**Expectation Maximization και Gaussian**

**Mixture Models**

**1 Εισαγωγή**

**1.1 Εισαγωγή στον αλγόριθμο Expectation Maximization**

Ο αλγόριθμος Expectation Maximization (EM) είναι μια επαναληπτική διαδικασία για την μοντελοποίηση μιας κατανομής η οποία έχει λανθάνουσες μεταβλητές ώστε να μεγιστοποιείται η πιθανοφάνεια (maximum likelihood estimate). Για δεδομένα *X* και λανθάνουσες *Z* και για ένα σύνολο παραμέτρων , ο αλγόριθμος μεγιστοποιεί την ποσότητα:

Η βελτιστοποίηση γίνεται σε δύο βήματα. Στο expectation step (E-step) χρησιμοποιούμε τις τιμές των παραμέτρων ώστε να υπολογίσουμε την εκ των υστέρων κατανομή των λανθάνουσων . Στο maximization step (M-step) υπολογίζουμε τη νέα τιμή του θ () δεδομένης της κατανομής αυτής . Ο αλγόριθμος συγκλίνει όταν η λογαριθμική πιθανοφάνεια σταματήσει να μεταβάλλεται (ή μεταβάλλεται πολύ λίγο).

**1.2 Εισαγωγή στα Gaussian Mixture Models**

Στη γενική του μορφή ένα mixture από *k* κανονικές κατανομές με μέσους , συμμεταβλητότητες και mixing coefficients (που ορίζουν τη συμμετοχή κάθε κατανομής στο mixture) ορίζεται με την εξίσωση:

Όπου τα mixing coefficients ορίζονται στο διάστημα [0, 1] και το άθροισμά τους είναι ίσο με τη μονάδα και είναι η multivariate κανονική κατανομή:

**1.3 Εισαγωγή στον αλγόριθμο EM για Gaussian Mixture Models**

Για την μοντελοποίηση σε Gaussian Mixture Models με τον αλγόριθμο EM οι παράμετροι του μοντέλου () θα είναι οι μέσοι και οι συμμεταβλητότητες των κατανομών (*μ*, *Σ*) καθώς και τα mixing coefficients (*π*). Ο αλγόριθμος θα μεγιστοποιήσει την ποσότητα:

Όπου x1, x2, … xN είναι οι τιμές των δεδομένων. Σε αυτήν την περίπτωση στο E-step υπολογίζεται για κάθε σημείο η πιθανότητα να ανήκει σε κάθε μία από τις *K* κατανομές. Στη συνέχεια, με βάση αυτές τις πιθανότητες, στο M-step υπολογίζονται οι νέες τιμές των παραμέτρων *π*, *μ*, *Σ*.

**1.4 Expectation Maximization και Gaussian Mixtures στην Python**

Για να χρησιμοποιήσουμε τον αλγόριθμο EM στην Python χρησιμοποιούμε τη βιβλιοθήκη GaussianMixture. Εφαρμόζουμε τον αλγόριθμο EM για Gaussian Mixture Models σε 1 διάσταση με τις εντολές:

>>> data = np.array(x.tolist()).reshape(-1,1)

>>> gm = GaussianMixture(n\_components=2, tol=0.001).fit(data)

Όπου με τις παραμέτρους n\_components και tol μπορούμε να επιλέξουμε τον αριθμό των κατανομών καθώς και τη μεταβολή της λογαριθμικής πιθανοφάνειας ώστε να συγκλίνει ο αλγόριθμος.

Για Gaussian Mixture Models σε 2 ή περισσότερες διαστάσεις χρησιμοποιούμε την εντολή:

>>> gm = GaussianMixture(n\_components=2, tol=0.001).fit(data)

Μπορούμε να δούμε τις τελικές παραμέτρους του μοντέλου με τις εντολές gm.means\_, gm.covariances\_ καθώς και την τελική λογαριθμική πιθανοφάνεια με την εντολή np.sum(gm.score\_samples(data)).

Για να σχεδιάσουμε τη συνάρτηση πυκνότητας πιθανότητας που προσέγγισε το μοντέλο εκτελούμε τις εντολές:

>>> pi, mu, sigma = gm.weights\_.flatten(), gm.means\_.flatten(), np.sqrt(gm.covariances\_.flatten())

>>> grid = np.arange(np.min(x), np.max(x), 0.01)

>>> plt.plot(grid, mix\_pdf(grid, mu, sigma, pi), label="GMM")

όπου:

>>> def mix\_pdf(x, loc, scale, weights):

>>> d = np.zeros\_like(x)

>>> for mu, sigma, pi in zip(loc, scale, weights):

>>> d += pi \* norm.pdf(x, loc = mu, scale = sigma)

>>> return d

Για το hard-assignment των clusters και τα κέντρα ως vectors χρησιμοποιούμε τις εντολές:

>>> clusters = gm.predict(gsdata)

>>> centers = gm.means\_

**2 Κατασκευή Gaussian Mixtures με Expectation Maximization**

Για την ομαδοποίηση δεδομένων με Gaussian Mixtures θα χρησιμοποιήσουμε ως εφαρμογή τα δεδομένα του αρχείου που δίνεται (gdata.txt) για ένα πρόβλημα ομαδοποίησης. Τα δεδομένα προέρχονται από δύο κανονικές κατανομές. Γνωρίζουμε ότι η τυπική απόκλιση των δύο κατανομών είναι ίση με 1. Ένα summary για τις δύο κατανομές είναι το παρακάτω:

D1\_X D2\_X

Min. :-1.9049 Min. : 3.992

1st Qu.: 0.2232 1st Qu.: 6.322

Median : 0.9065 Median : 6.944

Mean : 1.0001 Mean : 6.985

3rd Qu.: 1.6641 3rd Qu.: 7.681

Max. : 3.6587 Max. :10.810

Αρχικά, εισάγουμε τα δεδομένα:

>>> import pandas as pd

>>> gdata = pd.read\_csv("./gdata.txt")

>>> x = gdata.loc[:, "X"]

>>> y = gdata.loc[:, "Y"]

Στη συνέχεια, θα απαντήσουμε στα παρακάτω ερωτήματα:

α) Σχεδιάστε τα δεδομένα σε 1 διάσταση και σχεδιάστε επίσης τη συνάρτηση πυκνότητας πιθανότητάς τους με διαφορετικά χρώματα για κάθε κατανομή.

β) Υπολογίστε τις παραμέτρους (μέσες τιμές) μ1, μ2 των κανονικών κατανομών καθώς και τις λανθάνουσες μεταβλητές λ1, λ2 από τις οποίες προήλθαν τα δεδομένα. Εφαρμόστε τα βήματα του αλγορίθμου στην Python.

γ) Επαναλάβετε το ερώτημα (β) χρησιμοποιώντας τη βιβλιοθήκη GaussianMixture.

δ) Σχεδιάστε στο ίδιο διάγραμμα τη συνάρτηση πυκνότητας πιθανότητας που προσεγγίσατε καθώς και την πραγματική συνάρτηση πυκνότητας πιθανότητας των δεδομένων.

**2.1 Σχεδίαση δεδομένων**

Σχεδιάζουμε τα δεδομένα σε 1 διάσταση και τη συνάρτηση πυκνότητας πιθανότητάς τους με διαφορετικά χρώματα για κάθε κατανομή (ερώτημα (α)) με τις παρακάτω εντολές:

>>> import matplotlib.pyplot as plt

>>> import numpy as np

>>> import seaborn as sns

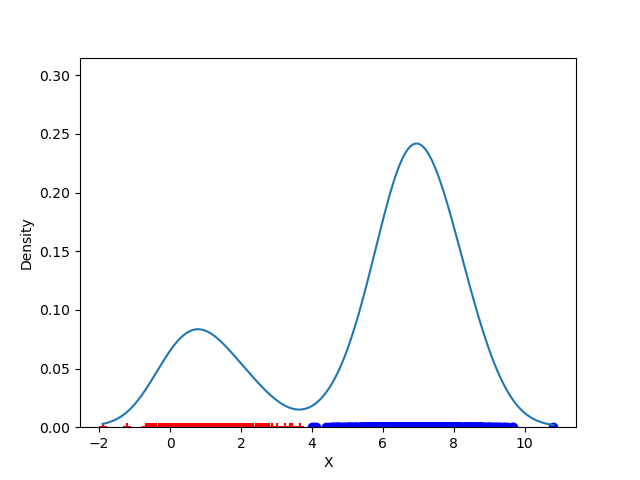
>>> plt.scatter(x[(y == 1)], np.zeros(len(x[(y == 1)].tolist())), c="red", marker="+")

>>> plt.scatter(x[(y == 2)], np.zeros(len(x[(y == 2)].tolist())), c="blue", marker="o")

>>> sns.histplot(x, kde=True, stat="density", linewidth=0, fill=False)

>>> plt.show()

Τα δεδομένα φαίνονται στο σχήμα:



**2.2 Υπολογισμός Παραμέτρων Κατανομών**

Θα χρησιμοποιήσουμε τον αλγόριθμο ΕΜ. Αρχικά επιλέγουμε τυχαίες αρχικές τιμές των μ1, μ2 έστω και και των λ1, λ2 έστω και . Έτσι, το αρχικό σετ παραμέτρων θα είναι .

*Expectation Step*

Για το expectation step του αλγορίθμου θέλουμε να υπολογίσουμε την πιθανότητα ότι ένα σημείο προήλθε από μία συγκεκριμένη κατανομή. Δηλαδή:

όπου και αρχικά .

Αν για παράδειγμα θεωρήσουμε ότι ένα σημείο που μας δίνεται είναι το , από τη συνάρτηση κανονικής κατανομής και για τα θ1, θ2 προκύπτει:

Στη συνέχεια οι πιθανότητες το σημείο να προήλθε από τις κατανομές 1 και 2 υπολογίζεται:

Τα παραπάνω σημαίνουν ότι το σημείο 2 έχει πιθανότητα 0.182 να ανήκει στην πρώτη κατανομή (D1) και πιθανότητα 0.818 να ανήκει στη δεύτερη κατανομή (D2).

Αφού υπολογίσουμε όλες τις πιθανότητες και για τα 1000 σημεία θα προχωρήσουμε στο maximization step του ΕΜ αλγορίθμου.

*Maximization Step*

Στο maximization step του αλγορίθμου θα αναζητήσουμε εκείνες τις παραμέτρους μ1, μ2 (τα σ1, σ2 είναι σταθερά ίσα με 1 από εκφώνηση) οι οποίες μεγιστοποιούν την πιθανότητα. Για τον υπολογισμό των παραμέτρων μ1, μ2 θα χρησιμοποιήσουμε τους παρακάτω τύπους:

Οι παραπάνω σχέσεις ουσιαστικά αποτελούν έναν σταθμισμένο μέσο όρο όλων των σημείων, όπου τα βάρη ισούνται με τις πιθανότητες το σημείο να ανήκει σε μία από τις δύο κατανομές.

Ακόμα, για τον υπολογισμό των mixing coefficients που μεγιστοποιούν τη συνάρτηση πιθανότητας θα χρησιμοποιήσουμε τους τύπους:

Αφού υπολογίσουμε τις νέες τιμές των παραμέτρων μ1, μ2, λ1, λ2, υπολογίζουμε τη λογαριθμική πιθανοφάνεια (log likelihood) των δεδομένων:

Συγκρίνουμε την τιμή που υπολογίσαμε με την προηγούμενη τιμή τους. Αν δεν έχει αλλάξει καθόλου ή έχει αλλάξει πολύ λίγο τότε ο αλγόριθμος έχει συγκλίνει και οι ζητούμενες παράμετροι των κατανομών είναι αυτές που υπολογίστηκαν. Διαφορετικά, επιστρέφουμε στο expectation step και στο maximization step ώστε να υπολογίσουμε εκ νέου τις πιθανότητες και τη συνέχεια μια νέα τιμή του Θ.

**2.3 Εφαρμογή αλγορίθμου στην Python**

Ο αλγόριθμος στην Python για το πρόβλημα που δίνεται (ερώτημα (β)) φαίνεται παρακάτω:

# Initialize the means and the latent variables

>>> from scipy.stats import norm

>>> import math

>>> mu = [0, 1]

>>> lamda = [0.5, 0.5]

>>> epsilon = 1e-08

>>> log\_likelihood = np.sum([math.log(i) for i in lamda[0] \* norm.pdf(x, loc=mu[0], scale=1) + lamda[1] \* norm.pdf(x, loc=mu[1], scale=1)])

# Loop until convergence

>>> while True:

>>> # Expectation step

>>> # Find distributions given mu, lambda (and sigma)

>>> T1 = norm.pdf(x, loc=mu[0], scale=1)

>>> T2 = norm.pdf(x, loc=mu[1], scale=1)

>>> P1 = lamda[0] \* T1 / (lamda[0] \* T1 + lamda[1] \* T2)

>>> P2 = lamda[1] \* T2 / (lamda[0] \* T1 + lamda[1] \* T2)

>>> # Maximization step

>>> # Find mu, lambda (and sigma) given the distributions

>>> mu[0] = np.sum(P1 \* x) / np.sum(P1)

>>> mu[1] = np.sum(P2 \* x) / np.sum(P2)

>>> lamda[0] = np.mean(P1)

>>> lamda[1] = np.mean(P2)

>>> # Calculate the new log likelihood (to be maximized)

>>> new\_log\_likelihood = np.sum([math.log(i) for i in lamda[0] \* norm.pdf(x, loc=mu[0], scale=1) + lamda[1] \* norm.pdf(x, loc=mu[1], scale=1)])

>>> # Print the current parameters and the log likelihood

>>> print("mu=", mu, " lambda=", lamda, " log\_likelihood=", new\_log\_likelihood)

>>> # Break if the algorithm converges

>>> if ((new\_log\_likelihood - log\_likelihood) <= epsilon): break

>>> log\_likelihood = new\_log\_likelihood

**2.4 Κατασκευή Μοντέλου με την Python**

Για το ερώτημα (γ) εισάγουμε αρχικά τη βιβλιοθήκη GaussianMixture:

>>> from sklearn.mixture import GaussianMixture

Εφαρμόζουμε τον αλγόριθμο expectation maximization για Gaussian Mixture Models σε 1 διάσταση με την εντολή:

>>> data = np.array(x.tolist()).reshape(-1,1)

>>> gm = GaussianMixture(n\_components=2).fit(data)

Μπορούμε να δούμε τις τελικές παραμέτρους του μοντέλου και την τελική λογαριθμική πιθανοφάνεια με τις εντολές:

>>> print(gm.means\_)

>>> print(gm.covariances\_)

>>> print(np.sum(gm.score\_samples(data)))

Τέλος σχεδιάζουμε στο ίδιο διάγραμμα τη συνάρτηση πυκνότητας πιθανότητας που προσέγγισε το μοντέλο με την πραγματική συνάρτηση πυκνότητας πιθανότητας των δεδομένων (ερώτημα (δ)) με τις εντολές:

>>> def mix\_pdf(x, loc, scale, weights):

>>> d = np.zeros\_like(x)

>>> for mu, sigma, pi in zip(loc, scale, weights):

>>> d += pi \* norm.pdf(x, loc = mu, scale = sigma)

>>> return d

>>> pi, mu, sigma = gm.weights\_.flatten(), gm.means\_.flatten(), np.sqrt(gm.covariances\_.flatten())

>>> grid = np.arange(np.min(x), np.max(x), 0.01)

>>> plt.scatter(x[(y == 1)], np.zeros(len(x[(y == 1)].tolist())), c="red", marker="+")

>>> plt.scatter(x[(y == 2)], np.zeros(len(x[(y == 2)].tolist())), c="blue", marker="o")

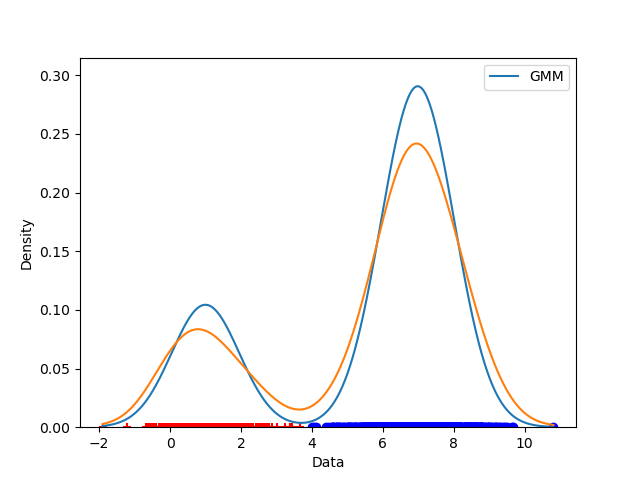
>>> plt.plot(grid, mix\_pdf(grid, mu, sigma, pi), label="GMM")

>>> sns.histplot(x, kde=True, stat="density", linewidth=0, fill=False)

>>> plt.legend(loc = 'upper right')

>>> plt.show()

Οπότε προκύπτει το παρακάτω διάγραμμα:



**3 Εφαρμογή Clustering με Gaussian Mixtures**

Για την κατασκευή ενός μοντέλου Gaussian Mixture Models θα χρησιμοποιήσουμε ως εφαρμογή τα δεδομένα του αρχείου που δίνεται (gsdata.txt) για ένα πρόβλημα ομαδοποίησης. Ένα summary για τις δύο κατανομές είναι το παρακάτω:

D1\_X1 D1\_X2 D2\_X1 D2\_X2

Min. : 3.328 Min. : 5.571 Min. :17.64 Min. : 5.469

1st Qu.: 8.292 1st Qu.: 9.012 1st Qu.:22.43 1st Qu.: 8.555

Median : 9.901 Median :10.228 Median :24.58 Median :10.068

Mean :10.068 Mean :10.218 Mean :24.91 Mean :10.022

3rd Qu.:11.876 3rd Qu.:11.383 3rd Qu.:27.44 3rd Qu.:11.479

Max. :17.324 Max. :14.803 Max. :31.27 Max. :13.471

Αρχικά, εισάγουμε τα δεδομένα:

>>> gsdata = read\_csv("./gsdata.txt")

>>> target = gsdata.loc[:, "Y"]

>>> gsdata = gsdata.drop(["Y"], axis=1)

Στη συνέχεια, θα απαντήσουμε στα παρακάτω ερωτήματα:

α) Σχεδιάστε τα δεδομένα με διαφορετικό χρώμα για κάθε κατανομή.

β) Εφαρμόστε Gaussian Mixture Models για να ομαδοποιήσετε τα δεδομένα σε 3 clusters με την παράμετρο epsilon ίση με 0.1.

γ) Σχεδιάστε τα δεδομένα με τη συνάρτηση πυκνότητας πιθανότητας που προέκυψε από την ομαδοποίηση.

δ) Εμφανίστε τα clusters (hard-assignment) καθώς και τα κέντρα τους.

ε) Υπολογίστε το silhouette.

στ) Σχεδιάστε το heatmap των δεδομένων αφού τα ταξινομήσετε σύμφωνα με την ομαδοποίηση που προέκυψε.

**3.1 Σχεδίαση Δεδομένων**

Αρχικά σχεδιάζουμε τα δεδομένα με την παρακάτω εντολή (ερώτημα (α)):

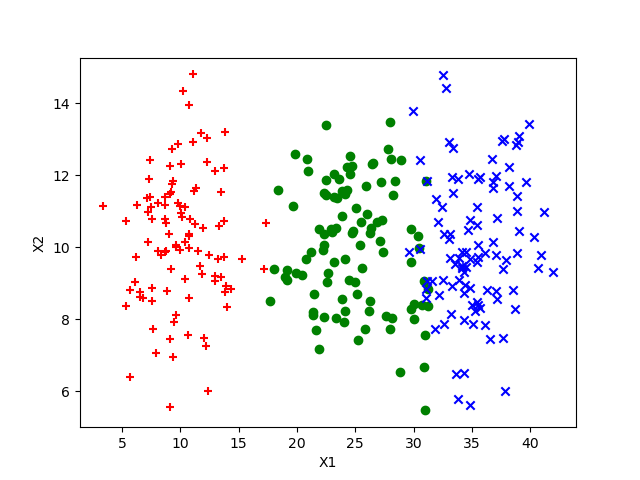
>>> plt.scatter(gsdata[(target == 1)].X1, gsdata[(target == 1)].X2, c="red", marker="+")

>>> plt.scatter(gsdata[(target == 2)].X1, gsdata[(target == 2)].X2, c="green", marker="o")

>>> plt.scatter(gsdata[(target == 3)].X1, gsdata[(target == 3)].X2, c="blue", marker="x")

>>> plt.show()

Οπότε προκύπτει το παρακάτω διάγραμμα:



**3.2 Κατασκευή Gaussian Mixture Models**

Εφαρμόζουμε τον αλγόριθμο expectation maximization για Gaussian Mixture Models σε 2 ή περισσότερες διαστάσεις με την εντολή (ερώτημα (β)):

>>> gm = GaussianMixture(n\_components=3, tol=0.1).fit(gsdata)

Σχεδιάζουμε τα δεδομένα με τη συνάρτηση πυκνότητας πιθανότητας που προσέγγισε το μοντέλο (ερώτημα (γ)) με τις εντολές:

>>> x = np.linspace(np.min(gsdata.loc[:, "X1"]), np.max(gsdata.loc[:, "X1"]))

>>> y = np.linspace(np.min(gsdata.loc[:, "X2"]), np.max(gsdata.loc[:, "X2"]))

>>> X, Y = np.meshgrid(x, y)

>>> XX = np.array([X.ravel(), Y.ravel()]).T

>>> Z = -gm.score\_samples(XX)

>>> Z = Z.reshape(X.shape)

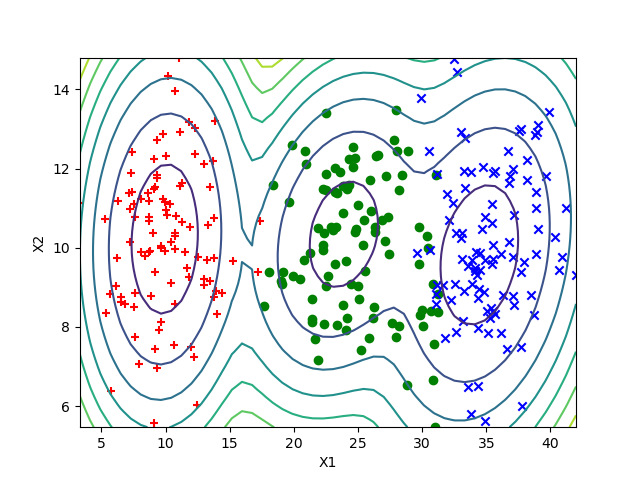
>>> plt.contour(X, Y, Z)

>>> plt.scatter(gsdata[(target == 1)].X1, gsdata[(target == 1)].X2, c="red", marker="+")

>>> plt.scatter(gsdata[(target == 2)].X1, gsdata[(target == 2)].X2, c="green", marker="o")

>>> plt.scatter(gsdata[(target == 3)].X1, gsdata[(target == 3)].X2, c="blue", marker="x")

>>> plt.show()



Για τα assigments και τα κέντρα εκτελούμε τις εντολές:

>>> clusters = gm.predict(gsdata)

>>> centers = gm.means\_

Με αυτές τις τιμές μπορούμε να κατασκευάσουμε το silhouette plot και το heatmap.

**3.3 Υπολογισμός Silhouette**

Για τον υπολογισμό του silhouette (ερώτημα (ε)) εκτελούμε τις εντολές:

>>> from sklearn.metrics import silhouette\_score

>>> print(silhouette\_score(gsdata, clusters))

**3.4 Κατασκευή Heatmap**

Για να κατασκευάσουμε το heatmap (ερώτημα (στ)) εκτελούμε τις εντολές:

>>> gsdata["cluster"] = clusters

>>> gsdata = gsdata.sort\_values("cluster").drop("cluster", axis=1)

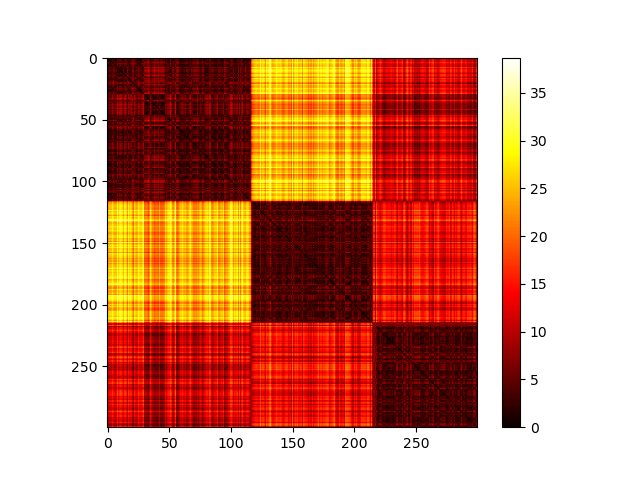
>>> from scipy.spatial import distance\_matrix

>>> dist = distance\_matrix(gsdata, gsdata)

>>> plt.imshow(dist, cmap='hot')

>>> plt.colorbar()

>>> plt.show()



**4 Εφαρμογή με Information Criteria**

Για την κατασκευή ενός μοντέλου Gaussian Mixtures θα χρησιμοποιήσουμε ως εφαρμογή τα δεδομένα εκπαίδευσης που δίνονται (icdata.txt) για ένα πρόβλημα ομαδοποίησης. Ένα summary για τα δεδομένα είναι το παρακάτω:

x

Min. :-2.2239

1st Qu.: 0.6396

Median : 5.1139

Mean : 5.0336

3rd Qu.: 9.1177

Max. :12.0908

Αρχικά, εισάγουμε τα δεδομένα:

>>> icdata = pd.read\_csv("./icdata.txt")

>>> x = icdata.loc[:, "X"]

>>> y = icdata.loc[:, "Y"]

Στη συνέχεια, θα απαντήσουμε στα παρακάτω ερωτήματα:

α) Σχεδιάστε τα δεδομένα με τη συνάρτηση πυκνότητας πιθανότητάς τους.

β) Εφαρμόστε τον αλγόριθμο EM ώστε να ομαδοποιήσετε τα δεδομένα χρησιμοποιώντας 2, 3, 4 και 5 κατανομές. Για κάθε περίπτωση σχεδιάστε τη συνάρτηση πυκνότητας πιθανότητας που προέκυψε και υπολογίστε τα κριτήρια AIC και BIC.

γ) Σχεδιάστε τις καμπύλες για τα AIC και BIC και αποφασίστε για τον αριθμό των κατανομών ώστε να μοντελοποιούνται ικανοποιητικά τα δεδομένα.

**4.1 Εισαγωγή Βιβλιοθηκών και Σχεδίαση Δεδομένων**

Σχεδιάζουμε τα δεδομένα με τις παρακάτω εντολές (ερώτημα (α)):

>>> plt.scatter(x[(y == 1)], np.zeros(len(x[(y == 1)].tolist())), c="red", marker="+")

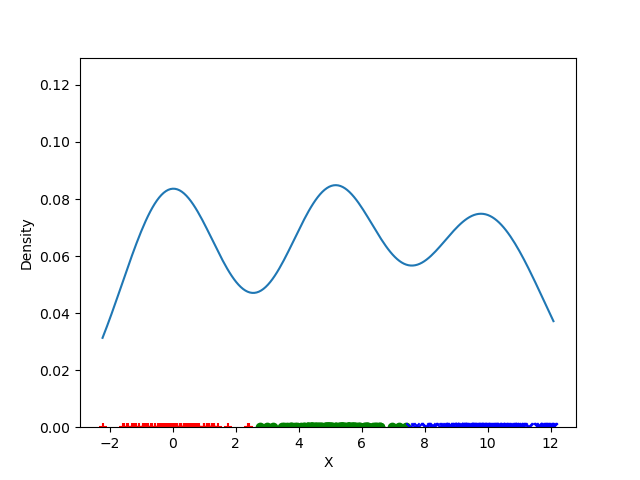
>>> plt.scatter(x[(y == 2)], np.zeros(len(x[(y == 2)].tolist())), c="green", marker="o")

>>> plt.scatter(x[(y == 3)], np.zeros(len(x[(y == 3)].tolist())), c="blue", marker="x")

>>> sns.histplot(x, kde=True, stat="density", linewidth=0, fill=False)

>>> plt.show()

Οπότε προκύπτει το παρακάτω διάγραμμα:



**4.2 Επιλογή πλήθους κατανομών με Information Criteria**

Για το ερώτημα (β) θα χρειαστεί να υπολογίσουμε το Akaike Information Criterion (AIC) και το Bayesian Information Criterion (BIC). Για ένα μοντέλο με t παραμέτρους που εφαρμόζεται σε πλήθος δεδομένων N και έχει συνάρτηση πιθανοφάνειας L τα κριτήρια αυτά υπολογίζονται με τους παρακάτω τύπους:

Ομαδοποιούμε τα δεδομένα χρησιμοποιώντας 2, 3, 4 και 5 κατανομές (ερώτημα (β)):

>>> fig, axs = plt.subplots(4, 1)

>>> fig.tight\_layout()

>>> n = [2, 3, 4, 5]

>>> AIC = []

>>> BIC = []

>>> for i in n:

>>> gm = GaussianMixture(n\_components=i).fit(data)

>>> pi, mu, sigma = gm.weights\_.flatten(), gm.means\_.flatten(), np.sqrt(gm.covariances\_.flatten())

>>> grid = np.arange(np.min(x), np.max(x), 0.01)

>>> axs[i - 2].scatter(x[(y == 1)], np.zeros(len(x[(y == 1)].tolist())), c="red", marker="+")

>>> axs[i - 2].scatter(x[(y == 2)], np.zeros(len(x[(y == 2)].tolist())), c="green", marker="o")

>>> axs[i - 2].scatter(x[(y == 3)], np.zeros(len(x[(y == 3)].tolist())), c="blue", marker="x")

>>> axs[i - 2].plot(grid, mix\_pdf(grid, mu, sigma, pi), label="GMM")

>>> axs[i - 2].set\_title("Density Curves (k=" + str(i) + ")")

>>> axs[i - 2].set\_xlabel("Data")

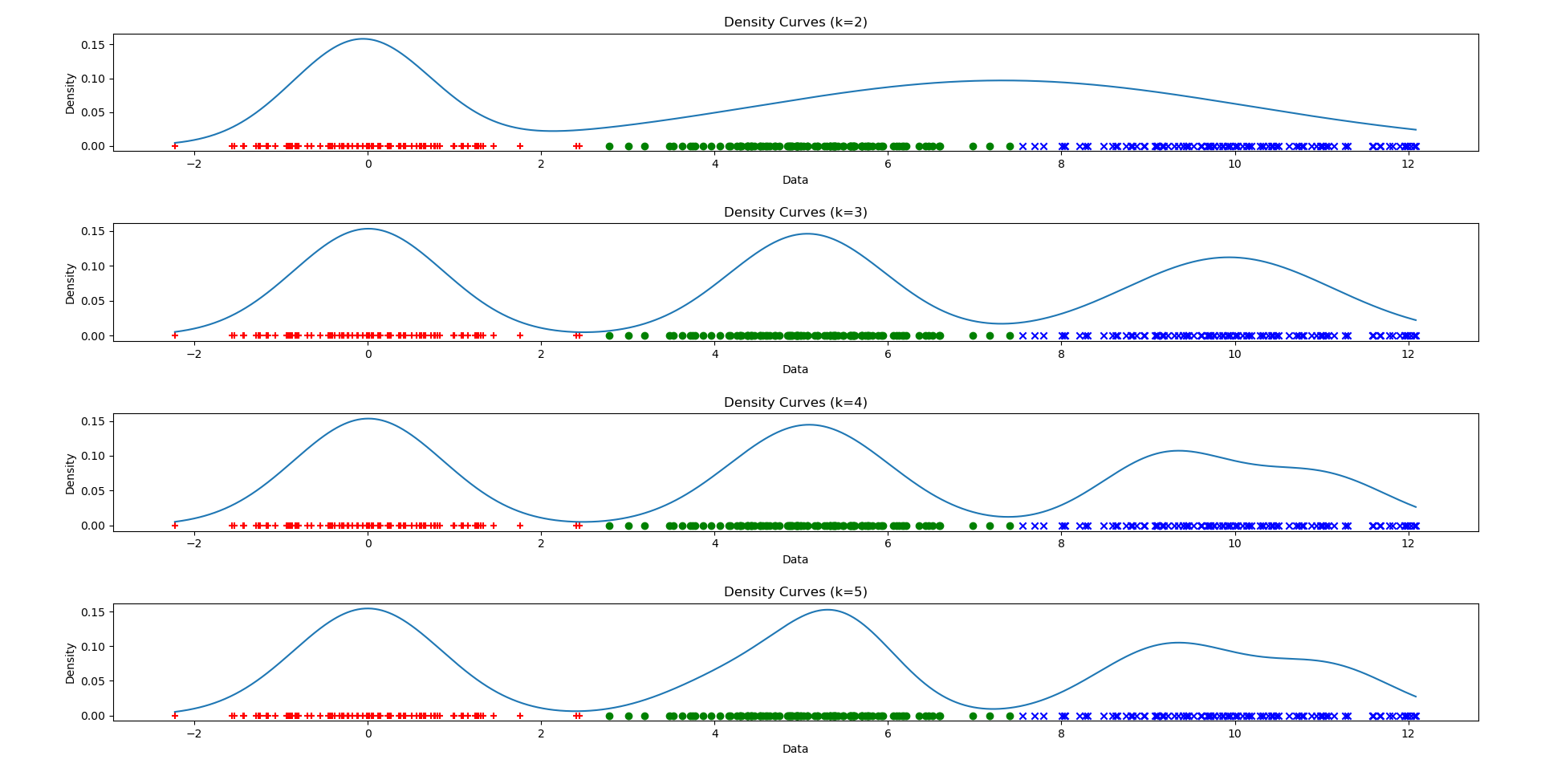
>>> axs[i - 2].set\_ylabel("Density")

>>> AIC.append(gm.aic(data))

>>> BIC.append(gm.bic(data))

>>> plt.show()

Όπως φαίνεται στον παραπάνω κώδικα, υπολογίζουμε επίσης τα AIC και BIC όπως ορίζονται. Επίσης, σχεδιάζουμε τη συνάρτηση πυκνότητας πιθανότητας σε κάθε περίπτωση.



Στη συνέχεια μπορούμε να σχεδιάσουμε το AIC (ερώτημα (γ)) με τις εντολές:

>>> plt.plot(n, AIC)

>>> plt.title("AIC")

>>> plt.xlabel("Index")

>>> plt.ylabel("AIC")

>>> plt.show()

Και αντίστοιχα για το BIC:

>>> plt.plot(n, BIC)

>>> plt.title("BIC")

>>> plt.xlabel("Index")

>>> plt.ylabel("BIC")

>>> plt.show()

Οπότε προκύπτουν τα παρακάτω διαγράμματα:

|  |  |
| --- | --- |
|  |  |