Soft Clustering with Gaussian Mixture Models

From Hard to Soft Clustering

So far, we have seen how to cluster objects using k-means:

1. Start with an initial set of cluster centers,
2. Assign each object to its closest cluster center
3. Recompute the centers of the new clusters
4. Repeat 2 $\longrightarrow$ 3 until convergence

In k-means clustering every object is assigned to a **single** cluster.

This is called **hard** assignment.

However, there may be cases where we either **cannot** use hard assignments or we do not **want** to do it.

In particular, we might have reason to believe that the best description of the data is a set of **overlapping** clusters.

Overlapping Clusters

Income Level Example

Consider a population in terms of a simple **binary** income level: higher or lower than some threshold.

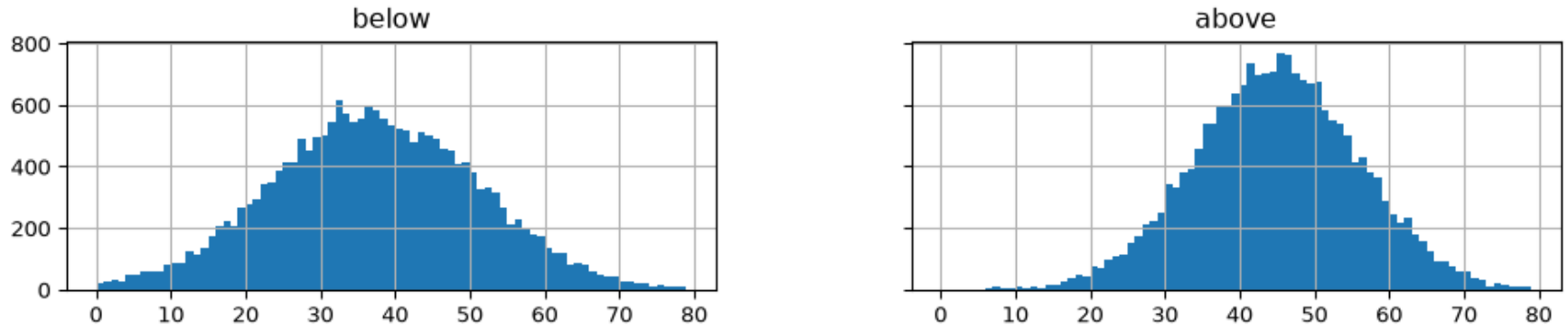
How might we model such a population in terms of the single feature **age**?

Income Level Sampling

Let's sample 20,000 *above income threshold* individuals and 20,000 *below income threshold* individuals.

From this sample we get have the following histograms.

▶ Code

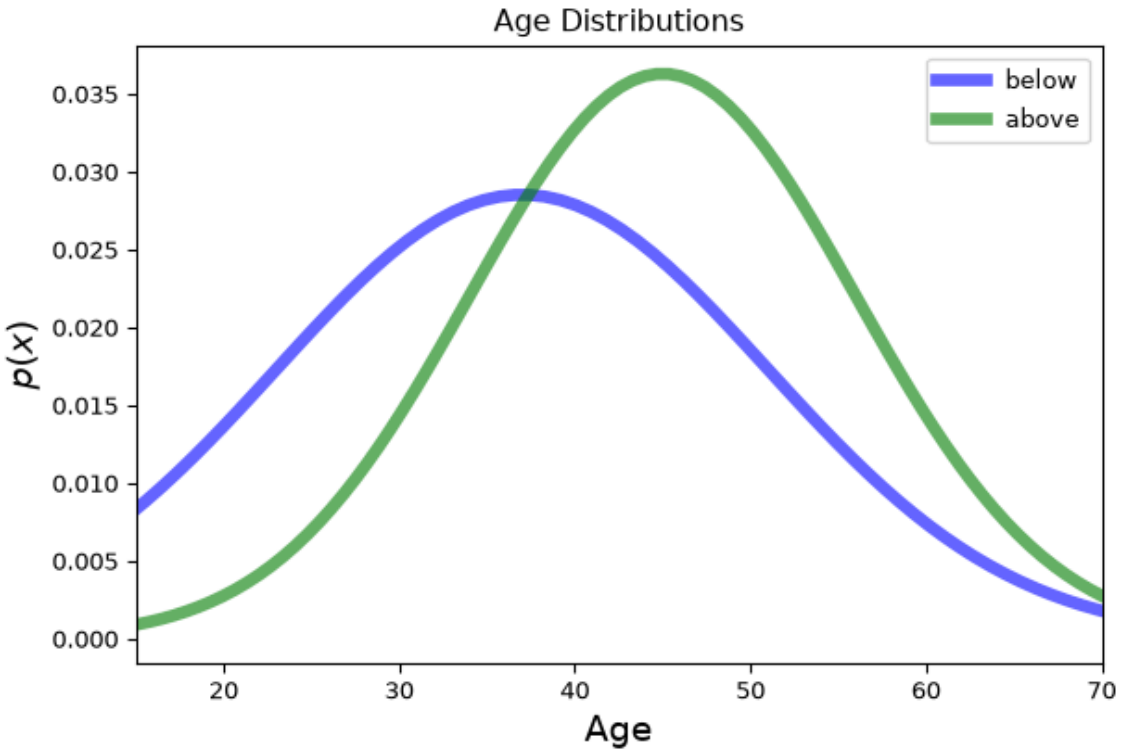


We find that ages of the below income threshold set have mean $\mu = 37$ with standard deviation $\sigma = 14$, while the ages of the above income threshold set have mean $\mu = 45$ with standard deviation $\sigma = 11$.

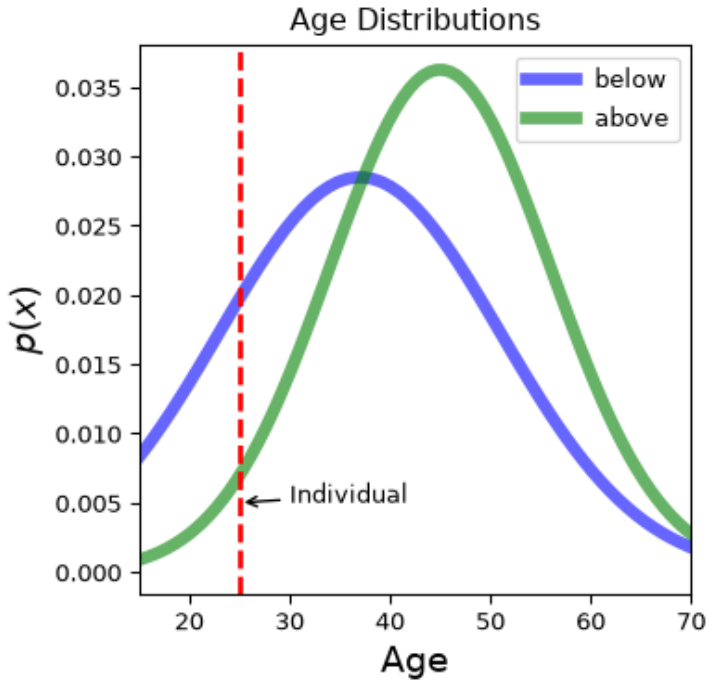
Income Level by Age

We can plot gaussian distributions for the two income levels.

▶ Code



Soft Clustering



Conditional Probability

More formally, we say that an object can belong to each particular cluster with some probability, such that the sum of the probabilities adds up to 1 for each object.

GMM Overview

Our goal with Gaussian mixture models (GMMs) is to show how to compute the probabilities that a data point belongs to a particular cluster.

The main ideas behind GMMs are

- Assume each cluster is normally distributed
- Use the Expectation Maximization (EM) algorithm to iteratively determine:
  - the parameters of the Gaussian clusters using *MLE*, and
  - the probabilities that a data point belongs to a particular cluster

We will see that GMMs are better suited for clustering non-spherical distributions of points.

GMMs build directly on *maximum likelihood estimation* from Probabilistic Modeling, so we start with a brief callback.

Recall: Maximum Likelihood Estimation

MLE in One Slide

Recall from Probabilistic Modeling:

From One Gaussian to a Mixture

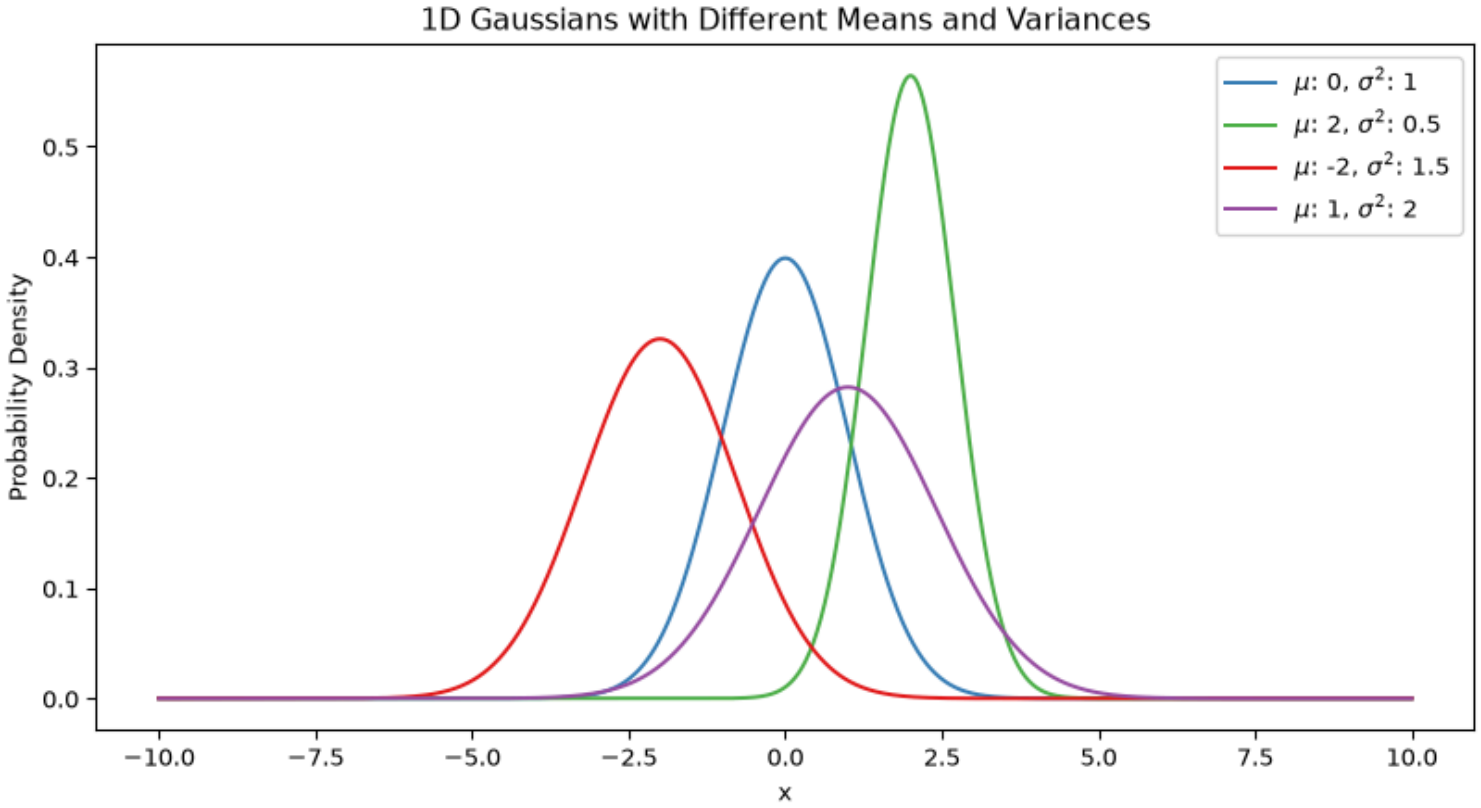
Keep those two formulas in front of you – they reappear inside EM.

Gaussian Mixture Models

Univariate Gaussians

In GMMs we assume that each of our clusters follows a Gaussian (normal) distribution with their own parameters.

▶ Code



Multivariate Gaussians

Given data points $\mathbf{x} \in \mathbb{R}^d$, i.e., d-dimensional points, then we use the multivariate Gaussian to describe the clusters. The formula for the multivariate Gaussian is

$$f(\mathbf{x}|\boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})},$$

where $\Sigma \in \mathbb{R}^{d\times d}$ is the covariance matrix, $|\Sigma|$, denote the determinant of Σ, and $\boldsymbol{\mu}$ is the d-dimensional vector of expected values.

The covariance matrix is the multidimensional analog of the variance. It determines the extent to which vector components are correlated.

See here for a refresher on the covariance matrix.

Notation

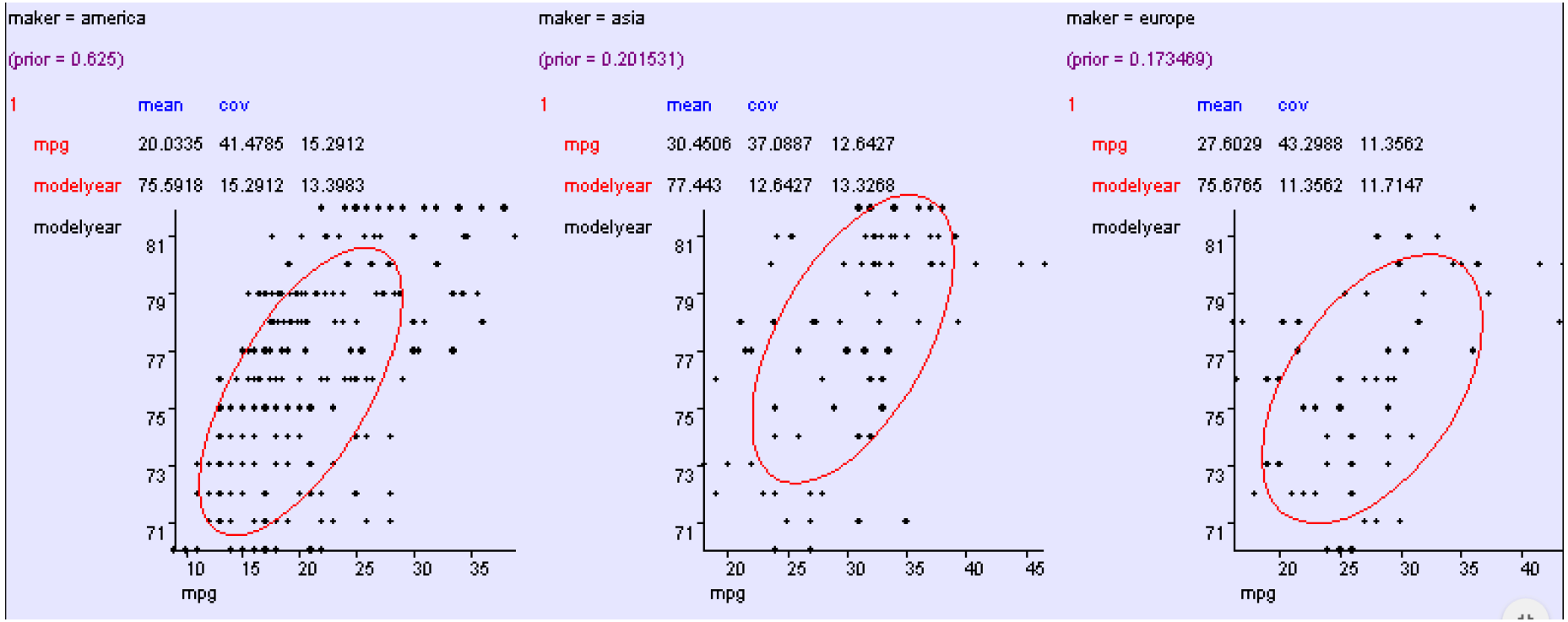
The notation $x \sim \mathcal{N}(\mu, \sigma^2)$ is used to denote a *univariate* random variable that is normally distributed with expected value μ and variance σ^2.

The notation $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ is used to denote a *multivariate* random variable that is normally distributed with mean $\boldsymbol{\mu}$ and covariance matrix Σ.

Multivariate Example: Cars

To illustrate a particular model, let us consider the properties of cars produced in the US, Europe, and Asia.

For example, let's say we are looking to classify cars based on their model year and miles per gallon (mpg).



Expectation-Maximization for Gaussian Mixture Models

The Clustering Problem

Suppose we observe n data points $X = \{x_1, x_2, \dots, x_n\}$ in $\mathbb{R}^d$ that appear to come from multiple clusters.

We want to model this data as a mixture of K Gaussian distributions, each representing a different cluster.

The Gaussian Mixture Model

A Gaussian Mixture Model (GMM) assumes each data point is generated by the following process:

1. Choose a cluster $k \in \{1, 2, \dots, K\}$ with probability π_k (where $\sum_{k=1}^{K} \pi_k = 1$)
2. Generate the point from a Gaussian distribution: $x_i \sim \mathcal{N}(\mu_k, \Sigma_k)$

The probability density of observing point x_i is:

$$p(x_i \mid \theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k)$$

where $\theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^{K}$ are the parameters we need to estimate.

The Likelihood Function

Assuming independent and identically distributed (i.i.d.) data, the likelihood function is:

$$L(X, \theta) = \prod_{i=1}^{n} p(x_i \mid \theta) = \prod_{i=1}^{n} \sum_{k=1}^{K} \pi_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k)$$

The log-likelihood function is:

$$\ell(X, \theta) = \log L(X, \theta) = \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k) \right)$$

Why is This Hard?

Assume $Z = \{z_1, z_2, \dots, z_n\}$ are the cluster assignments for each $i = 1, \dots, n$.

If we knew which cluster generated each point, finding the parameters of the MLE would be straightforward:

- μ_k = mean of points in cluster k
- Σ_k = covariance of points in cluster k
- π_k = fraction of the total points in cluster k

But we don't know the cluster assignments!

And if we knew the parameters, we could infer the cluster assignments.

This is a classic 🐓 and 🥚 problem.

The Log-Likelihood Challenge

The log-likelihood we want to maximize is:

$$\ell(X, \theta) = \log p(X \mid \theta) = \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k) \right)$$

The sum inside the logarithm makes this difficult to optimize directly.

We can’t separate the terms, and taking derivatives leads to a system of coupled equations with no closed-form solution.

EM to the Rescue

EM treats the cluster assignments $Z = \{z_1, z_2, \dots, z_n\}$ as latent (hidden) variables.

Instead of solving for a hard assignment of points to clusters, EM finds the parameters of the MLE by iterating between two steps:

1. **E-Step**: Computes soft assignments (responsibilities) - the probability each point belongs to each cluster given current parameters
2. **M-Step**: Updates parameters assuming these soft assignments are the true (weighted) cluster memberships

By iterating between these steps, EM monotonically increases the likelihood until convergence to a local maximum.

The Complete Data Likelihood

If we knew the cluster assignments $z_i \in \{1, \dots, K\}$ for each point, the complete data log-likelihood would be:

$$\log p(X, Z \mid \theta) = \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}(z_i = k) \left[\log \pi_k + \log \mathcal{N}(x_i \mid \mu_k, \Sigma_k)\right]$$

This is much easier to work with because the log operates on individual Gaussian densities rather than on a sum of densities.

See here for a detailed derivation.

> Tip: Try deriving this yourself!

Why EM Works for GMMs

- **E-Step**: Since we don't know z_i, we compute the expected value of the indicator $\mathbb{1}(z_i = k)$, which is the posterior probability $\gamma_{ik} = p(z_i = k \mid x_i, \theta^{(t)})$, where $\theta^{(t)}$ is the parameter vector at iteration t.
- **M-Step**: With these soft assignments, maximizing the expected complete log-likelihood yields closed-form updates that look like weighted versions of the standard MLE formulas

This elegant approach transforms an intractable optimization problem into a simple iterative procedure with intuitive updates at each step.

See here for a detailed derivation.

> Tip: Try deriving this yourself!

EM Algorithm for Gaussian Mixture Models

Setup and Goal

- We observe data points $x_1, x_2, \ldots, x_n$ in d-dimensional space
- We assume the data comes from K Gaussian distributions, each with mean μ_k, covariance Σ_k, and mixing weight π_k
- The latent variables are the cluster assignments z_i (which component generated each point)
- Goal: Find parameters $\theta = \{\mu_k, \Sigma_k, \pi_k\}_{k=1}^{K}$ that maximize the likelihood of the observed data

Initialization

- Randomly initialize the K means μ_k (or use k-means clustering)
- Initialize covariances Σ_k (often as identity matrices or sample covariance)
- Initialize mixing weights $\pi_k = \frac{1}{K}$ (uniform distribution over components)

E-Step: Compute Responsibilities

- For each data point x_i and each component k, compute the responsibility γ_{ik} (the probability that point i belongs to component k given current parameters)

$$\gamma_{ik} = \frac{\pi_k \cdot \mathcal{N}(x_i \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \cdot \mathcal{N}(x_i \mid \mu_j, \Sigma_j)}$$

- This is a soft assignment: each point gets a fractional membership in each cluster
- Responsibilities sum to 1 for each data point: $\sum_{k=1}^{K} \gamma_{ik} = 1$

M-Step: Update Parameters

- Update means as weighted averages:

$$\mu_k = \frac{\sum_{i=1}^{n} \gamma_{ik} x_i}{\sum_{i=1}^{n} \gamma_{ik}}$$

- Update covariances using weighted samples:

$$\Sigma_k = \frac{\sum_{i=1}^{n} \gamma_{ik} (x_i - \mu_k)(x_i - \mu_k)^T}{\sum_{i=1}^{n} \gamma_{ik}}$$

- Update mixing weights:

$$\pi_k = \frac{\sum_{i=1}^{n} \gamma_{ik}}{n}$$

Iteration and Convergence

- Alternate between E and M steps until convergence
- Monitor the log-likelihood: it’s guaranteed to increase (or stay the same) at each iteration
- Stop when log-likelihood change falls below a threshold or after a maximum number of iterations
- Note: EM finds a local maximum, so results depend on initialization (run multiple times with different initializations)

Key Intuition

- **E-step**: Given current cluster parameters, probabilistically assign points to clusters
- **M-step**: Given these soft assignments, update cluster parameters as if they were the true assignments
- This 🐓 and 🥚 problem is resolved through iteration

Income Level Example

We have

- Our single data point $x = 25$, and
- We have 2 Gaussians representing our clusters,
  - C_1 with parameters $\mu_1 = 37, \sigma_1^2 = 14$ and
  - C_2 with parameters $\mu_2 = 45, \sigma_2^2 = 11$,

Our latent variables are the probabilities that we classify someone as either above or below an income threshold, i.e.,

$$\gamma_1 = P(z = 1) = P(\text{above income threshold})$$

and

$$\gamma_2 = P(z = 2) = P(\text{below income threshold}).$$

Income Level Example, cont.

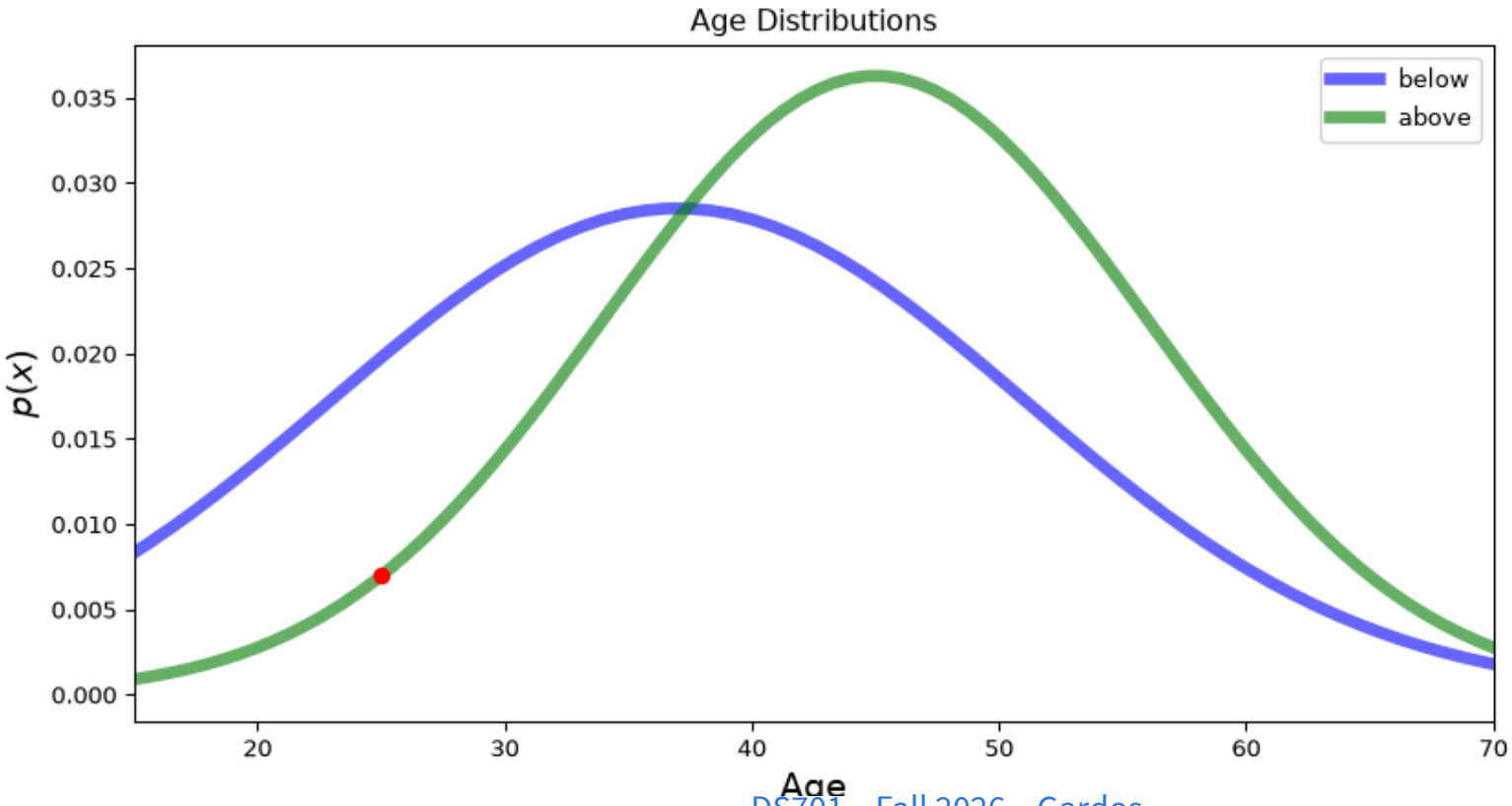
Our interest in this problem is computing the posterior probability, which is,

$$P(C_1|x) = P(z = 1|x) = P(\text{above income threshold}|\text{age } 25).$$

Income Level Example, cont.

If we know the parameters of the Gaussians, we can determine the value of $P(x|C_1)$ when $x = 25$. This is shown with the red dot.

▶ Code



Bayes' Rule then allows us to compute

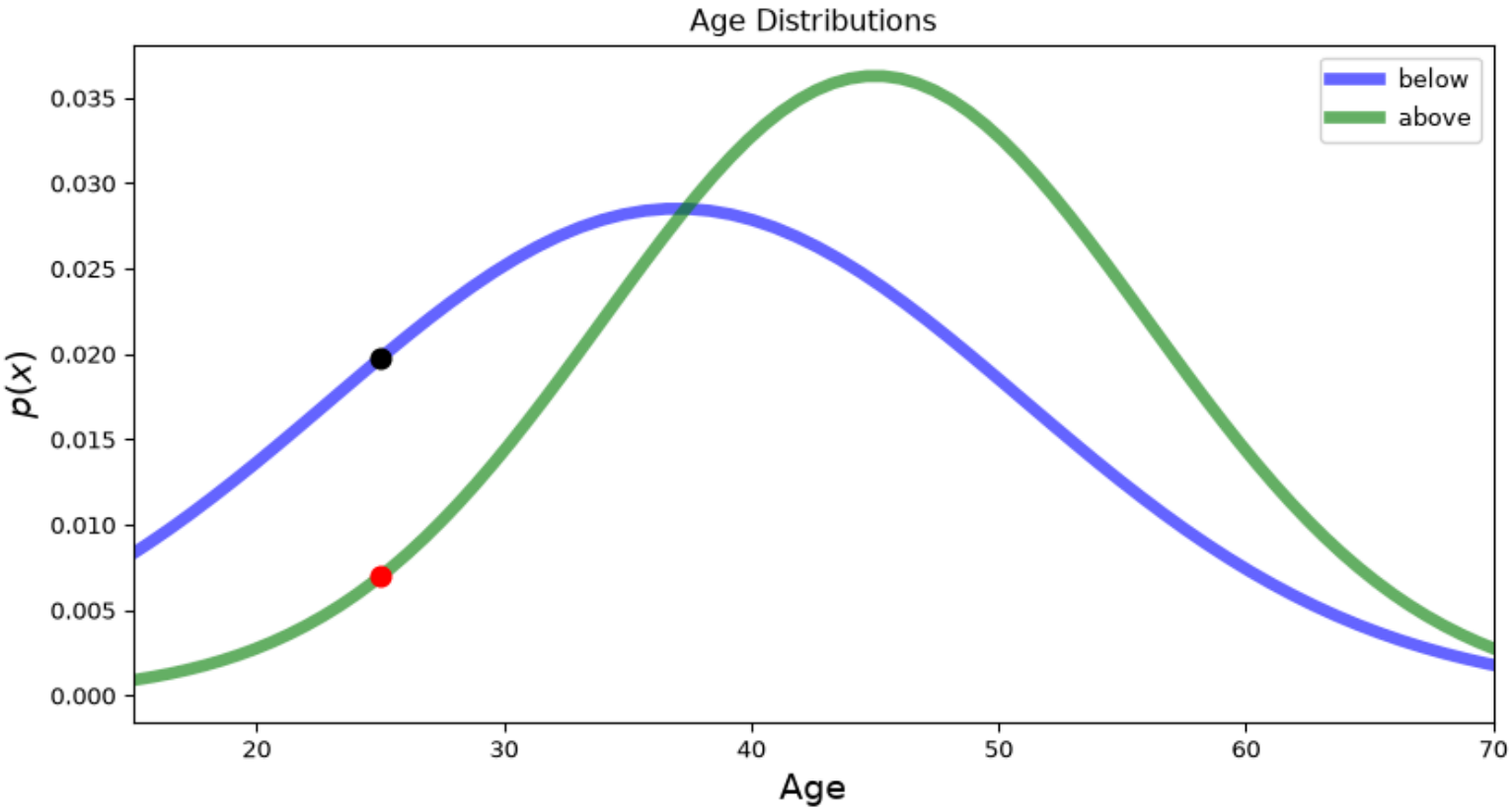
$$P(C_1|x) = \frac{P(x|C_1)}{P(x)} P(C_1).$$

We can always use the law of total probability to compute

$$\begin{aligned} P(x) &= P(x|C_1)P(C_1) + P(x|C_2)P(C_2), \\ &= P(z=1)P(x|z=1) + P(z=2)P(x|z=2), \\ &= \gamma_1 P(x|z=1) + \gamma_2 P(x|z=2), \\ &= \sum_{i=1}^{2} \gamma_i \cdot f(\mathbf{x}|\mu_i, \sigma_i). \end{aligned}$$

The final formula is illustrated in the following figure.

▶ Code



$$P(\text{above} \,|\, \text{age } 25) = \frac{\text{red}}{\text{red} \cdot P(\text{above}) + \text{black} \cdot P(\text{below})} \cdot P(\text{above}).$$

Convergence criteria

The convergence criteria for the Expectation-Maximization (EM) algorithm assess the *change across iterations* of either the

- likelihood function, or
- model parameters

with usually a *limit on the maximum number of iterations*.

Here are the common convergence criteria used:

Log-Likelihood Convergence

$$|\mathcal{L}^{(t)} - \mathcal{L}^{(t-1)}| < \text{tol}$$

Where:

- $\mathcal{L}^{(t)}$ is the log-likelihood at iteration t,
- tol is a small positive number (e.g., 10^{-4}).

Parameter Convergence

$$||\theta^{(t)} - \theta^{(t-1)}||_2 < \text{tol}$$

Where:

- $\theta^{(t)}$ represents the model parameters (means, covariances, and weights) at iteration t,
- $||\cdot||_2$ is the Euclidean (L2) norm,
- tol is a small positive number.

Maximum Number of Iterations

The EM algorithm is typically capped at a maximum number of iterations to avoid long runtimes in cases where the log-likelihood or parameters converge very slowly or never fully stabilize.

$$t > \text{max\_iterations}$$

Where:

- max_iterations is a predefined limit (e.g., 100 or 500 iterations).

Storage and Computational Costs

It is important to be aware of the computational costs of our algorithm.

Storage costs:

- There are N d-dimensional points
- There are K clusters
- There are $N \times K$ coefficients r_{nk}
- There are K d-dimensional cluster centers $\boldsymbol{\mu}_k$
- There are K $d \times d$ covariance matrices Σ_k
- There are K weights w_k

Computational costs:

- Computing each r_{nk} requires a sum of K evaluations of the Gaussian PDF.
- Updating $\boldsymbol{\mu}_k$ requires N vector summations

k-means vs GMMs

Let's pause for a minute and compare GMM/EM with k-means.

GMM/EM

1. Initialize randomly or using some rule
2. Compute the probability that each point belongs in each cluster
3. Update the clusters (weights, means, and variances).
4. Repeat 2-3 until convergence.

k-means

1. Initialize randomly or using some rule
2. Assign each point to a single cluster
3. Update the clusters (means).
4. Repeat 2-3 until convergence.

From a practical standpoint, the main difference is that in GMMs, data points do not belong to a **single** cluster, but have some probability of belonging to **each** cluster.

In other words, as stated previously, GMMs use soft assignment.

For that reason, GMMs are also sometimes called **soft k-means.**

However, there is also an important conceptual difference.

The GMM starts by making an **explicit assumption** about how the data were generated.

It says: “the data came from a collection of multivariate Gaussians.”

We made no such assumption when we came up with the k-means problem. In that case, we simply defined an objective function and declared that it was a good one.

Nonetheless, it appears that we were making a sort of Gaussian assumption when we formulated the k-means objective function. However, **it wasn’t explicitly stated.**

The point is that because the GMM makes its assumptions explicit, we can

- examine them and think about whether they are valid, and
- replace them with different assumptions if we wish.

For example, it is perfectly valid to replace the Gaussian assumption with some other probability distribution. As long as we can estimate the parameters of such distributions from data (e.g., have MLEs), we can use EM in that case as well.

Versatility of EM

A final statement about EM generally. EM is a versatile algorithm that can be used in many other settings. What is the main idea behind it?

Notice that the problem definition only required that we find the clusters, C_i, meaning that we were to find the (μ_i, Σ_i).

However, the EM algorithm posited that we should find as well the $P(C_j|x_i) = P(z = j|x_i)$, that is, the probability that each point is a member of each cluster.

This is the true heart of what EM does.

By **adding parameters** to the problem, it actually finds a way to make the problem solvable.

These are the latent parameters we introduced earlier. Latent parameters don't show up in the solution.

Examples

Example 1

Here is an example using **GMM**.

We're going to create two clusters, one spherical, and one highly skewed.

▶ Code

```
The covariance matrix of our skewed cluster will be:
 [[2.9  0.67]
 [0.67 0.17]]
```

▶ Code

▶ Code

▶ Code

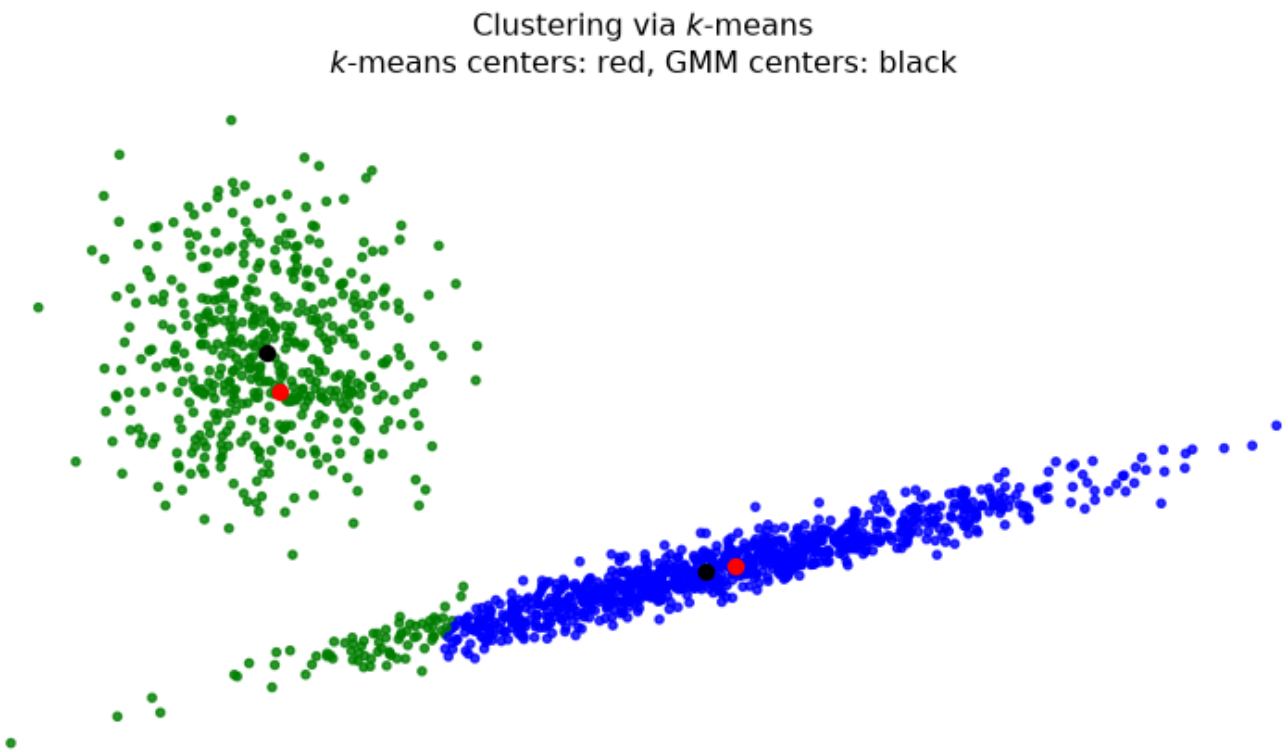
▶ Code

▶ Code

```
Cluster 0:
 weight: 0.667
 mean: [-0.04719455 -0.00741777]
```

Comparison with k-means

▶ Code



▶ Code

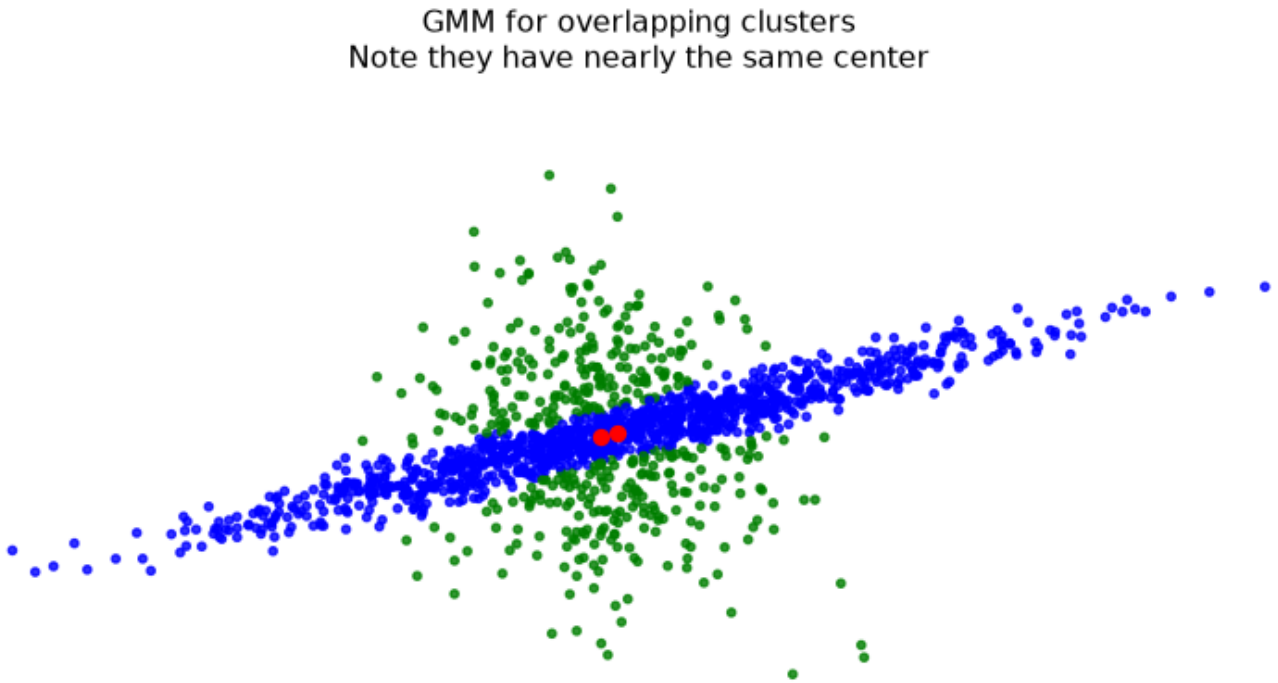
```
Cluster 0:
```

Overlapping Clusters

Now, let's construct **overlapping** clusters. What will happen?

▶ Code

▶ Code



How many parameters are estimated?

Most of the parameters in the model are contained in the covariance matrices.

In the most general case, for K clusters of points in d dimensions, there are K covariance matrices each of size $d \times d$.

So we need Kd^2 parameters to specify this model.

It can happen that you may not have enough data to estimate so many parameters.

Also, it can happen that you believe that clusters should have some constraints on their shapes.

Here is where the GMM assumptions become **really** useful.

Clusters with Equal Variance

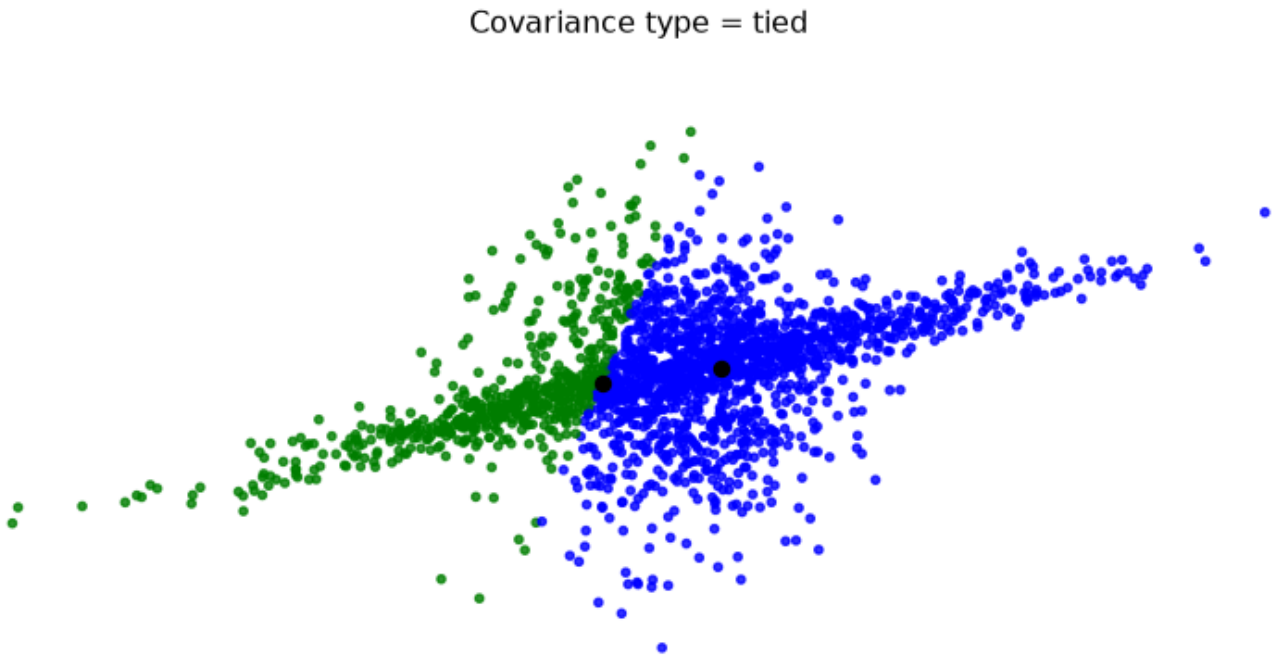
Let's say you believe all the clusters should have the same shape, but the shape can be arbitrary.

Then you only need to estimate **one** covariance matrix - just d^2 parameters.

This is specified by the GMM parameter `covariance_type='tied'`.

▶ Code

▶ Code



Non-Skewed Clusters

Perhaps you believe in even more restricted shapes: all clusters should have their axes aligned with the coordinate axes.

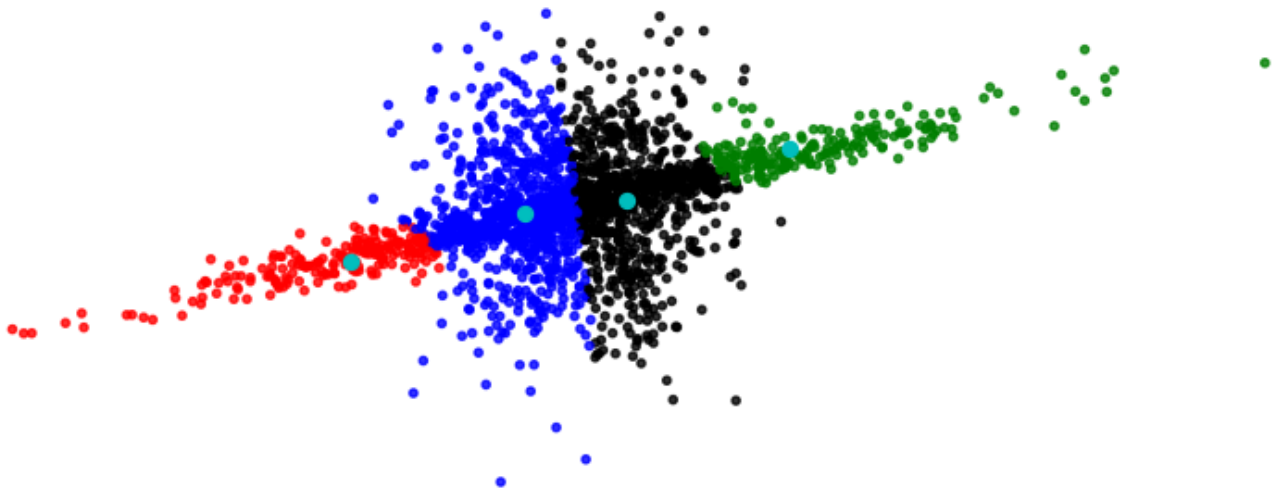
That is, clusters are not skewed.

Then you only need to estimate the diagonals of the covariance matrices - just Kd parameters.

This is specified by the GMM parameter `covariance_type='diag'`.

▶ Code

▶ Code



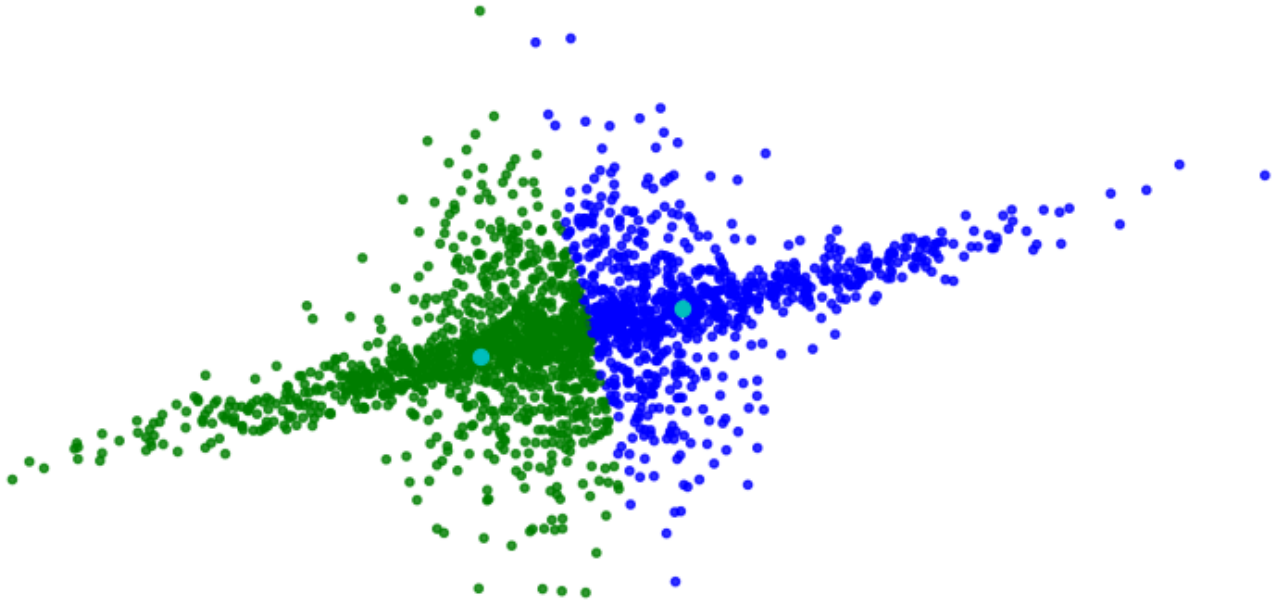
Round Clusters

Finally, if you believe that all clusters should be round, then you only need to estimate the K variances.

This is specified by the GMM parameter `covariance_type='spherical'`.

▶ Code

▶ Code



Summary

Today we covered:

- Maximum likelihood estimators
- Gaussian mixture models
- Expectation Maximization

A major benefit of GMMs is they are soft clustering technique that allows you to capture non-spherical clusters.