Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Τομέας Ηλεκτρονικής και Υπολογιστών

9ο Εξάμηνο
Φθινόπωρο 2021

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 θ, ο αλγόριθμος μεγιστοποιεί την ποσότητα:

$$\ln P(X|\theta) = \ln \sum_{Z} P(X, Z|\theta)$$

Η βελτιστοποίηση γίνεται σε δύο βήματα. Στο expectation step (E-step) χρησιμοποιούμε τις τιμές των παραμέτρων θ_{old} ώστε να υπολογίσουμε την εκ των υστέρων κατανομή των λανθάνουσων $P(Z|X, \theta_{old})$. Στο maximization step (M-step) υπολογίζουμε τη νέα τιμή του θ (θ_{new}) δεδομένης της κατανομής αυτής $\theta_{new} = arg\max_{\theta} \sum_{Z} P(Z|X, \theta_{old}) \ln P(X, Z|\theta)$. 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 μ_k, συμμεταβλητότητες Σ_k και mixing coefficients π_k (που ορίζουν τη συμμετοχή κάθε κατανομής στο mixture) ορίζεται με την εξίσωση:

$$P(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$$

Όπου τα mixing coefficients ορίζονται στο διάστημα [0, 1] και το άθροισμά τους είναι ίσο με τη μονάδα και $\mathcal{N}(x|\mu, \Sigma)$ είναι η multivariate κανονική κατανομή:

$$\mathcal{N}(x|\mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

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 (θ) θα είναι οι μέσοι και οι συμμεταβλητότητες των κατανομών (μ, Σ) καθώς και τα mixing coefficients (π). Ο αλγόριθμος θα μεγιστοποιήσει την ποσότητα:

$$\ln P(X|\pi,\mu,\Sigma) = \sum_{n=1}^{N} \ln\left(\sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k,\Sigma_k)\right)$$

Όπου x_1, x_2, … x_N 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) μ_1, μ_2 των κανονικών κατανομών καθώς και τις λανθάνουσες μεταβλητές λ_1, λ_2 από τις οποίες προήλθαν τα δεδομένα. Εφαρμόστε τα βήματα του αλγορίθμου στην Python.

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

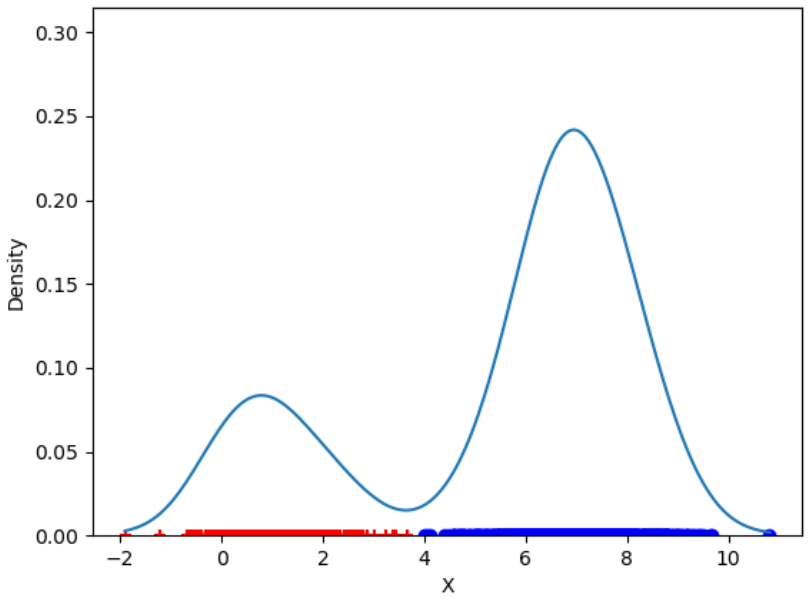
δ) Σχεδιάστε στο ίδιο διάγραμμα τη συνάρτηση πυκνότητας πιθανότητας που προσεγγίσατε καθώς και την πραγματική συνάρτηση πυκνότητας πιθανότητας των δεδομένων.

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 μ_1, μ_2 έστω $\mu_1 = 0$ και $\mu_2 = 1$ και των λ_1, λ_2 έστω $\lambda_1 = 0.5$ και $\lambda_2 = 0.5$. Έτσι, το αρχικό σετ παραμέτρων θα είναι $\Theta = \{\theta_1, \theta_2\} = \{(0,1,0.5), (1,1,0.5)\}$.

Expectation Step

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

$$P(D_j|x_i, \theta_j) = \frac{\lambda_j \cdot P(x_i|\theta_j)}{\lambda_1 \cdot P(x_i|\theta_1) + \lambda_2 \cdot P(x_i|\theta_2)}$$

όπου $j = 1,2$ και αρχικά $\lambda_1 = \lambda_2 = 0.5$.

Αν για παράδειγμα θεωρήσουμε ότι ένα σημείο που μας δίνεται είναι το $x_i = 2$, από τη συνάρτηση κανονικής κατανομής $P(x_i|\theta_j) = \frac{1}{\sqrt{2\pi\sigma_j}} e^{-\frac{(x_i-\mu_j)^2}{2\sigma_j}}$ και για τα θ_1, θ_2 προκύπτει:

$$P(2|\theta_1) = \frac{1}{\sqrt{2\pi \cdot 1}} e^{-\frac{(2-0)^2}{2\cdot 1}} = \frac{1}{\sqrt{2\pi}} e^{-2} = 0{,}054$$

$$P(2|\theta_2) = \frac{1}{\sqrt{2\pi \cdot 1}} e^{-\frac{(2-1)^2}{2\cdot 1}} = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}} = 0{,}242$$

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

$$P(D_1|2, \theta_1) = \frac{0.5 \cdot 0.054}{0.5 \cdot 0.054 + 0.5 \cdot 0.242} = 0.182$$

$$P(D_2|2, \theta_2) = \frac{0.5 \cdot 0.242}{0.5 \cdot 0.054 + 0.5 \cdot 0.242} = 0.818$$

Τα παραπάνω σημαίνουν ότι το σημείο 2 έχει πιθανότητα 0.182 να ανήκει στην πρώτη κατανομή (D_1) και πιθανότητα 0.818 να ανήκει στη δεύτερη κατανομή (D_2).

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 μ_1, μ_2 (τα σ_1, σ_2 είναι σταθερά ίσα με 1 από εκφώνηση) οι οποίες μεγιστοποιούν την πιθανότητα. Για τον υπολογισμό των παραμέτρων μ_1, μ_2 θα χρησιμοποιήσουμε τους παρακάτω τύπους:

$$\mu_1 = \frac{\sum_{i=1}^{1000} x_i \cdot P(D_1|x_i, \theta_1)}{\sum_{i=1}^{1000} P(D_1|x_i, \theta_1)}$$

$$\mu_2 = \frac{\sum_{i=1}^{1000} x_i \cdot P(D_2|x_i, \theta_2)}{\sum_{i=1}^{1000} P(D_2|x_i, \theta_2)}$$

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:

$$\lambda_1 = \frac{\sum_{i=1}^{1000} P(D_1|x_i, \theta_1)}{1000}$$

$$\lambda_2 = \frac{\sum_{i=1}^{1000} P(D_2|x_i, \theta_2)}{1000}$$

Αφού υπολογίσουμε τις νέες τιμές των παραμέτρων μ_1, μ_2, λ_1, λ_2, υπολογίζουμε τη λογαριθμική πιθανοφάνεια (log likelihood) των δεδομένων:

$$ln(P(X|\Theta)) = \sum_{i=1}^{1000} ln\left(\sum_{j=1}^{2} \lambda_j \cdot P(x_i|\theta_j)\right) = \sum_{i=1}^{1000} ln(\lambda_1 \cdot P(x_i|\theta_1) + \lambda_2 \cdot P(x_i|\theta_2))$$

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)
```

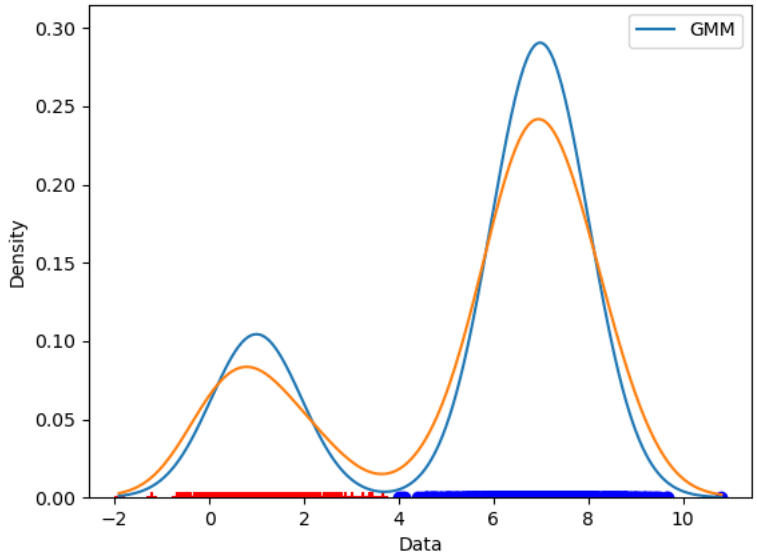
Μπορούμε να δούμε τις τελικές παραμέτρους του μοντέλου και την τελική λογαριθμική πιθανοφάνεια με τις εντολές:

```
>>> 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.

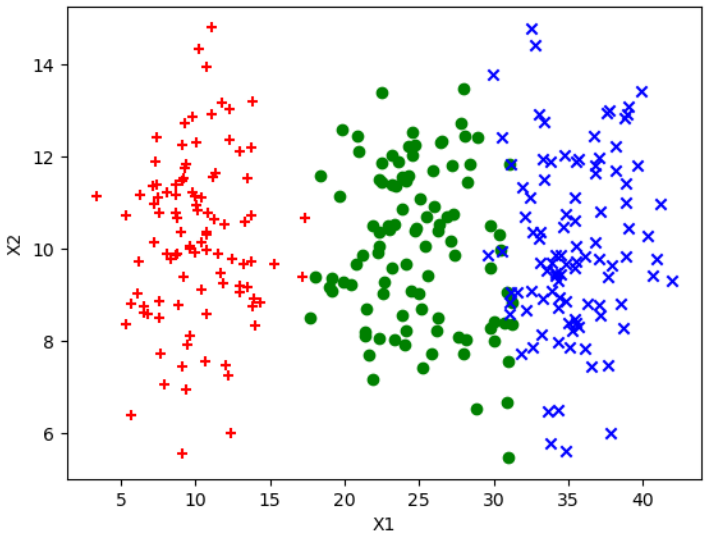
στ) Σχεδιάστε το heatmap των δεδομένων αφού τα ταξινομήσετε σύμφωνα με την ομαδοποίηση που προέκυψε.

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:



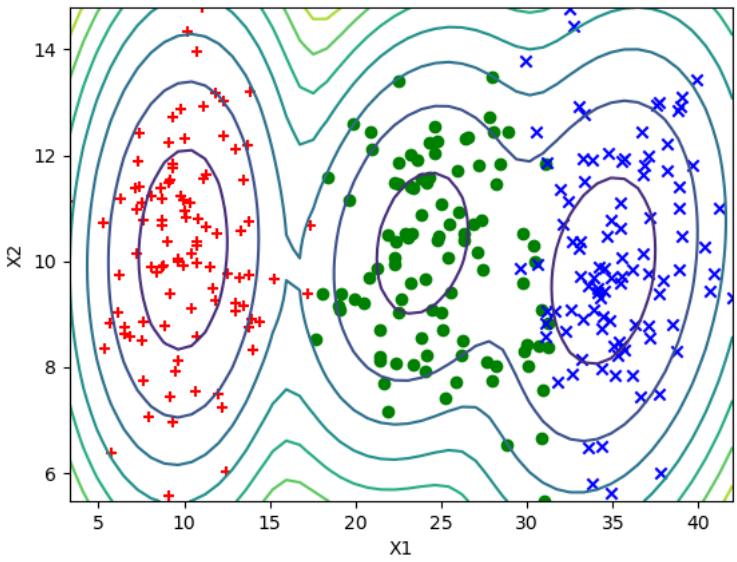
3.2 Κατασκευή Gaussian Mixture Models

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

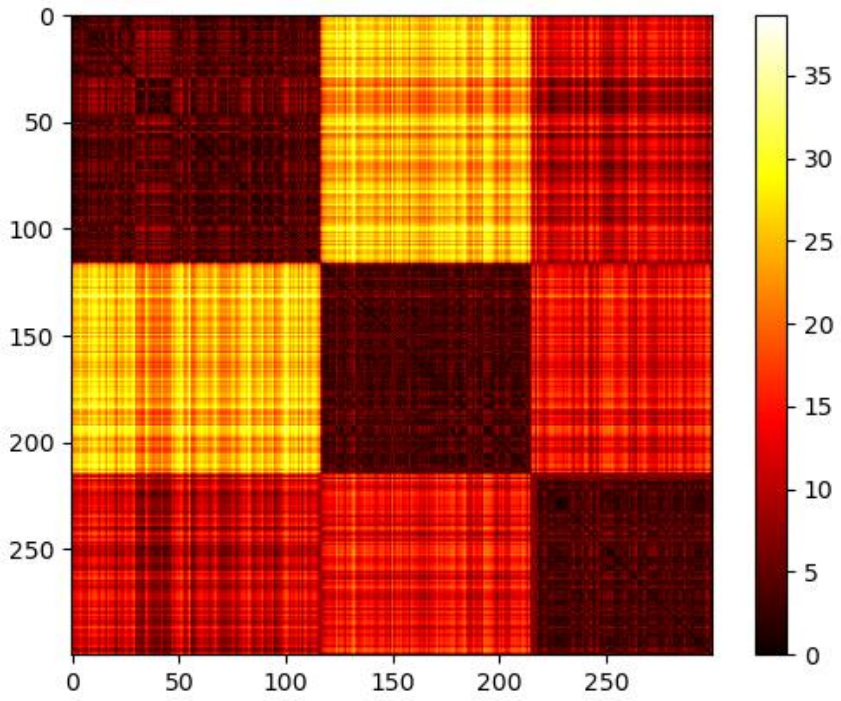
Για τον υπολογισμό του silhouette (ερώτημα (ε)) εκτελούμε τις εντολές:

```
>>> 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="+")
```

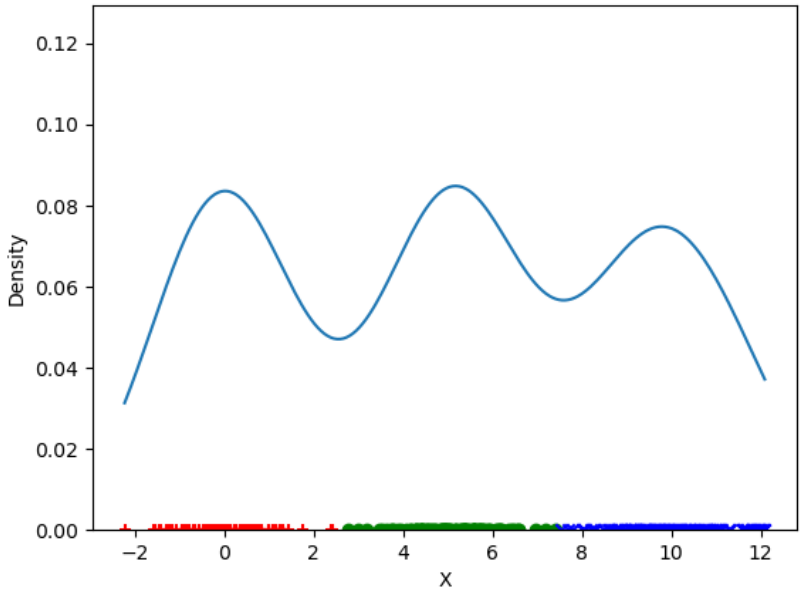
```
>>> plt.scatter(x[(y == 2)], np.zeros(len(x[(y == 2)].tolist())),
c="green", marker="o")
```

```
>>> 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:

$$AIC = 2 \cdot t - 2 \cdot \ln(L)$$

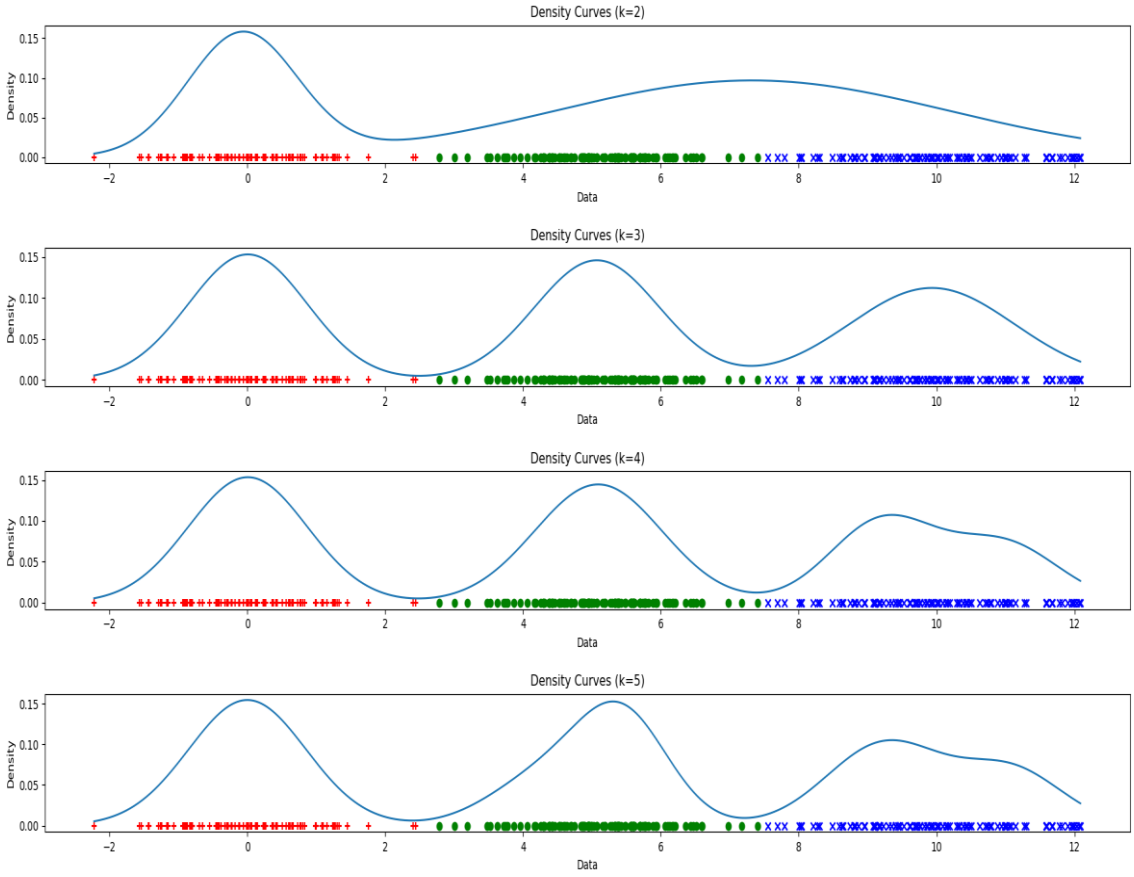
$$BIC = t \cdot ln(N) - 2 \cdot \ln(L)$$

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.



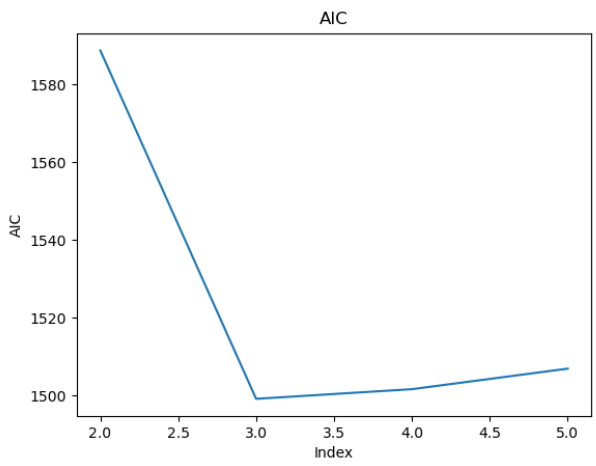
Στη συνέχεια μπορούμε να σχεδιάσουμε το AIC (ερώτημα (γ)) με τις εντολές:

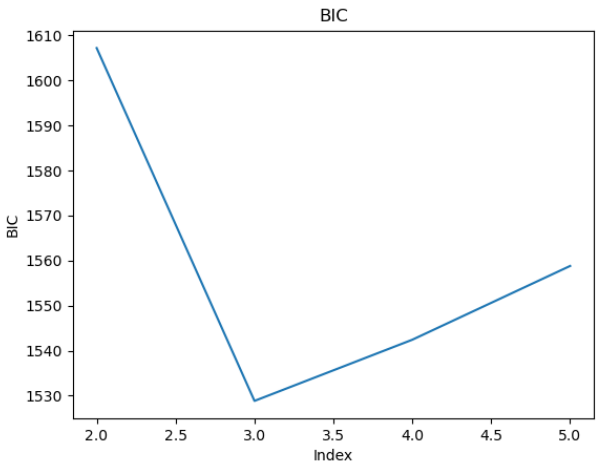
```
>>> plt.plot(n, AIC)
>>> plt.title("AIC")
>>> plt.xlabel("Index")
>>> plt.ylabel("AIC")
>>> plt.show()
```

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:


AIC
1580
1560
1540
1520
1500
AIC
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Index


BIC
1610
1600
1590
1580
1570
1560
1550
1540
1530
BIC
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Index