeton.edu/sites/default/files/uploads/karina_marvin.pdf
  - https://arxiv.org/pdf/1609.08520.pdf

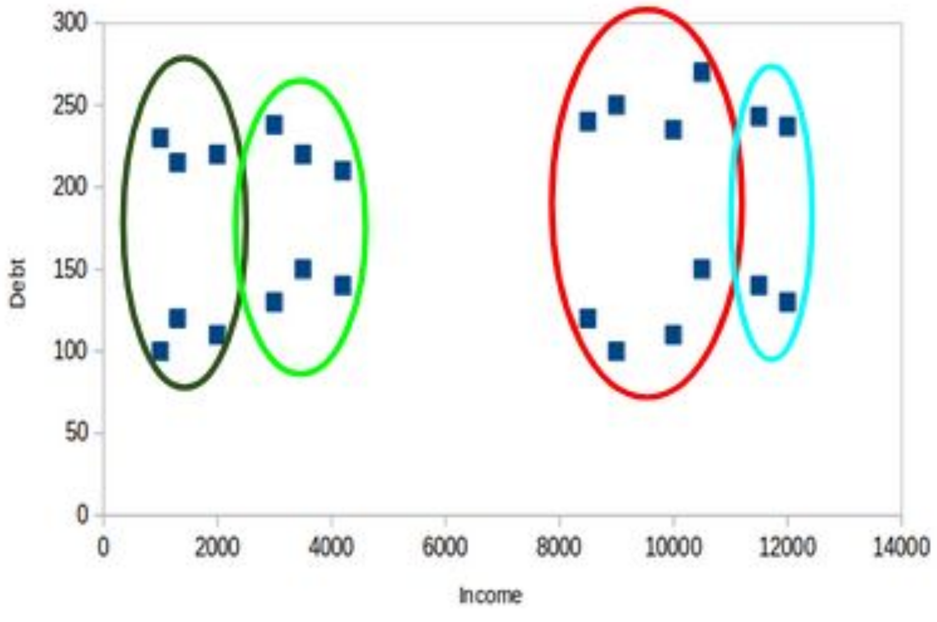
Example: Bank Credit Card Offers

- A bank wants to give credit card offers to its customers. Currently, they look at the details of each customer and based on this information, decide which offer should be given to which customer.
- Now, the bank can potentially have millions of customers. Does it make sense to look at the details of each customer separately and then make a decision? Certainly not! It is a manual process and will take a huge amount of time.
- So what can the bank do? One option is to segment its customers into different groups, and have one strategy per group, instead of per customer. For instance, the bank can group the customers based on their income:

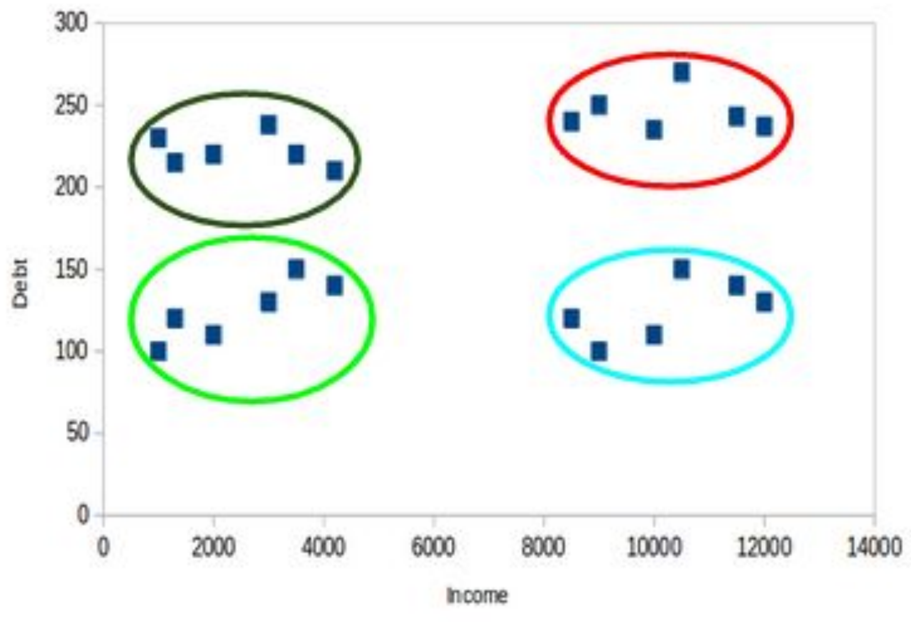


Properties of Clusters

- All the data points **in a cluster** should be **similar to each other**.
- The data points from **different clusters** should be as **different as possible**.
- Let's look at banking credit card example again with 2 dimensions, debt and income. Which are the better clusters?



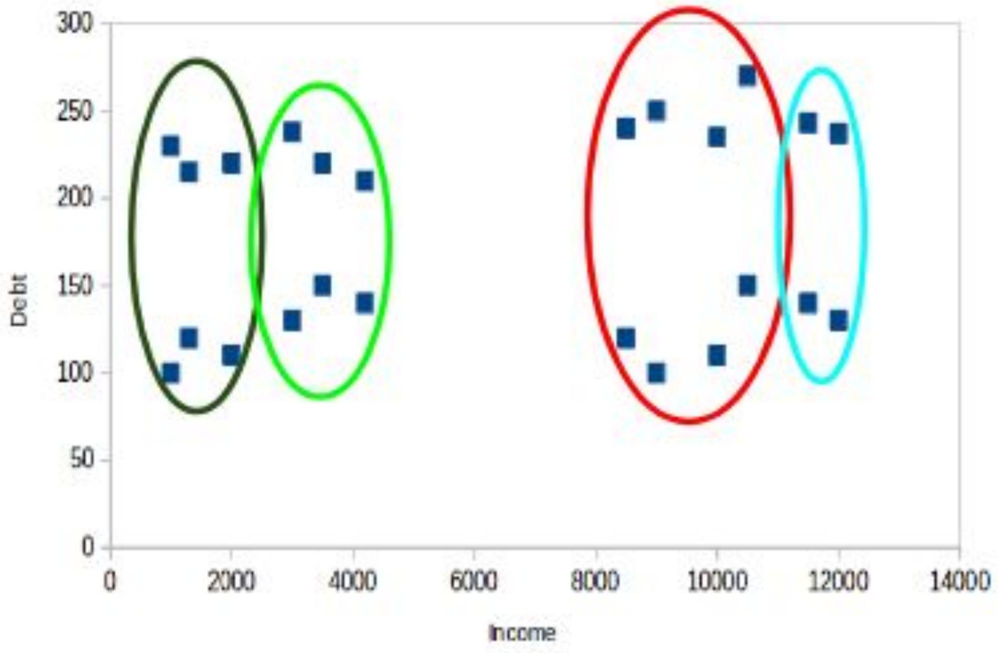
Case - I



Case - II

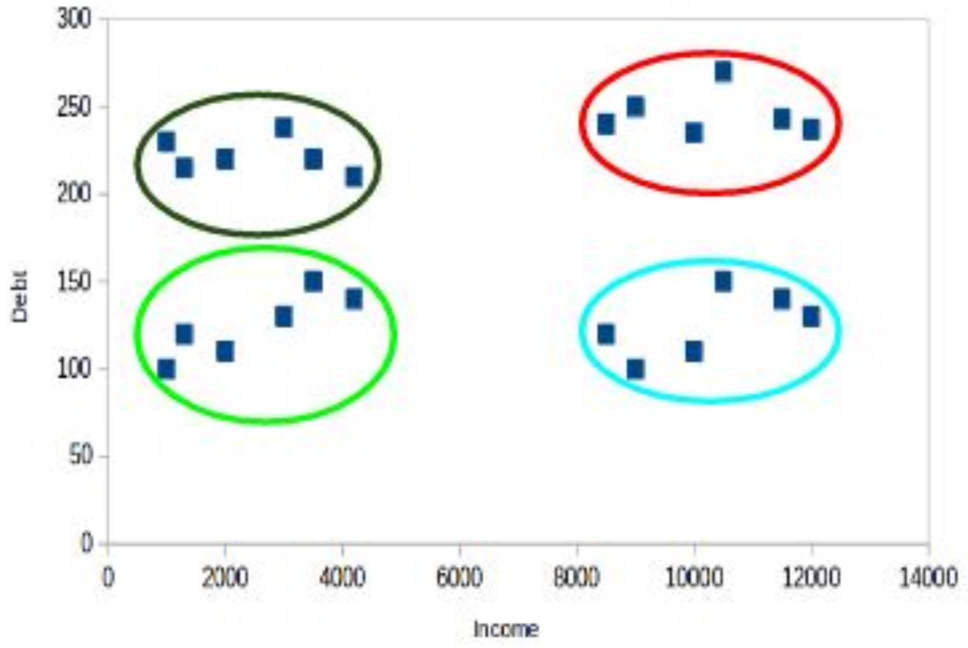
Case I

- Customers in the red and blue clusters are quite similar to each other.
- The top four points in the red cluster share similar properties as that of the top two customers in the blue cluster.
  - They have high income and high debt value.
- Here, we have clustered them differently, **violating the different as possible property** across clusters**.**



Case II

- Points in the red cluster are completely different from the customers in the blue cluster.
  - All red cluster customers have high income and high debt and customers in the blue cluster have high income and low debt value.
- We have a better clustering of customers in this case.

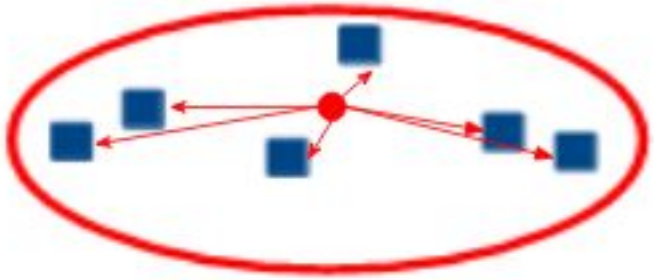


Evaluation Metrics for Clustering

- The primary aim of clustering is not just to make clusters, but to make **good** and **meaningful** ones.
- In the previous example, we used only two features and hence it was easy for us to visualize and decide which of these clusters is better.
- Unfortunately, that's not how real-world scenarios work. We will have a ton of features to work with. Let's take the customer segmentation example again – we will have features like customer's income, occupation, gender, age, and many more. Visualizing all these features together and deciding better and meaningful clusters would not be possible for us.
- This is where we can make use of evaluation metrics. Let's discuss a few of them and understand how we can use them to evaluate the quality of our clusters.

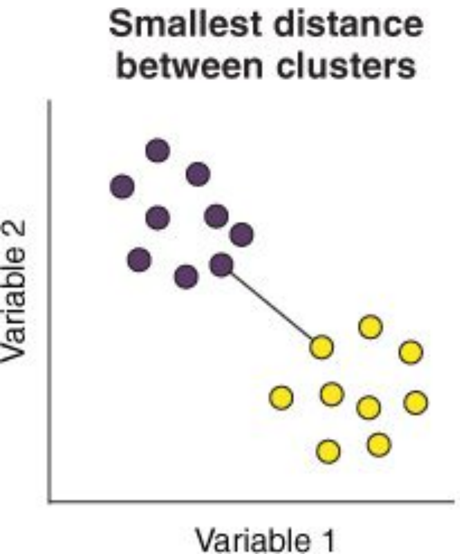
Inertia

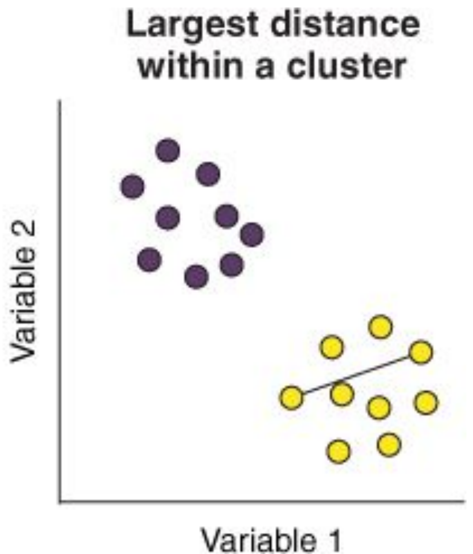
- Recall the first property of clusters we covered above.
- This is what inertia evaluates. **It tells us how far the points within a cluster are.** So, inertia actually calculates the sum of distances of all the points within a cluster from the centroid of that cluster.
- We calculate this for all the clusters and the final inertial value is the sum of all these distances. This distance within the clusters is known as intracluster distance. So, **inertia gives us the sum of intracluster distances.**
- We want the points within the same cluster to be similar to each other. Hence, the distance between them should be as low as possible.
- **Keeping this in mind, we can say that the lower the inertia value, the better our clusters are.**



Dunn Index

- Inertia is used to make more compact clusters.
- **The Dunn index takes into account the distance between two clusters**.
  - This distance between the centroids of two different clusters is known as **inter-cluster distance**.
  - Dunn index is the ratio of the minimum of inter-cluster distances and maximum of intracluster distances.
  - **We want to maximize the Dunn index**. The higher the value of the Dunn index, the better will be the clusters.





$$\text{Dunn Index} = \frac{\min(\text{Inter cluster distance})}{\max(\text{Intra cluster distance})}$$

Common Clustering Algos

- K-means
- Mean-Shift
- Agglomerative Hierarchical
- DBSCAN Algorithm
- Expectation-Maximization Clustering using Gaussian Mixture Models
- Overview
  - https://www.digitalvidya.com/blog/the-top-5-clustering-algorithms-data-scientists-should-know/

Clustering Algo Types

- Connectivity models
  - As the name suggests, these models are based on the notion that the data points closer in data space exhibit more similarity to each other than the data points lying farther away. Examples of these models are hierarchical clustering algorithm and its variants.
- Centroid models
  - These are iterative clustering algorithms in which the notion of similarity is derived by the closeness of a data point to the centroid of the clusters. K-Means clustering algorithm is a popular algorithm that falls into this category.
- Distribution models
  - These clustering models are based on the notion of how probable is it that all data points in the cluster belong to the same distribution (For example: Normal, Gaussian). These models often suffer from overfitting. A popular example of these models is Expectation-maximization algorithm which uses multivariate normal distributions.
- Density Models
  - These models search the data space for areas of varied density of data points in the data space. It isolates various different density regions and assign the data points within these regions in the same cluster. Popular examples of density models are DBSCAN and OPTICS.

Improving Supervised Learning Algorithms with Clustering

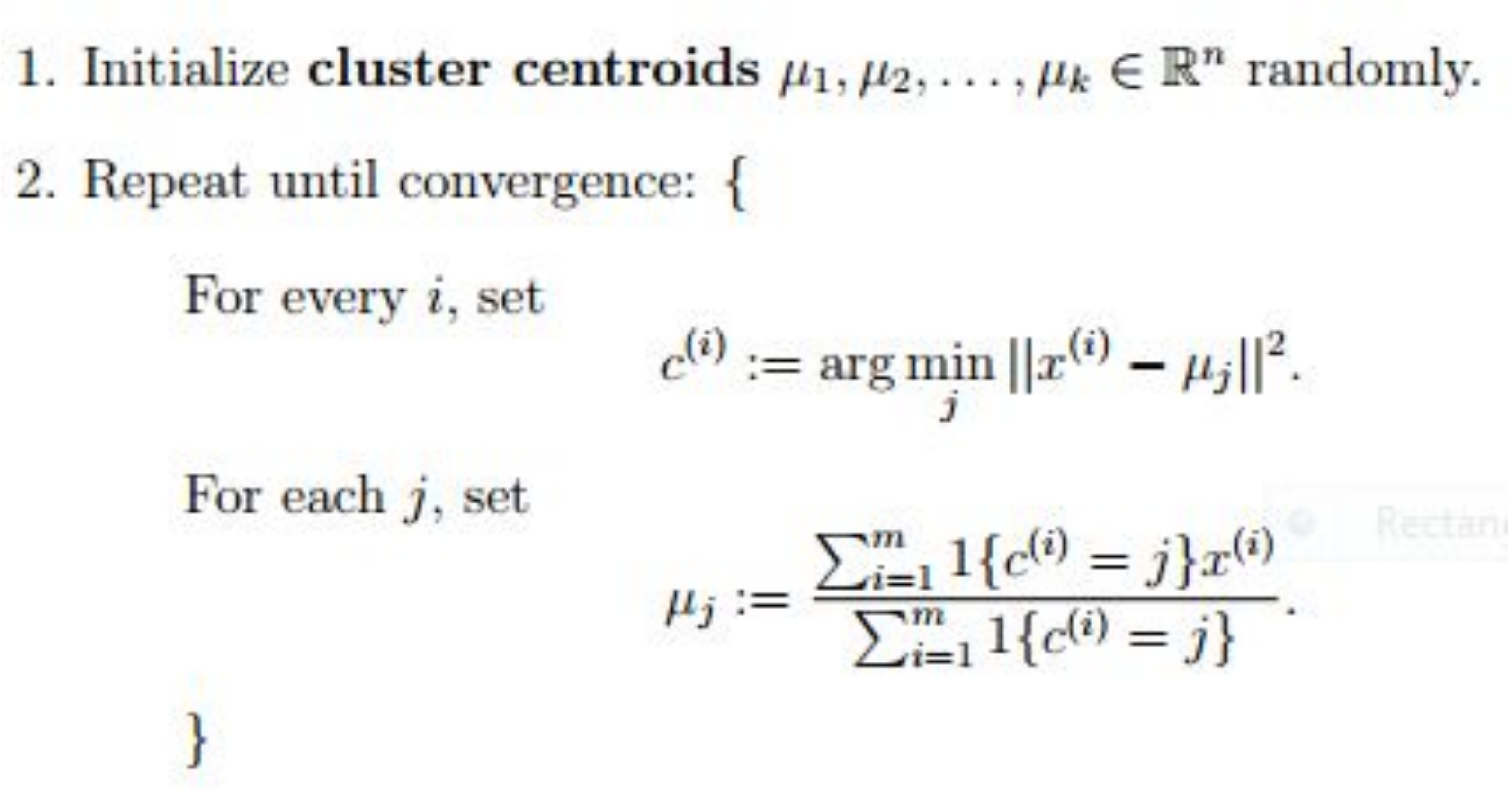
- Clustering is an unsupervised machine learning approach, but it can be used to improve the accuracy of supervised machine learning algorithms as well by clustering the data points into similar groups and using these cluster labels as independent variables in the supervised machine learning algorithm.
- Keep in mind the primary aim of clustering is not just to make clusters, but to make **good** (tight, distant) and **meaningful** ones.
  - You may be able to come up with good clusters, but not meaningful ones (e.g. the labels are not clear)

K-MEANS CLUSTERING

K-means Intro

- Recall the first property of clusters – it states that the points within a cluster should be similar to each other. So, our aim here is to minimize the distance between the points within a cluster.
- There is an algorithm that tries to minimize the distance of the points in a cluster with their centroid – the k-means clustering technique.
- K-means is a centroid-based algorithm, or a distance-based algorithm, where we calculate the distances to assign a point to a cluster. In K-Means, each cluster is associated with a centroid.
- The main objective of the K-Means algorithm is to minimize the sum of distances between the points and their respective cluster centroid.

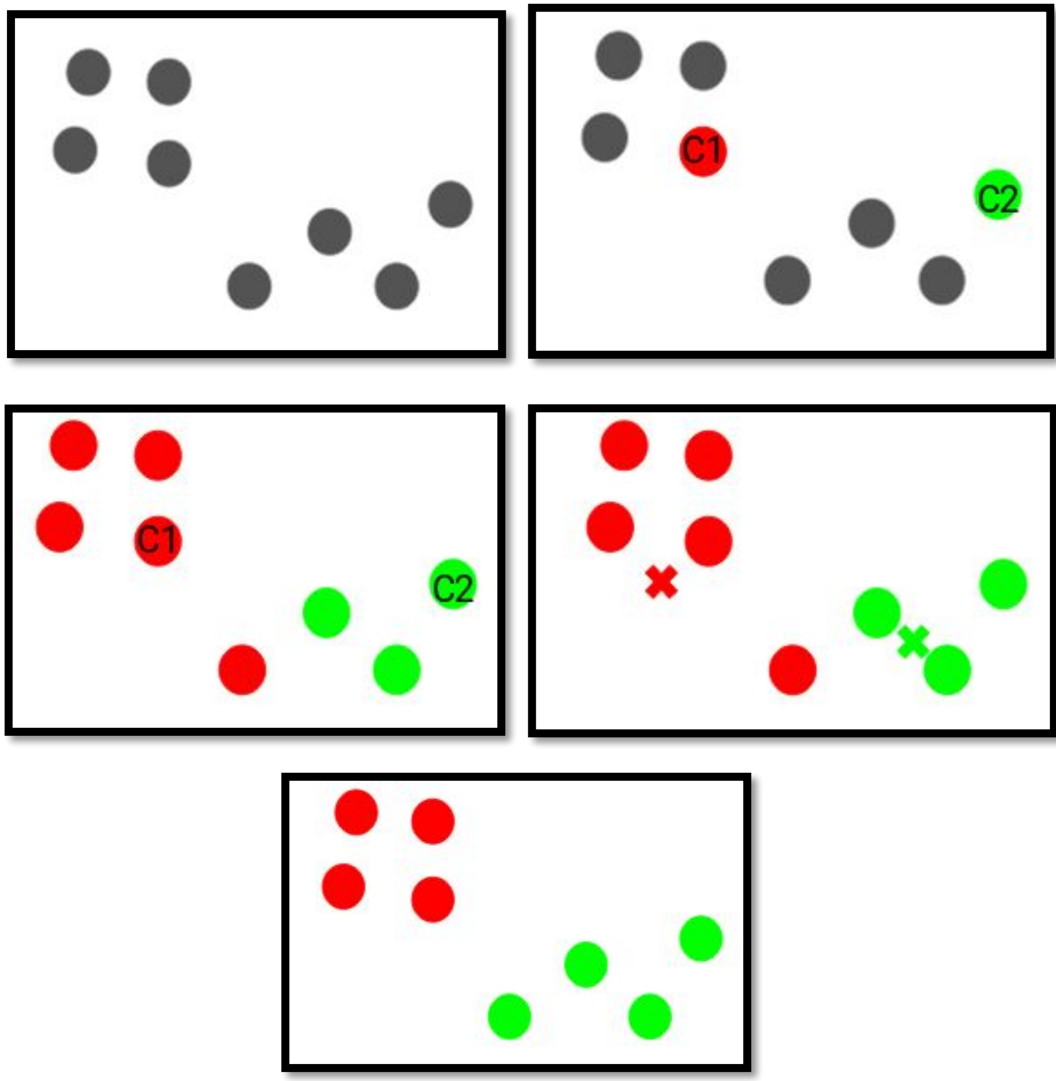
K-means Algo



```
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
            $\mu_j := \frac{\sum_{i=1}^m 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)} = j\}}.$
   }
```

K-means Steps

1. Choose the number of clusters k, say k = 2
2. Randomly select the centroid for each cluster (c1 and c2)
3. Assign all the points to the closest cluster centroid
4. Recompute the centroids of newly formed clusters
5. Repeat steps 3 and 4

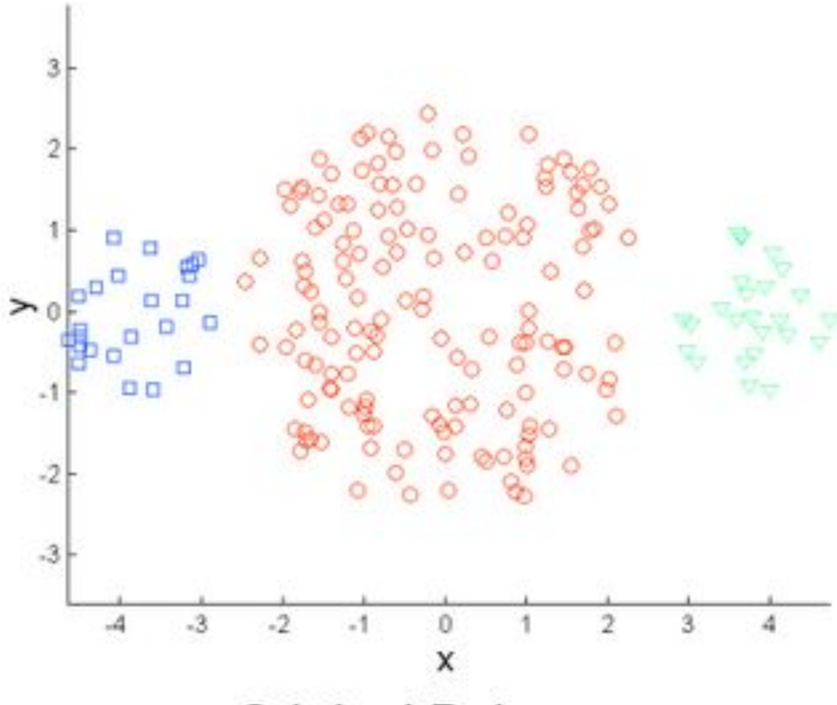


Stopping Criteria for K-Means

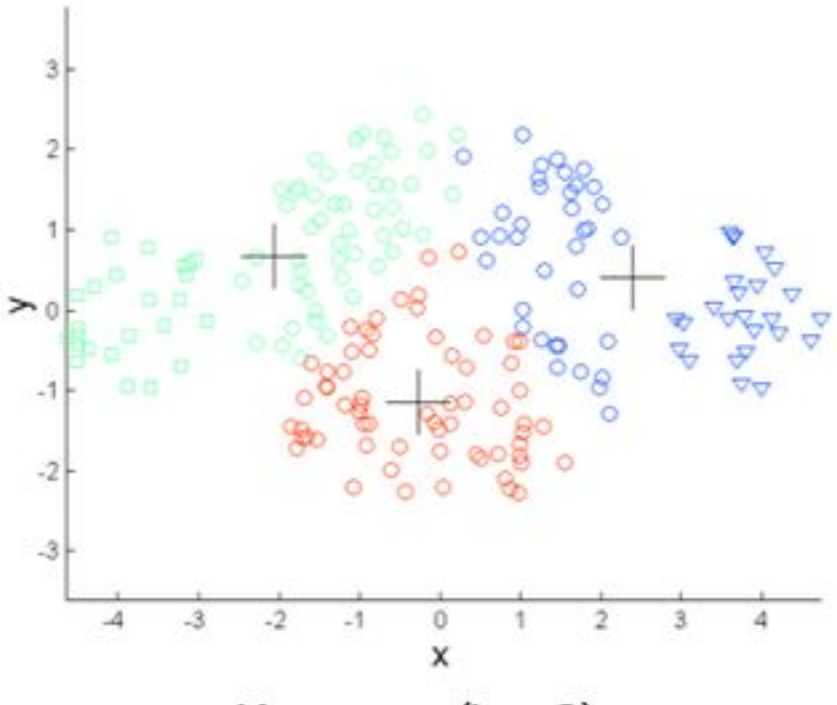
- There are essentially three stopping criteria that can be adopted to stop the K-means algorithm:
  - Centroids don’t change - centroids of newly formed clusters do not change
  - Clusters don’t change - points remain in the same cluster
  - Maximum number of iterations are reached

Challenges with the K-Means

- If the size of clusters is different



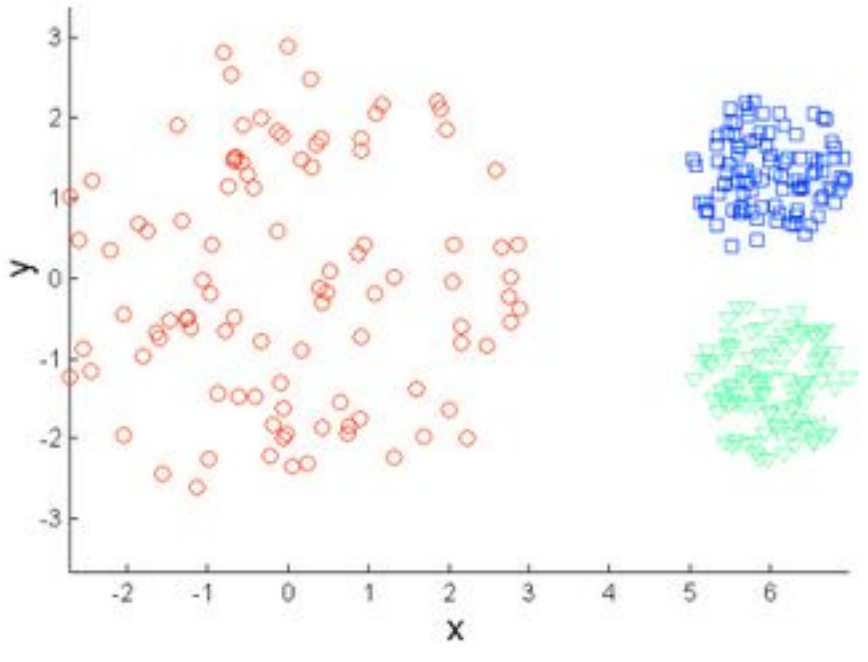
Original Points



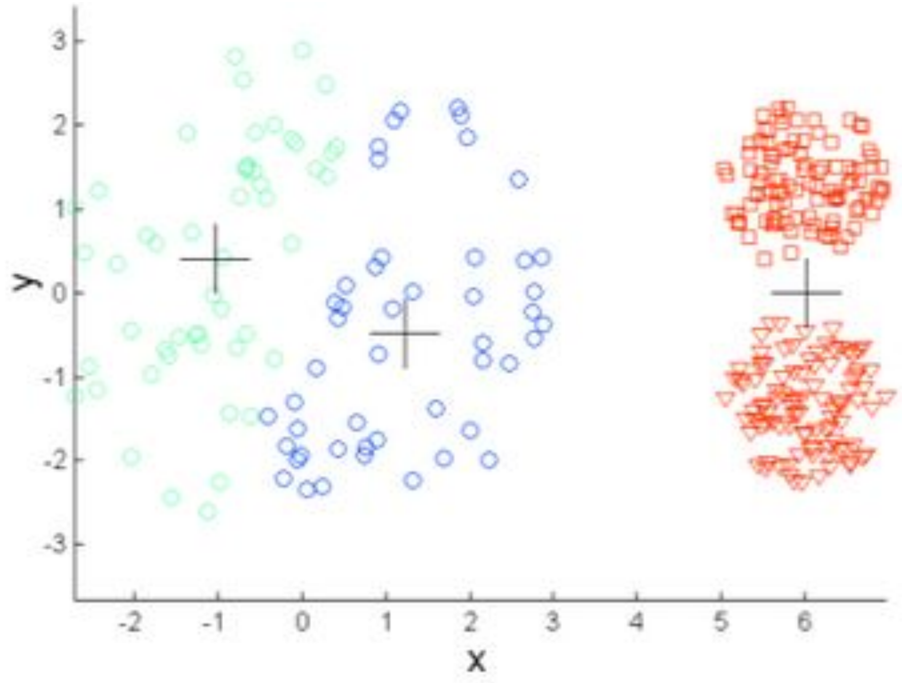
K-means (k = 3)

Challenges with the K-Means

- If the densities of the original points are different



Original Points



K-means (k = 3)

Challenges with the K-Means

- One solution is to use a higher number of clusters. So, in all the previous scenarios, instead of using 3 clusters, we can have a bigger number. Perhaps setting k=10 might lead to more meaningful clusters.
- Keep in mind computation costs of increasing k.

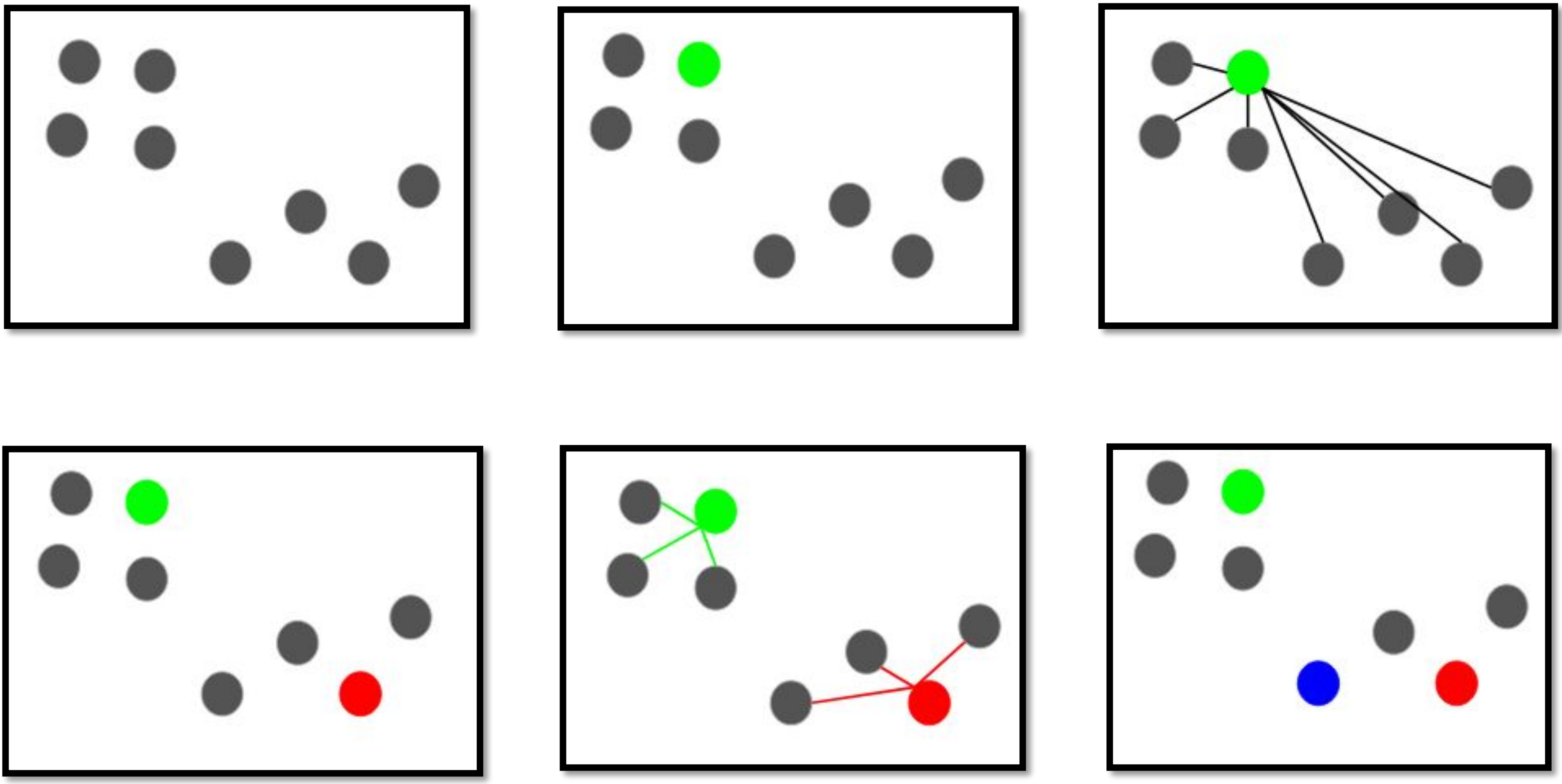
K-Means++

- Remember how we randomly initialize the centroids in k-means clustering?
- Well, this is also potentially problematic because we might get different clusters every time.
  - In some cases, if the initialization of clusters is not appropriate, K-Means can result in arbitrarily bad clusters.
- So, to solve this problem of random initialization, there is an algorithm called K-Means++ that can be used to choose the initial values, or the initial cluster centroids, for K-Means.
- Using K-Means++ to initialize the centroids tends to improve the clusters.

K-Means++ Steps

- Using the K-Means++ algorithm, we optimize the step where we randomly pick the cluster centroid.
- We are more likely to find a solution that is competitive to the optimal K-Means solution while using the K-Means++ initialization.
- The steps to initialize the centroids using K-Means++ are:
  1. The first cluster is chosen uniformly at random from the data points that we want to cluster. This is similar to what we do in K-Means, but instead of randomly picking all the centroids, we just pick one centroid here
  2. Next, we compute the distance (D(x)) of each data point (x) from the cluster center that has already been chosen
  3. Then, choose the next centroid so it will be the one whose squared distance ($D(x)^2$) is the farthest from the current centroid
  4. We then repeat steps 2 and 3 until k clusters have been chosen
- We can continue with the K-Means algorithm after initializing the centroids.
- Although it is computationally costly relative to random initialization, subsequent K-Means often converge more rapidly.

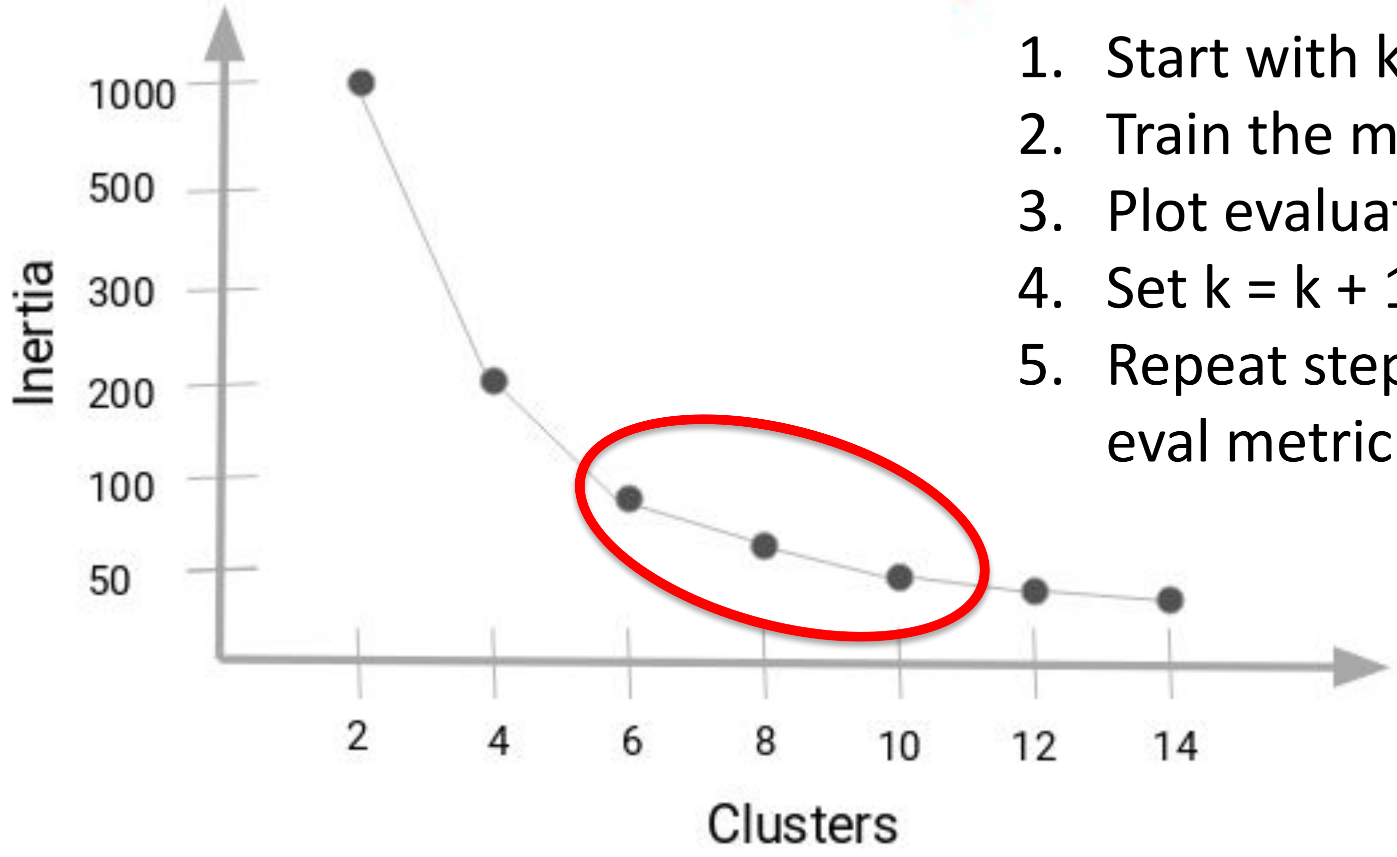
K-Means++ Example (k=3)



How to Choose the Right Number of Clusters in K-Means Clustering?

- The maximum possible number of clusters will be equal to the number of observations in the dataset.
- How do we know the right number of clusters?
- One thing we can do is plot a graph, also known as an **elbow curve**, where the x-axis will represent the number of clusters and the y-axis will be an evaluation metric, like Inertia or Dunn Index.

Inertia as Evaluation Metric



1. Start with k = 2
2. Train the model
3. Plot evaluation metric
4. Set k = k + 1
5. Repeat steps 2-5 until the eval metric levels off

Keep in mind computation costs as you increase and decide on k.