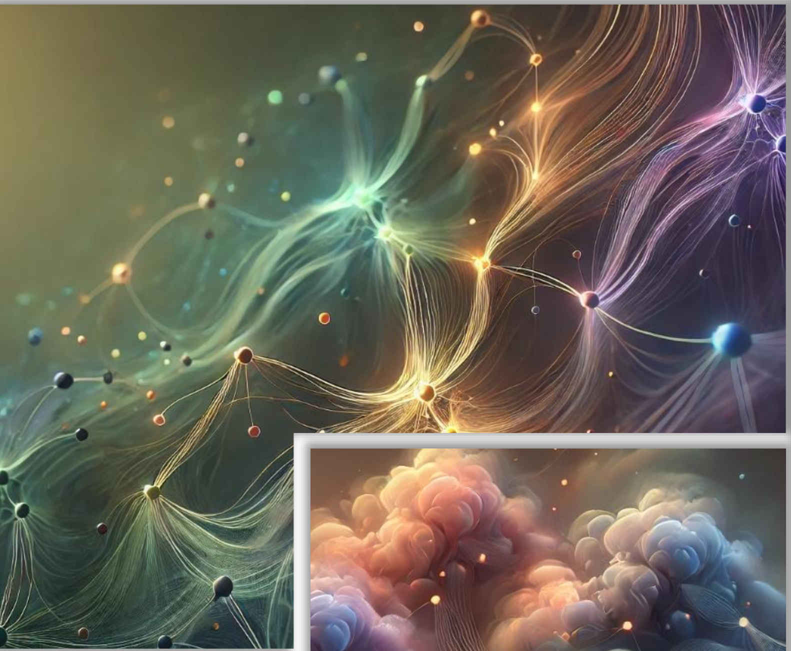


LEHIGH UNIVERSITY

Introduction to Unsupervised Learning

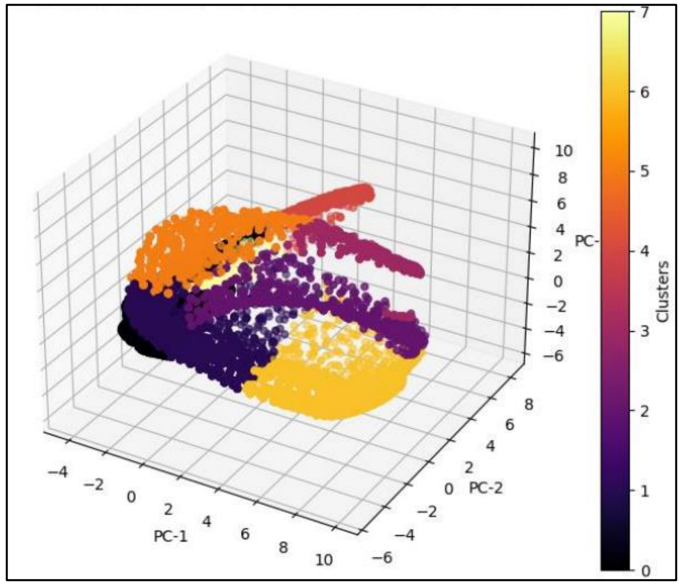
- ISE – 364 / 464
- Dept. of Industrial & Systems Engineering
- Griffin Dean Kent

Overview of Unsupervised Learning

Unsupervised Learning

A category of machine learning problems where one has a set of datapoints $\mathcal{D} := \{x^{(i)}\}_{i=1}^{m}$ that consist of only feature vectors $x^{(i)} \in \mathbb{R}^n$, for all $i \in \{1,2,\dots,m\}$, and does not include any corresponding target labels $y^{(i)}$ that one is trying to predict. As such, the **goal** of unsupervised learning techniques is to **"learn" some type of overall patterns** that are present in the features of the dataset, such as **natural groupings** or other similarities amongst the datapoints. So, in a way, unsupervised learning problems could be viewed as "discovering" or **creating a target variable y** with some number of classes and then group each data point into the class that it is most like. Most unsupervised learning techniques consist of different clustering methods, which are often paired with some type of dimensionality reduction to identify complex and unique patterns in a dataset.

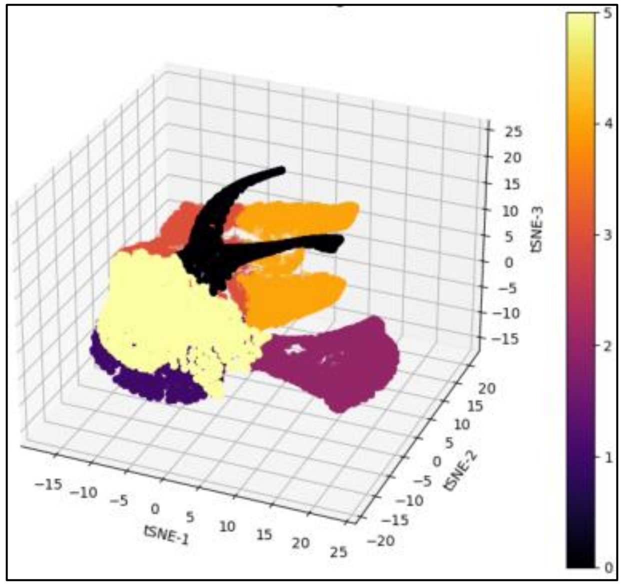
Example of Spectral Clustering



Some of the most common unsupervised learning algorithms consist of the following:

- K-Means Clustering
- Gaussian Mixture Models (GMMs)
- Density-Based Spatial Clustering of Applications with Noise (DBSCAN)
- Spectral Clustering

Example of a K-Means Clustering



Introduction to K-Means Clustering

K-Means Clustering

The simplest and most basic unsupervised learning method to perform clustering on a set of features. The idea behind this method is simple: given some **distortion** (or distance) **metric** $\boldsymbol{D}$ to evaluate, along with some pre-defined number of $\boldsymbol{K} \in \mathbb{N}$ **clusters** one is interested in generating, assign each datapoint $x^{(i)}$ to the cluster that it shows the most similar characteristics with.

Formally, we wish to assign each of the m unlabeled examples given by the dataset $\mathcal{D} = \{x^{(i)}\}_{i=1}^{m}$ to one of K clusters $\{1,2,\dots,K\}$. To understand which datapoint is assigned to which cluster , we can denote the following **latent binary indicator variables** to indicate the assignment of the datapoint $x^{(i)}$ to the k-th cluster:

$$r_{i,k} := \begin{cases} 1, & \text{if } x^{(i)} \text{ is assigned to cluster } k, \\ 0, & \text{otherwise.} \end{cases}$$

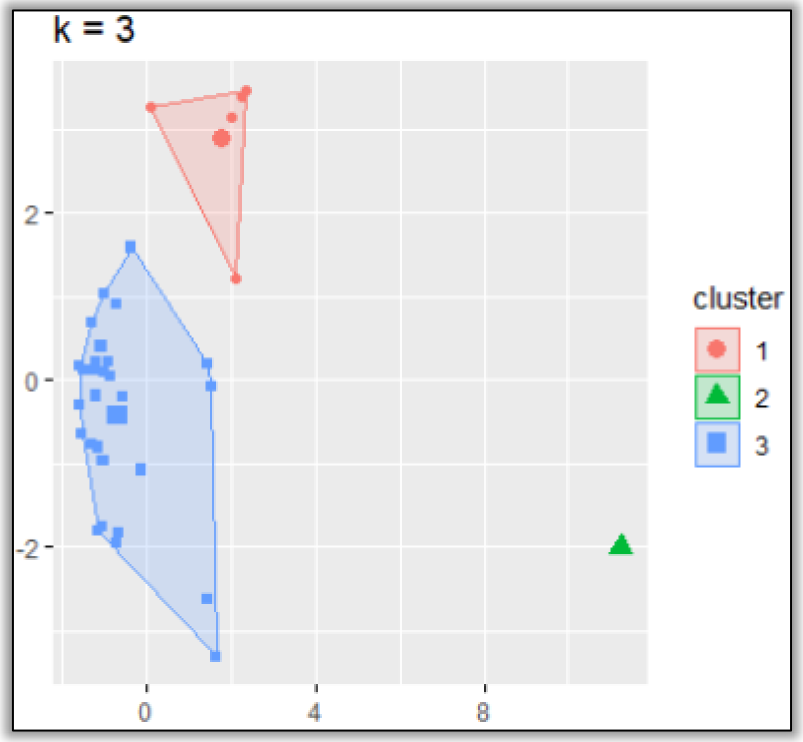
Further, each cluster $k \in \{1,2,\dots,K\}$ has a corresponding **center** (otherwise known as the **"centroid"**) of that cluster, which we denote by $\boldsymbol{\mu_k} \in \mathbb{R}^{\boldsymbol{n}}$. We can use these centroid vectors to represent each of the clusters (since this is ultimately what will be used to determine the distortion measure from the clusters to the datapoints).

Lastly, the distortion metric that is utilized in K-means clustering is simply the Euclidean distance, i.e., the distortion between a datapoint $x^{(i)}$ and some cluster centroid μ_k is given by

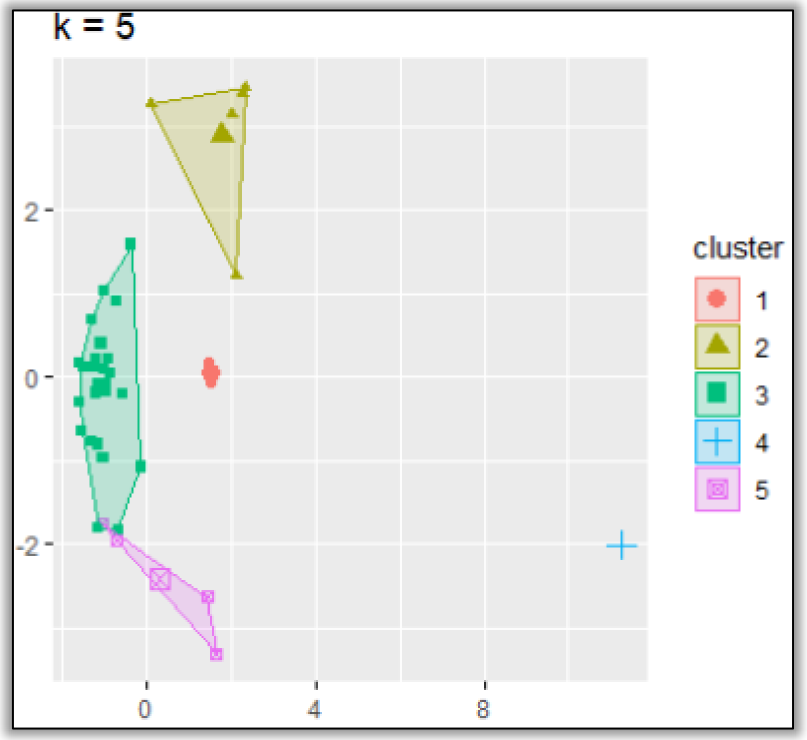
$$D\left(x^{(i)}, \mu_k\right) = \left\|x^{(i)} - \mu_k\right\|_2.$$

Lastly, this distortion metric is then used to assign each datapoint to the cluster that has the "nearest" centroid.

Examples of K-Means Clustering



$K = 3$ clusters



$K = 5$ clusters

Training a K-Means Clustering Model

Optimization Problem to Train a K-Means Clustering Model

In the K-means model, the goal is to **minimize the overall distortion measure defined over all datapoints**. Thus, given some dataset $\mathcal{D} = \{x^{(i)}\}_{i=1}^{m}$ and some initialized K centroid vectors, the K-means model minimizes the loss function

$$J(r, \mu) = \sum_{i=1}^{m} \sum_{k=1}^{K} r_{i,k} D\left(x^{(i)}, \mu_k\right) = \sum_{i=1}^{m} \sum_{k=1}^{K} r_{i,k} \left\| x^{(i)} - \mu_k \right\|_2^2 .$$

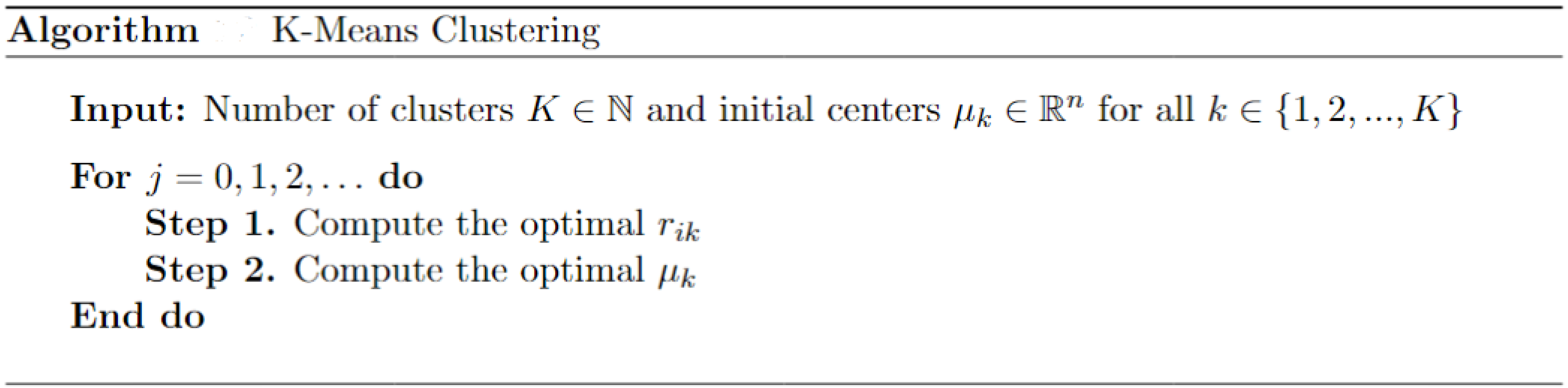
Therefore, the trained K-means model $h_\theta(x)$ is given by the optimal parameters

$$\theta^* = \operatorname*{argmin}_{r,\mu} J(r, \mu),$$

where the parameters are given by $\theta = [r^T, \mu^T]^T$.

- Notice that the **Hessian of the loss function J is positive definite (strictly convex) in the variables μ** for a given vale of r (under the assumption that each of the K clusters has at least 1 datapoint assigned to it) (Proof as Homework). However, J is not strictly convex in both μ and r (**J is in fact nonconvex in both variables r and μ**).
- As such, this optimization problem is solved in **two phases**: (**phase 1**) Update the cluster assignment variables r and then (**phase 2**) update the centroid locations μ
- It bears mentioning that the centroids are typically initialized as random vectors or are randomly assigned certain K distinct examples form the dataset.

The K-Means Clustering Algorithm



```
Algorithm  K-Means Clustering

    Input: Number of clusters $K \in \mathbb{N}$ and initial centers $\mu_k \in \mathbb{R}^n$ for all $k \in \{1, 2, ..., K\}$

    For $j = 0, 1, 2, \ldots$ do
        Step 1. Compute the optimal $r_{ik}$
        Step 2. Compute the optimal $\mu_k$
    End do
```

Updating the Assignment Variables $\boldsymbol{r_{i,k}}$

K-Means Phase (1) – Updating the Assignment Variables $\boldsymbol{r_{i,k}}$

The first step of the K-means clustering algorithm fixes the vales of the centroids μ_k for each $k \in \{1,2,\dots,K\}$. Then, for each given datapoint $x^{(i)}$, the relevant expression that needs to be minimized is the inner sum of J over the decision variables $r_{i,k} \in \{0,1\}$, i.e.,

$$\min_{r_{i,k}} J\left(r_{i,k}; \mu\right) = \sum_{k=1}^{K} r_{i,k} \left\| x^{(i)} - \mu_k \right\|_2^2 .$$

- This function J can be minimized by setting each $r_{i,k}$ to 1 if the k-th centroid μ_k is the closest of the centroids to the datapoint $x^{(i)}$.
- Therefore, to determine which centroid is closest to the datapoint $x^{(i)}$, one must compute all the Euclidean distances $\left\| x^{(i)} - \mu_k \right\|_2^2$ for each of the centroids and then simply choose the cluster that yields the smallest distance. The other "non-optimal" assignments $r_{i,k}$ are set to 0.

- Notice that minimizing J over $r_{i,k}$ is an integer (combinatorial) optimization problem since $r_{i,k} \in \{0,1\}$. As such, it is nondifferentiable and nonconvex.

Updating the Centroids $\boldsymbol{\mu_k}$

K-Means Phase (2) – Updating the Centroid Variables $\boldsymbol{\mu_k}$

The second step of the K-means clustering algorithm fixes the vales of the cluster assignments $r_{i,k}$ for each $i \in \{1,2,\dots,m\}$. Then, for each given cluster $k \in \{1,2,\dots,K\}$, the relevant expression that needs to be minimized is the inner sum of J over the decision variables $\mu_k \in \mathbb{R}^n$, i.e.,

$$\min_{\mu_k} J(\mu_k; r) = \sum_{i=1}^{m} r_{i,k} \left\| x^{(i)} - \mu_k \right\|_2^2 .$$

- Recall that, for fixed cluster assignments $r_{i,k}$, this function is **strictly convex in $\boldsymbol{\mu}$** (Hessian is positive definite) under the assumption that each of the K clusters has at least 1 datapoint assigned to it.
- Therefore, one can solve for the minimum value μ_k^* analytically, with an optimal value of (proof as Homework)

$$\mu_k^* := \frac{\sum_{i=1}^{m} r_{i,k} x^{(i)}}{\sum_{i=1}^{m} r_{i,k}} .$$

- Notice that since $r_{i,k} \in \{0,1\}$, the denominator will simply be the number of datapoints assigned to cluster k. Therefore, we can see that the optimal solution update of the centroids will simply be the average (mean) of the datapoints that are assigned to cluster k, hence the name **"K-Means"**.

- These two phases continue alternating back-and-forth until there are no more changes in the cluster assignments $r_{i,k}$.

Illustration of Training a K-Means Model

Steps

- Dataset of features $x \in \mathbb{R}^2$.

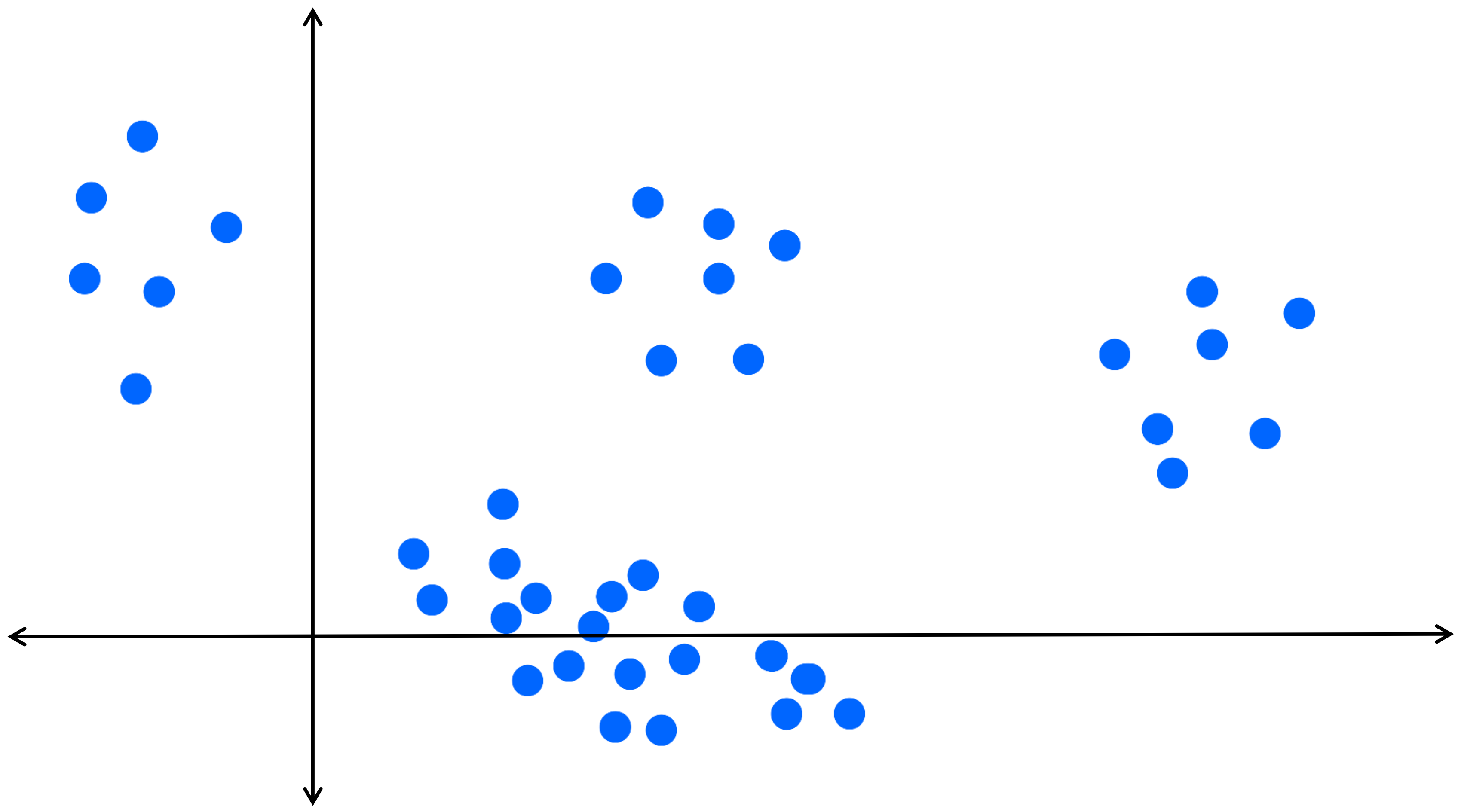


Illustration of Training a K-Means Model

Steps

- Dataset of features $x \in \mathbb{R}^2$.
- Choose the number of clusters to be $K = 4$.
- Randomly initialize the centroids $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$, $\boldsymbol{\mu}_3$, and $\boldsymbol{\mu}_4$ (all in $\mathbb{R}^2$).

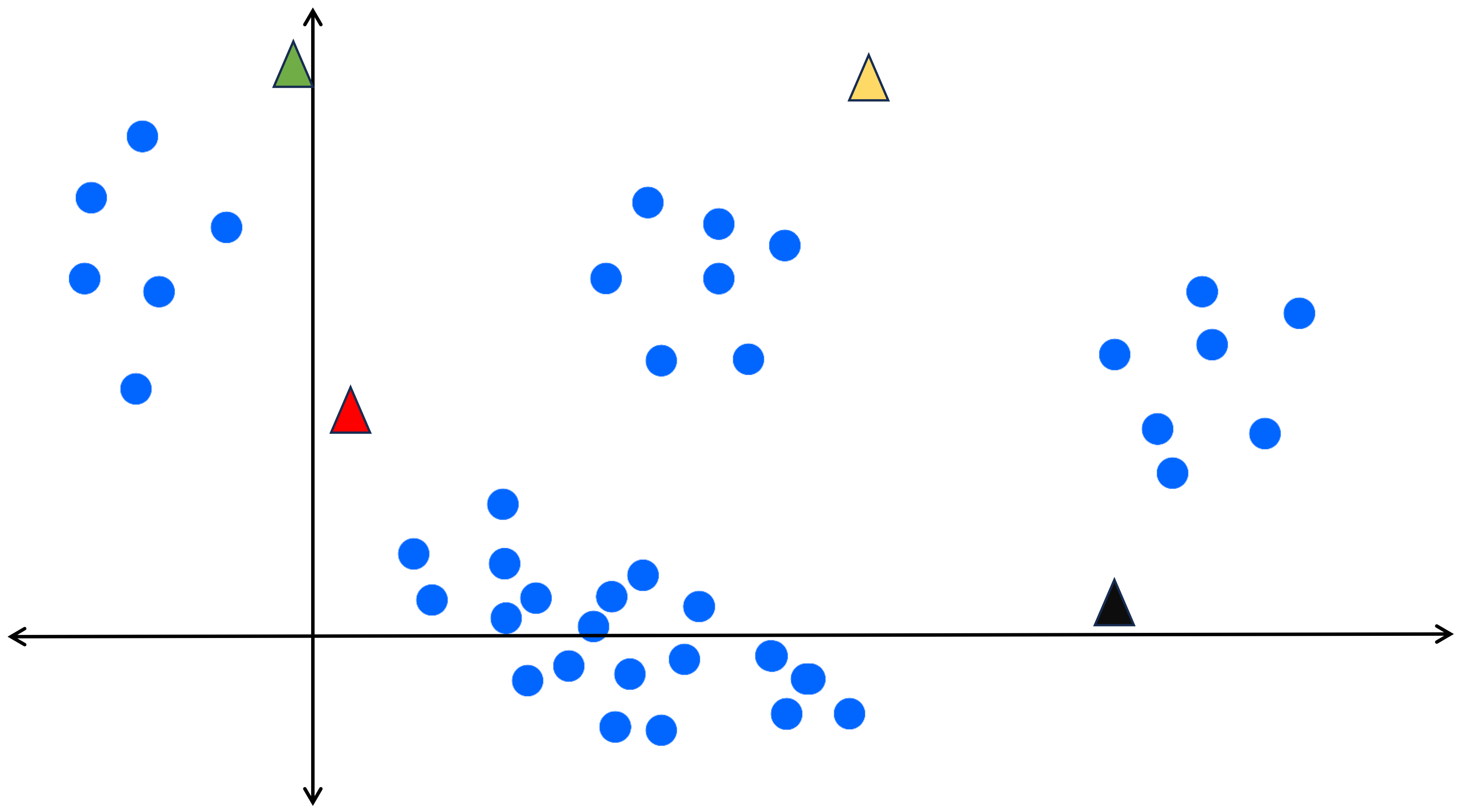


Illustration of Training a K-Means Model

Steps

- Dataset of features $x \in \mathbb{R}^2$.
- Choose the number of clusters to be $K = 4$.
- Randomly initialize the centroids $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$, $\boldsymbol{\mu}_3$, and $\boldsymbol{\mu}_4$ (all in $\mathbb{R}^2$).
- (**Phase 1**) Assign datapoints to their nearest centroids (i.e., update cluster assignments $r_{i,k}$).

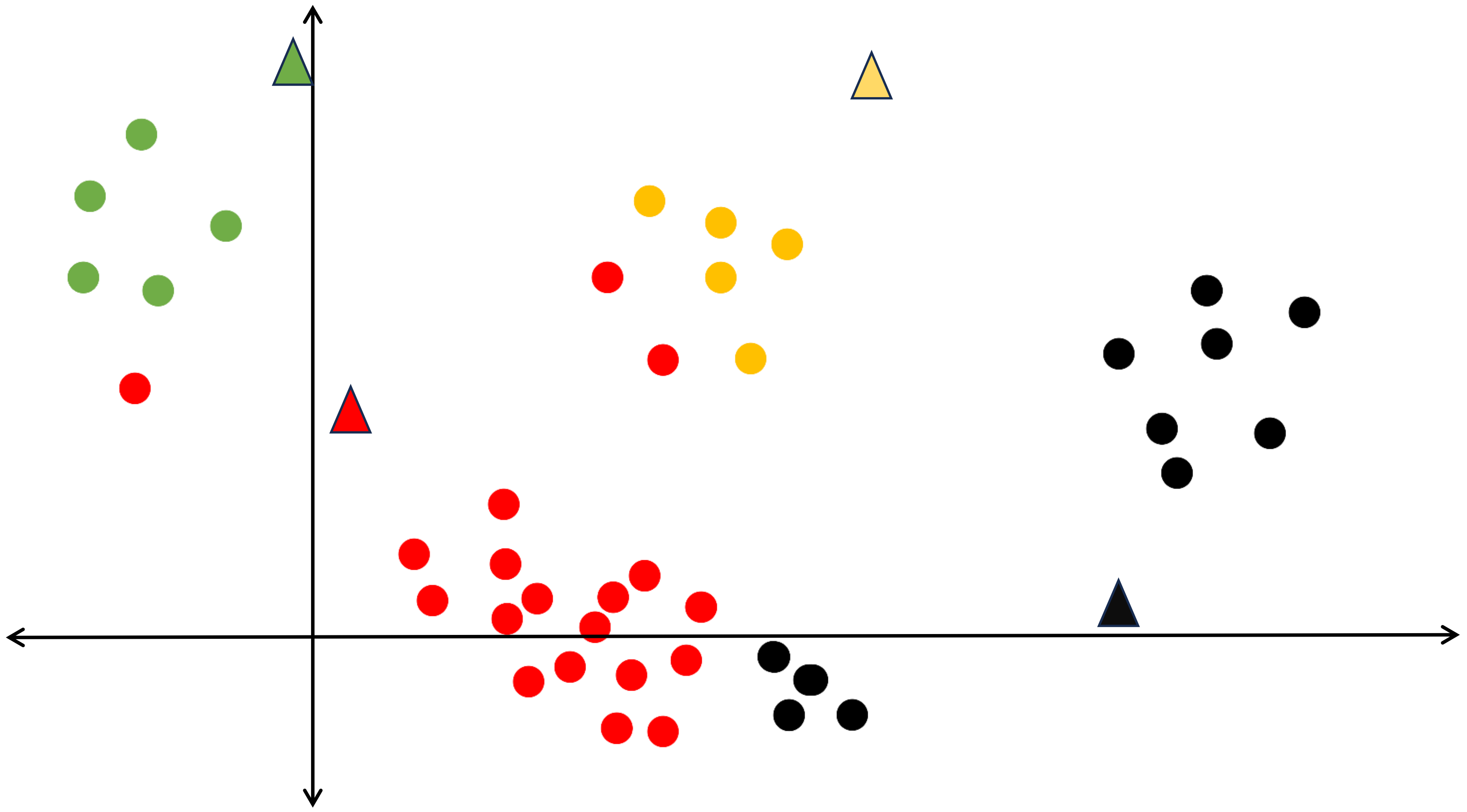


Illustration of Training a K-Means Model

Steps

- Dataset of features $x \in \mathbb{R}^2$.
- Choose the number of clusters to be $K = 4$.
- Randomly initialize the centroids $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$, $\boldsymbol{\mu}_3$, and $\boldsymbol{\mu}_4$ (all in $\mathbb{R}^2$).
- (**Phase 1**) Assign datapoints to their nearest centroids (i.e., update cluster assignments $r_{i,k}$).
- (**Phase 2**) Update the location of the centroids to the average of their datapoints.

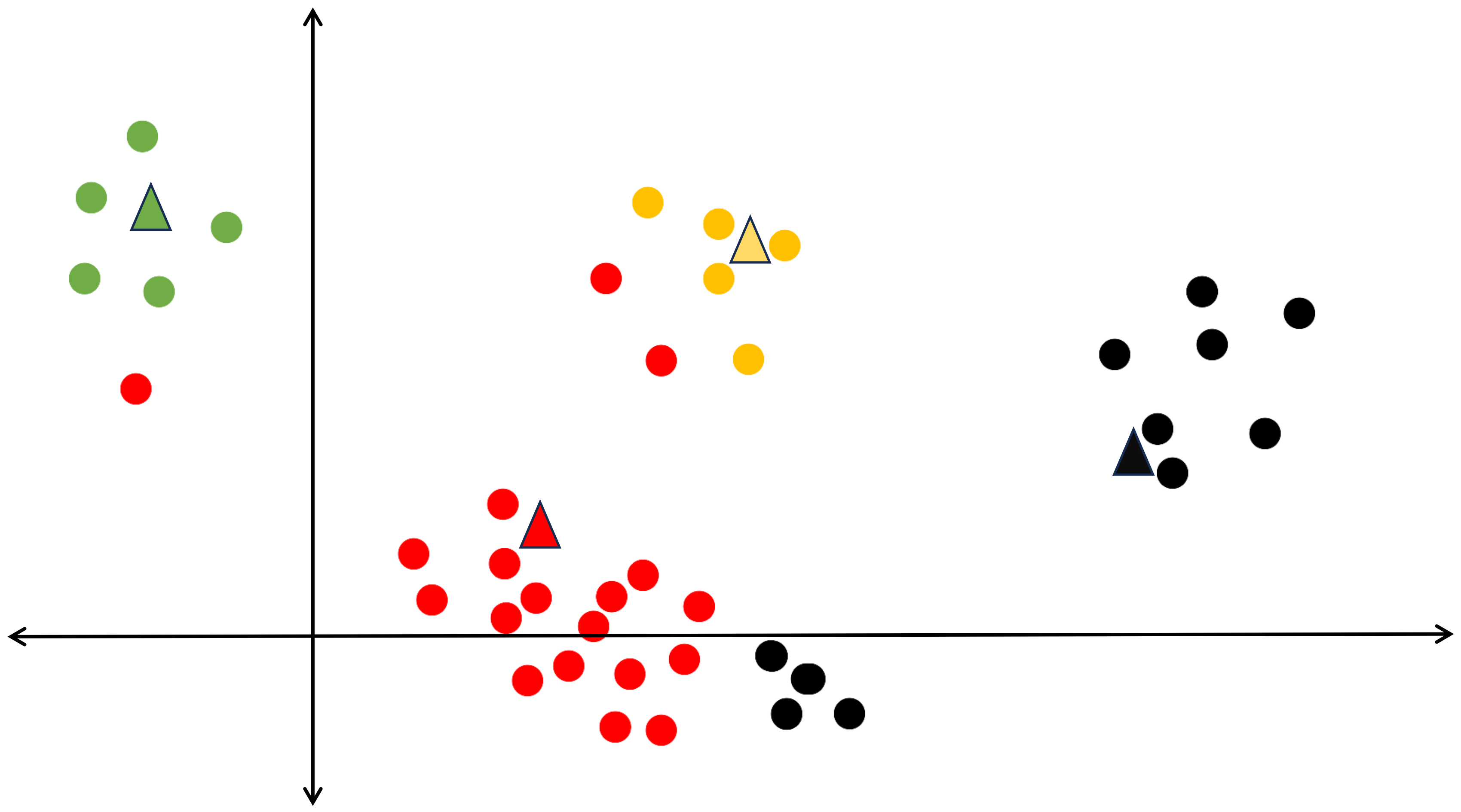


Illustration of Training a K-Means Model

Steps

- Dataset of features $x \in \mathbb{R}^2$.
- Choose the number of clusters to be $K = 4$.
- Randomly initialize the centroids $\boldsymbol{\mu_1}$, $\boldsymbol{\mu_2}$, $\boldsymbol{\mu_3}$, and $\boldsymbol{\mu_4}$ (all in $\mathbb{R}^2$).
- (**Phase 1**) Assign datapoints to their nearest centroids (i.e., update cluster assignments $r_{i,k}$).
- (**Phase 2**) Update the location of the centroids to the average of their datapoints.
- (**Phase 1**) Update cluster assignments $r_{i,k}$.

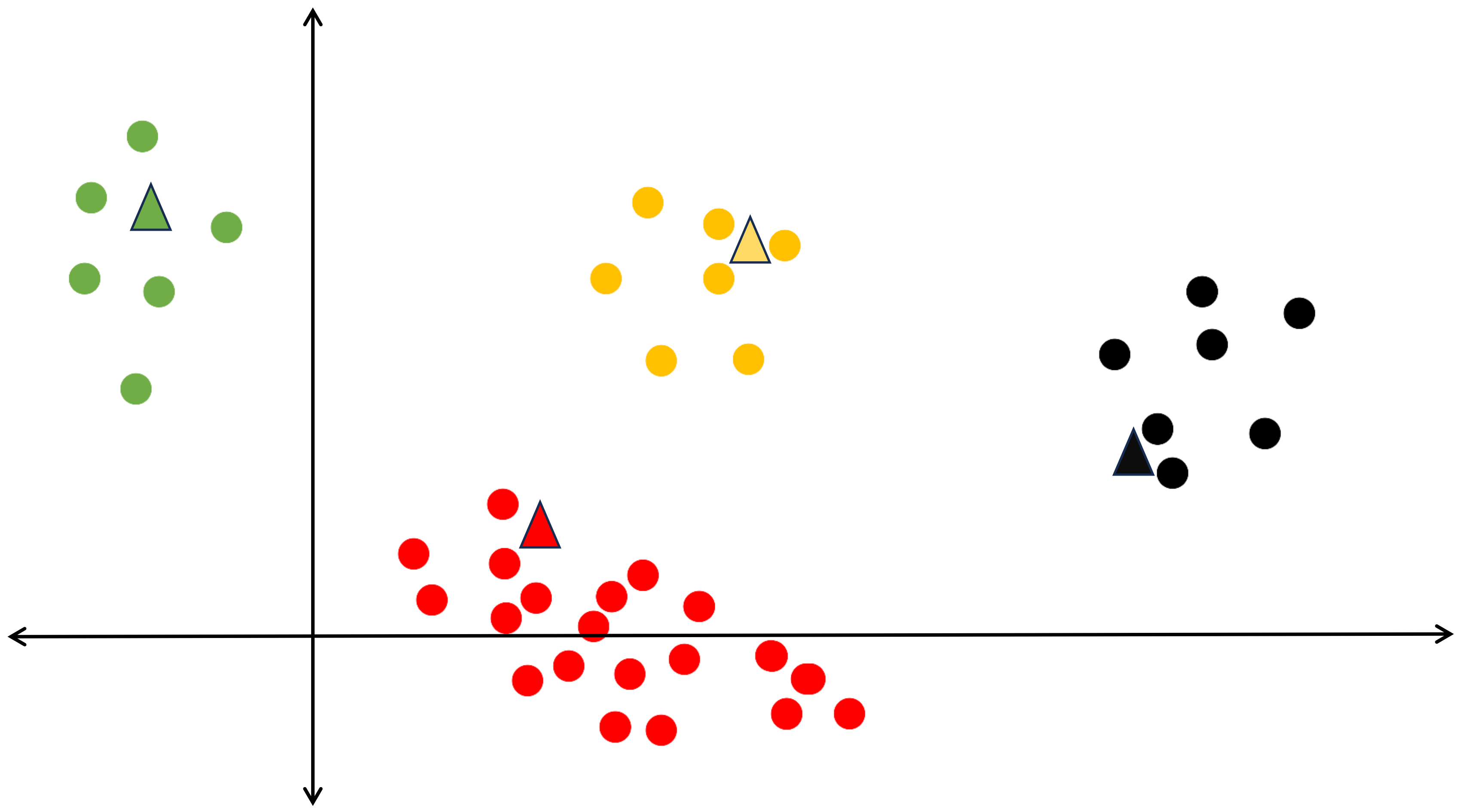


Illustration of Training a K-Means Model

Steps

- Dataset of features $x \in \mathbb{R}^2$.
- Choose the number of clusters to be $K = 4$.
- Randomly initialize the centroids $\boldsymbol{\mu_1}$, $\boldsymbol{\mu_2}$, $\boldsymbol{\mu_3}$, and $\boldsymbol{\mu_4}$ (all in $\mathbb{R}^2$).
- (**Phase 1**) Assign datapoints to their nearest centroids (i.e., update cluster assignments $r_{i,k}$).
- (**Phase 2**) Update the location of the centroids to the average of their datapoints.
- (**Phase 1**) Update cluster assignments $r_{i,k}$.
- (**Phase 2**) Update the location of the centroids.

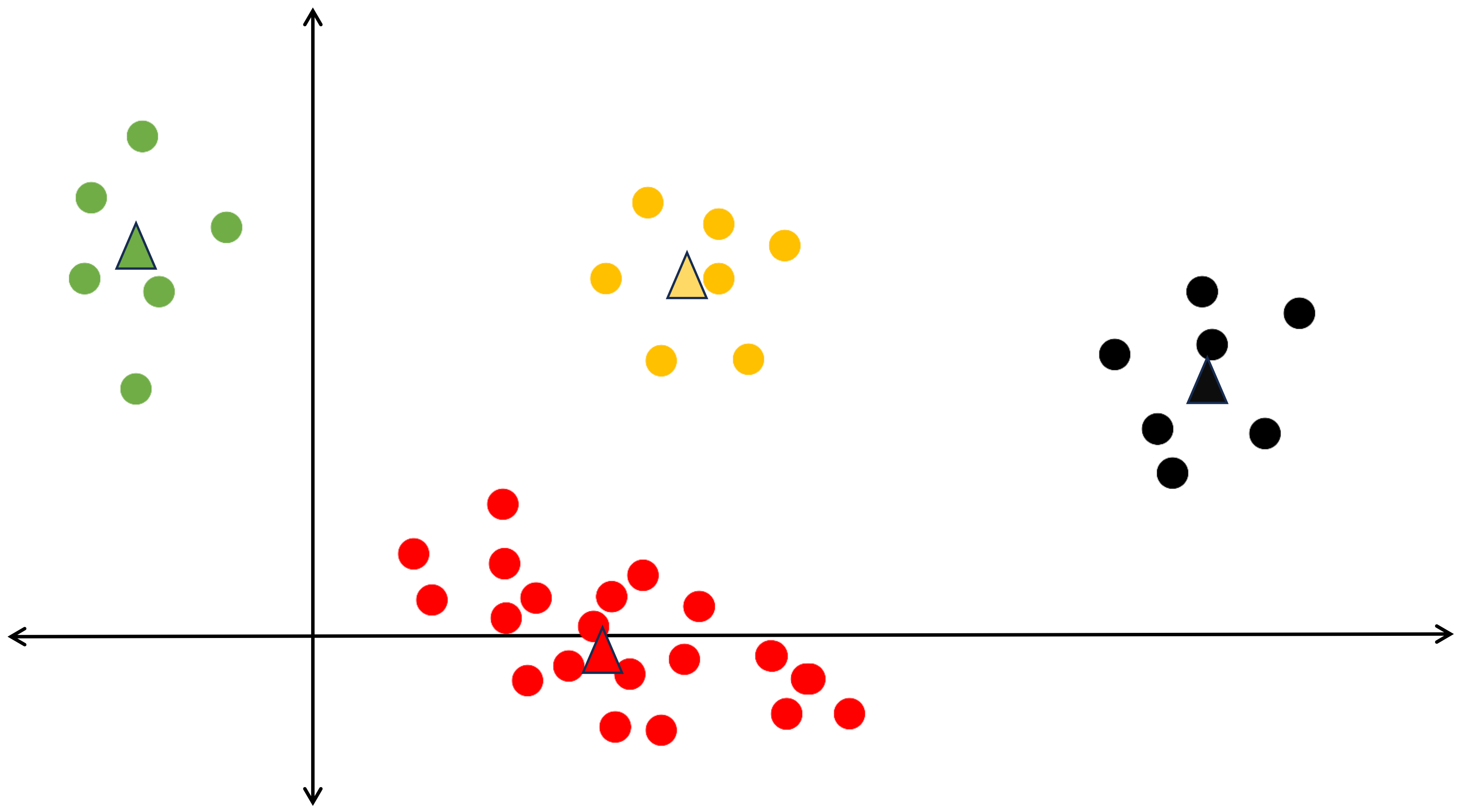
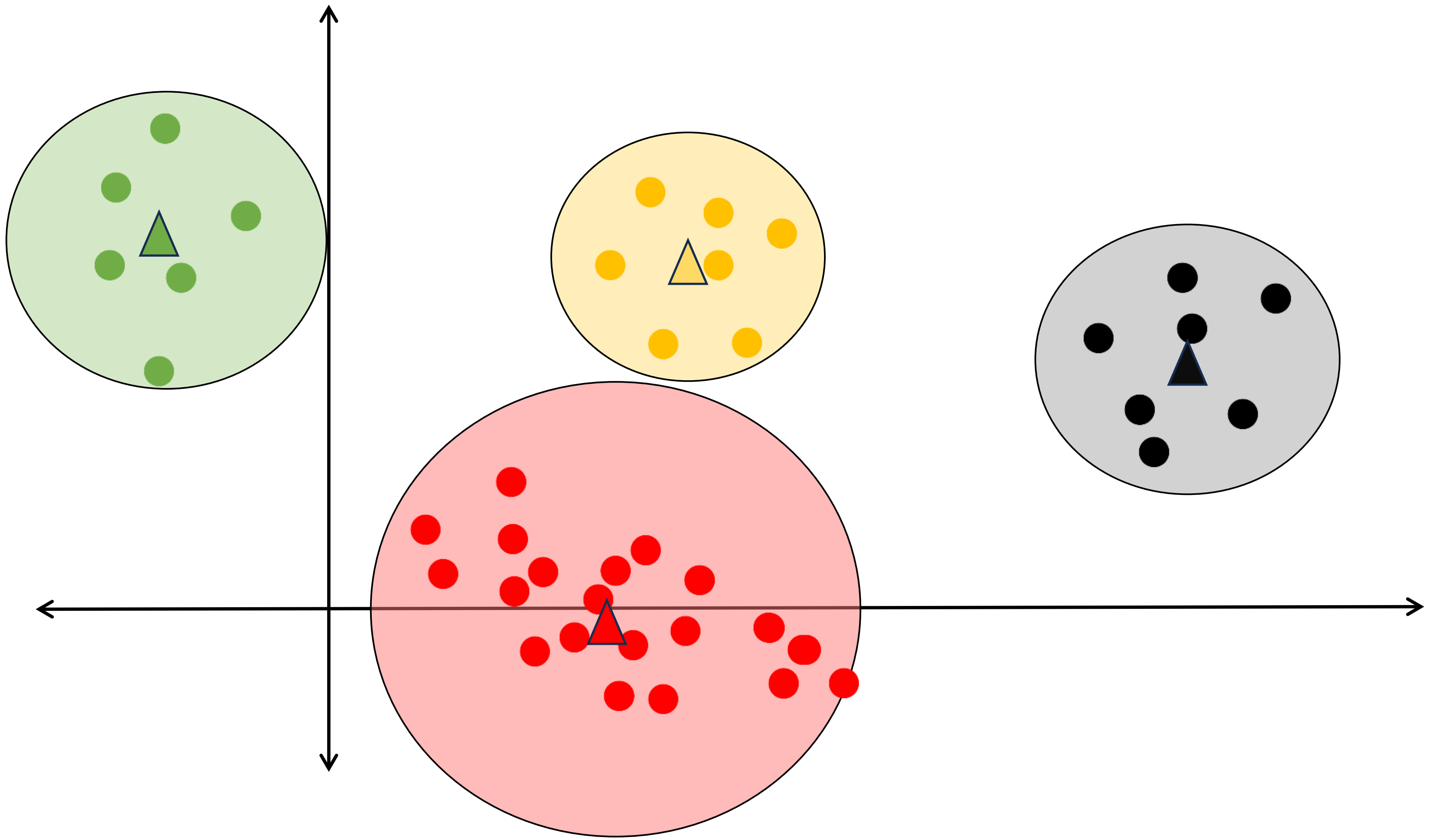


Illustration of Training a K-Means Model

Steps

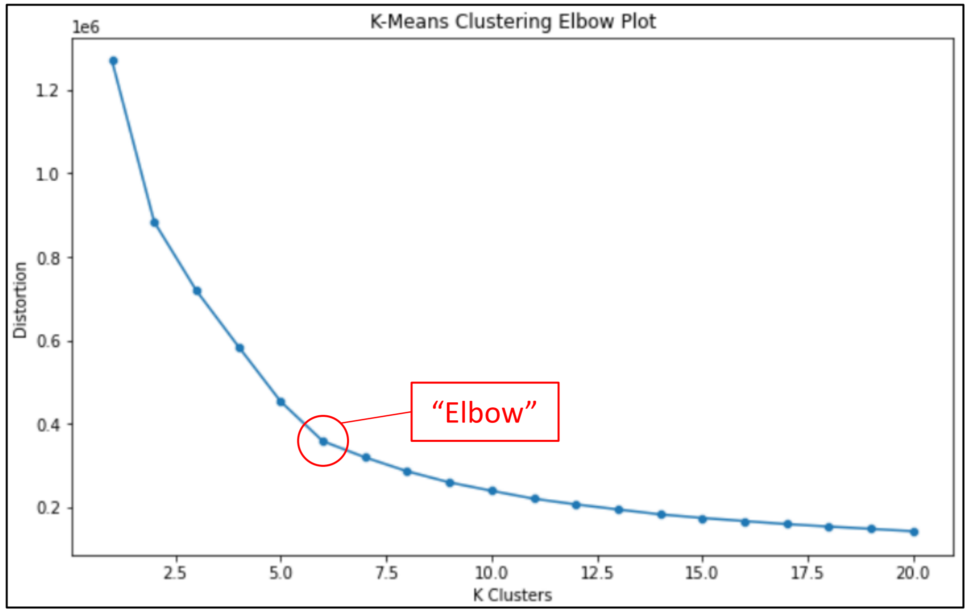
- Dataset of features $x \in \mathbb{R}^2$.
- Choose the number of clusters to be $K = 4$.
- Randomly initialize the centroids $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$, $\boldsymbol{\mu}_3$, and $\boldsymbol{\mu}_4$ (all in $\mathbb{R}^2$).
- (**Phase 1**) Assign datapoints to their nearest centroids (i.e., update cluster assignments $r_{i,k}$).
- (**Phase 2**) Update the location of the centroids to the average of their datapoints.
- (**Phase 1**) Update cluster assignments $r_{i,k}$.
- (**Phase 2**) Update the location of the centroids.
- **Finish** (because no more cluster assignments will change.



Hyperparameter Tuning

Lack of Global Optimality

- One important thing to realize is that, because the **loss function for K-means is non-convex** in the decision variables, the solution that is ultimately obtained is **not guaranteed to be the best (global) solution overall**.
- To help alleviate this, and to get a **“good” heuristic solution**, what is done in practice is to use **multiple different starting points** for the centroids (many different random starting points). Then, among all the resulting clusterings, the one that yields the minimum total distortion is chosen.

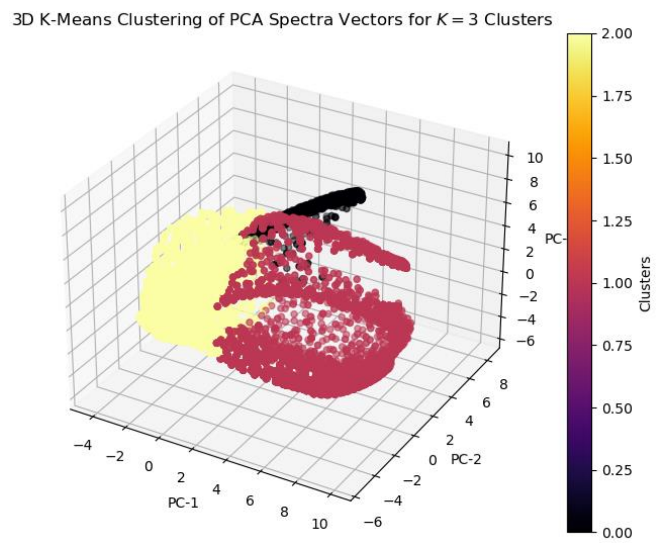


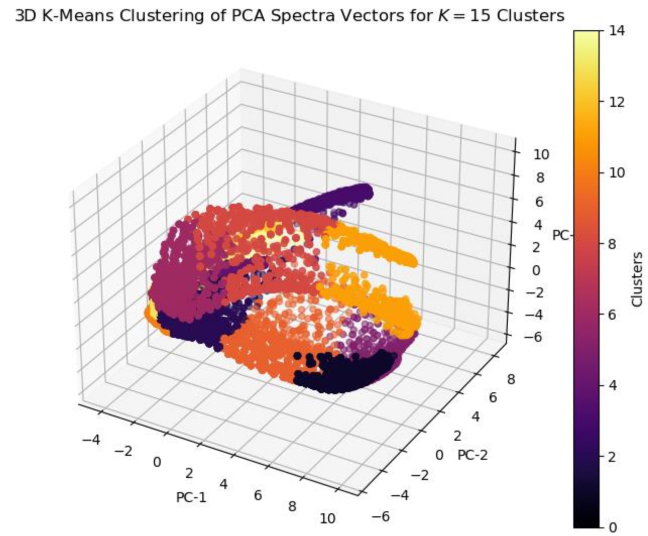
Hyperparameter Tuning the Number of Clusters

- Perhaps the most relevant question when it comes to K-means clustering (and clustering in general) is how many clusters should we generate in the first place?
- Typically, what is most common practice is to generate what is referred to as an “**elbow plot**”. This is a series of experiments that generates different numbers of K clusters (ex., in the figure on the left, $K \in \{2,3,\dots,20\}$) and then plots the total distortion for all of them. The idea is then to choose the number of clusters that indicates an “**elbow**” in the plot (This is usually the point at which the most gain has been achieved in minimizing the distortion while ideally not being too complex – not using too many clusters).

Illustration of the Limitations of K-Means

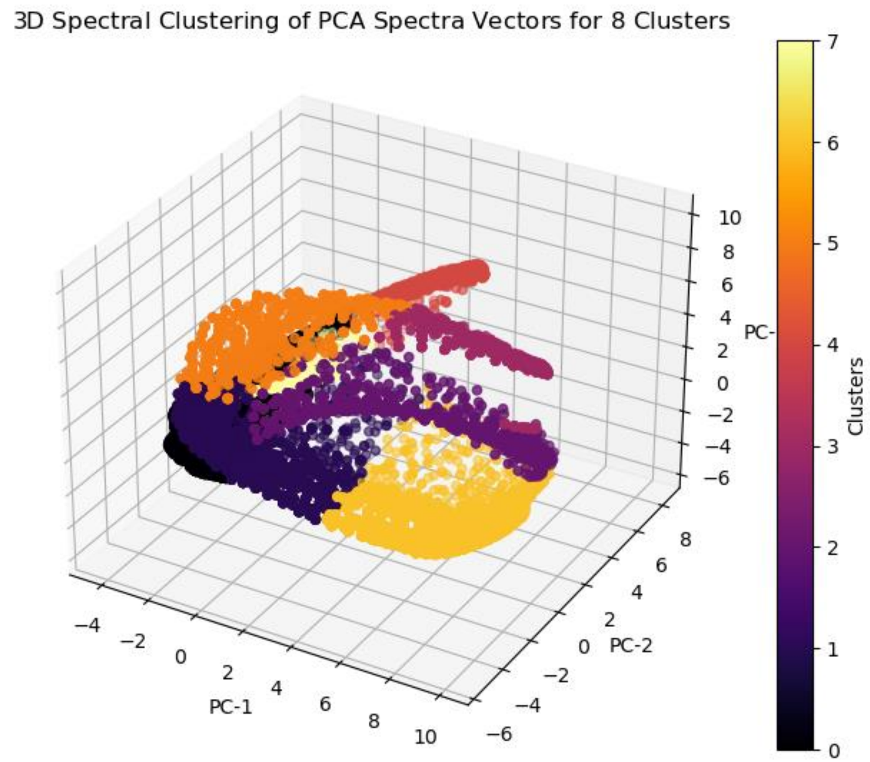
K-Means Clusters





- An example of two different K-means clusterings using the top 3 principal component embeddings of a dataset. The top plot displays a $K = 3$ clustering and the bottom displays a $K = 15$ clustering.
- Notice the limitations on the type of patterns that K-means can identify.
- K-means can only identify clusters in a spherical pattern (since it utilizes the Euclidean norm as the distortion).
- As such, K-means does not excel at identifying highly nonconvex patterns.

Spectral Clusters



- When dealing with data that exhibits highly nonconvex patterns, other methods are more suitable (such as spectral clustering).

Connection to Gaussian Mixture Models

- As we have already discussed, the **K-means** algorithm **assumes** that the data can be grouped into spherical clusters (due to the use of the Euclidean ℓ_2-norm). Is there a probabilistic interpretation that this assumption is equivalent to?
- Assuming spherical clusters is equivalent to assuming that the K clusters are multivariate Gaussian distributions with corresponding means of $\mu_k \in \mathbb{R}^n$ and covariance matrices equal to the identity matrix, i.e., $\Sigma_k = I \in \mathbb{R}^{n \times n}$ (this guarantees that the features, when viewed as random variables, will be independent which will yield the spherical nature). Thus, K-means assumes that each cluster k has a distribution $\mathcal{N}(\mu_k, I)$.
- Considering this realization, a natural question might be, could we develop better clusters by not restricting the covariance to the identity matrix (i.e., allow for non-spherical clusters)?

Gaussian Mixture Models (GMMs)

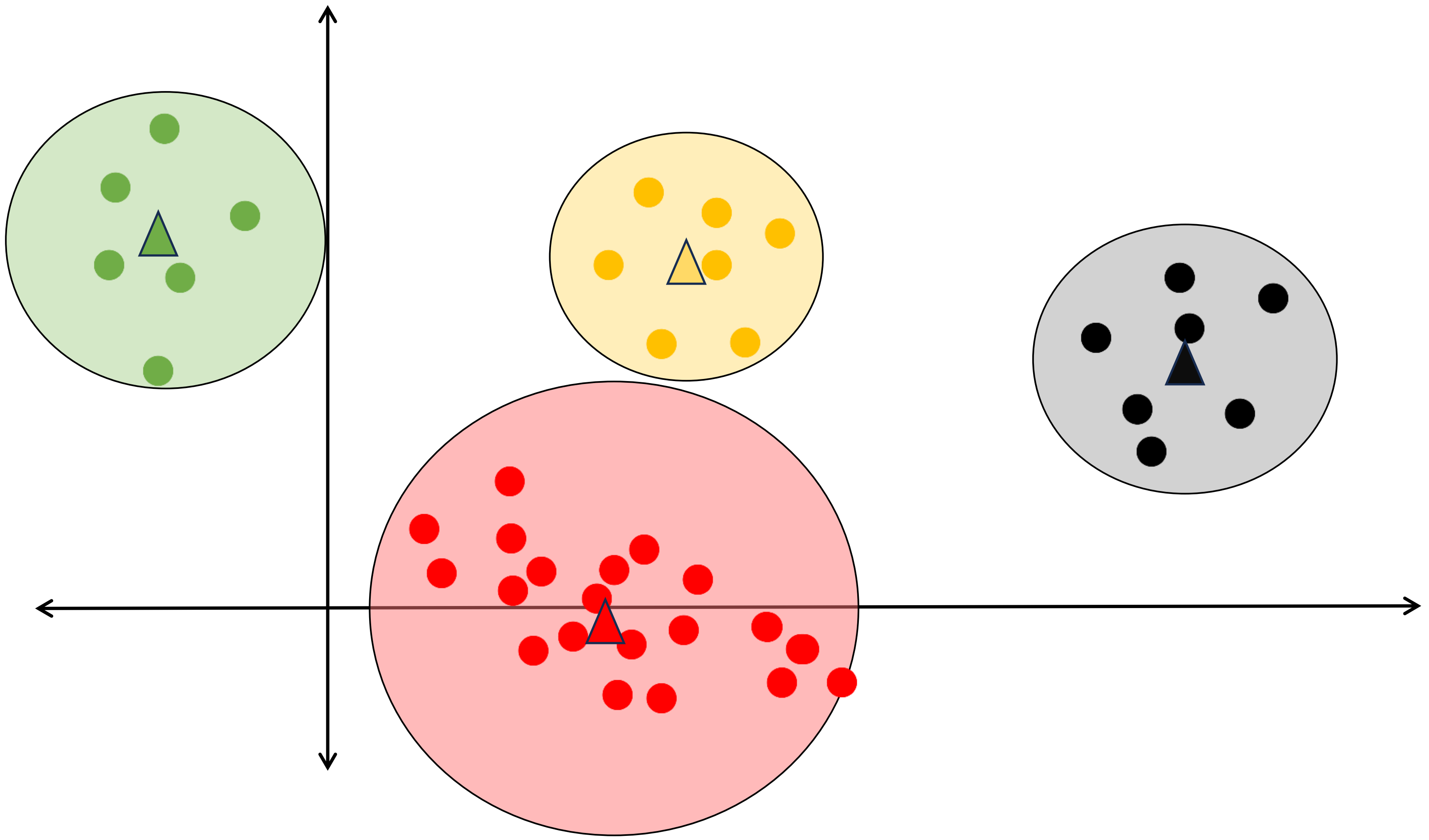
This method extends the ide of viewing each of the ***K* clusters as multivariate Gaussian distributions**, only we no longer impose the assumption on the covariance matrices that restricts their value to the identity matrix. Thus, we instead assume that the data are generated from K different Gaussian distributions, given by

$$\mathcal{N}(x; \mu_k, \Sigma_k), \qquad \text{for all } k \in \{1,2,\dots,K\}.$$

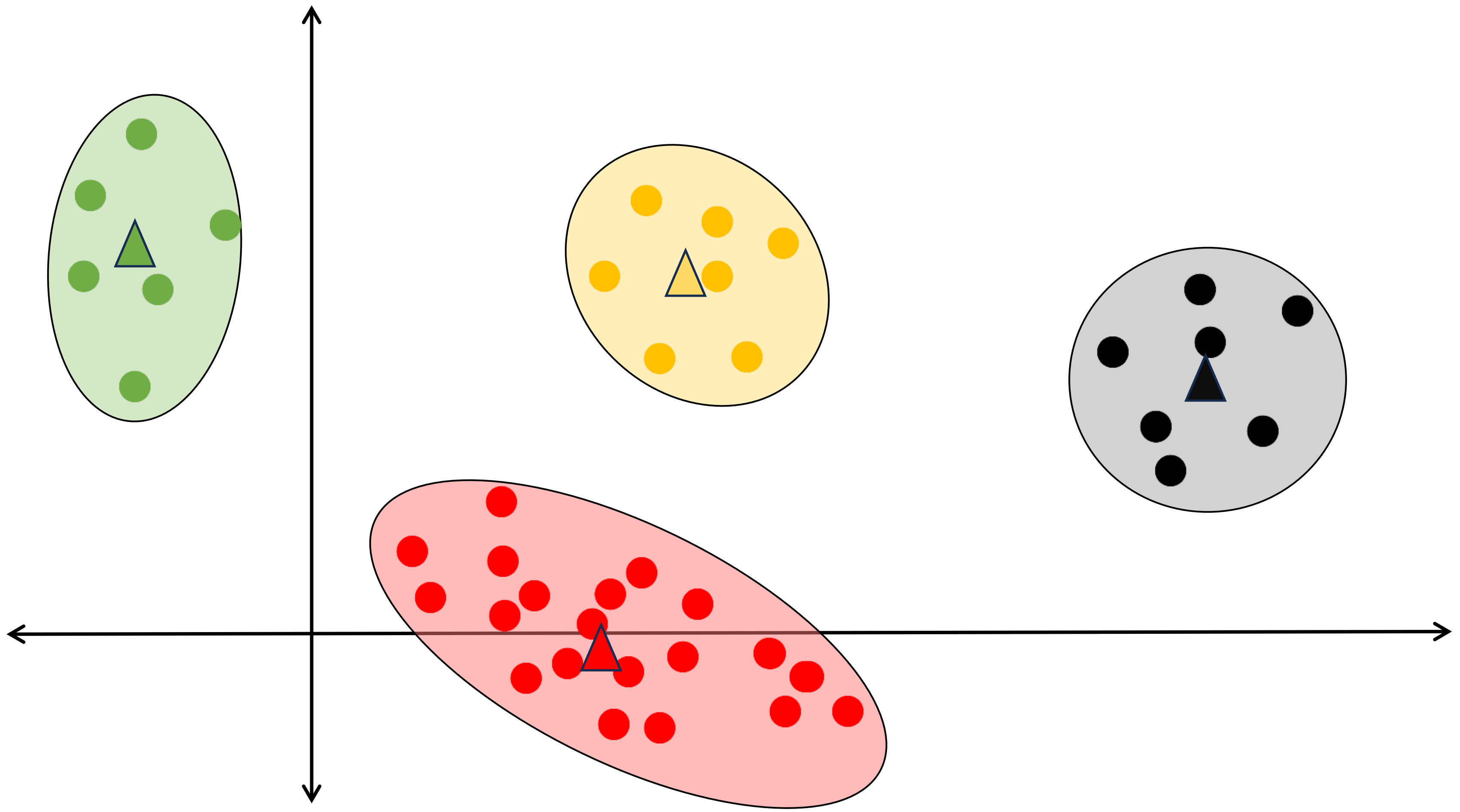
Where $\mu_k \in \mathbb{R}^n$ and $\Sigma_k \in \mathbb{R}^{n \times n}$ denote the mean vector and covariance matrix corresponding to the k-th Gaussian. Further, the notation $\mathcal{N}(x; \mu_k, \Sigma_k)$ denotes that the datapoint x follows the Gaussian distribution $\mathcal{N}(\mu_k, \Sigma_k)$, i.e., $x \sim \mathcal{N}(\mu_k, \Sigma_k)$. We can then "mix" the K Gaussians together (by letting $\phi_k \in (0,1)$ denote the probability of a datapoint coming from the k-th respective cluster) as a weighted sum of the Gaussians. Thus, the probability of observing any datapoint $x^{(i)}$ is

$$\mathbb{P}\left(x^{(i)}\right) = \sum_{k=1}^{K} \phi_k \mathcal{N}\left(x^{(i)}; \mu_k, \Sigma_k\right).$$

Example of Original K-Means Spherical Clusters



Example of General GMM Clusters



Alternative Formulation of the Probability of GMMs

- Let's define the **latent binary indicator variables $\boldsymbol{r_{i,k}} \in \{\mathbf{0}, \mathbf{1}\}$**, which will take a value of 1 if the datapoint $x^{(i)}$ is assigned to the k-th cluster and a value of 0 otherwise. Then, we can define the GMM mixture weights ϕ_k as the prior probabilities of $r_{i,k}$, i.e., $\mathbb{P}(r_{i,k} = 1) = \phi_k$ for all datapoints $i \in \{1,2,\dots,m\}$.

- Further, since each datapoint $x^{(i)}$ can only come from **one** of the Gaussian clusters, we can see that the random vector $r_i = [r_{i,1}, r_{i,2}, \dots, r_{i,K}]^T$ has a **multinomial distribution** with probability parameters $(\phi_1, \phi_2, \dots, \phi_K)$, where ϕ_k denotes the probability that the k-th element of the vector r_i takes a value of 1 and the remaining elements take values of 0.

- Using this new notation, we can rewrite the probability of observing any datapoint $x^{(i)}$, originally given by $\mathbb{P}(x^{(i)}) = \sum_{k=1}^{K} \phi_k \mathcal{N}(x^{(i)}; \mu_k, \Sigma_k)$, as the equation $\mathbb{P}(x^{(i)}) = \sum_{r_i} \mathbb{P}(r_i)\mathbb{P}(x^{(i)}|r_i)$, where **$\mathbb{P}(\boldsymbol{r_i})$ is the probability of the multinomial random variable $\boldsymbol{r_i}$** (that is, the probability that datapoint $x^{(i)}$ comes from cluster k) given by

$$\mathbb{P}(r_i) = \prod_{k=1}^{K} \phi_k^{r_{i,k}} = \prod_{k=1}^{K} \phi_k^{\mathbb{I}[r_{i,k}=1]},$$

and where **$\mathbb{P}(\boldsymbol{x^{(i)}}|\boldsymbol{r_i})$ denotes the distribution of the $\boldsymbol{k}$-th Gaussian that the datapoint $\boldsymbol{x^{(i)}}$ is drawn from** and is given by

$$\mathbb{P}(x^{(i)}|r_i) = \prod_{k=1}^{K} \mathcal{N}(x^{(i)}; \mu_k, \Sigma_k)^{r_{i,k}} = \prod_{k=1}^{K} \mathcal{N}(x^{(i)}; \mu_k, \Sigma_k)^{\mathbb{I}[r_{i,k}=1]}.$$

Defining Posterior Probabilities of GMMs

Up to this point in dealing with GMMs, we have rewritten the probability of observing a datapoint $x^{(i)}$ from a mixture of k Gaussians by utilizing latent variables $r_{i,k}$ in the following way:

$$\mathbb{P}(x^{(i)}) = \sum_{k=1}^{K} \phi_k \mathcal{N}(x^{(i)}; \mu_k, \Sigma_k) \quad \longrightarrow \quad \sum_{r_i} \prod_{k=1}^{K} \left(\phi_k \mathcal{N}(x^{(i)}; \mu_k, \Sigma_k)\right)^{\mathbb{I}[r_{i,k}=1]}.$$

This **new form allows us to explicitly incorporate the latent cluster assignment variables $r_{i,k} \in \{0,1\}$** which will be useful information when solving the corresponding likelihood function as well as the posterior probabilities.

- (Note) The summation over each possible r_i above is used to marginalize over the latent variables r_i in order to obtain the final probability function $\mathbb{P}(x^{(i)})$. To clarify, this summation is over every possible value of r_i, i.e., for all

$$r_i \in \{[1,0,\dots,0],[0,1,\dots,0],\dots,[0,0,\dots,1]\}.$$

Posterior Probabilities of GMMs

Similar to generative ML models, one can utilize Bayes' Theorem to compute the posterior probabilities $\gamma_{r_k} = \mathbb{P}(r_k = 1 | x)$ of any datapoint $x \in \mathbb{R}^n$ being generated from the k-th Gaussian cluster given the model parameters $\theta \coloneqq [\phi_1, \phi_2, \dots, \phi_K, \mu_1, \mu_2, \dots, \mu_K, \Sigma_1, \Sigma_2, \dots, \Sigma_K]$. Further, here we denote $r_k \in \{0,1\}$ for all $k \in \{1,2,\dots,K\}$. The posterior probabilities are formally defined as

$$\gamma_{r_k} = \mathbb{P}(r_k = 1|x) = \frac{\mathbb{P}(r_k = 1, x)}{\mathbb{P}(x)} = \frac{\mathbb{P}(x|r_k = 1)\mathbb{P}(r_k = 1)}{\mathbb{P}(x)}.$$

Notice that **$\mathbb{P}(r_k = 1)$ is the prior probability that x is generated by the k-the Gaussian**. Similarly, **γ_{r_k} denotes the posterior probability that x is generated from the k-th Gaussian after observing the data x**.

Likelihood Function for GMMs

As with most generative ML models, we wish to go about obtaining the best choice of parameters θ by maximizing the corresponding likelihood function.

Likelihood Function for GMM

Assume that the datapoints, given by the set $\mathcal{D} = \{x^{(i)}\}_{i=1}^{m}$, are i.i.d. and sampled from a Gaussian Mixture Model defined by the originally stated probability function

$$\mathbb{P}\left(x^{(i)}\right) = \sum_{k=1}^{K} \phi_k \mathcal{N}\left(x^{(i)}; \mu_k, \Sigma_k\right).$$

Then, one can define the **likelihood** function of observing the data $\{x^{(i)}\}_{i=1}^{m}$, given the set of model parameters $\theta \coloneqq [\phi_1, \phi_2, \dots, \phi_K, \mu_1, \mu_2, \dots, \mu_K, \Sigma_1, \Sigma_2, \dots, \Sigma_K]$, as

$$L\left(\theta; \{x^{(i)}\}_{i=1}^{m}\right) = \mathbb{P}\left(\{x^{(i)}\}_{i=1}^{m}; \theta\right) = \prod_{i=1}^{m} \mathbb{P}\left(x^{(i)}\right) = \prod_{i=1}^{m} \sum_{k=1}^{K} \phi_k \mathcal{N}\left(x^{(i)}; \mu_k, \Sigma_k\right).$$

Further, the **log-likelihood** function is given by

$$\ell\left(\theta; \{x^{(i)}\}_{i=1}^{m}\right) = \sum_{i=1}^{m} \log\left[\sum_{k=1}^{K} \phi_k \mathcal{N}\left(x^{(i)}; \mu_k, \Sigma_k\right)\right].$$

Maximum Likelihood of GMMs

Maximum Likelihood Estimation (MLE) Solutions for a GMM

The optimization problem of maximizing the log-likelihood function for GMMs can be defined by the constrained optimization problem given by

$$\max_{\theta} \quad \ell\left(\theta; \{x^{(i)}\}_{i=1}^{m}\right) = \sum_{i=1}^{m} \log\left[\sum_{k=1}^{K} \phi_k \mathcal{N}\left(x^{(i)}; \mu_k, \Sigma_k\right)\right]$$

$$\text{s.t.} \quad \sum_{k=1}^{K} \phi_k = 1.$$

It can be shown that this problem has the following **parameter solutions** (for some chosen constant posterior probabilities $\gamma_{r_{i,k}}$):

$$\mu_k := \frac{\sum_{i=1}^{m} \gamma_{r_{i,k}} x^{(i)}}{\sum_{i=1}^{m} \gamma_{r_{i,k}}}, \qquad \Sigma_k := \frac{1}{\sum_{i=1}^{m} \gamma_{r_{i,k}}} \sum_{i=1}^{m} \gamma_{r_{i,k}} x^{(i)}, \qquad \phi_k := \frac{\sum_{i=1}^{m} \gamma_{r_{i,k}}}{m}.$$

- Notice that these expressions are fundamentally derived by utilizing the **first-order necessary optimality condition** of the log-likelihood (when the posterior probabilities $\gamma_{r_{i,k}}$ are held constant).
- This can be done **only because the log-likelihood is concave** in the parameters θ when the posterior probabilities $\gamma_{r_{i,k}}$ are held constant.
- If the posteriors are not held constant, then the **overall log-likelihood function is not concave in general**.

The EM Algorithm for Training GMMs

- You may have noticed that there is a problem with the derived parameter solutions that were state on the previous slide. What is the issue?
- The problem is that the solutions depend on the posterior probabilities $\gamma_{r_{i,k}} = \mathbb{P}(r_{i,k} = 1|x)$, which in-turn depend on the values of the parameters themselves…
- This is why the derived solutions on the previous slide are "*for some chosen constant posterior probabilities $\gamma_{r_{i,k}}$*", i.e., when the posterior probabilities $\gamma_{r_{i,k}}$ are held as constant values, then the optimal parameter values (for this particular version of the likelihood) are given by the expressions on the previous slide. However, these are not the true optimal solutions that maximize the whole likelihood function. This is where the EM algorithm comes in.

The Expectation Maximization (EM) Algorithm

The EM algorithm was developed in the 1970s and is utilized for many applications in ML. This overall idea of this algorithm (when applied to maximizing the likelihood of GMMs) consists of two steps: (1) the **expectation step** (E-step) and (2) the **maximization step** (M-step), which can both be described as follows:

- **(E-step):** This step **computes the posterior probabilities** $\gamma_{r_{i,k}} = \mathbb{P}(r_{i,k} = 1|x)$ by holding the current values of the model parameters $\theta \coloneqq [\phi_1, \phi_2, \dots, \phi_K, \mu_1, \mu_2, \dots, \mu_K, \Sigma_1, \Sigma_2, \dots, \Sigma_K]$ as constant.
- **(M-step):** Using the newly computed posterior probabilities $\gamma_{r_{i,k}}$, the algorithm then holds these probability values as constants (not dependent on the parameters θ) and **updates the parameters θ via the maximum likelihood equations** that were stated on the previous slide.

These two steps are then alternated between iteratively until the change in the parameters stops (with some tolerance).

Summarizing K-Means & GMMs

K-Means Overview

- Simple and computationally efficient.
- Assumes spherical clusters. As such, is not suitable for non-spherical shapes.
- Cannot handle overlapping clusters.
- Faster and more scalable to very large datasets.
- No global optimality guarantees. Typically utilizes a multi-start method to obtain better solutions.
- Cluster centroids are either typically (1) assigned to be randomly chosen datapoints or (2) randomly initialized to values within the domain of the dataset.

Gaussian Mixture Model (GMM) Overview

- Very popular way of generating very versatile cluster shapes.
- Flexible in modeling more general oval / elliptical shapes and overlapping clusters.
- Greater representational power via more complexity.
- The added complexity requires more compute power due to additional parameters.
- Less scalable for very large datasets.
- Still no global optimality guarantees. Due to the higher computational cost, multi-start methods can be cost-prohibitive.
- As such, cluster centers are often initialized to be the solution obtained from a K-Means model.

Attribution & License