# Gaussian Mixture Model (GMM)

Generative probabilistic model describing the distribution of the data.

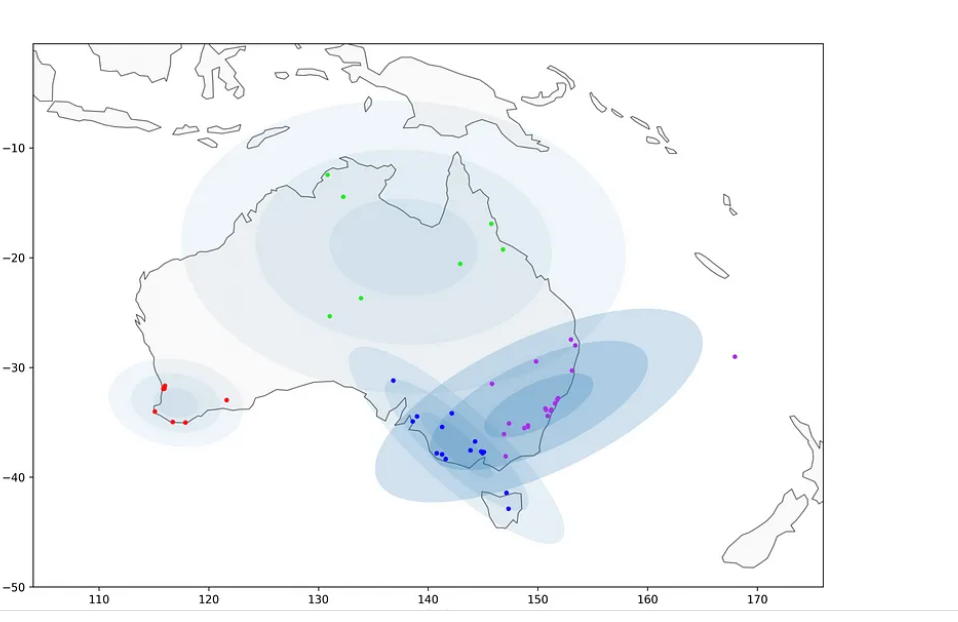
# A generative model is inherently a probability distribution for the dataset. Probably the most known and used algorithm for clustering is K-Means. But it has its limitations. One of them is that the clusters will be separated based on an optimal radius value from the cluster center, calculated based on the Euclidean distance to the point. Ergo, if your cluster is not defined as a circular shape, you can find trouble to separate it properly. GMM, on the other hand, works with other formats, being the elliptical shape the most common.

As you might have figured, Gaussian Mixture Models assume that your data follows Gaussian (a.k.a. Normal) distribution. Since there can be multiple such distributions within your data, you get to specify their number, which is essentially the number of clusters that you want to have.

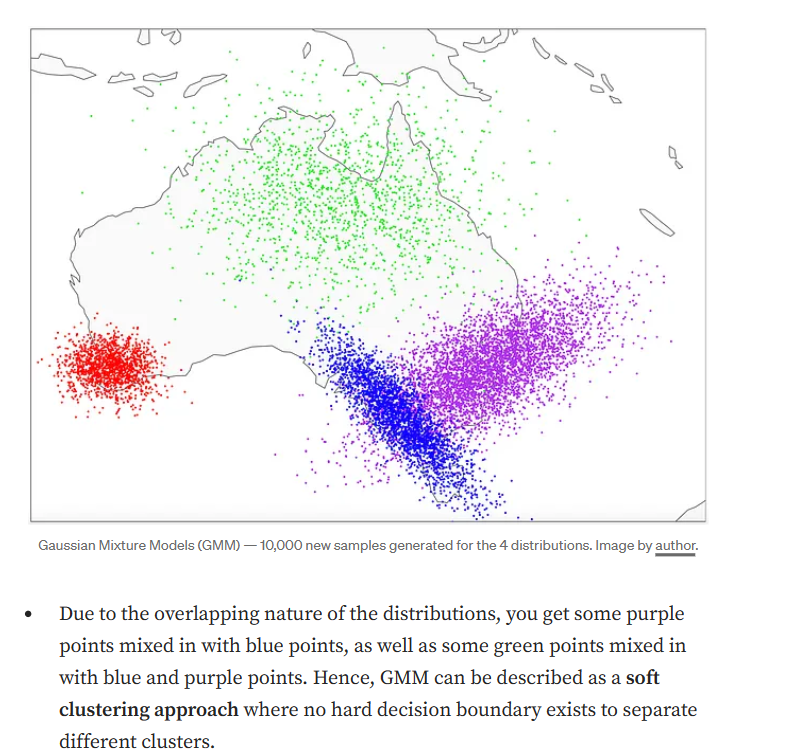
Also, since separate distributions can overlap, the model output is not a hard assignment of points to specific clusters. It is based on a **probability**that the point belongs to a said distribution. Say, if point A has a probability of 0.6 belonging to “Cluster 0” and a probability of 0.4 belonging to “Cluster 1,” then the model would recommend “Cluster 0” to be the label for that point (since 0.6>0.4).

To aid the explanation further, let’s look at a few graphs.

* The below image shows 4 clusters of Australian cities identified by GMM. Note how each cluster has its own mean (center), covariance (shape), and size. Also, there is a notable overlap between different clusters (purple, blue, and green).

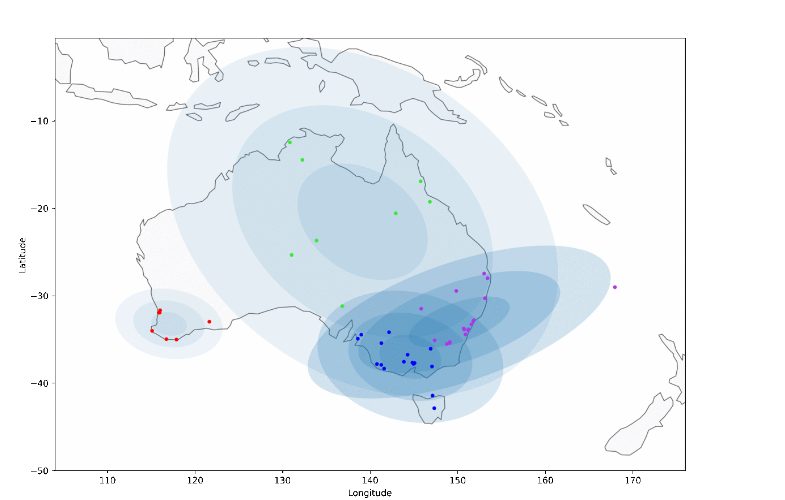


* As mentioned earlier, the cluster label assignment was based on picking the highest probability of a specific data point belonging to a specific cluster. However, that does not mean that the point is definitely part of that cluster (distribution).
* See what happens when we ask the model to generate new data points (samples) for the distributions it has found above:

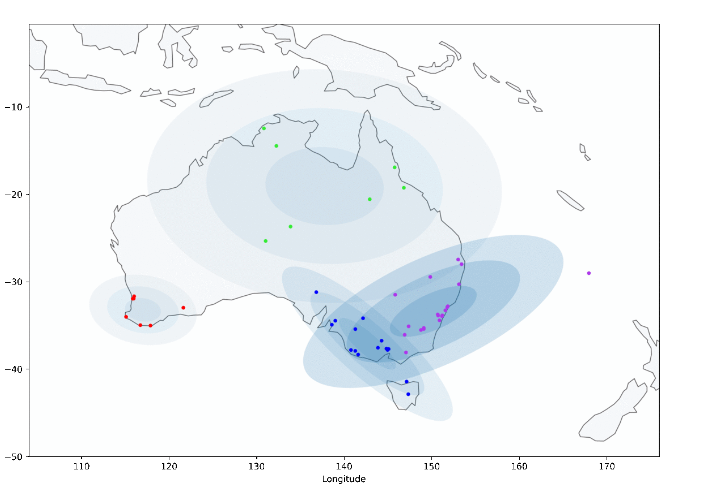


## **Expectation-Maximization (EM)**

To understand how GMM works in practice, we need to look at the Expectation-Maximization (EM) algorithm. The EM uses an iterative method to calculate and recalculate the parameters of each cluster (distribution), i.e., mean, variance/covariance, and size.



After multiple iterations the clusters change their mean, variance/covariance, and size to get better separated clusters:



At the outset, the model initializes a specified number of clusters with a set of parameters that can either be random or specified by the user. Smart initialization options are also available in some implementations:

[sklearn.mixture.GaussianMixture — scikit-learn 1.3.1 documentation](https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html#sklearn.mixture.GaussianMixture.sample)

GMM attributes:

For each cluster:

gmm.means\_: Location

, gmm.covars\_: Shape

, gmm.weights\_: Size.

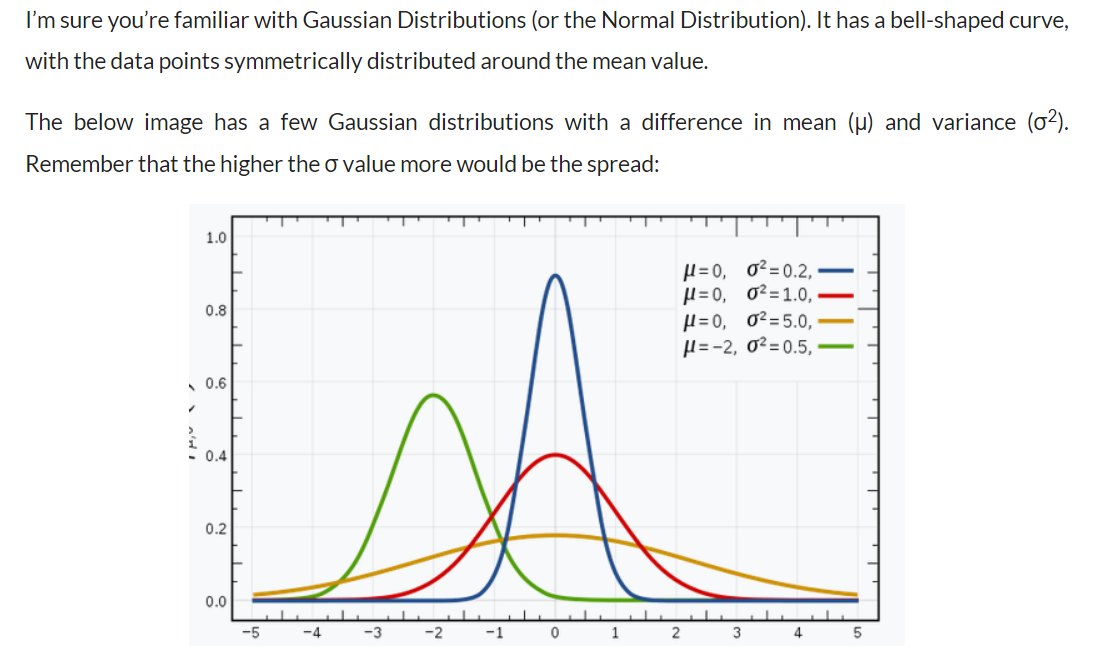
For the above graph, I have specified my own set of mean values (starting centers) to initialize clusters, which helped me to create a nicer visualization. It has also sped up the convergence when comparted to random initialization.

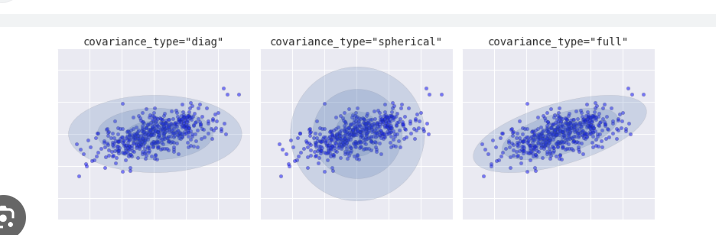
However, you have to be very careful with initialization, because GMM’s final result tends to be quite sensitive to the initial starting parameters. Hence, it is recommended to either use smart initialization or to randomly initialize many times and then pick the best result.

So, with clusters initialized, we have the mean (μ), covariance (Cov), and size (𝜋) available to use.

* **Expectation (E-step)** — for each data point, a “responsibility” **r** is calculated, which is, in simple terms, a probability of that data point belonging to a cluster **c**. This is done for each point with regard to each cluster.
* **Maximization (M-step)**— then “responsibilities” are used to recalculate the mean, covariance, and size of each cluster (distribution), At this step, you can also think of “responsibility” as a weight. The less likely it is that the data point belongs to a cluster, the smaller the weight it will carry in the recalculation of μ, Cov, and 𝜋. In the image above, you can see how the position, shape, and size of the clusters change with each iteration.

The process of E-step and M-step is repeated many times until no further improvements can be made, i.e., convergence is achieved.

**

****

# Full will be considered the elliptical shape, and is the default in the covariance\_type that defines the shape.

# The GaussianMixture comes with different options to constrain the covariance of the difference classes estimated: spherical, diagonal, tied or full covariance.

### **What are the differences between Gaussian mixture models and other types of clustering algorithms such as K-means?**

Here are some of the key differences between Gaussian mixture models and the K-means algorithm used for clustering:

* A Gaussian mixture model is a type of clustering algorithm that assumes that the data point is generated from a mixture of Gaussian distributions with unknown parameters. The goal of the algorithm is to estimate the parameters of the Gaussian distributions, as well as the proportion of data points that come from each distribution. In contrast, K-means is a clustering algorithm that does not make any assumptions about the underlying distribution of the data points. Instead, it simply partitions the data points into K clusters, where each cluster is defined by its centroid.
* While Gaussian mixture models are more flexible, they can be more difficult to train than K-means. K-means is typically faster to converge and so may be preferred in cases where the runtime is an important consideration.
* In general, K-means will be faster and more accurate when the data set is large and the clusters are well-separated. Gaussian mixture models will be more accurate when the data set is small or the clusters are not well-separated.
* Gaussian mixture models take into account the variance of the data, whereas K-means does not.
* Gaussian mixture models are more flexible in terms of the shape of the clusters, whereas K-means is limited to spherical clusters.
* Gaussian mixture models can handle missing data, whereas K-means cannot. This difference can make Gaussian mixture models more effective in certain applications, such as data with a lot of noise or data that is not well-defined.

Just as in the k-means expectation–maximization approach, to change the parameters of each distribution until the probabilities are optimize. The same with k-means it is tunning the centroids until the Euclidean distances are optimize.

### **Choosing the covariance type**

If you look at the details of the preceding fits, you will see that the covariance\_type option was set differently within each. This hyperparameter controls the degrees of freedom in the shape of each cluster; it is essential to set this carefully for any given problem. The default is covariance\_type="full", which means that the size of the cluster along each dimension can be set independently, with the resulting ellipse constrained to align with the axes. Ellipses parallel to the axes.

A slightly simpler and faster model is covariance\_type="spherical", which constrains the shape of the cluster such that all dimensions are equal. The resulting clustering will have similar characteristics to that of k-means, though it is not entirely equivalent. A more complicated and computationally expensive model (especially as the number of dimensions grows) is to use covariance\_type=" diag", which allows each cluster to be modeled as an ellipse with arbitrary orientation. Any orientation.