# Module: Adv. Machine Learning

## Live Session-5

## Statistical clustering
## (Gaussian Mixture Model)

# Clustering:

- Data are often given as points (or vectors) $x^n$ in a Euclidean vector space and often form groups that are close to each other, so called clusters.

- In data analysis one is, of course, interested to discover such a structure, a process called clustering.

# K-mean clustering

1. First we initialize k points, called means, randomly.
2. We categorize each item to its closest mean and we update the mean's coordinates, which are the averages of the items categorized in that mean so far.
3. We repeat the process for a given number of iterations and at the end, we have our clusters.

The idea is to represent each cluster $k$ by a center point $c_k$ and assign each data point $x_n$ to one of the clusters $k$, which can be written in terms of index sets $C_k$.

The center points and the assignment are then chosen such that the distance between data points and center points

$$E := \sum_{k=1}^{K} \sum_{n \in \mathcal{C}_k} \|\mathbf{x}_n - \mathbf{c}_k\|^2$$

is minimized.

# K Mean Algorithm

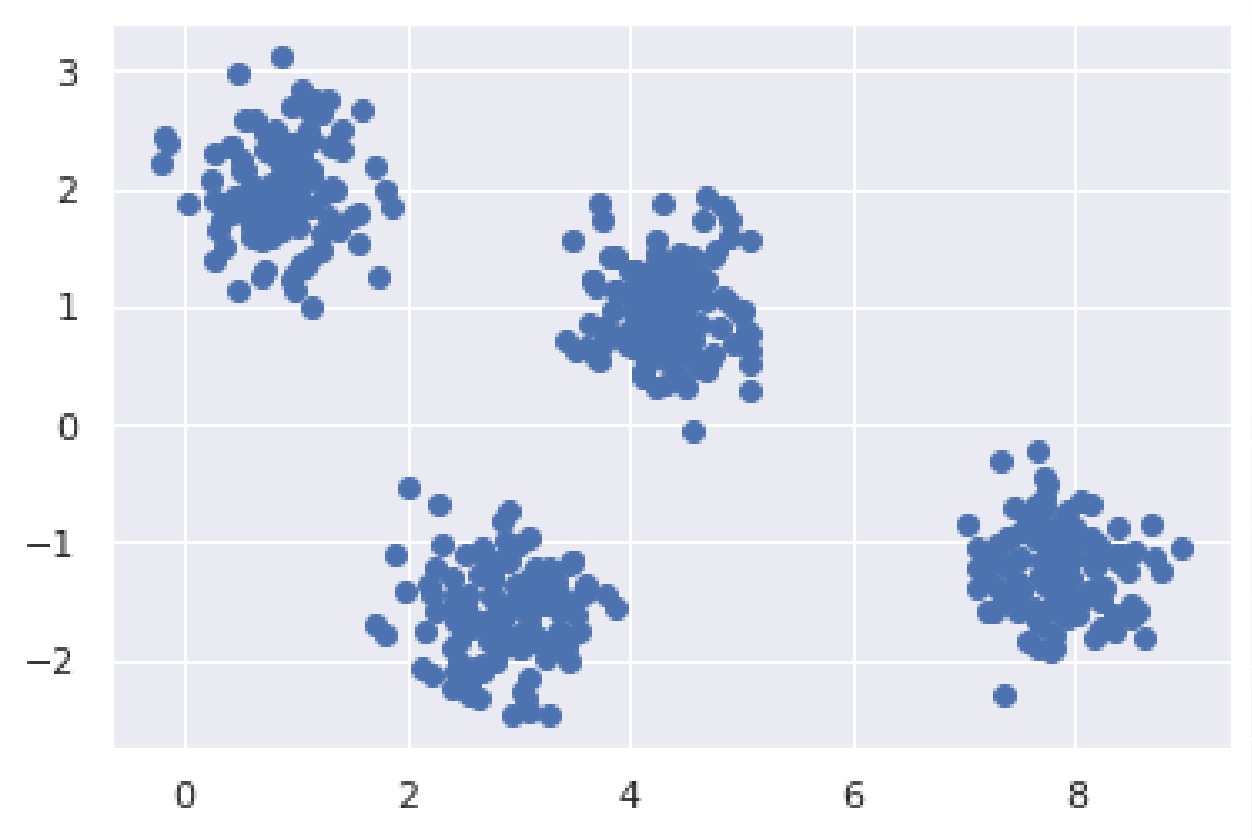
The K-means algorithm now consists of applying these two optimizations in turn until convergence.

The initial center locations could be chosen randomly from the data points. A drawback of this and many other clustering algorithms is that the number of clusters is not determined.

One has to decide on a proper K in advance, or one simply runs the algorithm with several different K-values and picks the best according to some criterion.

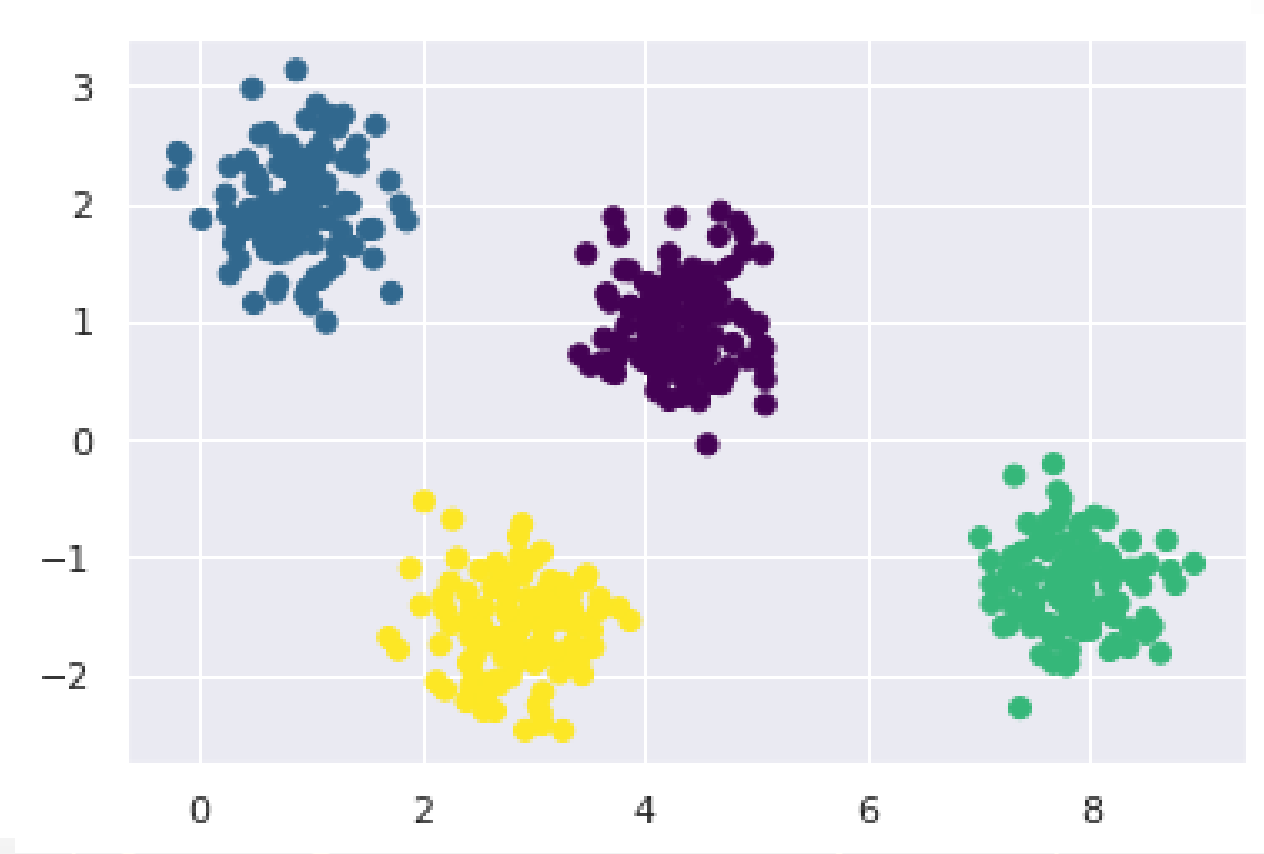
# Why GMM:

Given simple, well-separated data, *k*-means gives good clustering results.



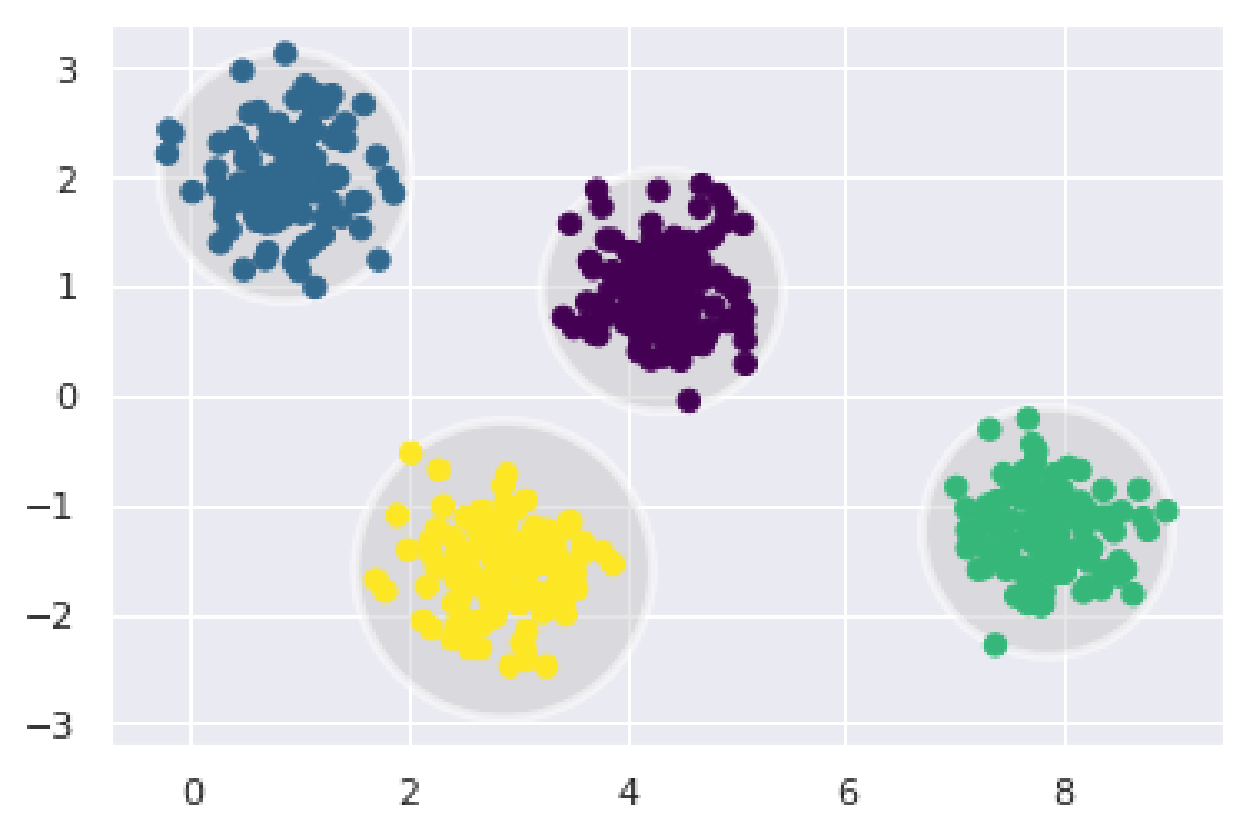
# Why GMM:

Given simple, well-separated data, *k*-means gives good clustering results.



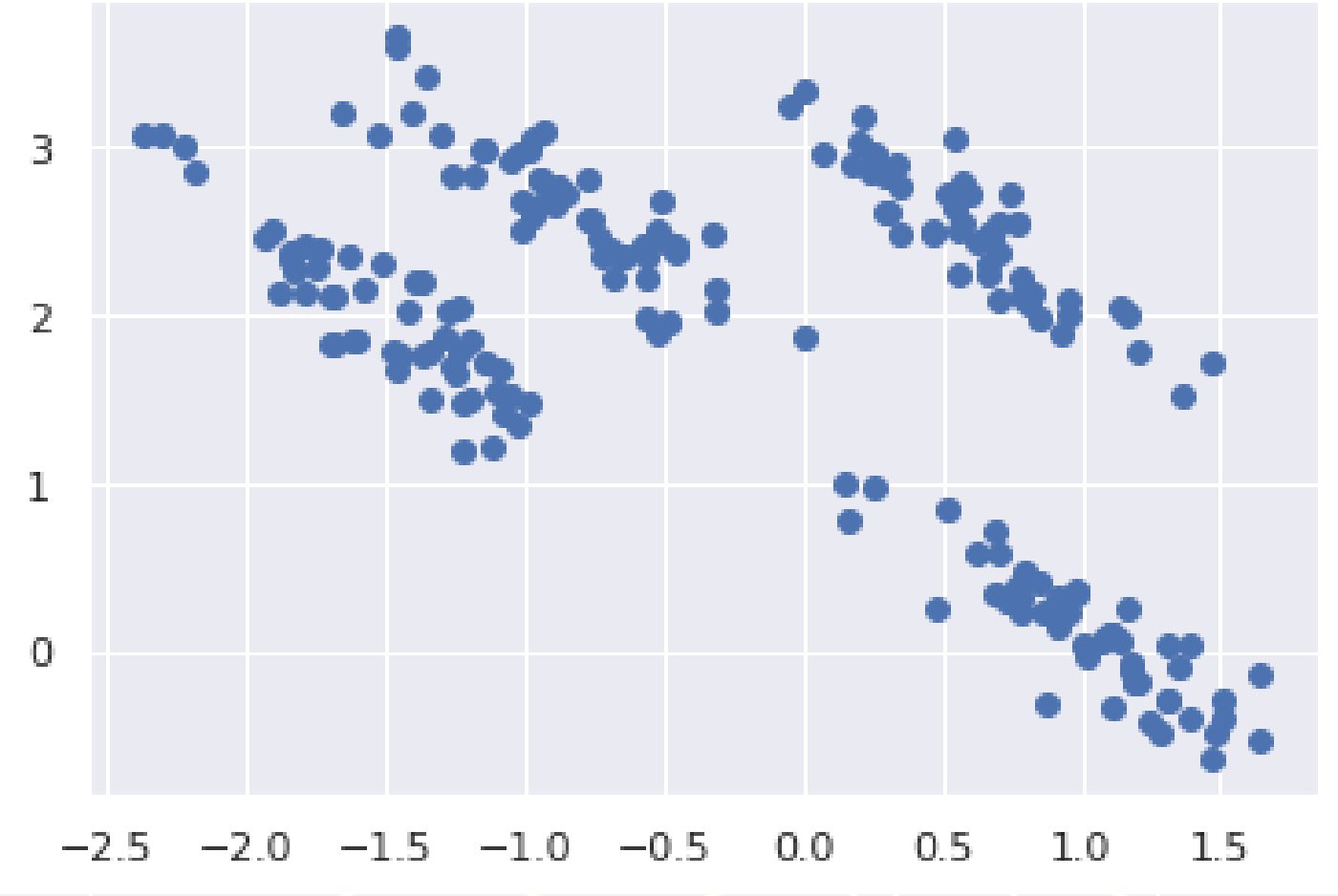
# Why GMM:

Given simple, well-separated data, *k*-means gives good clustering results.



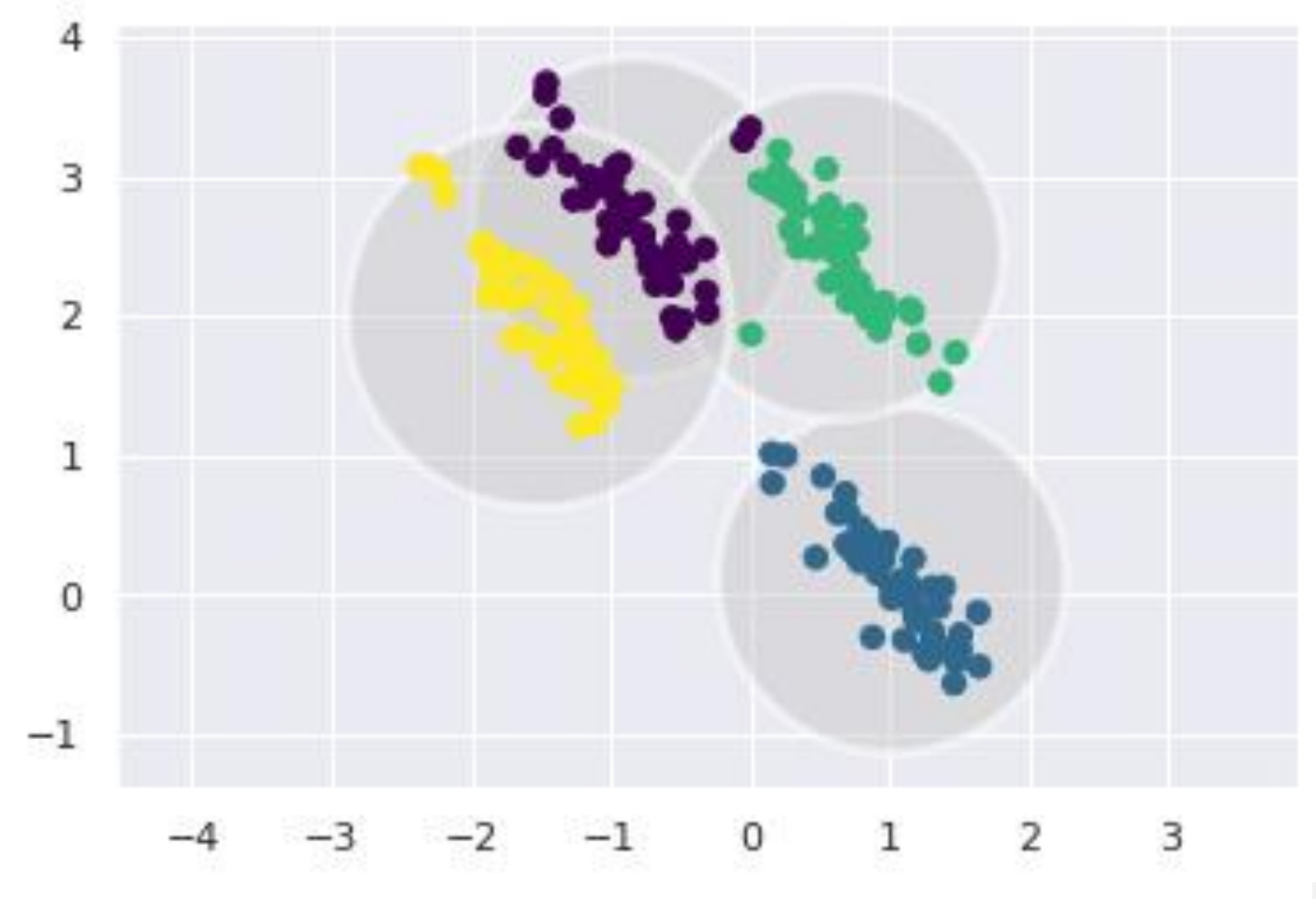
# Why GMM:

Consider the following data:



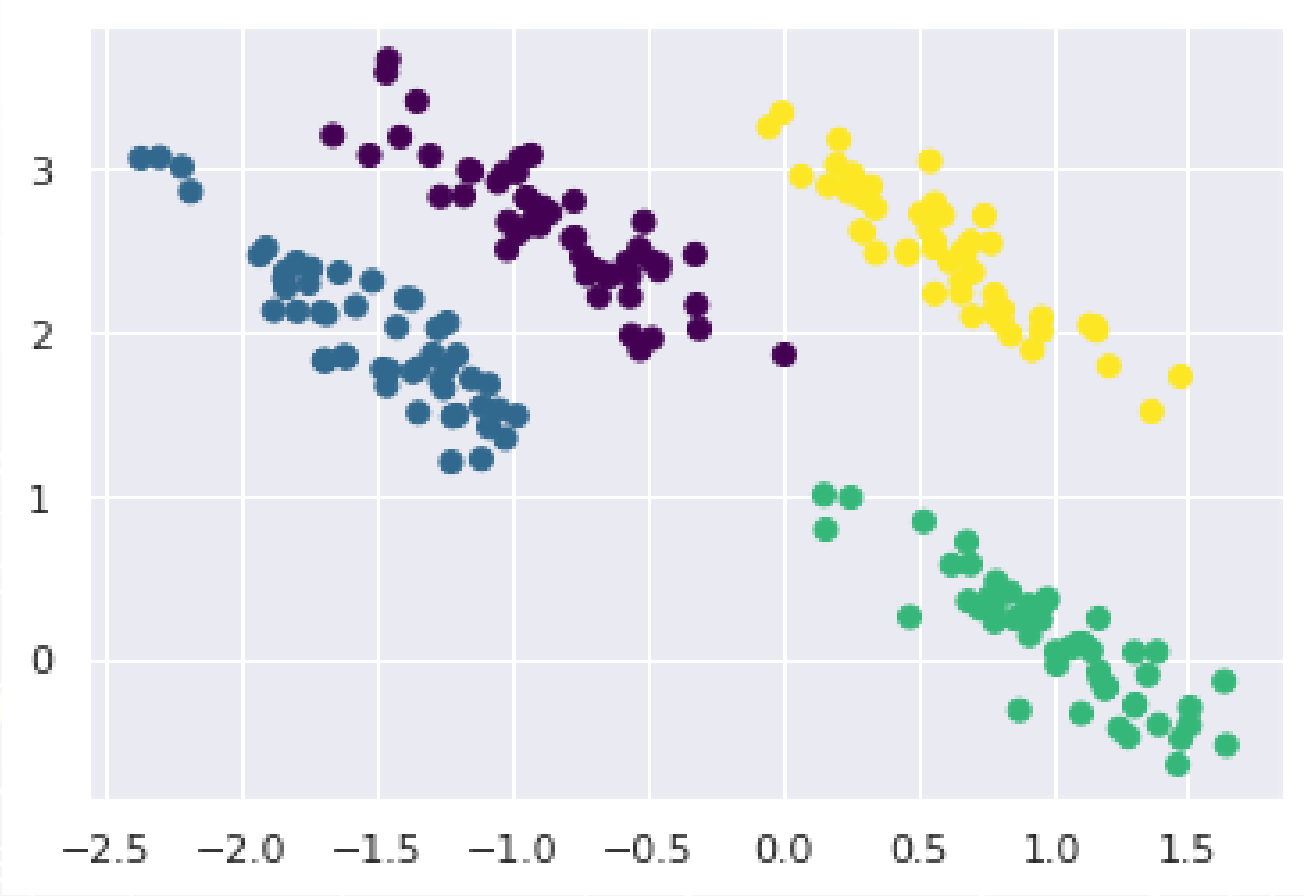
# Why GMM:

K-Means Results:

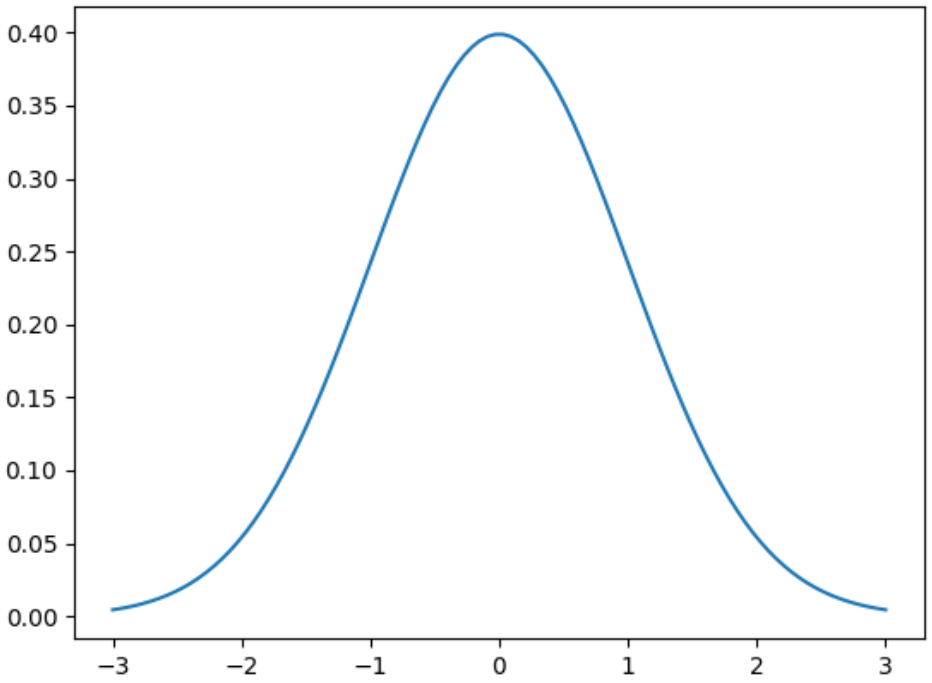


# Why GMM:

Solution:



# Gaussian Distribution Model:



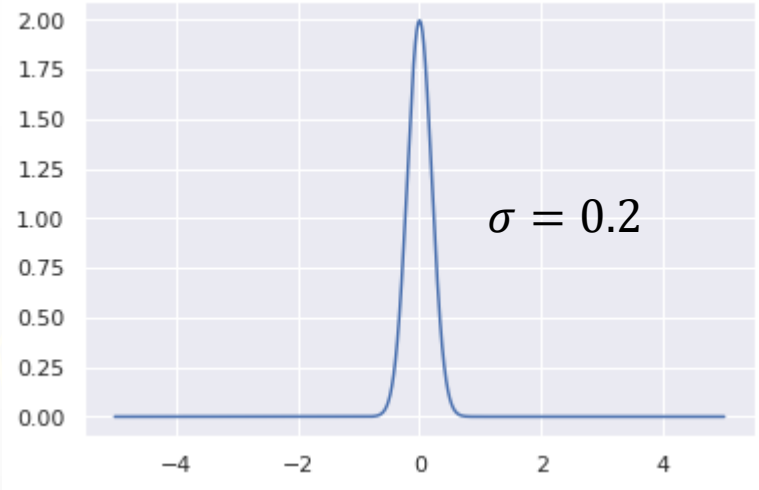

$$N(x;|\,\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}\,\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

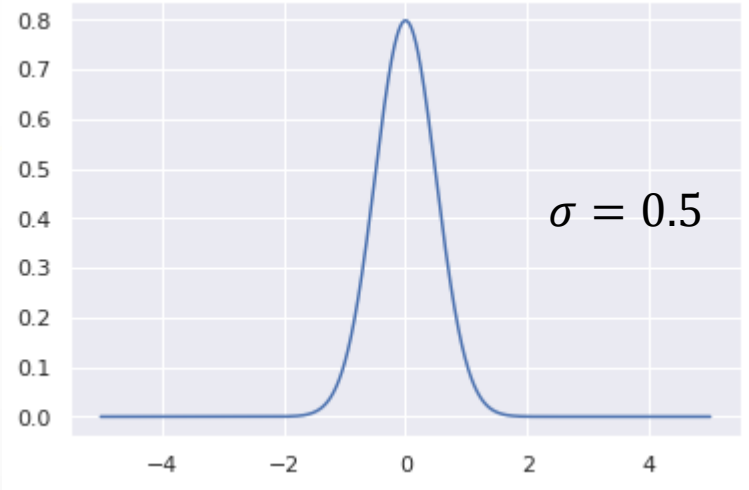
# Gaussian Distribution: Two Parameters

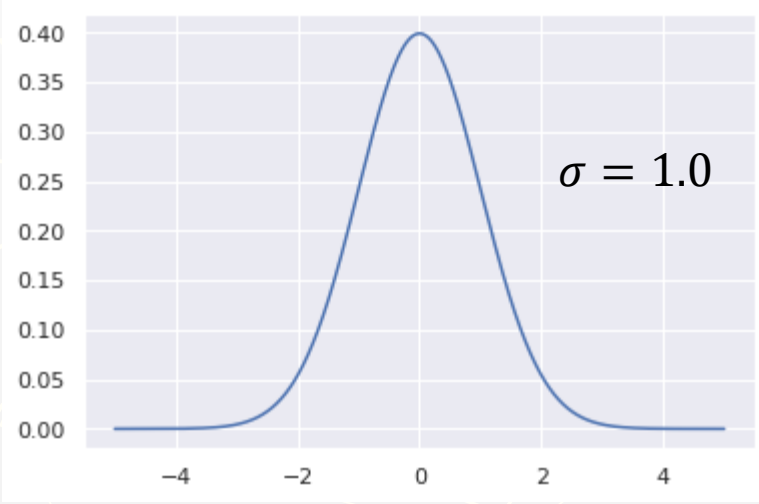
The mean of the Gaussian simply shifts the center of the Gaussian, i.e., the "bump" or top of the bell. In the image above, $\mu$=0, so the largest value is at $x$=0.

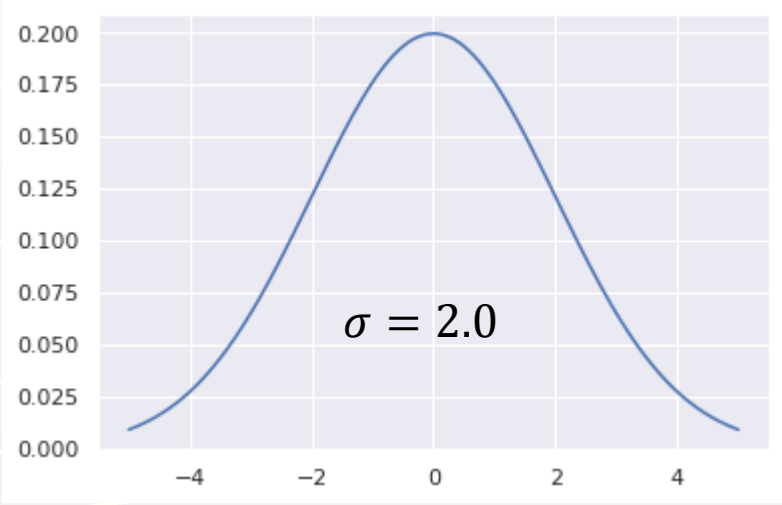
The standard deviation is a measure of the *spread* of the Gaussian. It affects the "wideness" of the bell. Using a larger standard deviation means that the data are more spread out, rather than closer to the mean.

# Gaussian Distribution: Two Parameters



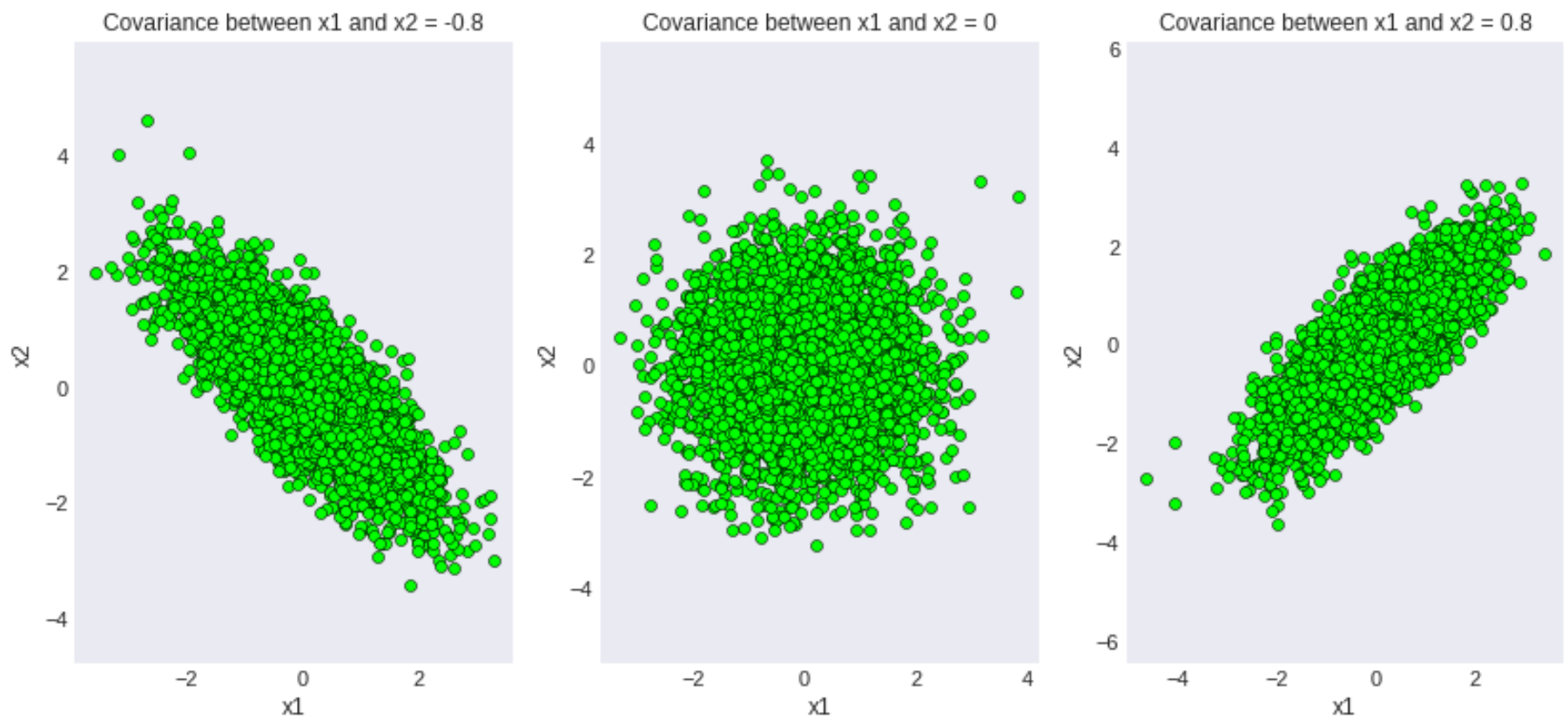



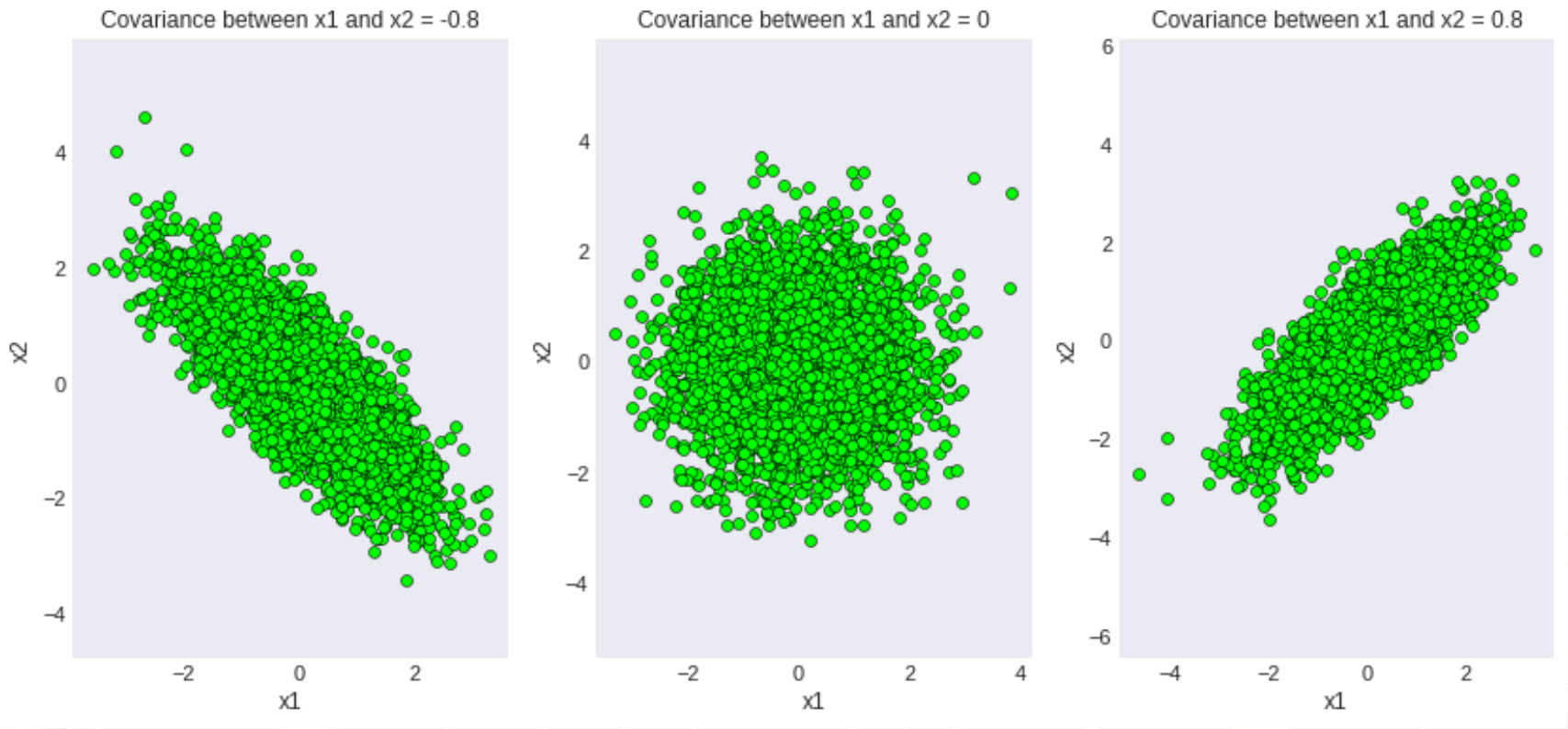



# Bivariate Gaussian Distribution:

$$G(X|\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)|\mathbf{\Sigma}|}} \exp\left(-\frac{1}{2}(X-\mu)^T\mathbf{\Sigma}^{-1}(X-\mu)\right)$$

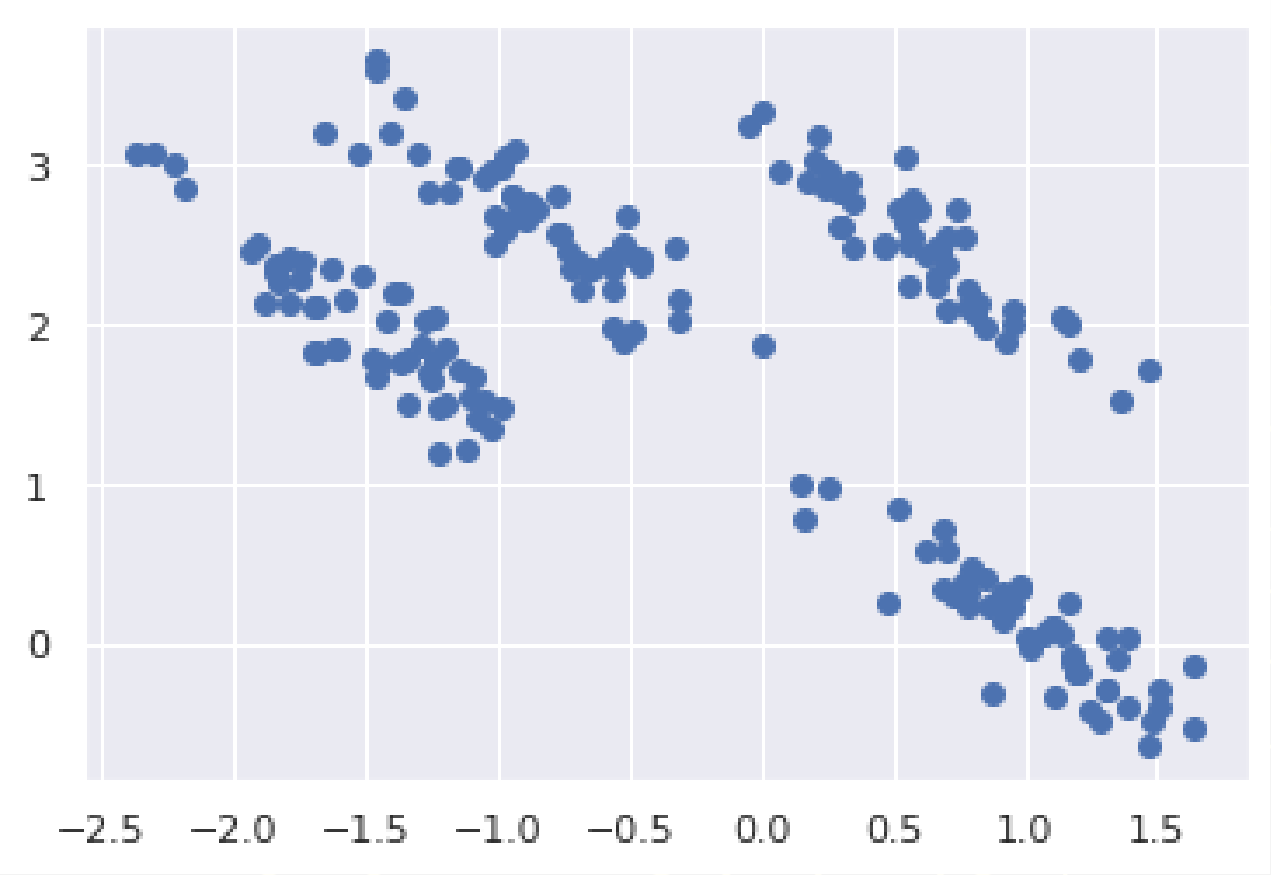


# Gaussian Distribution: Two Parameters



# Gaussian Mixture Model:

The real dataset can be seen as a mixture of several Gaussian distributions.



# Gaussian Mixture Model:

The real dataset can be seen as a mixture of several Gaussian distributions.

Suppose the dataset is formed with *k* clusters (suppose we knew it).

Then, we need to estimate mean and standard deviation for each cluster.

The probability density is defined as a linear function of densities of all these *k* distributions, i.e.

$$p(X) = \sum_{k=1}^{K} \pi_k G(X|\mu_k, \Sigma_k)$$

# Gaussian Mixture Model: Theory

# Gaussian Mixture Model: Theory

# Gaussian Mixture Model: Theory

# Gaussian Mixture Model: EM algorithm

Since the closed form solution is not available, the Expectation-Maximization (EM) algorithm is a **Two-Steps** iterative way to find maximum-likelihood estimates for model parameters

- **Estimation step:**
  - initialize $\mu_k$, $\sum_k$ and $\pi_k$ by some random values, or by K means clustering results or by hierarchical clustering results.
  - Then for those given parameter values, estimate the value of the latent variables (i.e $\gamma_k$)
- **Maximization Step:**
  - Update the value of the parameters( i.e. $\mu_k$, $\sum_k$ and$\pi_k$) calculated using ML method.

# Gaussian Mixture Model: Implementation

class sklearn.mixture.GaussianMixture(n_components=3, covariance_type='full', tol=0.001, max_iter=100, init_params='kmeans', weights_init=None, means_init=None, precisions_init=None, random_state=None, warm_start=False, verbose=0, verbose_interval=10

| | |
|---|---|
| **n_components** | The number of mixture components. |
| **covariance_type** | {'full', 'tied', 'diag', 'spherical'}, default='full'<br>full: each component has its own general covariance matrix<br>tied: all components share the same general covariance matrix<br>diag: each component has its own diagonal covariance matrix<br>spherical: each component has its own single variance |
| **tol** | default=.001<br>The convergence threshold. EM iterations will stop when the lower bound average gain is below this threshold. |
| **max_iter** | *int, default=100*<br>The number of EM iterations to perform |
| **init_params** | *{'kmeans', 'random'}, default='kmeans'*<br>The method used to initialize the weights, the means, etc. |

# Gaussian Mixture Model: Implementation

class sklearn.mixture.GaussianMixture(n_components=3, covariance_type='full', tol=0.001, reg_covar=1e-06, max_iter=100, n_init=1, init_params='kmeans', weights_init=None, means_init=None, precisions_init=None)

| | |
|---|---|
| **weights_init** | array-like of shape (n_components, ), default=None<br>The user-provided initial weights. If it is None, weights are initialized using the init_params method. |
| **means_init** | The user-provided initial means, If it is None, means are initialized using the init_params method. |
| **precisions_init** | The user-provided initial precisions (inverse of the covariance matrices). If it is None, precisions are initialized using the 'init_params' method. |

# THANKYOU