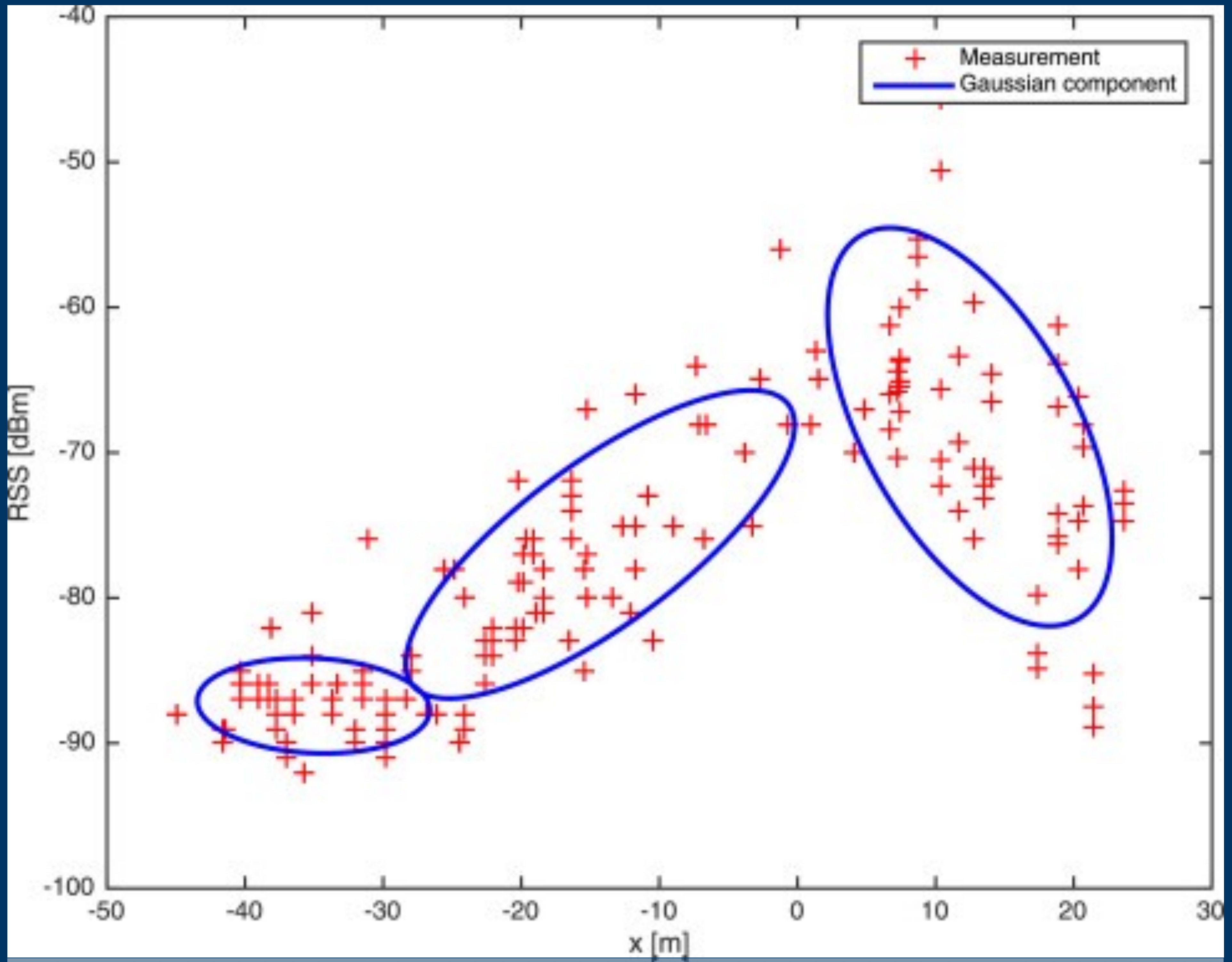
Unsupervised Learning: Gaussian Mixture Models (GMMs)



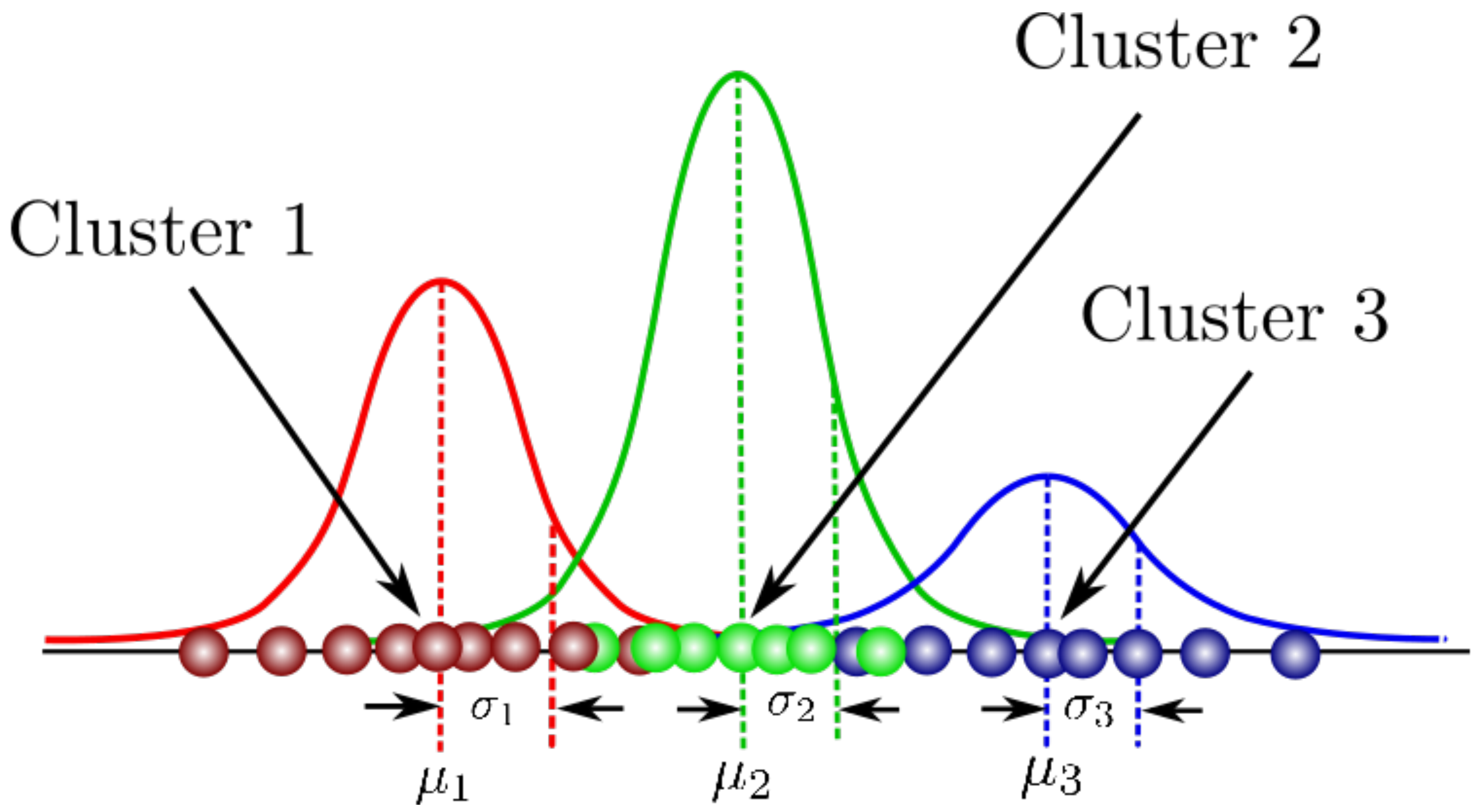
CSCI-P556 Applied Machine Learning
Lecture 23

D.S. Williamson

Agenda and Learning Outcomes

Today's Topics

- **Topics**:
  - Unsupervised Learning: Gaussian Mixture Models
- **Announcements:**
  - HW#3 Due this week
  - HW#4 Posted next week



Unsupervised Learning

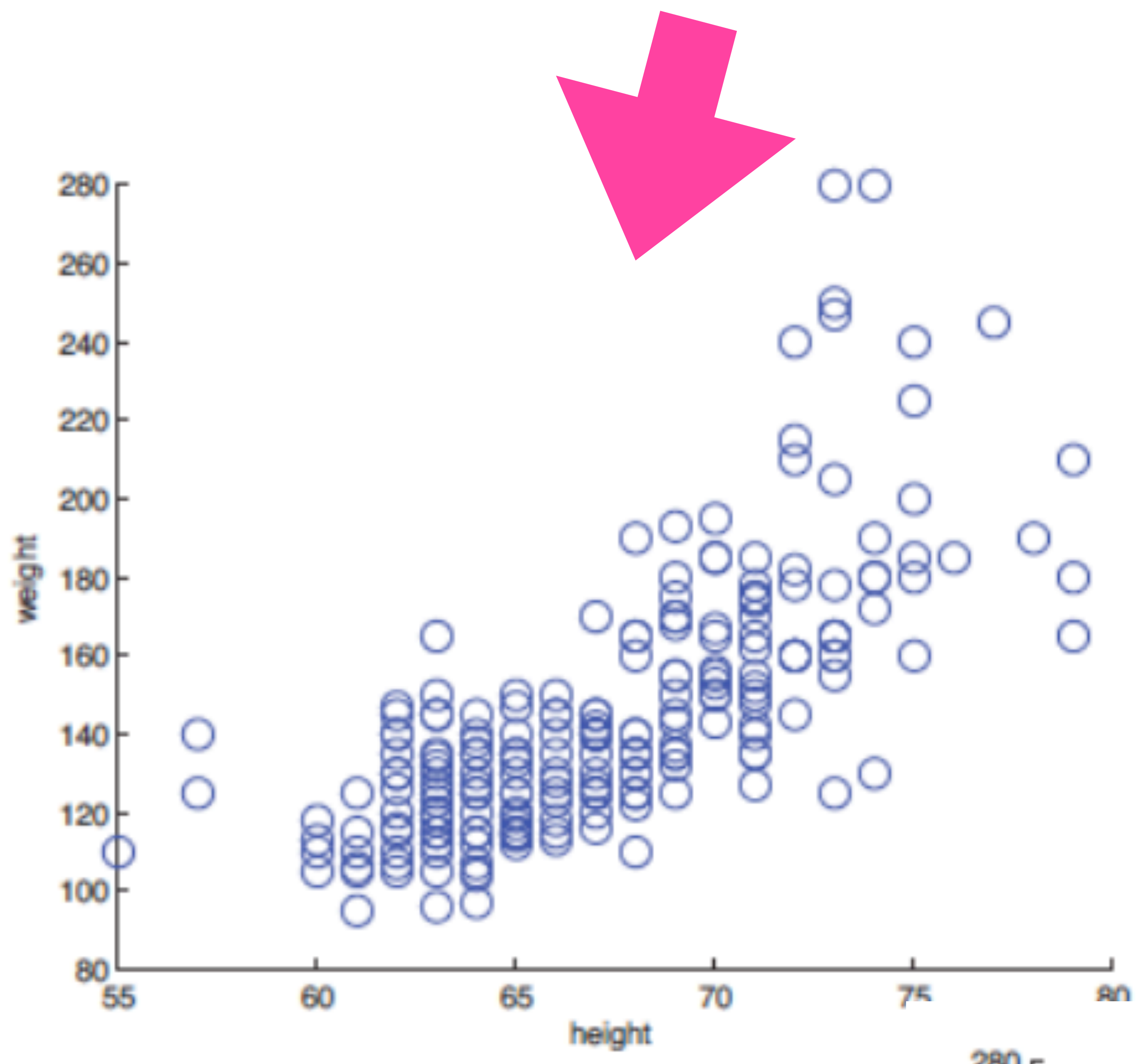
Discover patterns or structure in the data

- Only have (or use) the data information (e.g. ignore labels)

$$D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_i, \ldots, \boldsymbol{x}_N\}$$

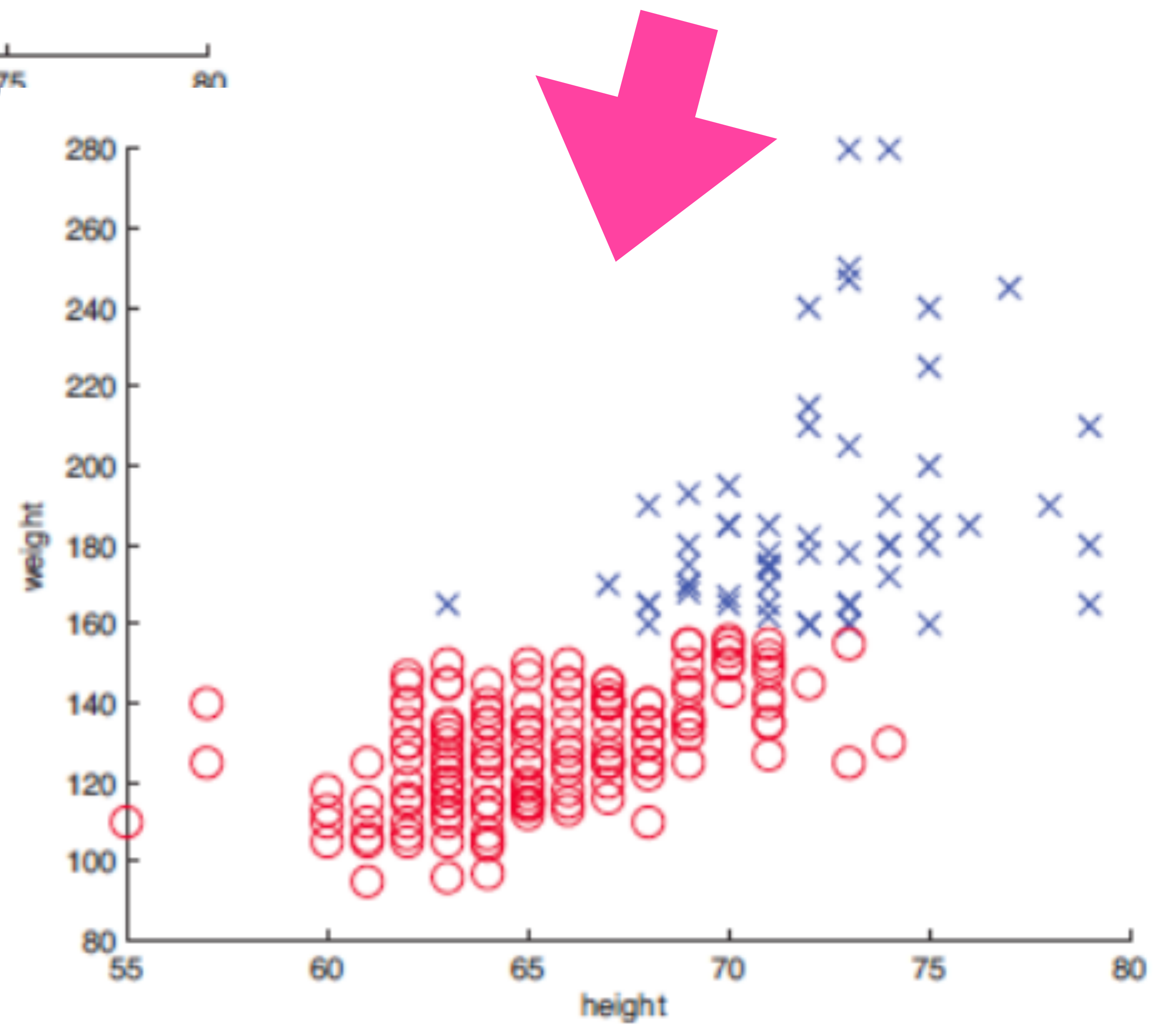
- Examples of unsupervised learning
  - **Clustering**: K-means, vector quantization, Gaussian mixture models
  - **Dimensionality Reduction**: principal components analysis, nonnegative matrix factorization
  - **Topic Modeling:** often used in NLP

Input Data



(a)

Date Clustered into two Groups



(b)

K-Means Algorithm

Iterative Steps

1. Choose a set of ***K* cluster centers randomly** from the **input data; or randomly initialize**

2. Assign the *N* input patterns (individually) to the *K* clusters using the squared Euclidean distance rule. **x** is assigned to C_j if:

$$||\mathbf{x} - \mathbf{u}_j||^2 \leq ||\mathbf{x} - \mathbf{u}_i||^2 \text{ for } i \neq j$$

3. Update cluster centers based on data that is assigned to each cluster

The number of data samples in the j-th cluster

$$\mathbf{u}_j = \frac{1}{||C_j||} \sum_{i \in C_j} \mathbf{x}_i$$

4. If any cluster center changes, go to step 2; otherwise stop. Can also specify a tolerance threshold for stopping

The K-means algorithm always converges, but the global minimum is not assured

Hard vs. Soft Clustering

K-Means Performs Hard Clustering

- K-Means is an example of ***hard clustering***, each example in the dataset is assigned to exactly one cluster
- Hard clustering may be problematic if data points are somewhat in between the centers of multiple clusters
    - They can conceivable be in multiple clusters.
- ***Soft clustering***, conversely, assigns examples to one or more clusters

Gaussian Mixture Models

A Soft Clustering Approach

- GMMs are a soft clustering approach that ***assign conditional probabilities for each example***.
    - Each cluster follows a Gaussian distribution with specific mean and variance
    - Each example can have a probability > 0 for multiple clusters
- GMMs are also useful in better modeling probability distributions of real data (more on this later)

Gaussian Distribution Review

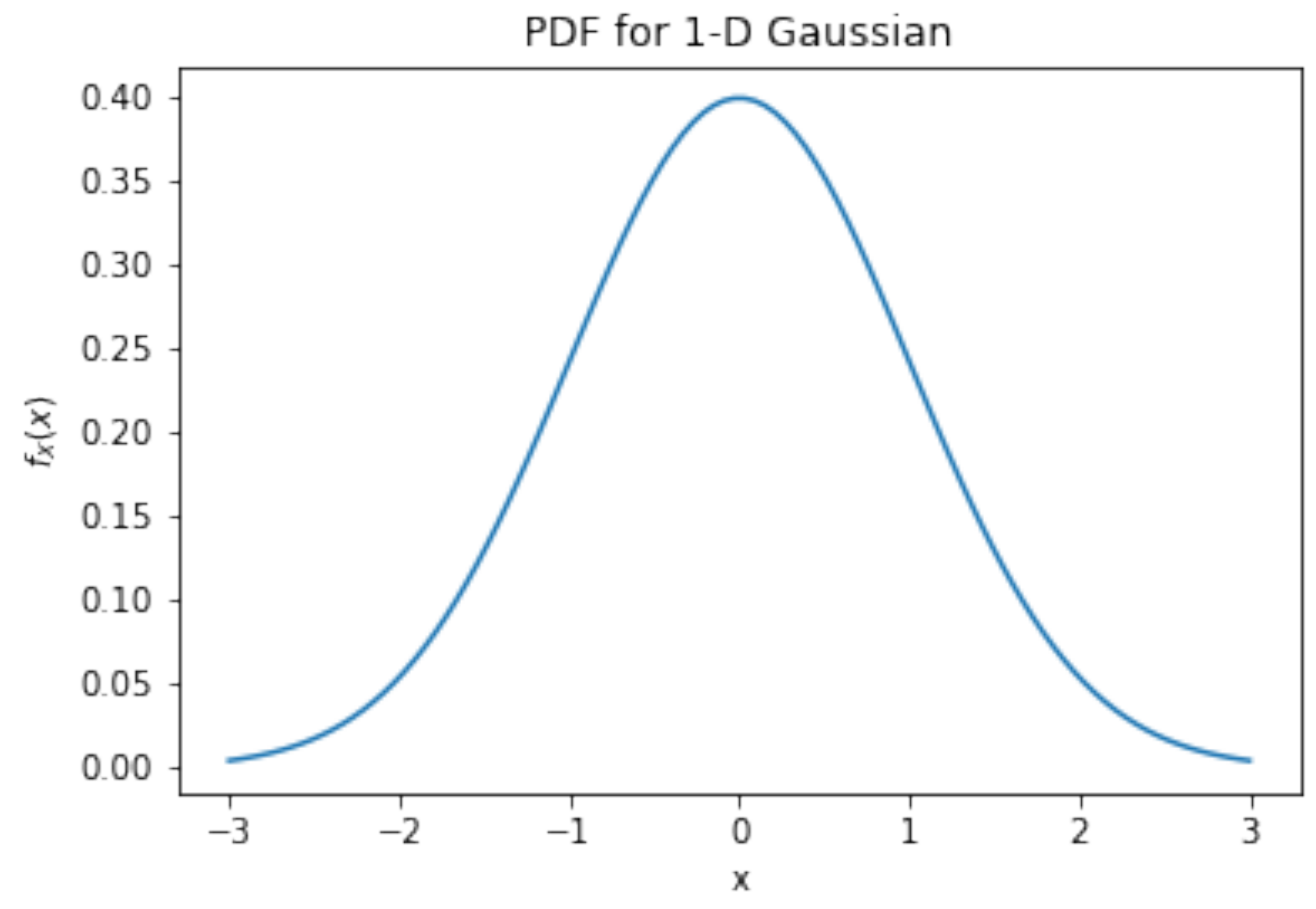
Probability Density Function

- A 1-D random variable, X, follows a Gaussian distribution if its PDF takes the following form:

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$$

μ: average (mean) value of X

σ^2: variance (spread) of X



- **Example 1-D Gaussian Random Variances:** Class grades, height, IQ, income, …

Gaussian Distribution Review

Probability Density Function

- An *N*-D random variable, **X**, follows a Gaussian distribution if its PDF takes the following form:

$$f_{\mathbf{X}}(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^N |\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\mathbf{x}-\mu)^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\mu)}$$

μ: N-D average (mean) vector for $\mathbf{X}$

$\boldsymbol{\Sigma}$: $N \times N$ co-variance matrix of $\mathbf{X}$

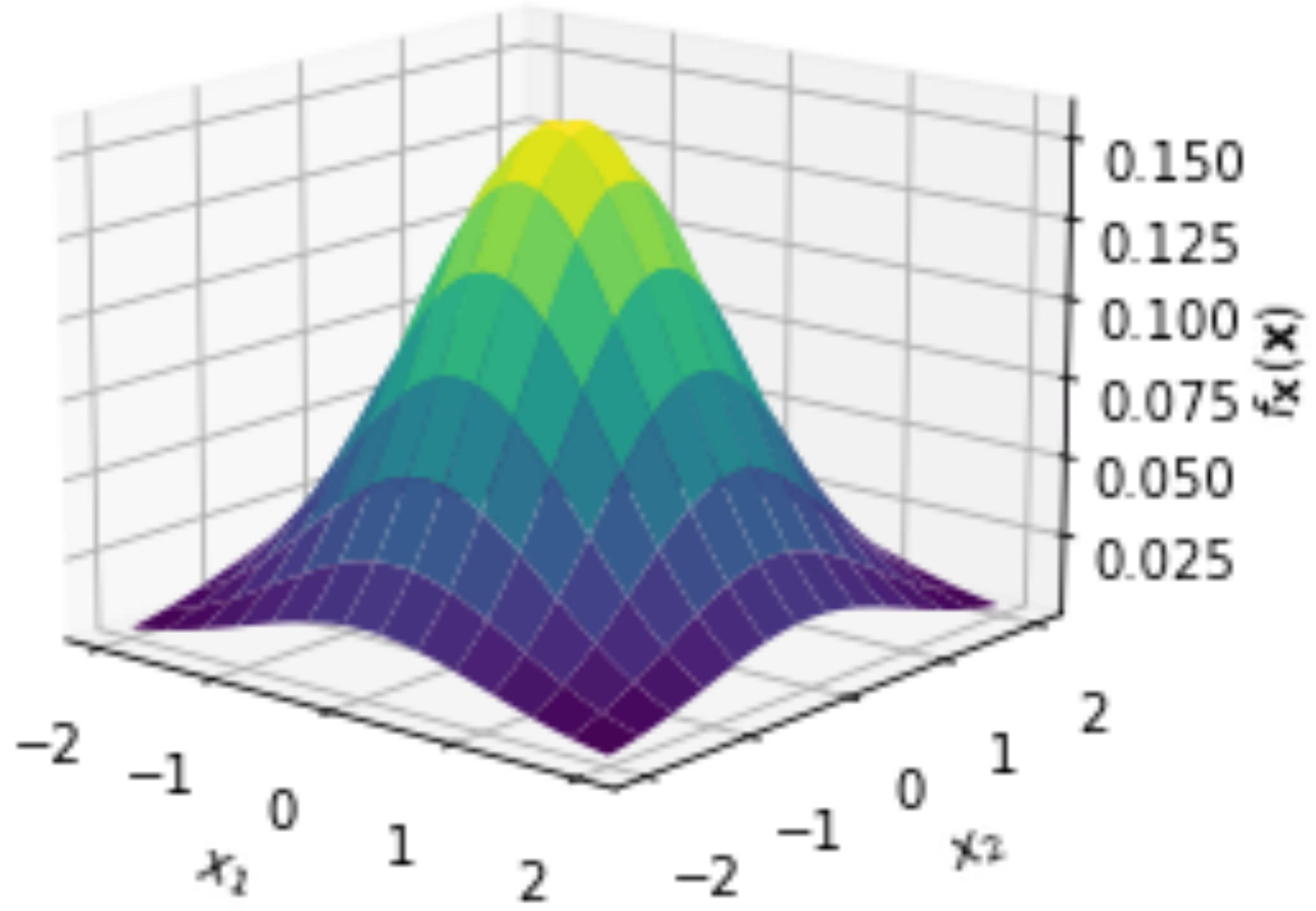
2-D Gaussian Random Variable

Side View

- Suppose **X** is a 2-D Gaussian Random Variable with the mean vector and covariance matrix below. It's PDF will look like:

$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



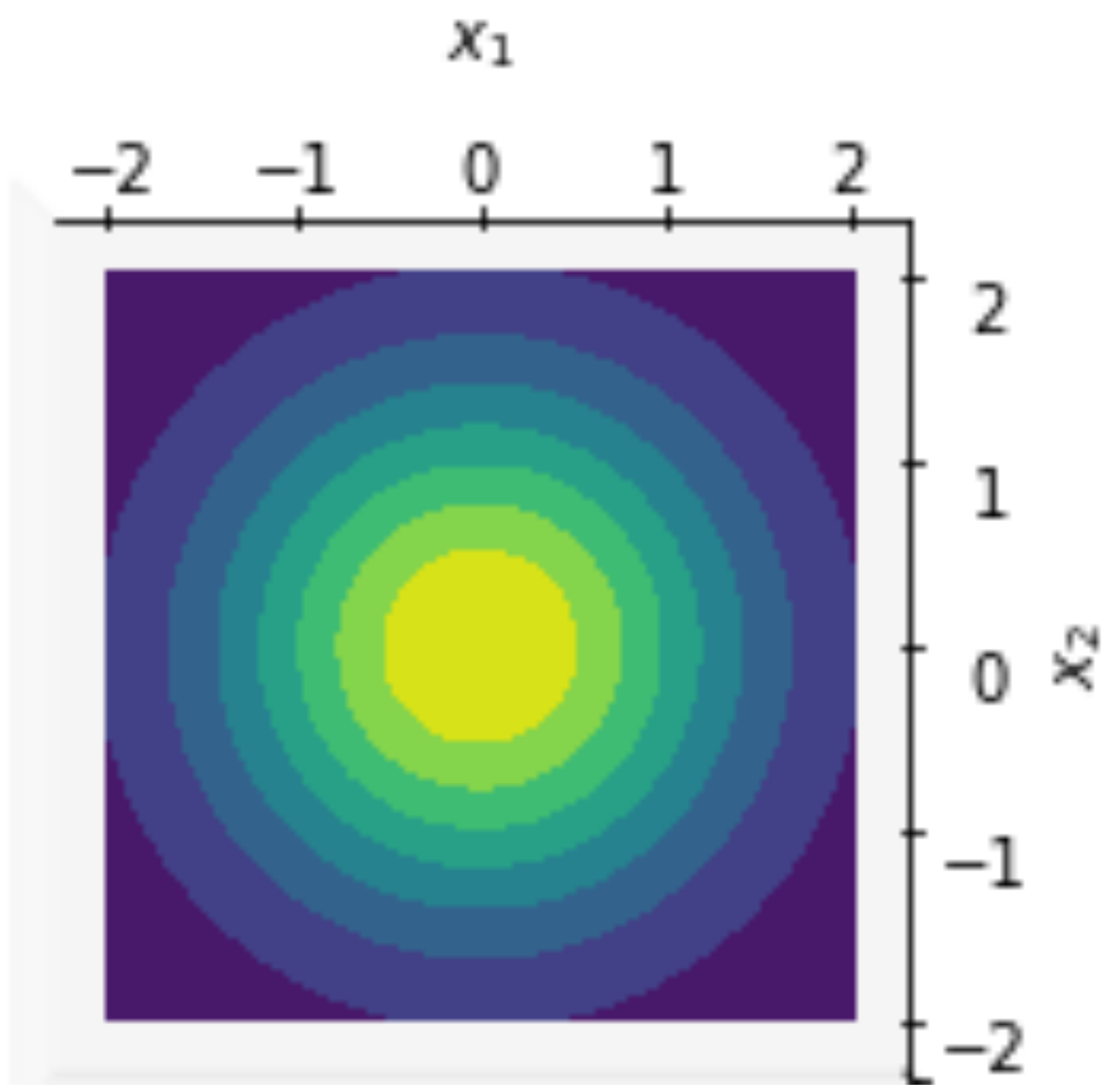
2-D Gaussian Random Variable

Top-down View

- Suppose **X** is a 2-D Gaussian Random Variable with the mean vector and covariance matrix below. It's PDF will look like:

$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



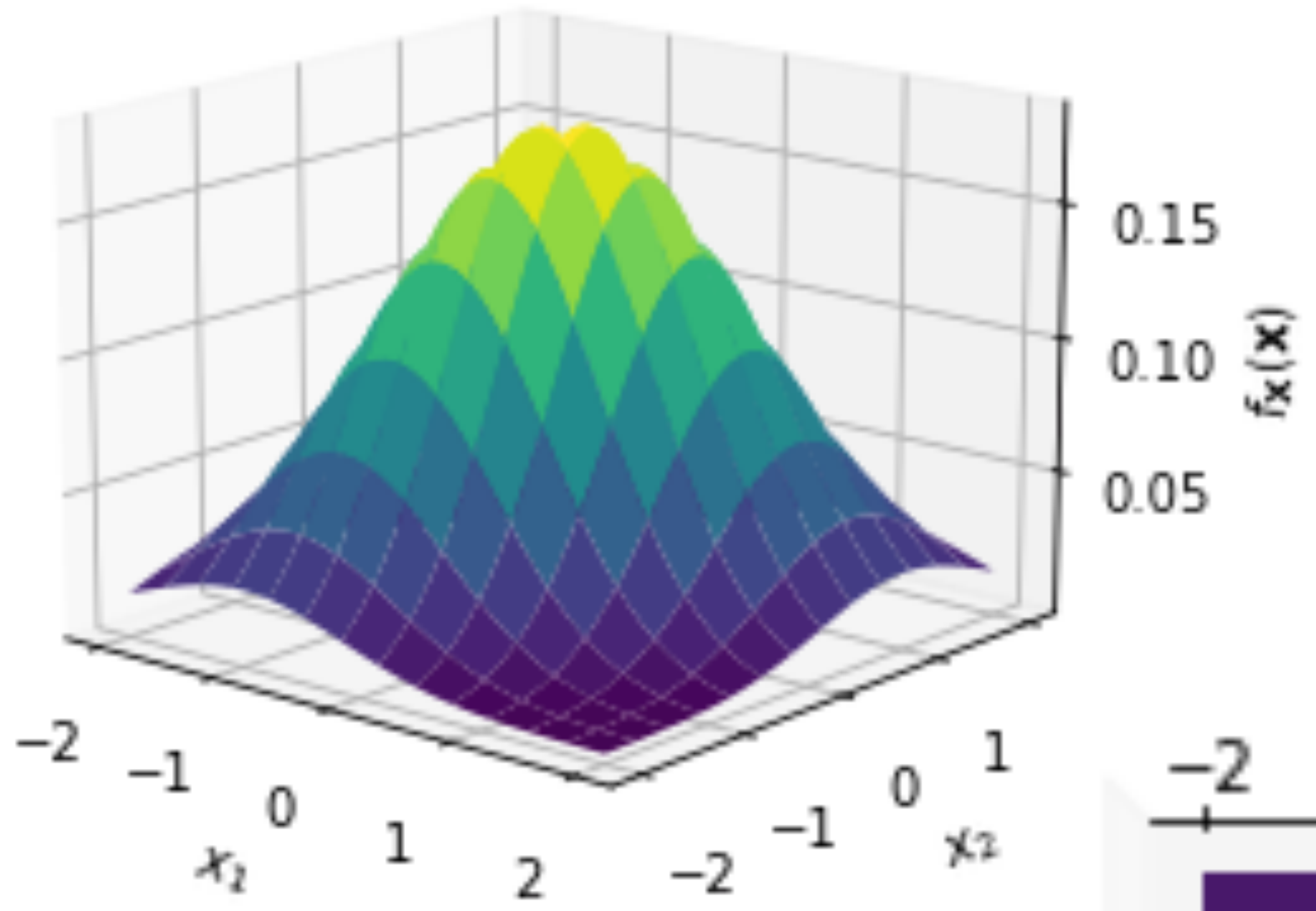
2-D Gaussian Random Variable

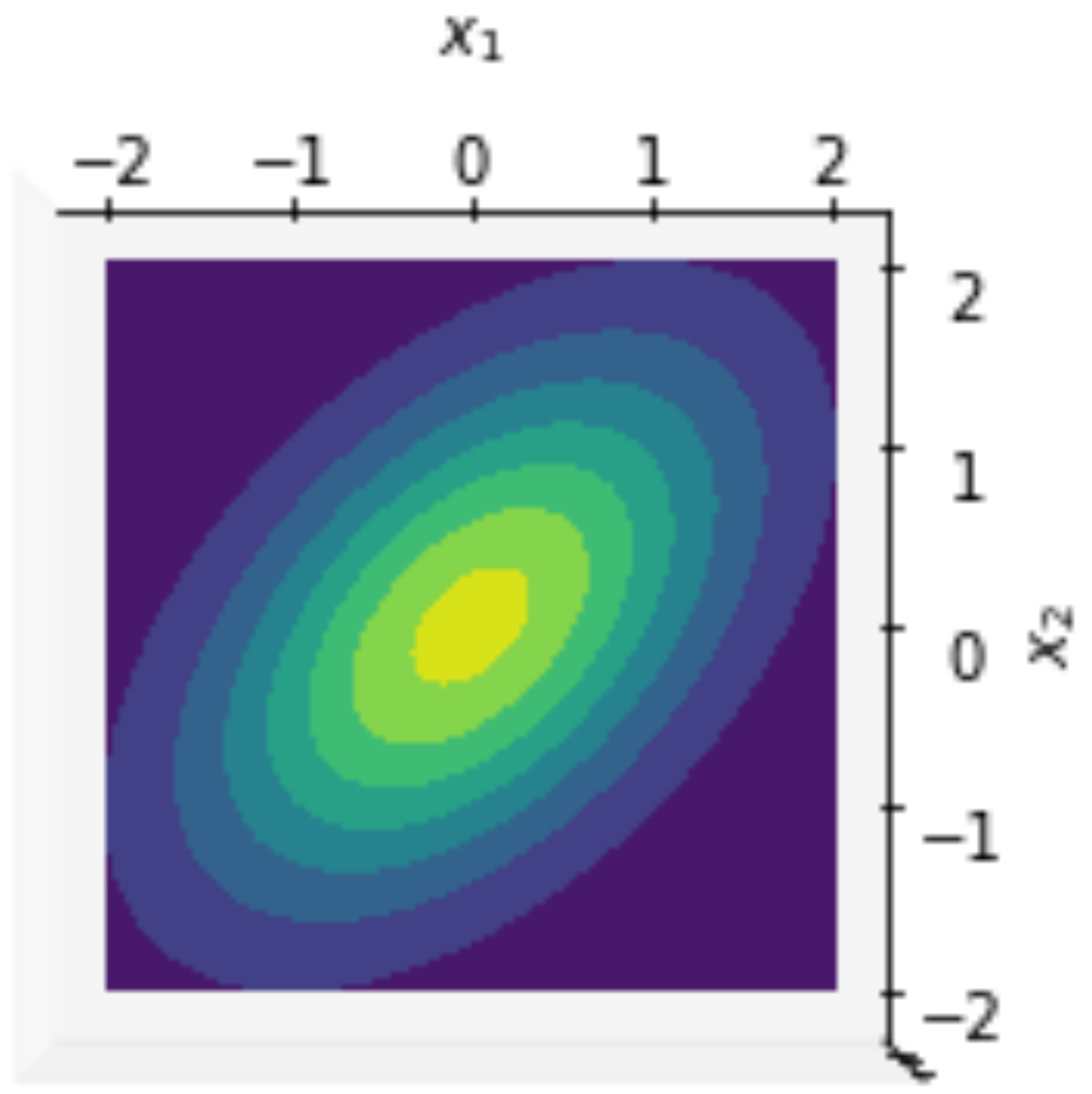
Now with co-variance terms

- Suppose **X** is a 2-D Gaussian Random Variable with the mean vector and covariance matrix below. It's PDF will look like:

$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

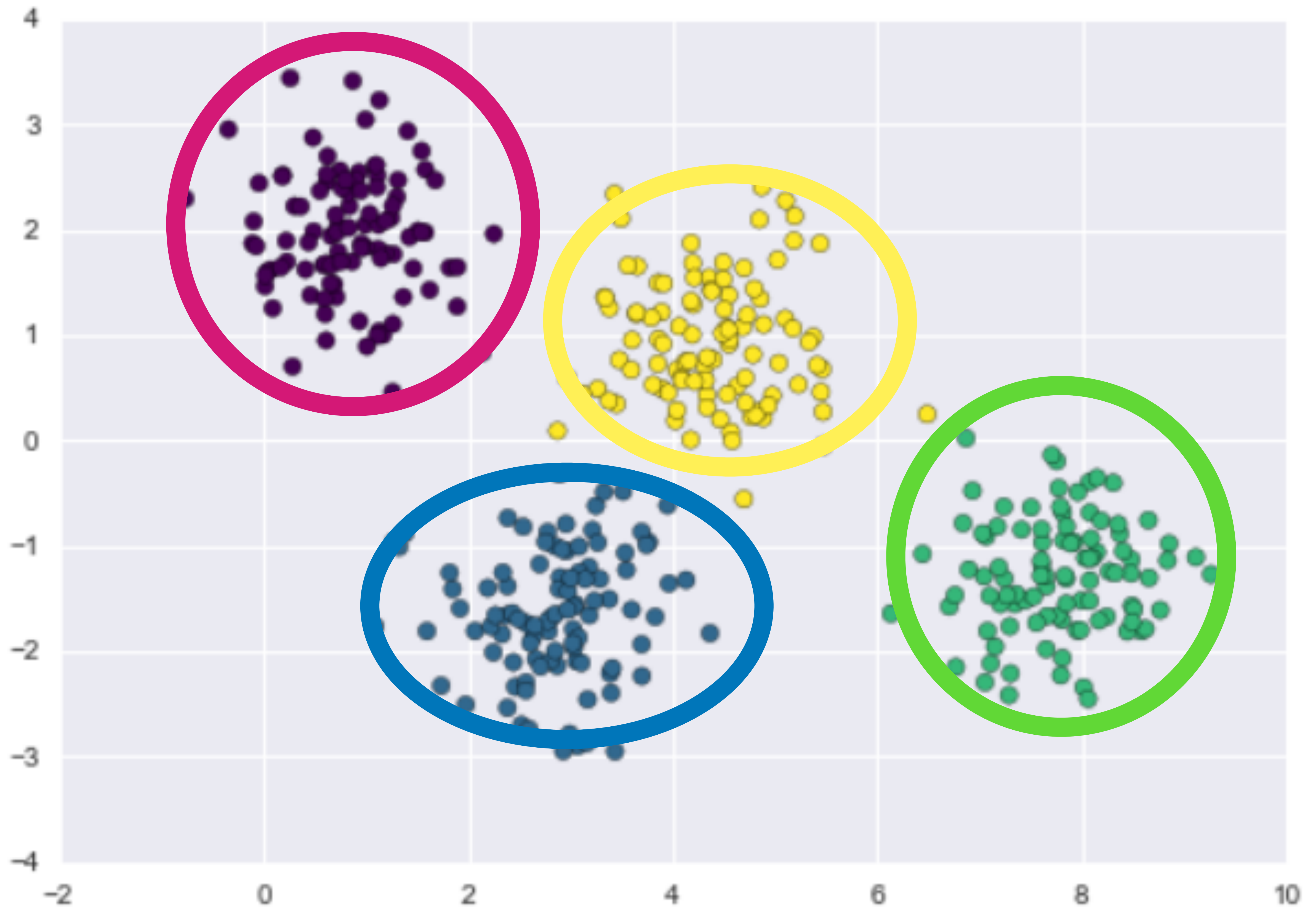
$$\Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$





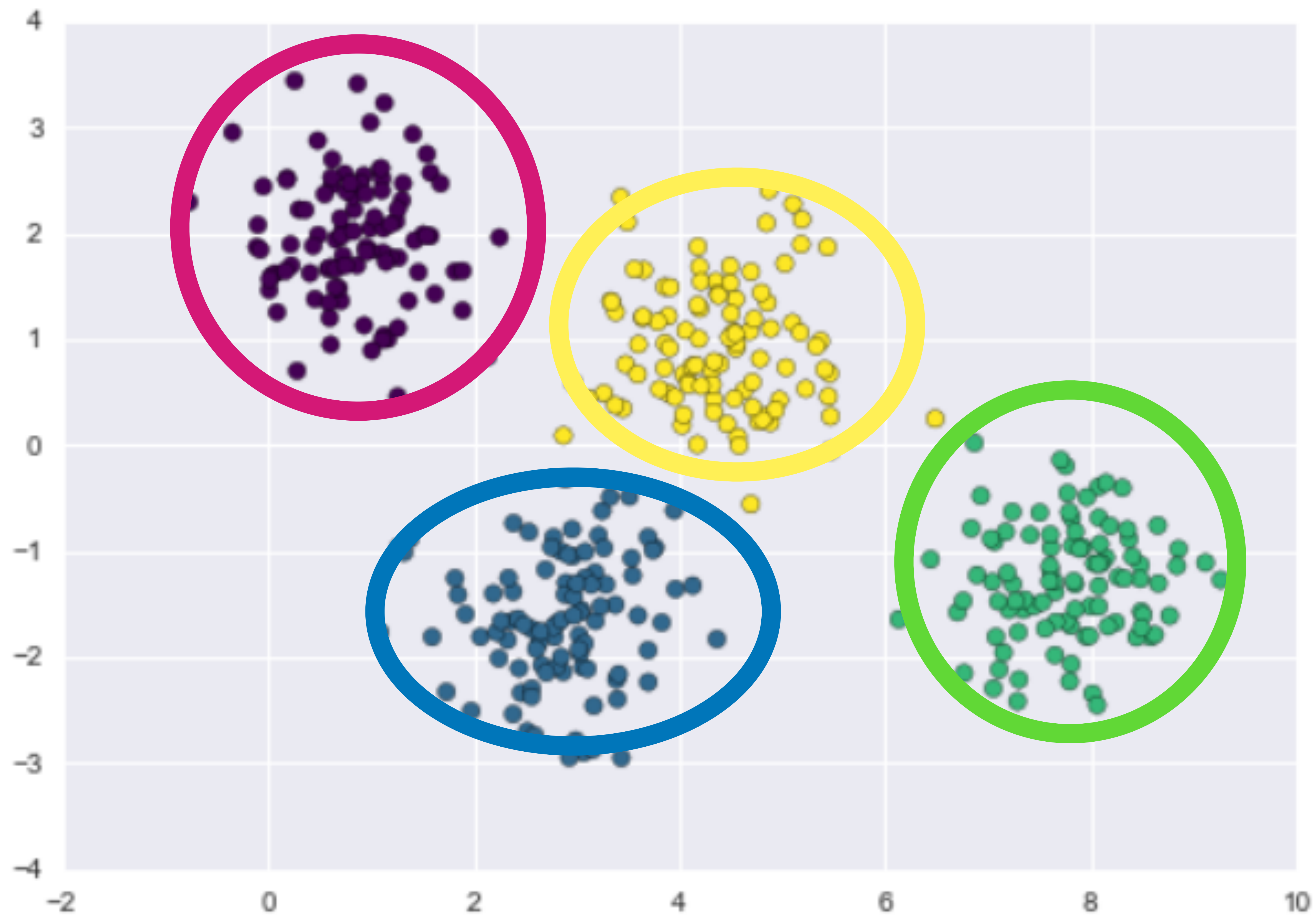
Gaussians for Clustering

- The Gaussians can be used to model soft clustering
- Each data point has a probability of being in the cluster
- If there are multiple clusters, then multiple Gaussians will be used.



Gaussian Mixture Models

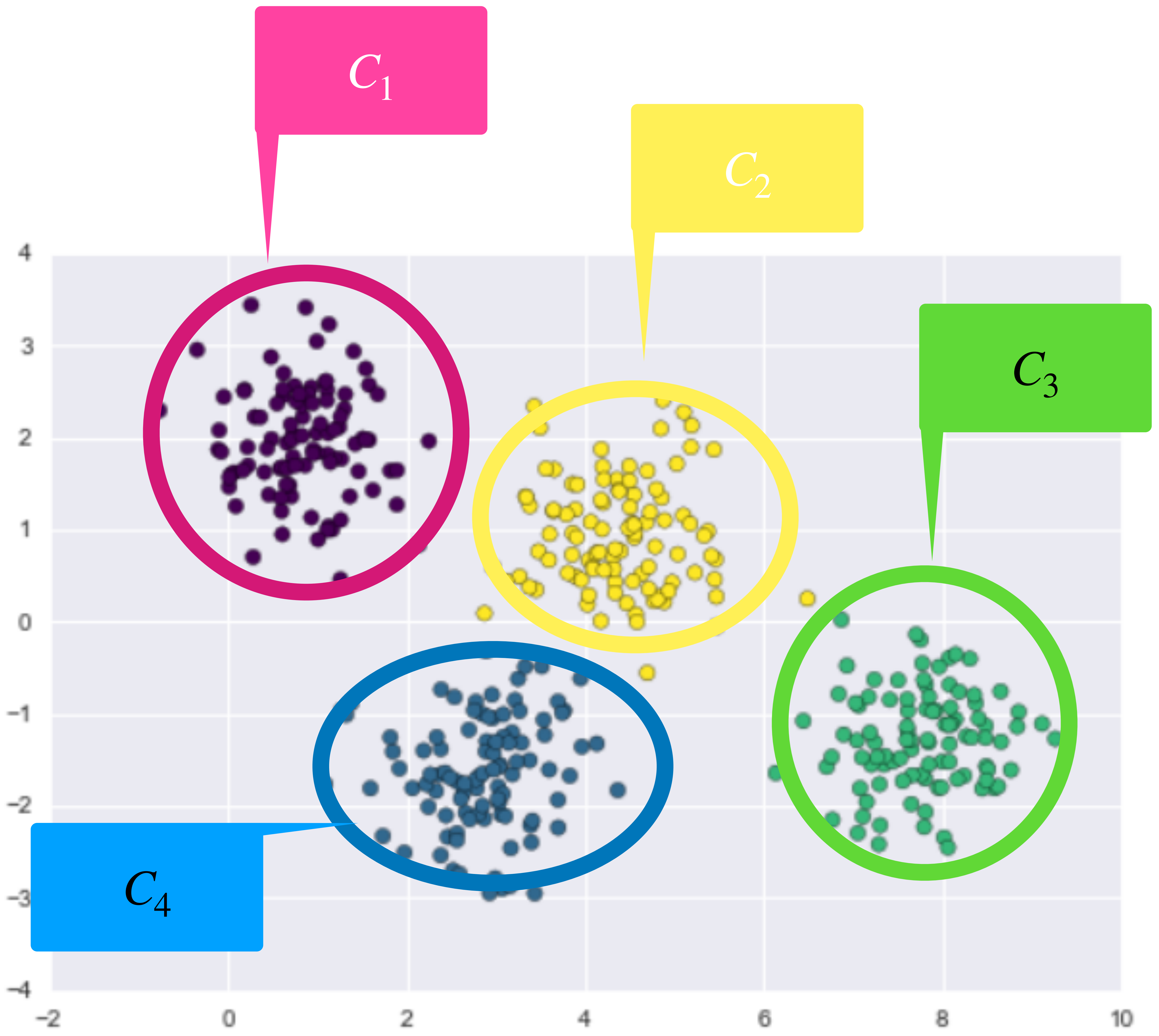
- **Main Question:**
  - How do we learn which Gaussian each data point belongs to? **Expectation-Maximization (EM) algorithm**
- Other Questions we will answer:
  - How many Gaussians?
  - How to find means of each?
  - How to find (co-) variance of each?



Gaussian Mixture Models

Notation

- **Clusters are represented by C_i.** If there are K=4 clusters then $\{C_1, C_2, C_3, C_4\}$
- The conditional probability of each sample given the cluster $P(\mathbf{X}_j \mid C_i)$ follows a Gaussian distribution with:
  - Mean vector: μ_i
  - Covariance matrix: Σ_i
- Each cluster has a certain probability as well, $P(C_i)$

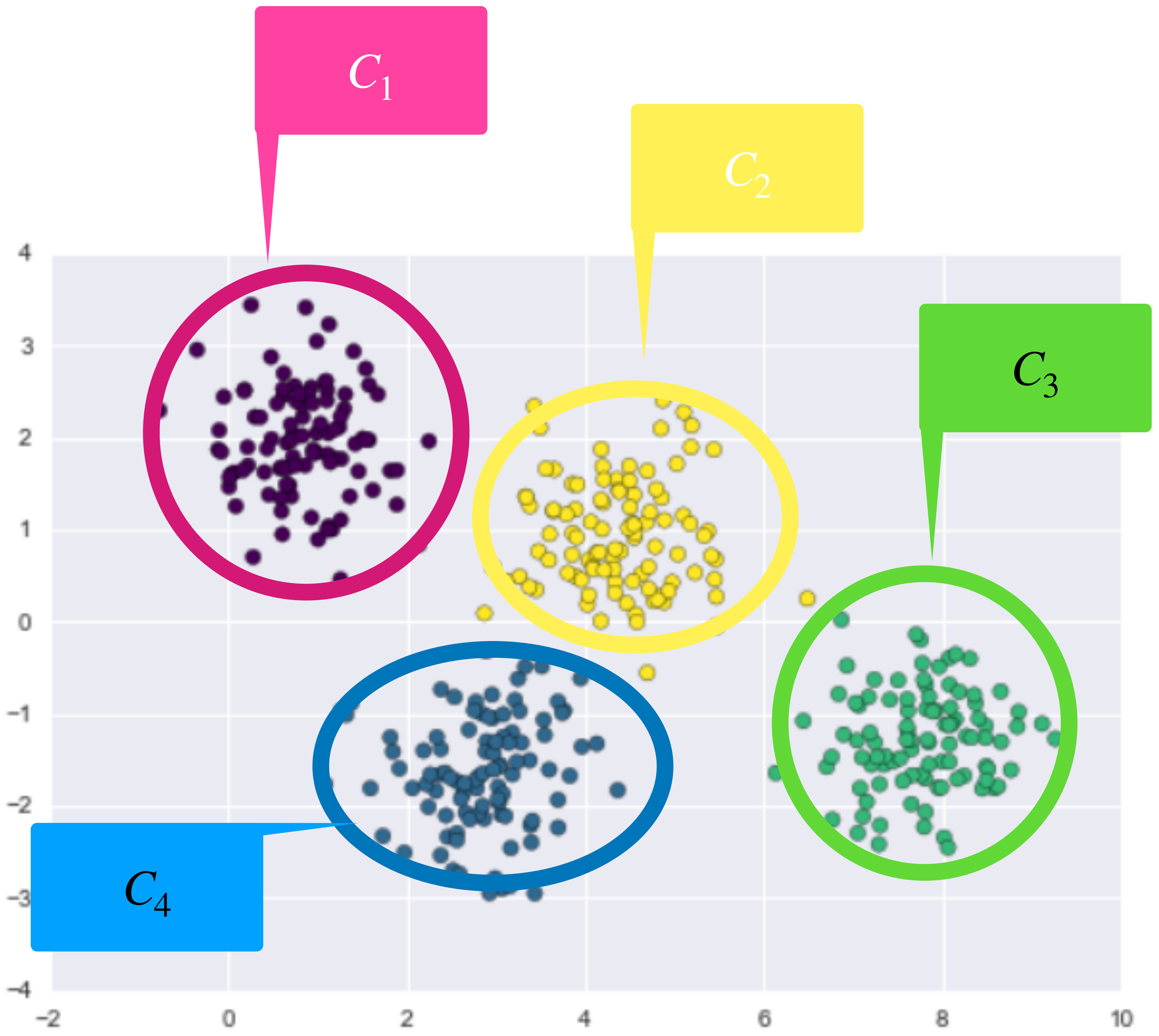


Gaussian Mixture Models

Notation

- Our goal is to find the parameters (e.g. probabilities) that maximize the likelihood of these Gaussians accurately representing **ALL** the data

$$P(\mathbf{X}) = \sum_{j=1}^{D} \sum_{i=1}^{K} \underbrace{P(C_i)}_{\text{which Gaussian (or class)?}} \underbrace{P(\mathbf{X}_j \mid C_i)}_{\text{likelihood given Gaussian}}$$



Expectation-maximization (EM) algorithm

- Two parts, iterated over and over again until convergence
- **Part 1: Expectation**
  - What's our best guess for every data point as to which Gaussian cluster (mixture) it comes from
  - In general, compute the conditional probability of the data point for a given cluster $P(\mathbf{X}_j | C_i)$
- **Part 2: Maximization:**
  - Given our expectations, figure out the parameters (mean and variance) for the Gaussian distributions
  - In general, compute new parameters based on the above conditional probability

EM: Initialization

1. Set $P(C_i)$ to a random value, making sure the sum over all Gaussians is 1 (e.g. $\sum_{i=1}^{K} P(C_i) = 1$)

2. Set means (μ_i) and variances ($\mathbf{\Sigma}_i$) for each Gaussian (e.g. cluster) to random values. This helps to get them started in the right direction.

   - With this, you can now compute the probability of ***X****j* given ***C****i*, for each class

$$P(\mathbf{X}_j \mid C_i) = \frac{1}{\sqrt{(2\pi)^N |\mathbf{\Sigma}_i|}} e^{-\frac{1}{2}(\mathbf{x}-\mu_i)^T \mathbf{\Sigma}_i^{-1}(\mathbf{x}-\mu_i)}$$

EM: Expectation Step

Compute Posterior Probability

- **Task #1**: for every point ***X***$_j$, compute the probability that cluster i (e.g. C_i) generated that point.

From Bayes Rule

$$P(C_i \mid \mathbf{X}_j) = \frac{P(\mathbf{X}_j \mid C_i)P(C_i)}{P(\mathbf{X}_j)}$$

$$= \frac{P(\mathbf{X}_j \mid C_i)P(C_i)}{\sum_{i=1}^{K} P(\mathbf{X}_j \mid C_i)P(C_i)}$$

Law of Total Probability

EM: Expectation Step

Computer Number of Samples per Cluster

- **Task #2**: Compute the (effective) number of data samples that are assigned to each cluster, N_i

$$N_i = \sum_{j=1}^{D} P(C_i \mid X_j)$$

EM: Maximization Step

Re-estimate the parameters

- For every cluster, compute a new class mean, variance, and prior probability

$$\mu_i = \frac{1}{N_i} \sum_{j=1}^{D} P(C_i \,|\, \mathbf{X}_j) \cdot \mathbf{X}_j$$

new mean vector: weighted average of points assigned to class *i*

$$\mathbf{\Sigma}_i = \frac{1}{N_i} \sum_{j=1}^{D} P(C_i \,|\, \mathbf{X}_j)(\mathbf{X}_j - \mu_i)(\mathbf{X}_j - \mu_i)^T$$

new variance matrix: calculated in same weighted manner

$$P(C_i) = \frac{N_i}{\sum_{i=1}^{K} N_i}$$

new class prior: proportion of weighted samples attributed to class

EM: Iterations of EM algorithm

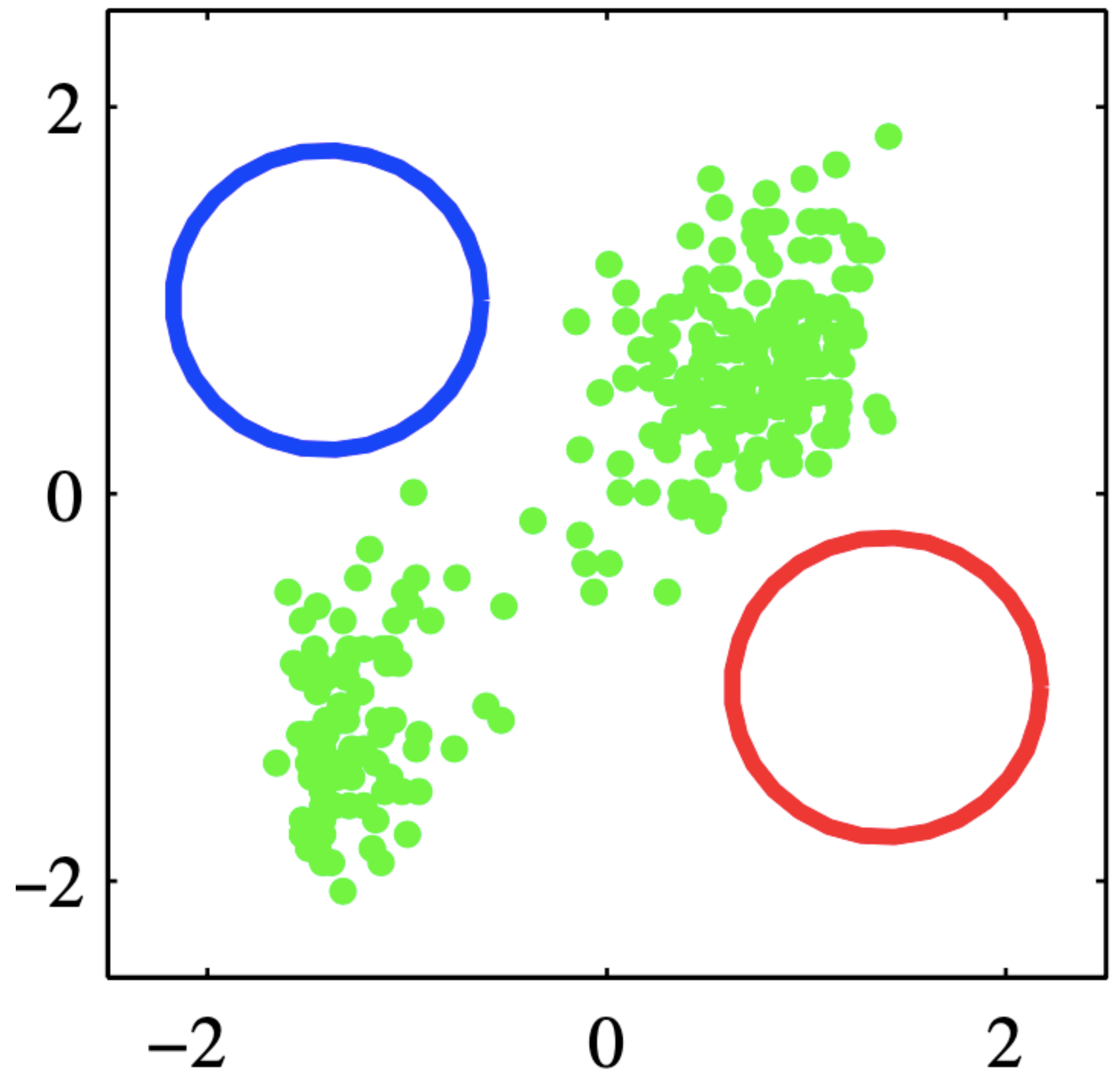
Summary of EM

1. Initialize probabilities and parameters
2. Perform Expectation Step
    1. Compute posterior probability of cluster given each data point, $P(C_i \mid \mathbf{X}_j)$
    2. Compute effective number of data points in each cluster, N_i
3. Perform Maximization Step
    1. Update mean of each cluster,μ_i
    2. Update variance of each cluster, $\mathbf{\Sigma}_i$
    3. Update prior probability of each cluster, $P(C_i)$
4. Return to step 2, unless convergence condition has been satisfied

Graphical Depiction of EM

Clustering Data using Two Gaussians

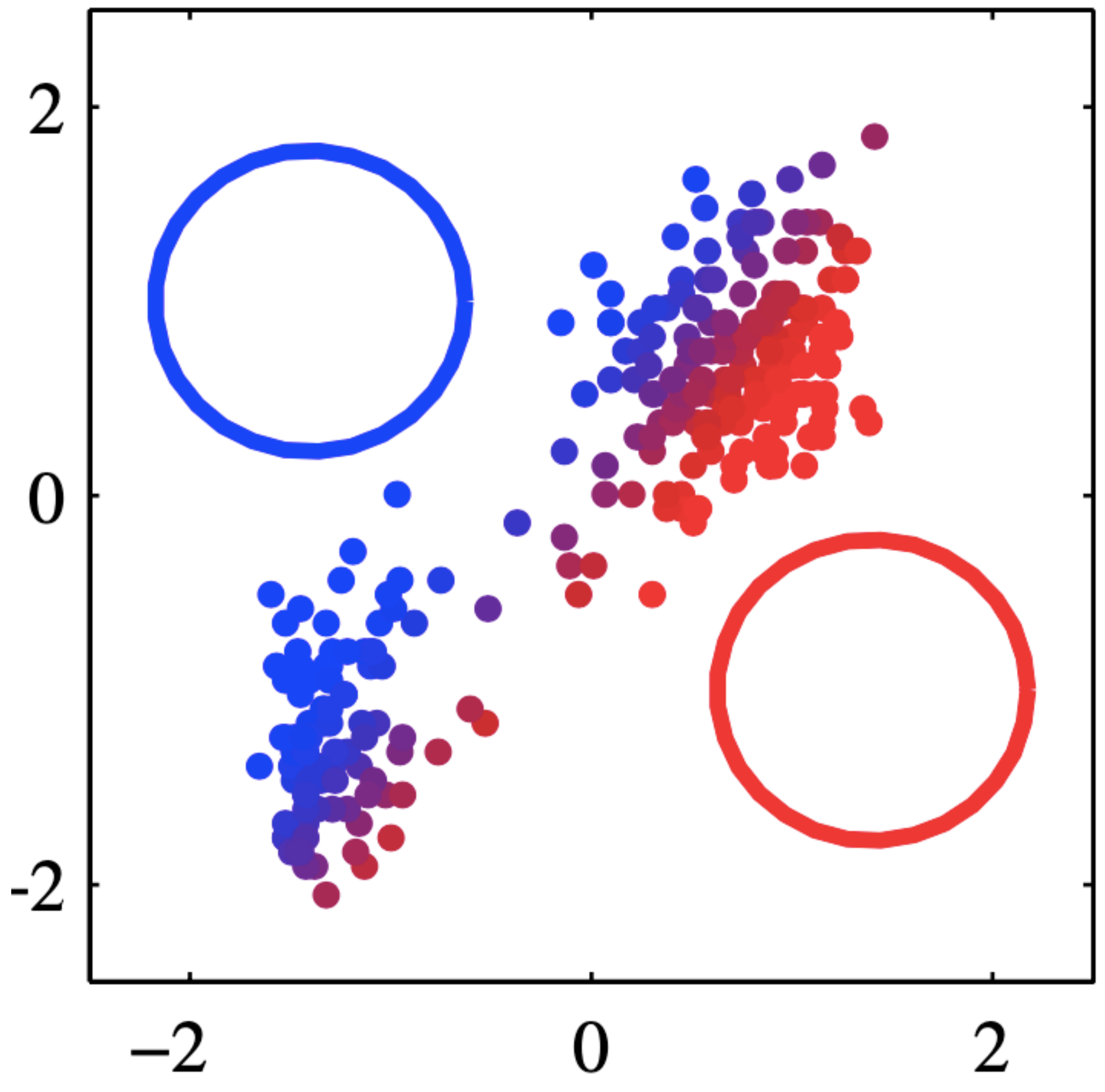
- 2-D Data points shown in green
- Initial 2-D Gaussians, one in Blue, one in Red
- Showing Gaussian contours out to one standard deviation



Graphical Depiction of EM

Clustering Data using Two Gaussians

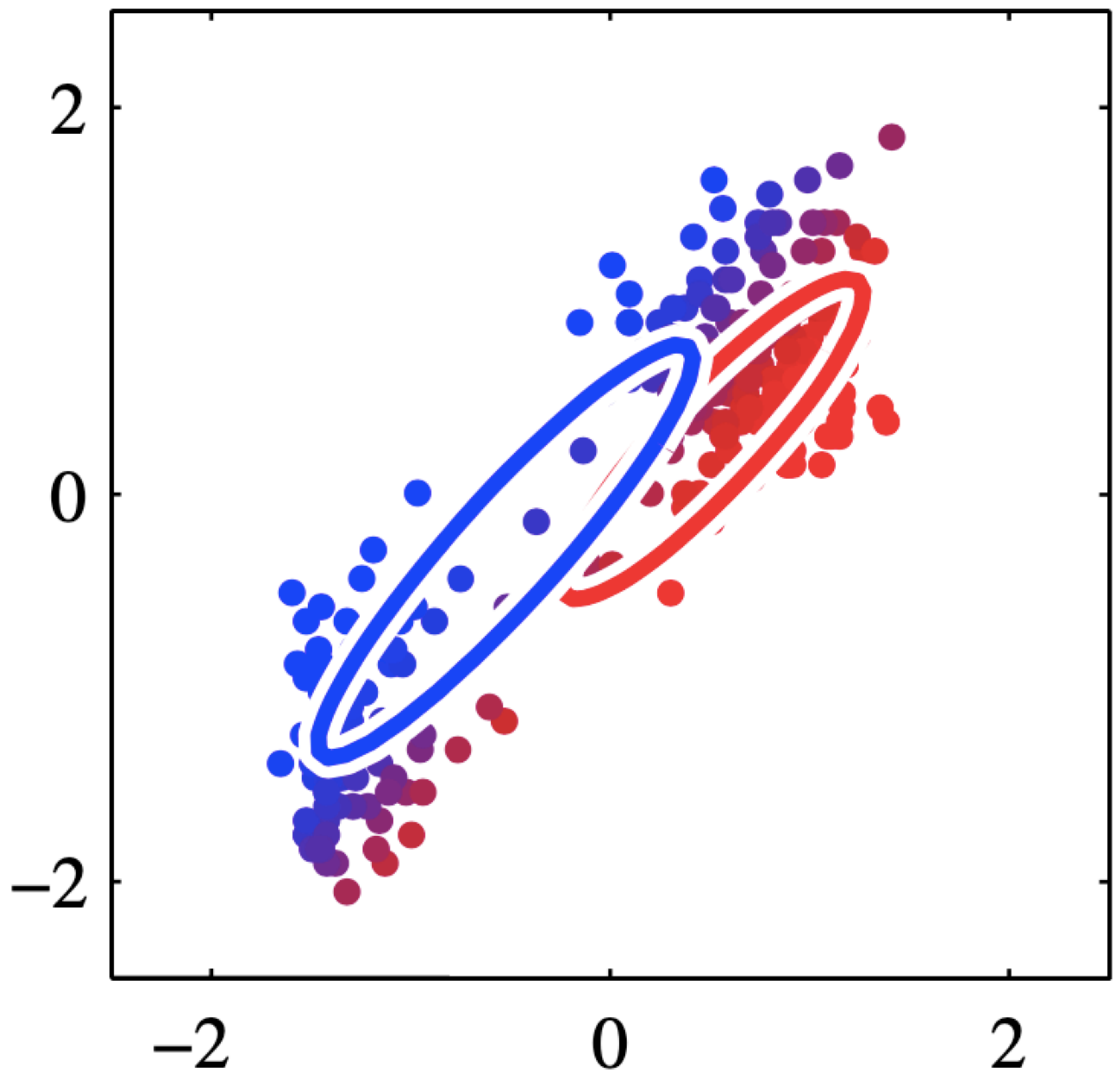
- Result after the initial E-step
    - Compute posterior probability of cluster given each data point, $P(C_i \mid \mathbf{X}_j)$
    - Color indicates probability that it was generated by the cluster
    - Points that could have come from either cluster are shown in Purple



Graphical Depiction of EM

Clustering Data using Two Gaussians

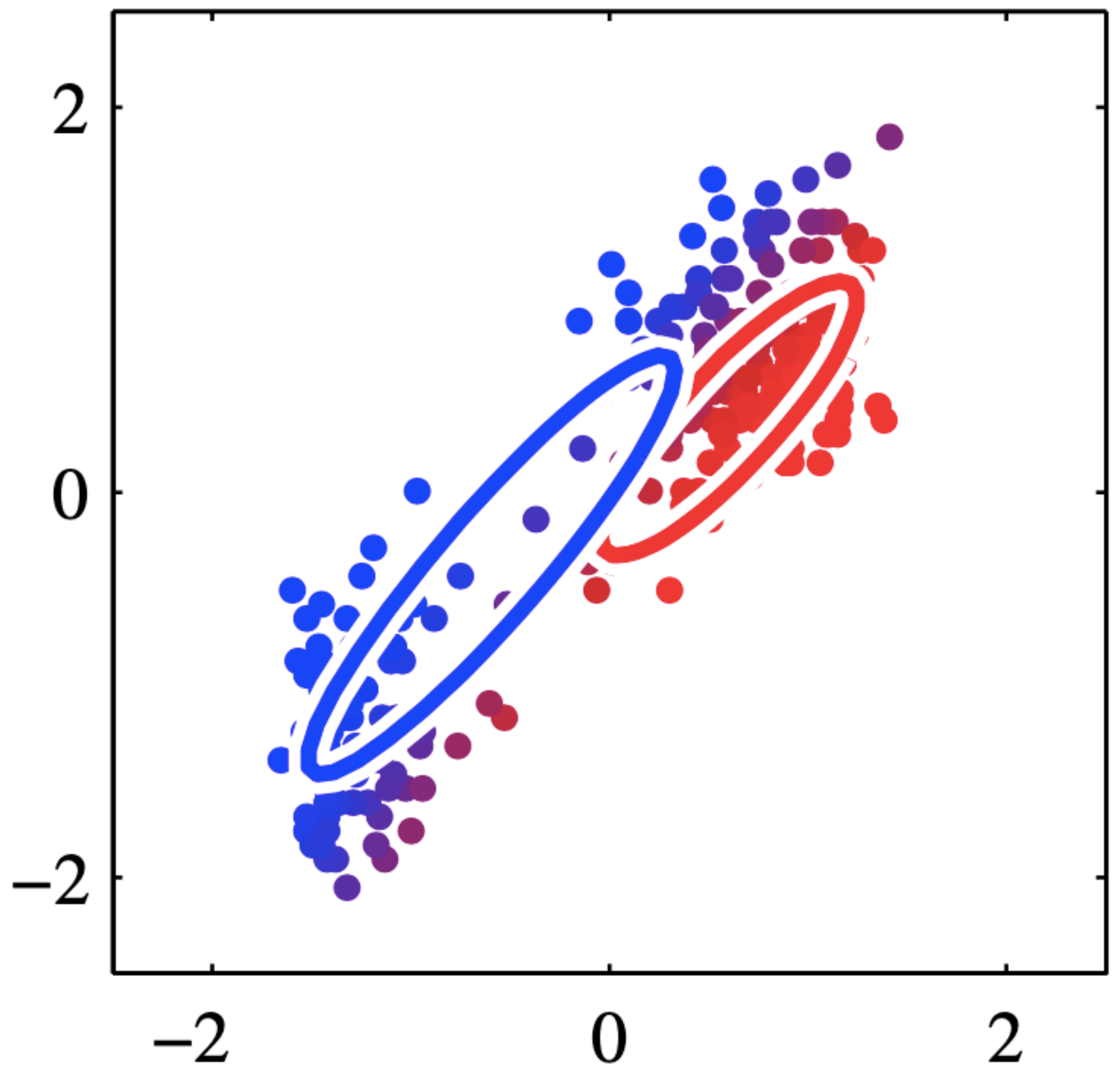
- Result after the initial M-step
  - For every cluster, compute a new class mean, variance, and prior probability
- The clusters have moved and better represent their data



Graphical Depiction of EM

Clustering Data using Two Gaussians

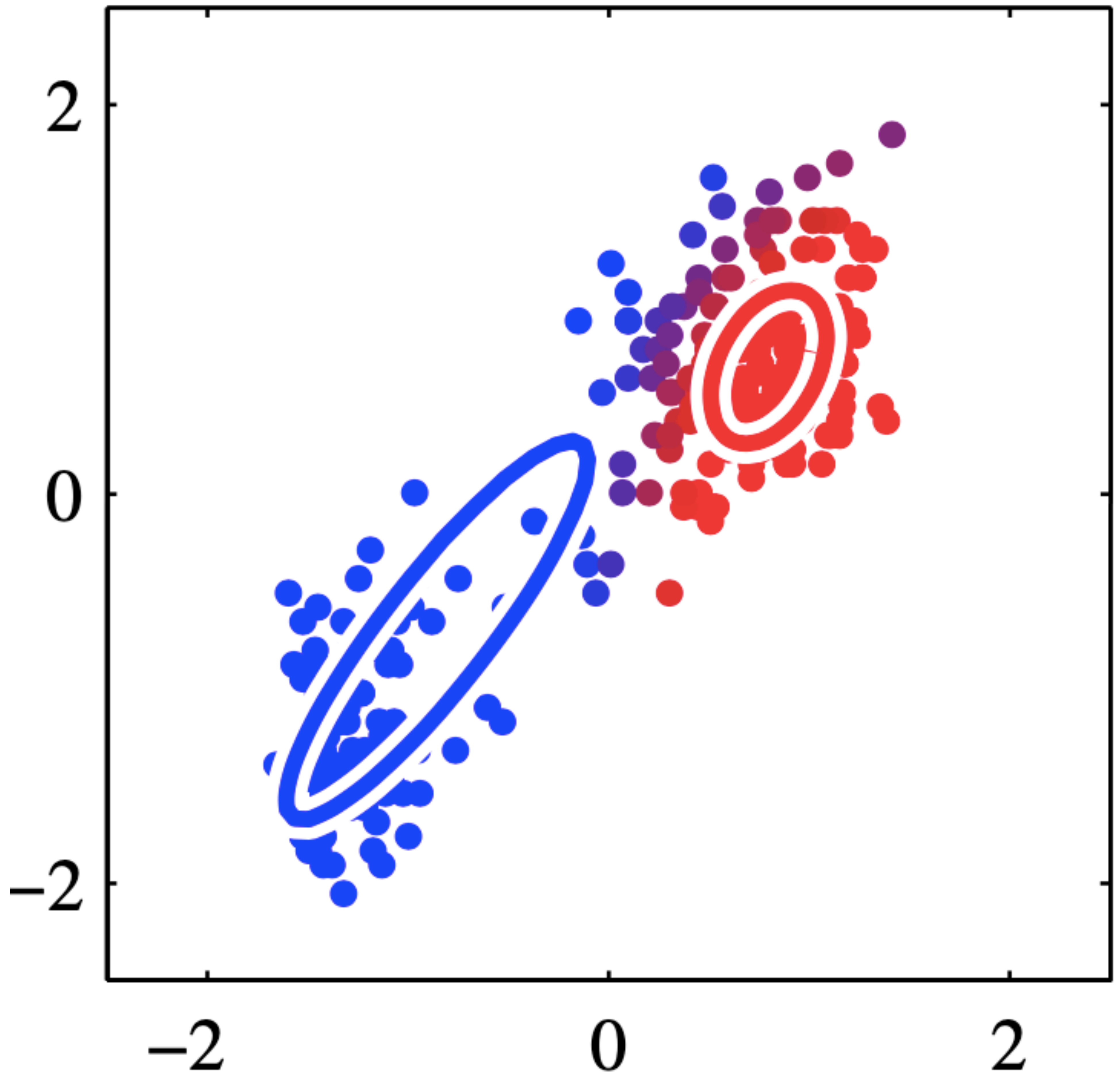
- Result after the completing the 2nd E and M steps



Graphical Depiction of EM

Clustering Data using Two Gaussians

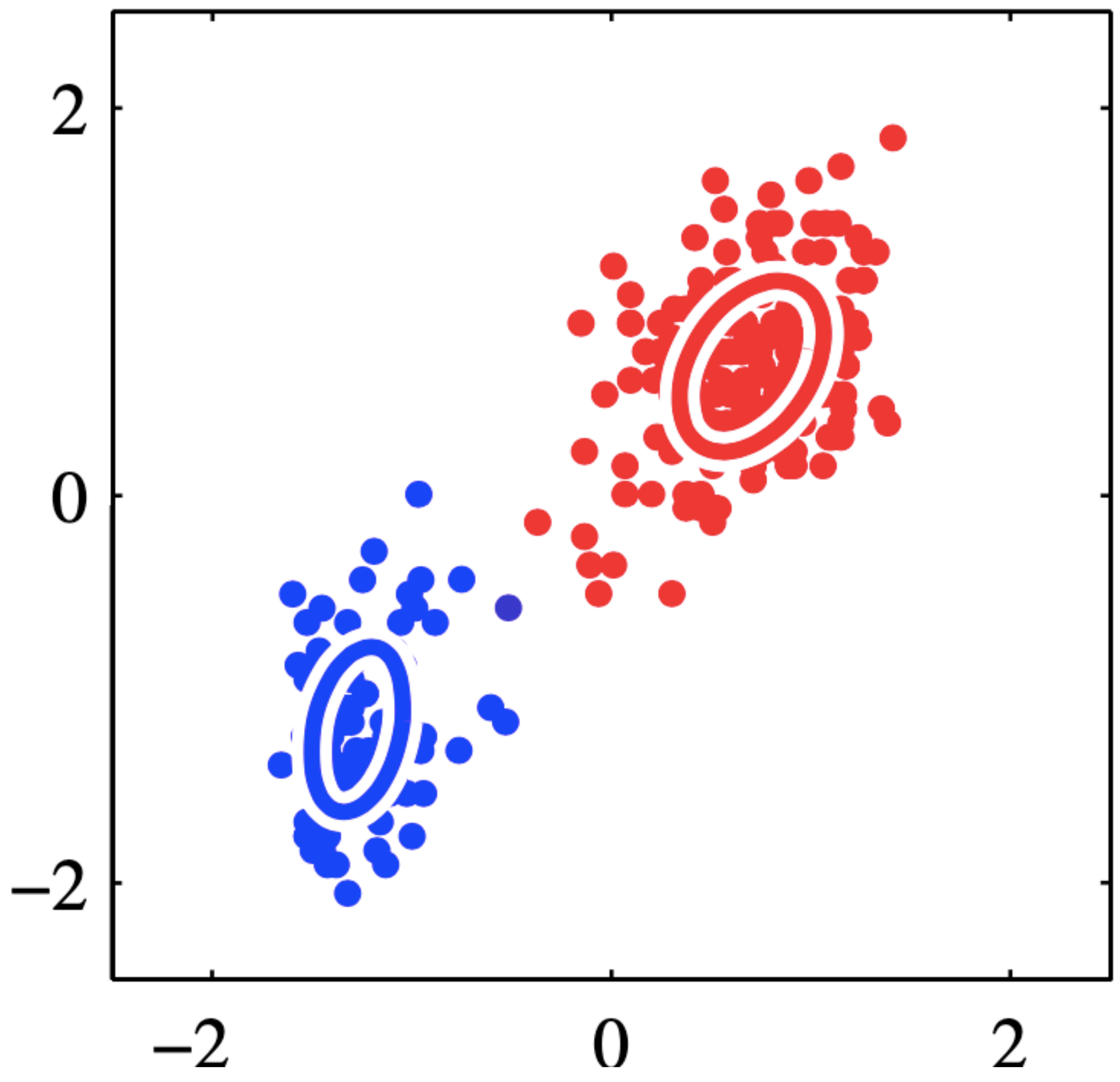
- Result after the completing the 5th E and M steps



Graphical Depiction of EM

Clustering Data using Two Gaussians

- Result after the completing the 20th E and M steps



EM Performance Evaluation

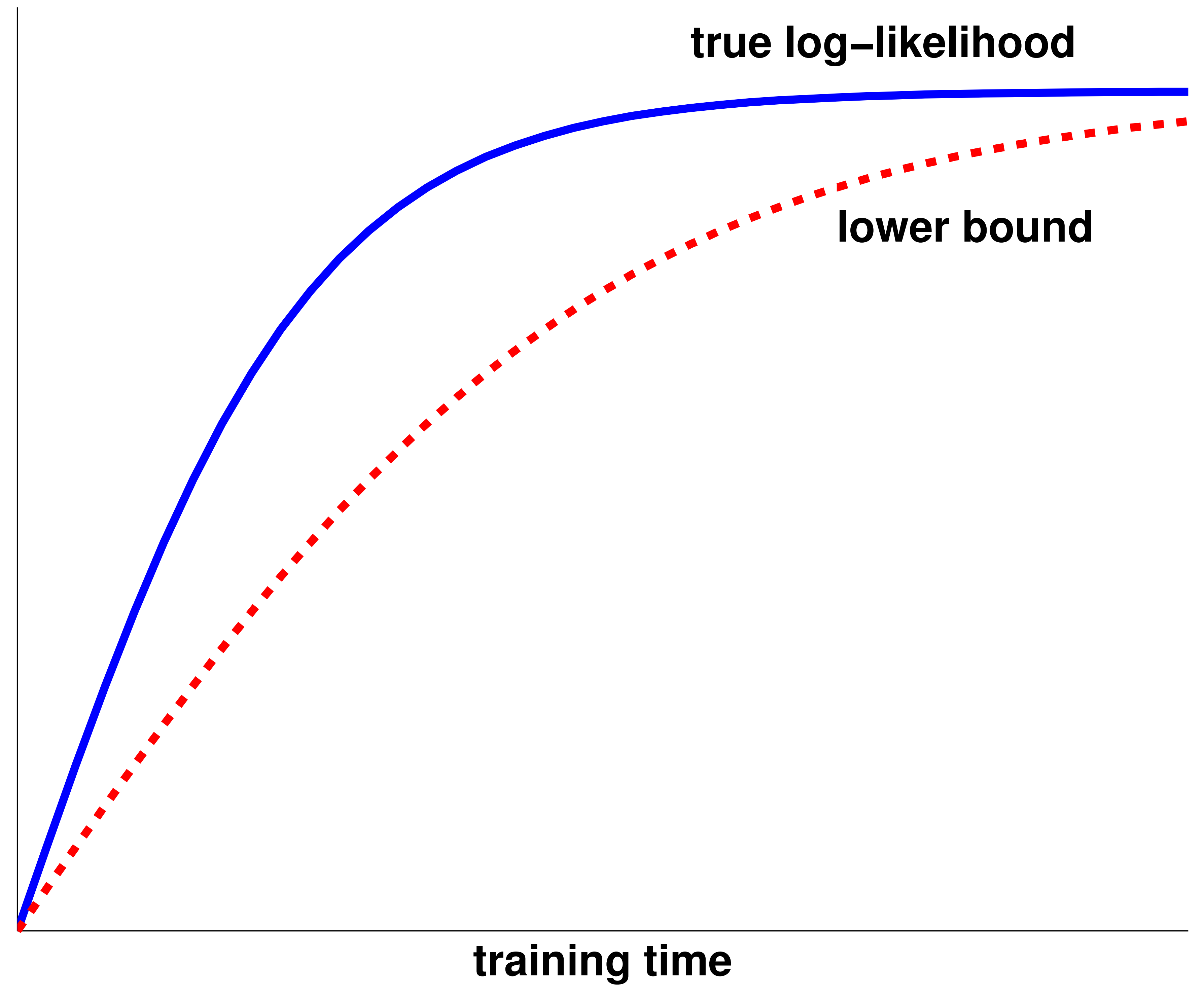
How to judge performance

- The log likelihood is used to evaluate performance
- For a set of data **X**, we define the log likelihood given as (assuming independence):

$$\ln P(\mathbf{X}) = \sum_{j=1}^{D} \ln \left[\sum_{i=1}^{K} P(C_i) P(\mathbf{X}_j \mid C_i) \right]$$

EM Performance Evaluation

- **Key points:**
  - EM algorithm monotonically increases the data log likelihood until it reaches a local optimum
  - **If you do not observe monotonic increase, then you must have an error in your math and/or code!**
  - **In practice, the algorithm converges when the change in log likelihood (or the parameters) falls below some threshold**

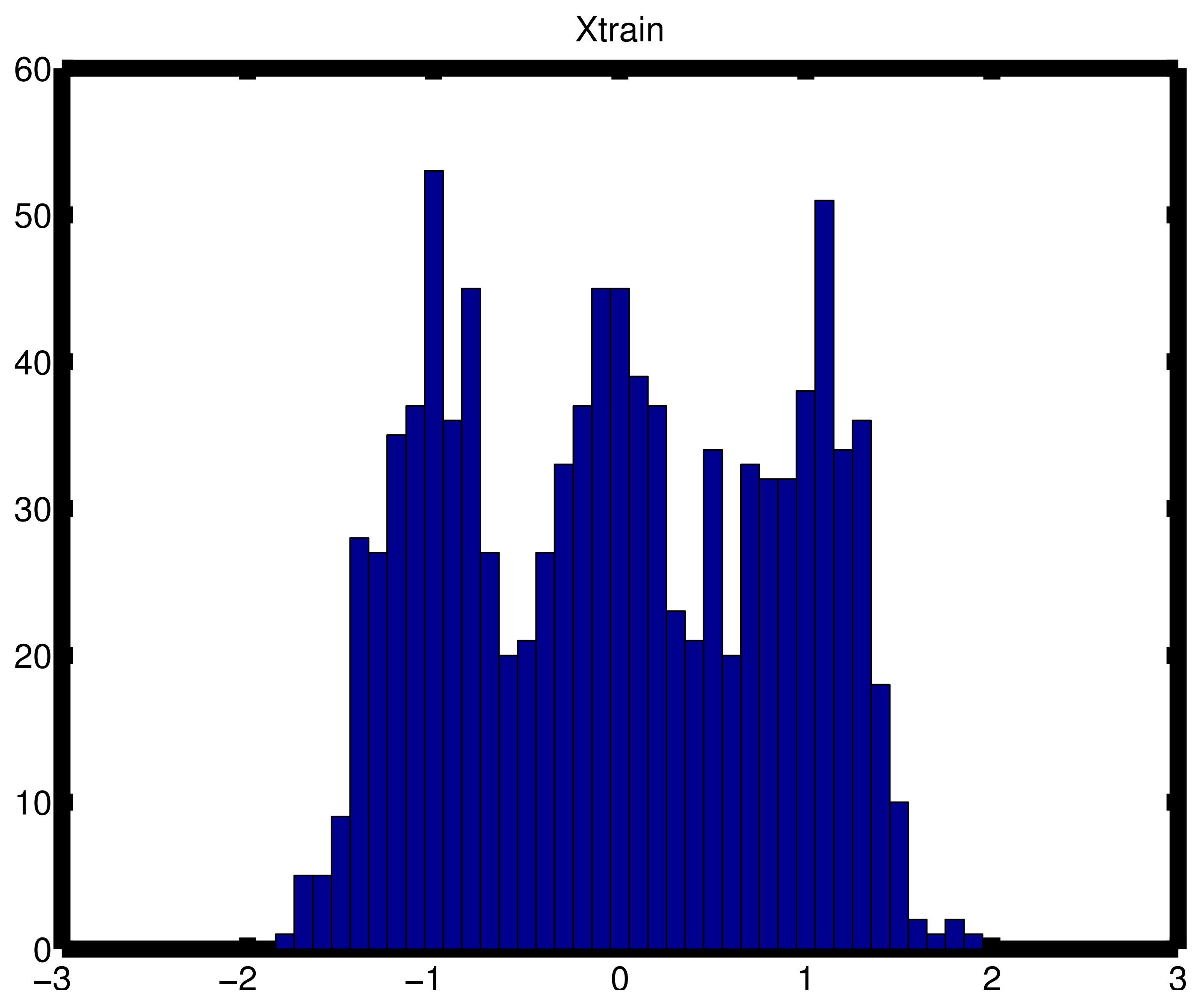


EM Design Decision

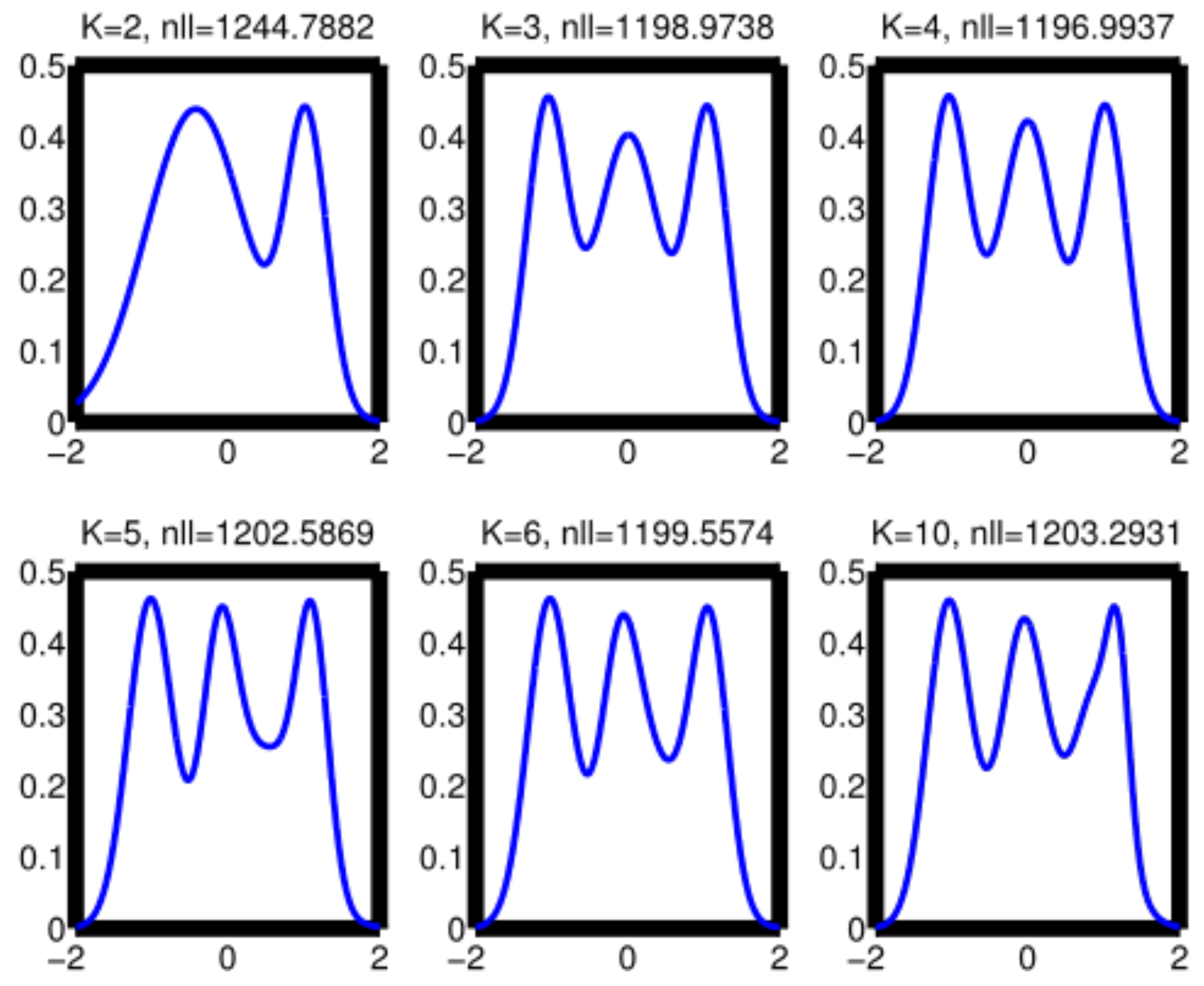
How do you determine how many Gaussians to use?

- Simple approach:
  - Try different number of Gaussians and pick the one with the largest (log) likelihood (or smallest negative log likelihood)
    - Pros: Easy to do
    - Cons: slow and time-consuming
  - Alternatively, use the **elbow method** *as before for K-Means*

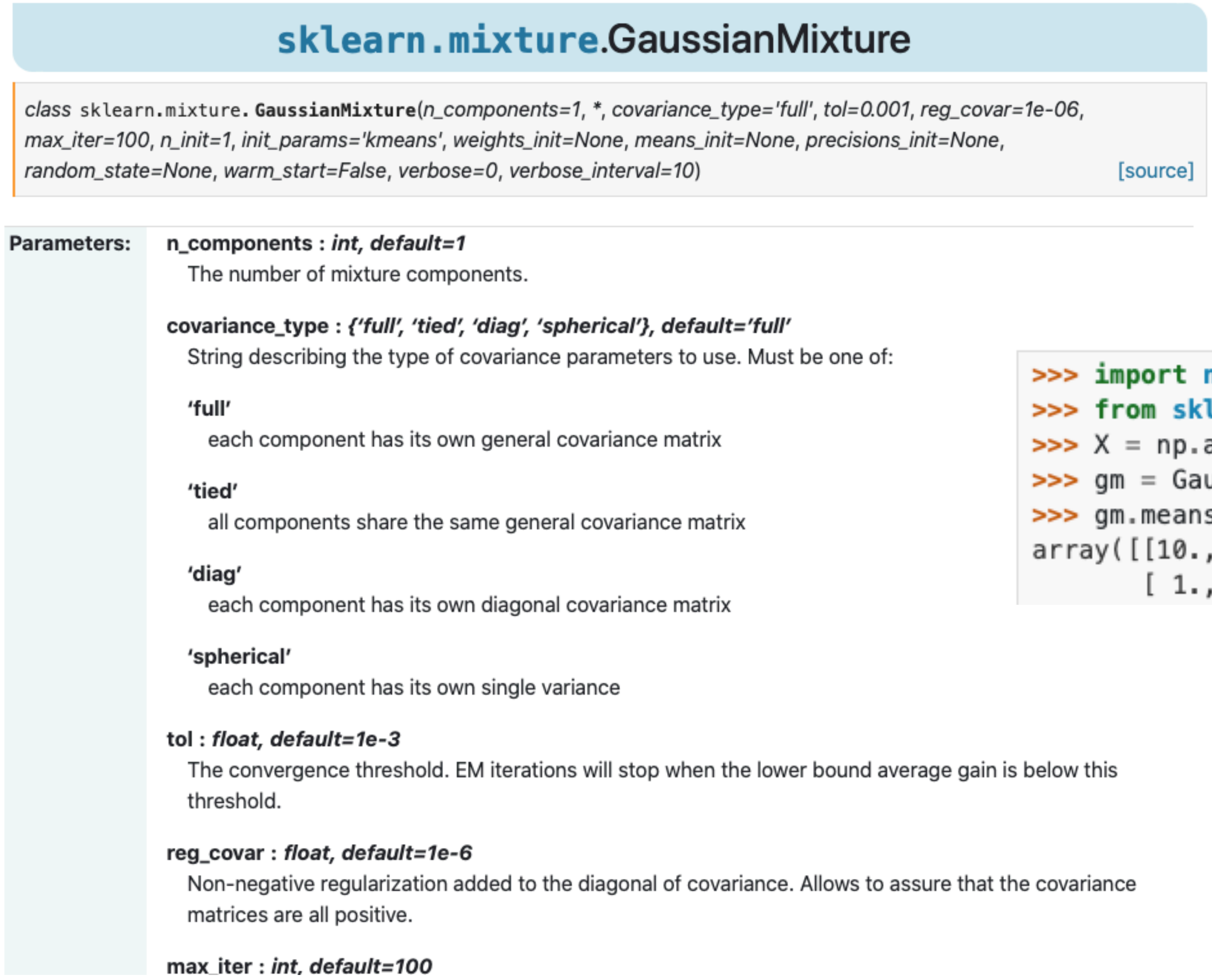
True "Distribution"



Modeled "Distributions"



Gaussian Mixture Model in Python



sklearn.mixture.GaussianMixture

class `sklearn.mixture.`**`GaussianMixture`**(*n_components=1, *, 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, random_state=None, warm_start=False, verbose=0, verbose_interval=10*) [source]

Parameters:

n_components : *int, default=1*
The number of mixture components.

covariance_type : *{'full', 'tied', 'diag', 'spherical'}, default='full'*
String describing the type of covariance parameters to use. Must be one of:

'full'
each component has its own general covariance matrix

'tied'
all components share the same general covariance matrix

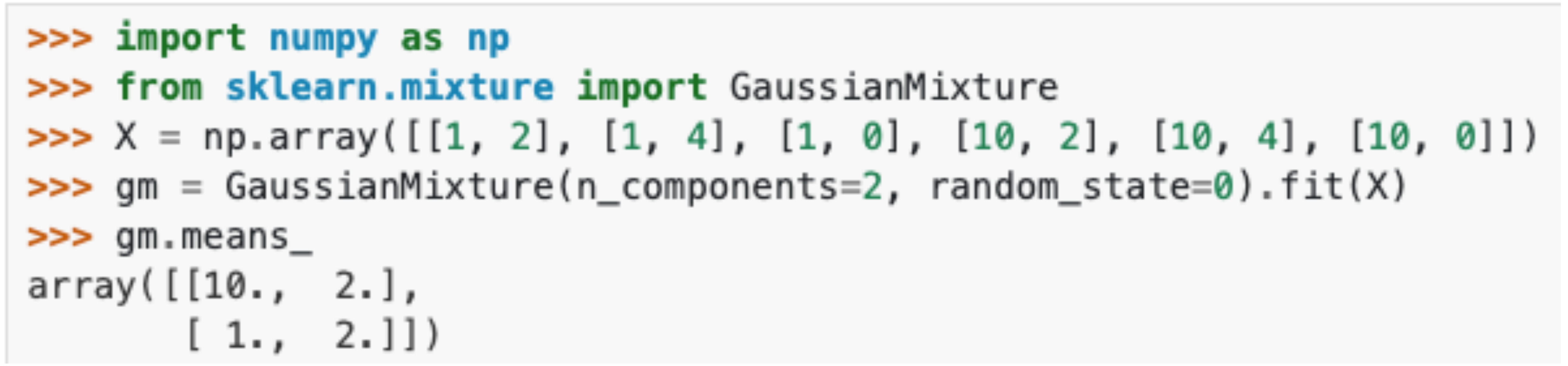
'diag'
each component has its own diagonal covariance matrix

'spherical'
each component has its own single variance

tol : *float, default=1e-3*
The convergence threshold. EM iterations will stop when the lower bound average gain is below this threshold.

reg_covar : *float, default=1e-6*
Non-negative regularization added to the diagonal of covariance. Allows to assure that the covariance matrices are all positive.

max_iter : *int, default=100*



```
>>> import numpy as np
>>> from sklearn.mixture import GaussianMixture
>>> X = np.array([[1, 2], [1, 4], [1, 0], [10, 2], [10, 4], [10, 0]])
>>> gm = GaussianMixture(n_components=2, random_state=0).fit(X)
>>> gm.means_
array([[10.,  2.],
       [ 1.,  2.]])
```

Next Class: Unsupervised Learning

Dimensionality Reduction