19ECCN1702 - Machine Learning

UNIT 4-UNSUPERVISED LEARNING

Unit I INTRODUCTION **9 Hours**

Introduction to Machine Learning - Types of Machine Learning systems - Challenges in Machine Learning - Overfitting and Under fitting - Testing and Validating the model - Bias and Variance

Unit II MACHINE LEARNING FRAMEWORK **9 Hours**

Problem Formulation - Get the data - analyze and visualize the data - Prepare the data for ML algorithms - sample complexity - Hypothesis space - Model evaluation and Improvement: Cross validation - Grid search - Evaluation Metrics - Kernel functions

Unit III SUPERVISED LEARNING **9 Hours**

Linear and Logistic Regression – Eigen Values and Eigen vectors - Naïve Bayes Classifier: Maximum Likelihood, Minimum Description Length – Gradient Descent - Decision Trees - Ensembles of Decision Trees - Support Vector Machine(SVM)

Unit IV UNSUPERVISED LEARNING 9 Hours

Clustering: k-Means clustering- Agglomerative Clustering - DBSCAN- Gaussian Mixtures- precision and recall - Collaborative filtering and Content Filtering

Unit V NEURAL NETWORK AND DEEP LEARNING 9 Hours

Biological Neuron - Logical computation with Neuron - Perceptron - Sigmoid and softmax functions - Multi Layer Perceptron(MLP) with Back propagation - Regression MLPs - Classification MLPs - Fine Tuning NN models - Convolutional Neural Network: Architecture of Visual cortex - Convolutional Layers - Stacking Multiple Feature Maps- CNN architectures

Course Outcomes	**Cognitive Level**
At the end of this course, students will be able to:	
CO1:Describe the types and challenges in Machine learning for exploring the machine learning concepts	Understand
CO2:Illustrate the machine learning framework for implementation of machine learning projects	Apply
CO3:Interpret the supervised learning techniques for classification	Apply
CO4:Demonstrate the un-supervised learning methods for clustering and classification	Apply
CO5:Construct the Neural network and deep learning models for classification	Apply

Text Book(s):

T1. AurélienGéron," Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow", Second edition, O'Reilly Media, Inc,2019

T2. Andreas C. Müller and Sarah Guido, "Introduction to Machine Learning with Python A Guide for Data Scientists", First Edition,O'Reilly,2017

Reference Book(s):

R1. Ethem Alpaydin, "Introduction to Machine Learning 3e (Adaptive Computation and Machine Learning Series)", 3rd Edition, MIT Press, 2014

R2. Jason Bell, "Machine learning - Hands on for Developers and Technical Professionals",1st Edition, Wiley, 2014

R3. Peter Flach, "Machine Learning: The Art and Science of Algorithms that Make Sense of Data", 1st Edition, Cambridge University Press, 2012.

Web References:

1. https://www.kaggle.com/kanncaa1/machine-learning-tutorial-for-beginners
2. https://nptel.ac.in/courses/106/106/106106139/
3. https://archive.ics.uci.edu/ml/datasets.php

Topics

- K means clustering
- Agglomerative clustering
- DBSCAN
- Gaussian Mixtures
- Precision and Recall
- Collaborative filtering and Content filtering

Gaussian Mixtures

A Gaussian Mixture Model (GMM) is a probabilistic model that assumes that the instances were generated from a mixture of several Gaussian distributions whose parameters are unknown.

Covariance

Population Covariance Formula

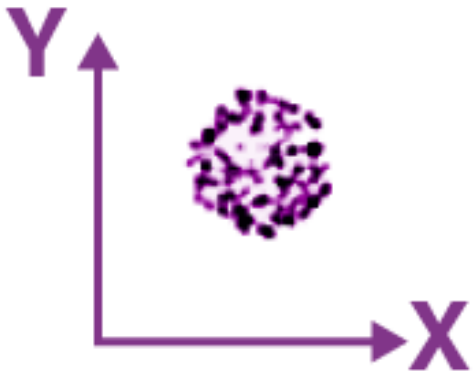
$$\mathbf{Cov(x,y) = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{N}}$$

Sample Covariance

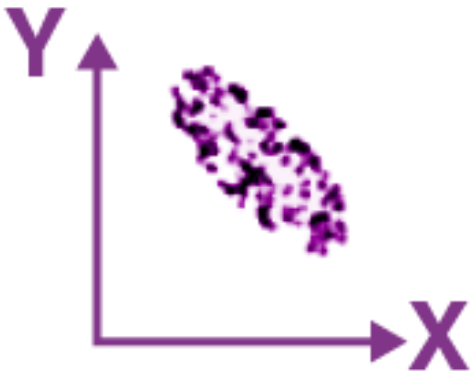
$$\mathbf{Cov(x,y) = \frac{\sum(x_i - \bar{x})(y_i - y)}{N - 1}}$$

Covariance

cov(X,Y)>0



cov(X,Y)≈0

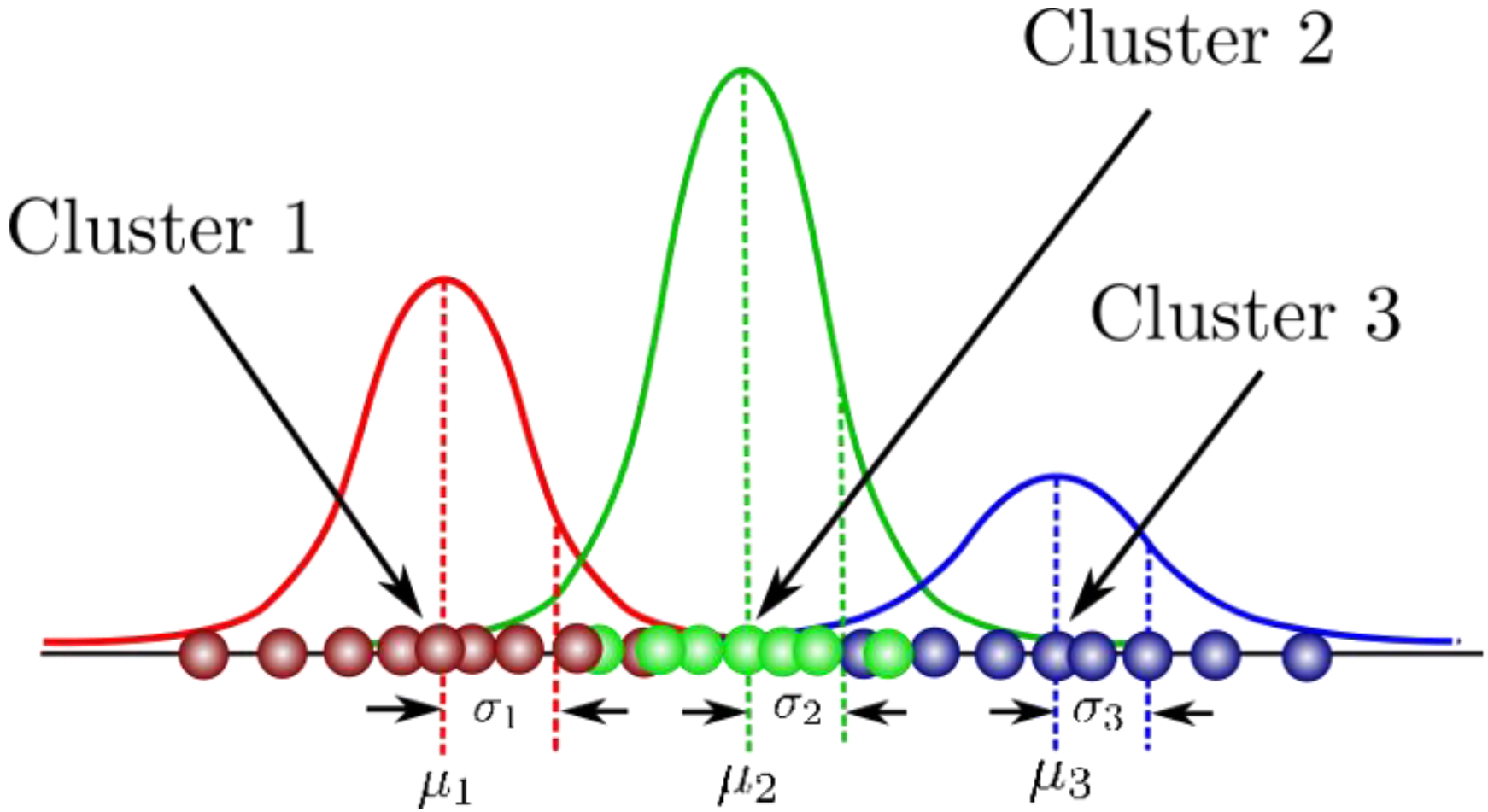


cov(X,Y)<0

Covariance

- If cov(X, Y) is greater than zero, then we can say that the covariance for any two variables is positive and both the variables move in the same direction.
- If cov(X, Y) is less than zero, then we can say that the covariance for any two variables is negative and both the variables move in the opposite direction.
- If cov(X, Y) is zero, then we can say that there is no relation between two variables.

Gaussian Functions



$$\sum_{k=1}^{K} \pi_k = 1 \qquad (1)$$

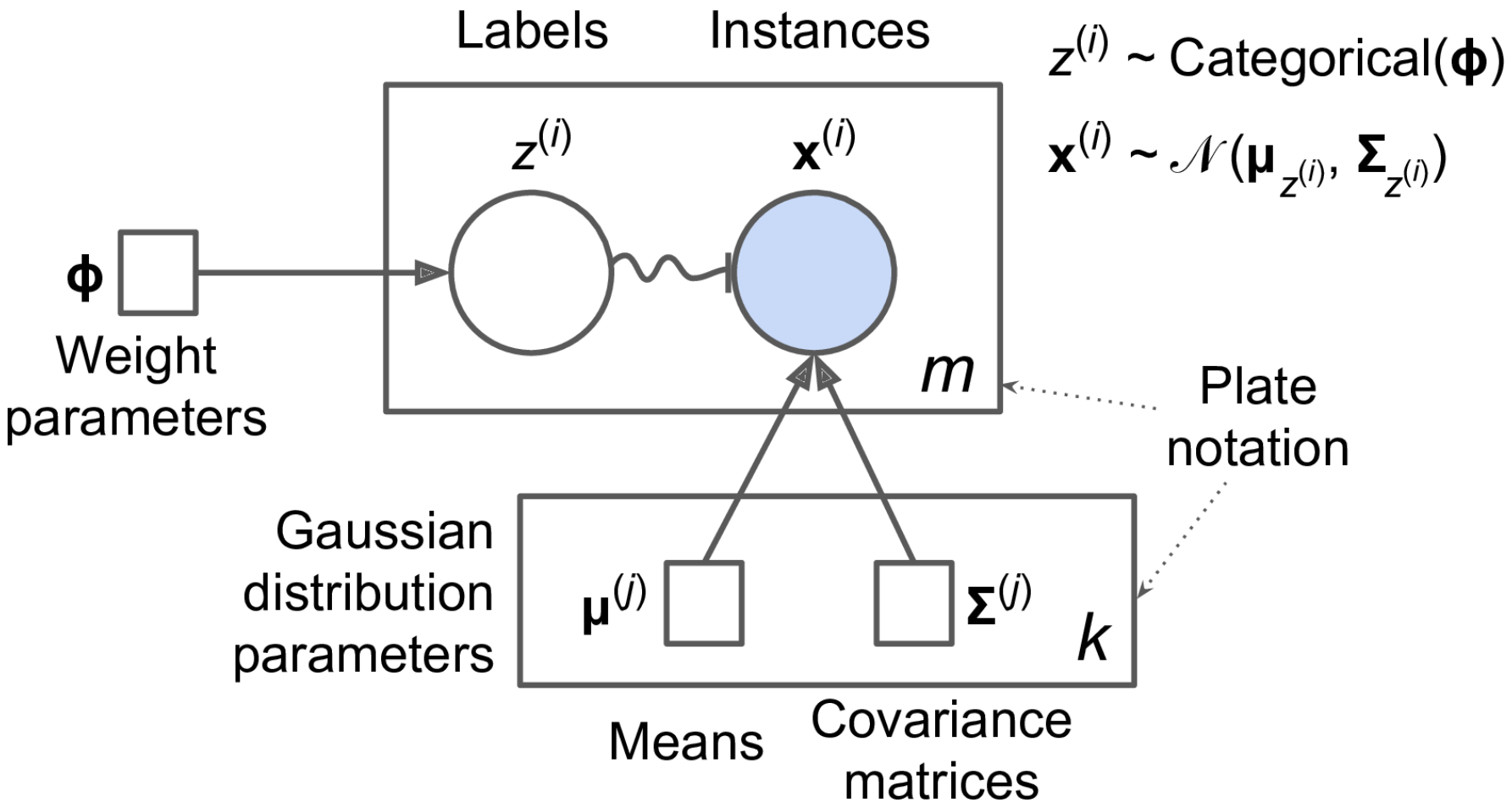
$K = 3$

- A mean μ that defines its centre.
- A covariance Σ that defines its width. This would be equivalent to the dimensions of an ellipsoid in a multivariate scenario.
- A mixing probability π that defines how big or small the Gaussian function will be.

- Gaussian PDF:

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

Gaussian Mixture Model



Gaussian Mixture Model

- For each instance, a cluster is picked randomly among k clusters. The probability of choosing the j th cluster is defined by the cluster's weight $\phi(j)$. The index of the cluster chosen for the i th instance is noted $z(i)$.
- If $z(i)=j$, meaning the ith instance has been assigned to the jth cluster, the location $\mathbf{x}(i)$ of this instance is sampled randomly from the Gaussian distribution with mean $\boldsymbol{\mu}(j)$ and covariance matrix $\boldsymbol{\Sigma}(j)$. This is noted

$$\mathbf{x}^{(i)} \sim \mathcal{N}\left(\mu^{(j)}, \Sigma^{(j)}\right).$$

Interpretation

The circles represent random variables.

The squares represent fixed values (i.e., parameters of the model).

The number indicated at the bottom right hand side of each plate indicates how many times its content is repeated, so there are m random variables $z^{(i)}$ (from $z^{(1)}$ to $z^{(m)}$) and m random variables $\mathbf{x}^{(i)}$, and k means $\boldsymbol{\mu}^{(j)}$ and k covariance matrices $\boldsymbol{\Sigma}^{(j)}$, but just one weight vector $\boldsymbol{\phi}$ (containing all the weights $\phi^{(1)}$ to $\phi^{(k)}$).

Interpretation

The solid arrows represent conditional dependencies. For example, the probability distribution for each random variable $z^{(i)}$ depends on the weight vector $\boldsymbol{\phi}$. Note that when an arrow crosses a plate boundary, it means that it applies to all the repetitions of that plate, so for example the weight vector $\boldsymbol{\phi}$ conditions the probability distributions of all the random variables $\mathbf{x}^{(1)}$ to $\mathbf{x}^{(m)}$.

The squiggly arrow from $z^{(i)}$ to $\mathbf{x}^{(i)}$ represents a switch: depending on the value of $z^{(i)}$, the instance $\mathbf{x}^{(i)}$ will be sampled from a different Gaussian distribution. For example, if $z^{(i)}=j$, then $\mathbf{x}^{(i)} \sim \mathcal{N}\left(\mu^{(j)}, \Sigma^{(j)}\right)$.

Shaded nodes indicate that the value is known, so in this case only the random variables $\mathbf{x}^{(i)}$ have known values: they are called *observed variables*. The unknown random variables $z^{(i)}$ are called *latent variables*.

Scikit-Learn

```
from sklearn.mixture import GaussianMixture

gm = GaussianMixture(n_components=3, n_init=10)
gm.fit(X)
```

```
>>> gm.weights_
array([0.20965228, 0.4000662 , 0.39028152])
>>> gm.means_
array([[ 3.39909717,  1.05933727],
       [-1.40763984,  1.42710194],
       [ 0.05135313,  0.07524095]])
>>> gm.covariances_
array([[[ 1.14807234, -0.03270354],
        [-0.03270354,  0.95496237]],

       [[ 0.63478101,  0.72969804],
        [ 0.72969804,  1.1609872 ]],

       [[ 0.68809572,  0.79608475],
        [ 0.79608475,  1.21234145]]])
```

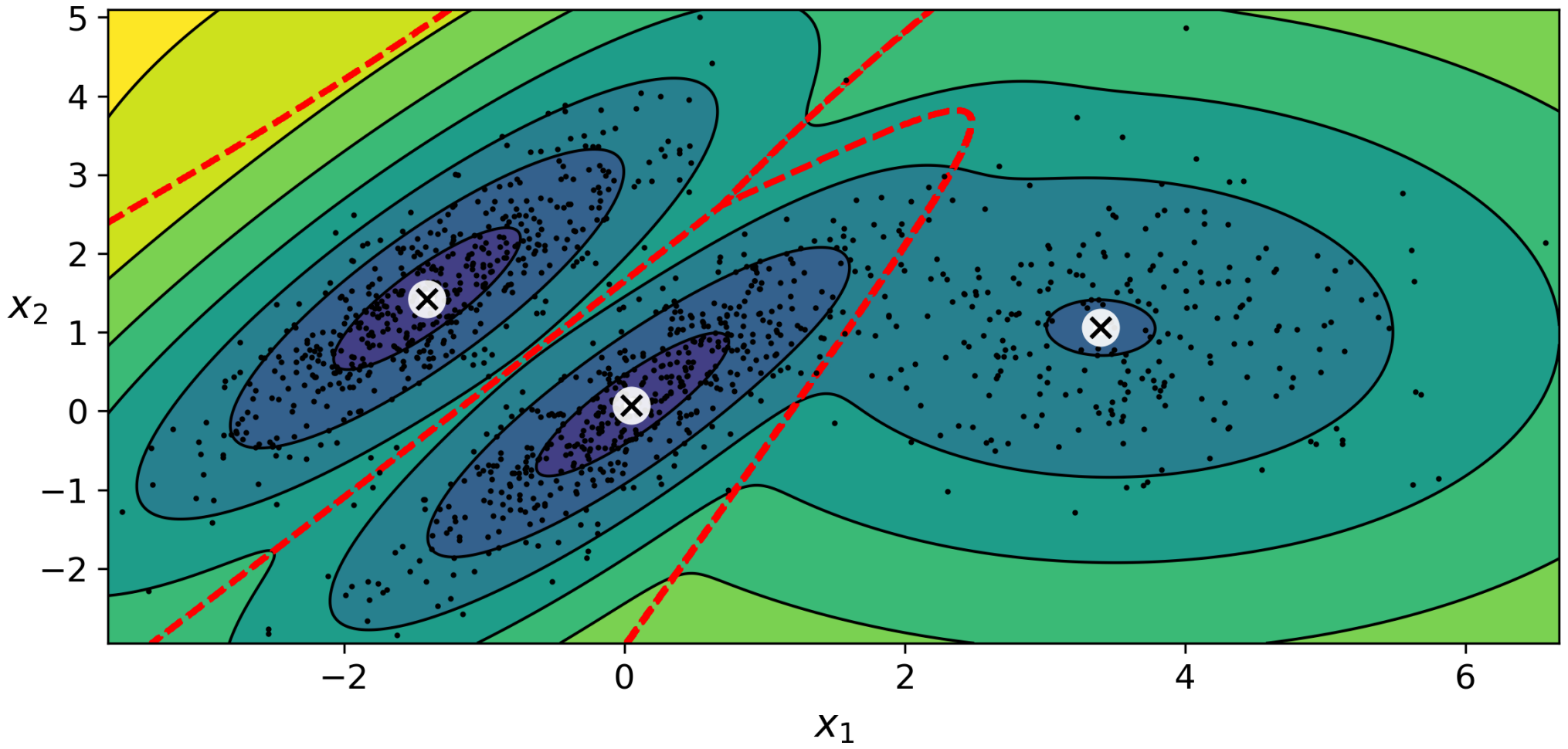
Expectation Maximization (EM) algorithm

- It initializes the cluster parameters randomly, then it repeats two steps until convergence, first assigning instances to clusters (this is called the *expectation step*) then updating the clusters (this is called the *maximization step*).

Indeed, in the context of clustering you can think of EM as a generalization of K-Means which not only finds the cluster centers ($\boldsymbol{\mu}^{(1)}$ to $\boldsymbol{\mu}^{(k)}$), but also their size, shape and orientation ($\boldsymbol{\Sigma}^{(1)}$ to $\boldsymbol{\Sigma}^{(k)}$), as well as their relative weights ($\phi^{(1)}$ to $\phi^{(k)}$).

Soft Cluster Assignments

- For each instance during the expectation step, the algorithm estimates the probability that it belongs to each cluster (based on the current cluster parameters).
- Then, during the maximization step, each cluster is updated using *all* the instances in the dataset, with each instance weighted by the estimated probability that it belongs to that cluster.
- These probabilities are called the *responsibilities* of the clusters for the instances. During the maximization step, each cluster's update will mostly be impacted by the instances it is most responsible for.



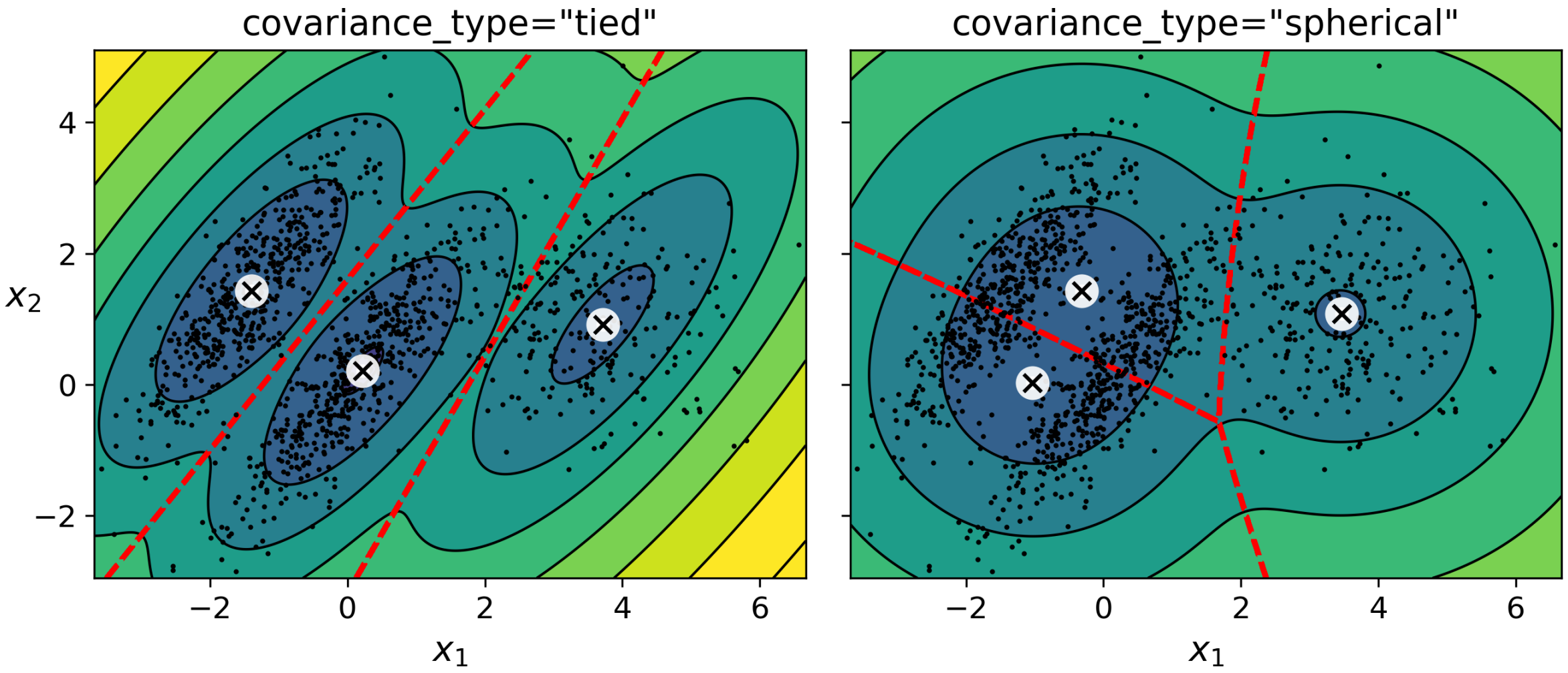
Cluster means, decision boundaries and density contours of a trained Gaussian mixture model

- When there are many dimensions, or many clusters, or few instances, EM can struggle to converge to the optimal solution.
- You might need to reduce the difficulty of the task by limiting the number of parameters that the algorithm has to learn:.
- one way to do this is to limit the range of shapes and orientations that the clusters can have.
- This can be achieved by imposing constraints on the covariance matrices.

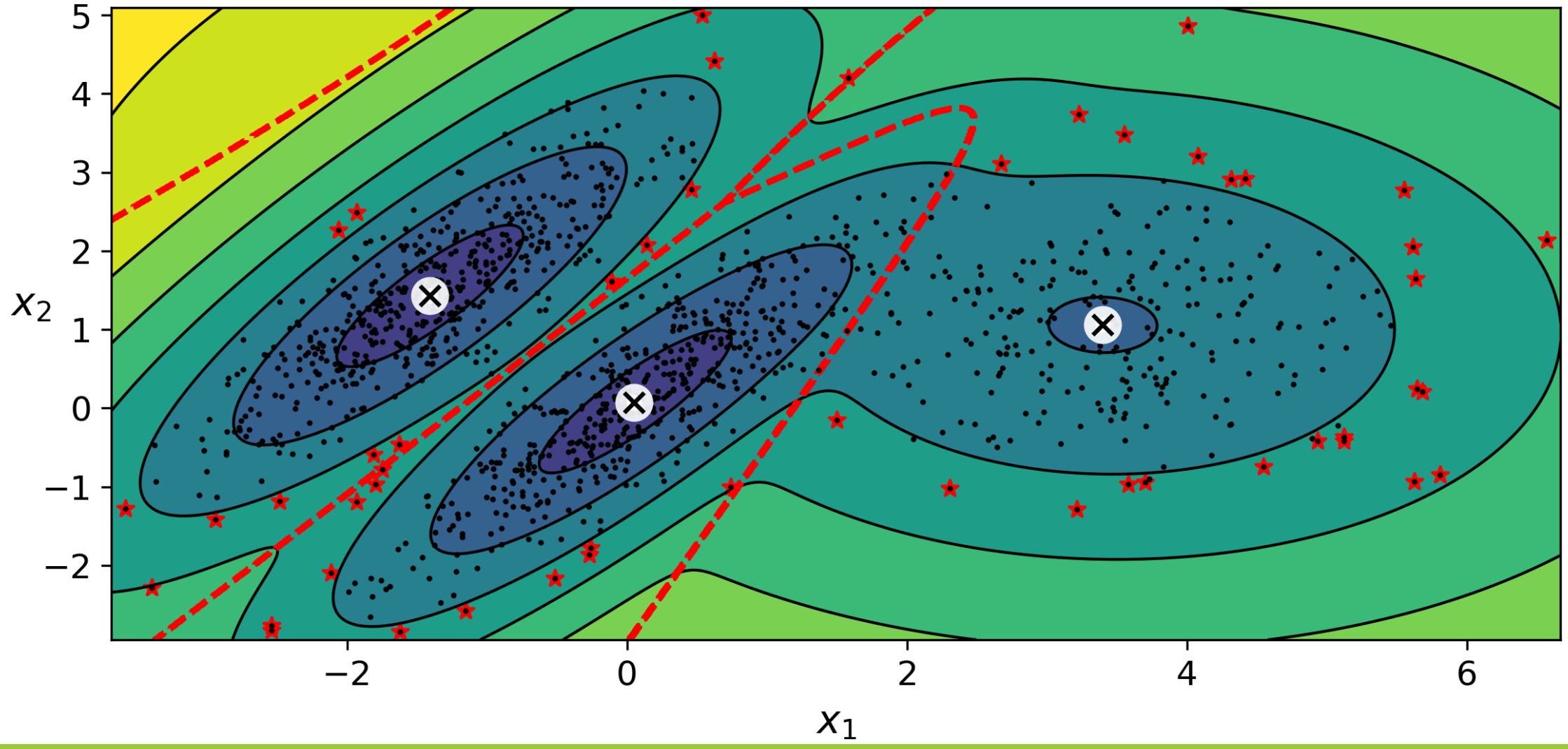
To do this, just set the *covariance type hyper parameter* to one of the following values:

- `"spherical"`: all clusters must be spherical, but they can have different diameters (i.e., different variances).
- `"diag"`: clusters can take on any ellipsoidal shape of any size, but the ellipsoid's axes must be parallel to the coordinate axes (i.e., the covariance matrices must be diagonal).
- `"tied"`: all clusters must have the same ellipsoidal shape, size and orientation (i.e., all clusters share the same covariance matrix).

covariance_type_diagram



Anomaly Detection using Gaussian Mixtures



Anomaly Detection

Anomaly detection (also called *outlier detection*) is the task of detecting instances that deviate strongly from the norm.

These instances are of course called *anomalies* or *outliers*, while the normal instances are called *inliers*.

Anomaly detection is very useful in a wide variety of applications, for example in fraud detection, or for detecting defective products in manufacturing, or to remove outliers from a dataset before training another model, which can significantly improve the performance of the resulting model