University of Prishtina

Faculty of Electrical and Computer Engineering

Kodra e Diellit, p.n.
10000 - Prishtinë, Kosova

Lënda: Dizajni dhe Analiza e Algoritmeve

Expectation–Maximization algorithm

Lista e shpërndarjes:

Emri (alfab.)	**Department**	**Location**
Avni Rexhepi	Komp.	FIEK
Dardan Shabani	Komp.	FIEK

Prishtinë, Maj, 2021

Menaxhimi i dokumentit

Personat e autorizuar për të bërë ndryshime:

Rrezearta Thaqi	Komp.	FIEK
Zahir Maliqi	Komp.	FIEK
Lum Pireva	Komp.	FIEK

Dokumenti u krijua me veglat:

Microsoft Word
Python
Jupyter Notebook

Përmbajtja

1. Qëllimi

Ky raport ka për qëllim dokumentimin e projektit i cili ka të bëjë me dizajnin dhe analizën e algoritmit Expectation–maximization. Së pari do të bëhet një hyrje dhe do të shpjegohet algoritmi se si punon, hapat që ndërmerren në këtë algoritëm, vetitë e tij, aplikimin dhe pastaj do të demonstrohet me anë të kodit. Në fund do të paraqitet edhe nje shembull i implementimit të këtij algoritmi si aplikacion.

2. Hyrje

Algoritmi Expectation-maximazation përdoret për të gjetur parametrat e gjasave maksimale të një modeli në rastet ku ekuacionet nuk mund të zgjidhen drejtpërdrejt. Këto modele përfshijnë variabla latente përveç parametrave të panjohur dhe vëzhgimeve të njohura të të dhënave. Kjo është p.sh. një vlerë e panjohur në një set të dhënash, ose modeli mund të formulohet më thjesht duke supozuar ekzistencën e pikave të të dhënave të ardhshme të pa analizuara. Një mixture model mund të përshkruhet thjesht duke supozuar që secila data point e observuar ka një data point korresponduese të pa observuar ose një variabël latente, duke spefikuar mixture komponenten të cilës i takon secila data point.
Gjetja e një zgjidhjeje për gjasë maksimale kërkon marrjen e derivateve të funksionit të gjasës respektivisht të gjithë vlerave të panjohura, parametrave dhe variablave latente, dhe njëkohësisht zgjidhjen e ekuacioneve që rezultojnë në këtë rast. Në modelet statike me variabla latente, kjo zakonisht është e pamundur. Në vend të kësaj, rezultati është një set i ekuacioneve të ndërthurura në të cilat zgjidhja e parametrave kërkon vlerat e variablave latente dhe anasjelltas, por duke zëvendësuar një set të ekuacioneve në tjetrën krijohet një ekuacion i pazgjidhshëm.
Iteracionet (përsëritjet) e këtij algoritmi alternojnë mes performimit të një exceptation step (E), i cili krijon një funksion për pritjen e gjasës së llogaritur duke përdorur vlerën e tanishme për parametrat, dhe një maximization step (M), i cili llogarit parametrat duke rritur gjasën e gjetur në hapin E. Këto llogaritje të parametrave pastaj përdoren për të përcaktuar shpërndarjen e variablave latente në hapin e ardhshëm E. [6]

3. Algoritmi Expectation-maximization

Algoritmi Expectation-maximization vazhdon nga këndvështrimi se gjendet një mënyrë për t'i zgjidhur numerikisht dy sete të ekuacioneve. Njëri mund thjesht të zgjedhë vlera arbitrare për një nga dy sete të të panjohurave, t'i përdorë ato për të llogaritur setin e dytë, pastaj t'i përdorë këto vlera të reja për të bërë një llogaritje më të mirë të setit të pare, dhe pastaj të alternojë në mes të këtyre dyjave derisa vlerat rezultuese të konvergjojnë në pika të caktuara.
Është e dukshme se ndoshta kjo nuk do të funksionojë , por mund të vërtetohet se mundet në këtë kontekst, dhe se derivati i gjasës është përafërsisht zero në atë pikë, që do të thotë se pika është ose një maksimum ose një minimum. Në përgjithësi, mund të bëhen shumë maksimume pa asnjë mundësi që mund të gjendet maksimumi i përgjithshëm.

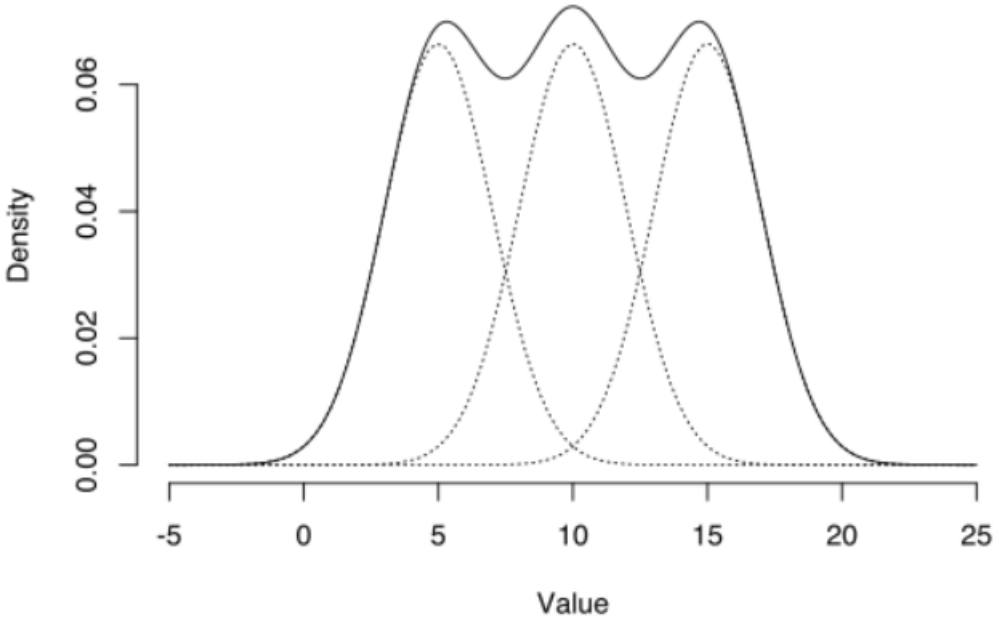

Fig.1. Expectation-Maximization tek Gaussian Mixture Models

Në përgjithësi, hapat bazik të këtij algoritmi janë:

Bëhet një hamendësim për parametrat e modelit dhe krijohet një shpërndarje e probabilitetit. Kjo quhet "E-Step" për shpërndarjen "Expected".
Të dhënat e reja te gjetura përfshihen në model.
Shpërndarja e probabilitetit nga hapi E shkëputet për të përfshirë të dhënat e reja. Ky hap nganjëherë quhet hapi M (M-Step).
Hapat prej 2 deri në 4 përsëriten derisa të arrihet stabiliteti, p.sh derisa të arrihet një shpërndarje që nuk ndryshon nga hapi E në hapin M. [7]

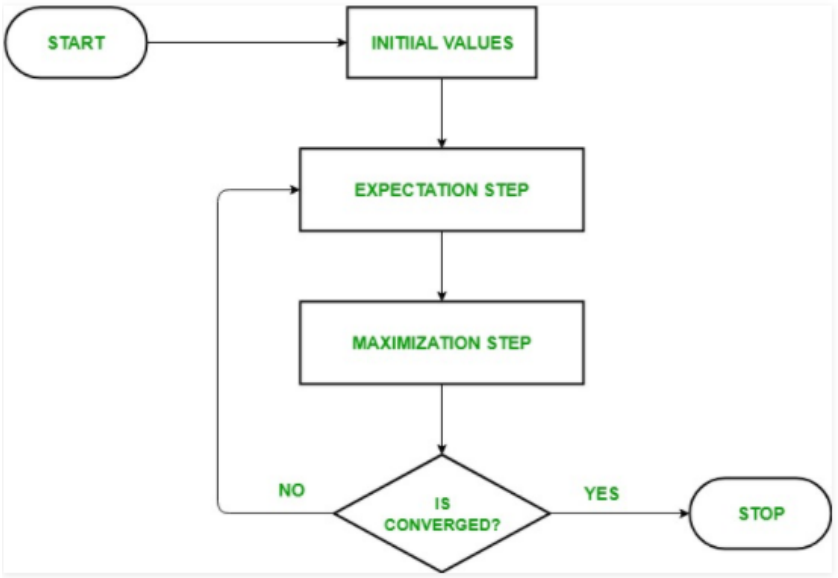

Fig.2. Hapat e algoritmit Expectation-Maximization

3.1. Përshkrimi i algoritmit

Nëse jepet një model statik i cili gjeneron një set të të dhënave të observuara, një i të dhënave latente të paobservuara ose vlera që mungojnë, dhe një vektor me parametra të panjohur, së bashku me një funksion të probabilitetit (gjasës), vlera e gjasës maksimale e parametrave të panjohur përcaktohet duke maksimizuar gjasën më të ulët të të dhënave të observuara.
Megjithatë, kjo sasi shpesh është e pazgjidhshme, p.sh. është një sekuencë e ngjarjeve, ashtu që numri i vlerave rritet në mënyrë eksponenciale me gjatësinë e sekuencës, kalkulimi i saktë i shumës do të jetë shumë i vështirë.
Algoritmi Expectation maximization kërkon të gjejë vlerën e gjasës maksimale duke aplikuar në mënyrë iterative hapat E dhe M:

Hapi E (Expectation) : Defino $\mathbf{Q(\theta \mid \theta^{(t)})}$ si vlerën e pritur të funksionit të gjasës të θ, respektivisht me shpërndarjen e tanishme të **Z** të dhënë **X**-it dhe vlerat e tanishme të parametrave $\boldsymbol{\theta^{(t)}}$.

Hapi M (Maximization) : Gjej parametrat që e rrisin (maksimizojnë) këtë sasi:

$$\boldsymbol{\theta^{(t+1)} = \arg\max Q(\theta \mid \theta^{(t)})}$$

Modelet tipike tek të cilat aplikohet ky algoritëm e përdorin **Z** si variabël latente duke indikuar pjesëmarrjen në një nga setet e grupeve:

1. Data points të observuara **X** mund të jenë diskrete (marrin vlera në një set të fundëm ose infinit të numërueshëm) ose kontinuale (marrin vlera në një set infinit të panumërueshëm). Asocuar me secilën data point mund të jetë një vektor observimeve.

2. Vlerat e munguara (të njohura si variabla latente) **Z** janë diskrete, të vizatuara nga një numër fiks i vlerave, dhe me një variabël latenete për njësi të observimit.

3. Parametrat janë kontinualë, dhe janë të dy llojeve: Parametrat të cilët janë të asocuar me të gjitha data points, dhe ato të asocuara me një vlerë specifike të variablës latente.

Megjithatë, është e mundur që EM të aplikohet në lloje tjera të modeleve. [6]

3.2. Vetitë e algoritmit Expectation-Maximization

Edhe nëse një iteracion EM rrit të funksionin e gjasës të të dhënave të observuara, nuk ka siguri që sekuenca konvergjon në një vlerësues (llogaritës) të gjasave maksimale. Për shpërndarje multimodale, kjo do të thotë se një EM algoritëm mund të konvergjojë në një maksimum lokal të funksionit të gjasës së të dhënave të observuara, duke u varur nga vlerat fillestare.
Algoritmi Expectation-Maximization është i dobishëm sidomos kur gjasa i takon familjes eksponenciale: hapi E bëhet shuma e pritjeve të statistikave, dhe hapi M përfshin maksimizimin (rritjen) e një funksioni linear. Në këso raste zakonisht është e mundur që të derivojmë përditësimet e shorehjeve të mbyllura për çdo hap, duke përdorur formulën Sundberg. [6]

3.3. Procedura maximization – maximization

Algoritmi Expectation-Maximization mund të paraqitet si dy hapa maximization të alternuar, që është si një shembull i zbritjes së koordinatave. Konsiderojmë funksionin:

$$\mathbf{F(q, \theta) := E_q\,[\log L(\theta; x, Z)] + H(q),}$$

ku q është një shpërndarje arbitrare e probabilitetit mbi të dhënat e observuara z dhe H(q) është një entropi e shpërndarjes q. Ky funksion mund të shkruhet si:

$$\mathbf{F(q, \theta) = -D_{KL}\,(q \,||\, p_{Z|X}\,(\cdot\,|x; \theta)) + \log L(\theta;x),}$$

ku $\mathbf{p_{Z|X}\,(\cdot\,|x; \theta)}$ është shpërndarja e të dhënave të observuara. [6]

Pastaj hapat mund të paraqiten si:

Hapi *Expectation:* Zgjedh **q** për të maksimizuar (rritur) F:

$$\mathbf{q^{(t)} = \arg\max_q F\,(q, \theta^{(t)})}$$

Hapi *Maximization:* Zgjedh **θ** për të rritur F:

$$\mathbf{\theta^{(t+1)} = \arg\max_\theta F(q^{(t)}, \theta)}$$

3.4. Kompleksiteti i algoritmit

Algoritmi Expectation-maximazation për të dhëna të mëdha në çdo përsëritje ka $n \leftarrow O(\lceil n / 2 \rceil)$ dhe çdo përsëritje pasuese zgjat sa gjysma e kohës, $i \leftarrow i / 2$.
Ne e dimë nga teoria themelore e kompleksitetit që $cO(n) = O(n)$, për disa numra pozitiv c. Nga pikëpamja praktike e të dhënave megjithatë, është një ndryshim dramatik i dëshmuar me eksperimente sepse në shumë raste, EM thjesht nuk mund të konvergjojë. Le të jetë $l = \lceil n / 2 \rceil$ që tregonë numrin e të dhënave të vendosura në nyje, $t = \lfloor n / 2 \rfloor$ që tregonë numrin e të dhënave qe nuk janë të vendosura në nyje dhe k – numri i atributeve . Koha e përgjithshme dhe më e mirë e ekzekutimit të EM për të dhëna të mëdha është $O(i_2k(d^3 + nd^2) + li_2logn)$ dhe $O(i_2knd + li_2logt)$, përkatësisht ($i_2 \ll i$). Pjesët logaritmike vijnë nga azhurnimet (update-imet) e grumbujve. Nëse supozojmë se algoritmi EM përdor algjebër lineare (gjë që bën) , atëherë kompleksiteti i tij duhet të jetë ***O (m*n ^ 3)***, ku m është numri i përsëritjeve dhe n është numri i parametrave. [1]

3.5. Përparësitë e këtij algoritmi krahas një algoritmi tjetër

Grumbullimi (clustering) është një mjet i rëndësishëm për fushën e data mining dhe algoritmet që i ndajnë të dhënat e të njëjtës natyrë. Për dallim nga algoritmi për klasifikim, grumbullimi i përkon algoritmeve të tipit 'unsupervised'. Dy përfaqësuesit e algoritmeve të grumbullimit janë K-means algoritmi dhe Expectation Maximization (EM) algoritmi. Këto dy algoritme janë të njëjtë në mënyrën që të dy e lejojne rafinimin e modelit të një procesi të përsëritshëm për ta gjetur ndeshjen (congestion) më të mirë. Sidoqoftë, K-means algoritmi dallon për nga metoda që e përdor për ta llogaritur distancen Euklidiane duke llogaritur distancën mes çdo dy artikujve të të dhënave, ndërsa EM përdor metoda statistikore. Algoritmi EM zakonisht përdoret për të siguruar funksionet në mënyrë më efektive.

Clustering do të thotë ndarja e një dataseti të madh në një shumicë të grupeve të të dhënave, të cilat ndajnë disa tipare të secilit nëngrup. Ai kryhet duke llogaritur ngjashmëritë bazuar në metodën për matjen e distancës. Të dy mund të ndahen në grumbullim të pjesshëm dhe grumbullim hierarkik në të dhëna.

Overview i Clustering algoritmit

Clustering mund të konsiderohet problemi më i madh i 'unsupervised learning', dhe - si çdo problem i këtij lloji, synon ta gjëjë një strukturë në një koleksion të të dhënash të pa etiketuara (unlabelled). Prandaj, një cluster është një koleksion i objekteve që janë të ngjashme mes vete dhe janë të ndryshme mes objekteve që i takojnë clusterëve të tjerë.

Një pyetje e rëndësishme është për të vendosur se çka e përbën një grumbullim të mirë, pasi që zakonisht pranohet që nuk ka një kriter absolut të mirë që mund të ishte e pavarur nga qëllimi final i grumbullimit. Si pasojë, është përdoruesi që duhet të jap kriterin që i përshtatet mirë nevojave të tij të veçanta, dhe rezultati i algoritmit të grumbullimit mund të interpretohet në mënyra të ndryshme. Ekzistojne disa tipe të grumbullimit. Njëra mënyrë është grupimi i të dhënave në një mënyrë

ekskluzive, ashtu që nëse një artikull i të dhenave i takon një grumbullimi, atëherë ai nuk mund të futet në një grumbullim tjetër.
Një mënyrë tjetër, është e ashtuquajtura grumbullimi me mbivendosje, duke përdorur bashkësitë e paqarta për të grumbulluar në atë mënyrë qe secili artikull i të dhënave mund t'i takojë dy apo më shumë grumbullimesh.

Me anë të testimeve të shumta të realizuara, është ekzaminuar performanca e dy algoritmeve më të përdorura për grumbullim: K-means dhe EM.
Rezultatet kanë treguar se shpejtësia procesuese është më e madhe më anë të algoritmit EM se sa me K-means, mirëpo saktësia e klasifikimit të të dhënave ishte 94.7476% për K-means, e cila është 7.3171% më mirë se algoritmi EM. Për ta shkurtuar kohën e procesimit me anë të algoritmit K-means, duhet të bëhen optimizime të mëtejshme. [2]

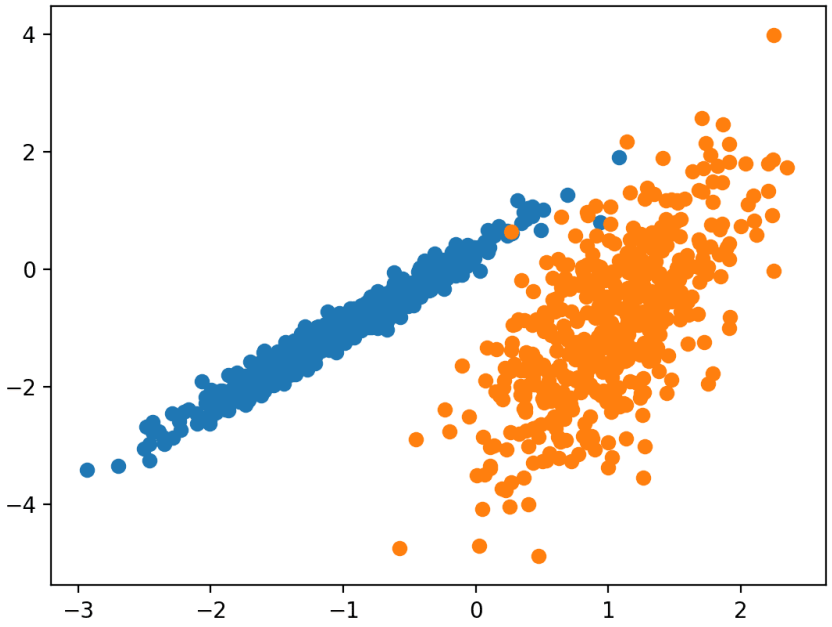

Fig.3. Shembull i ekzekutimit të Clustering algoritmit

3.6. Kufizimet e algoritmit

Algoritmi EM mund të jetë shumë i ngadaltë, madje edhe në kompjuterin më të shpejtë.
Funksionon më mirë kur kemi vetëm një përqindje të vogël të të dhënave që mungojnë
dhe dimensioni i tyre nuk është shumë i madh.

Scalability
EM ka shkallëzim linear me numrin e rekordeve dhe atributeve. Numri i përsëritjeve të konvergjencës tenton të rritet me rritjen e madhësisë së të dhënave (si rreshtat ashtu edhe kolonat).

Konvergjenca
EM mund të jetë e ngadaltë për probleme komplekse dhe mund të vendosë një ngarkesë të konsiderueshme në burimet llogaritëse.

Dimensioni i lartë
EM ka kapacitet të kufizuar për modelimin e të dhënave me dimensione të larta (të gjera).
Prania e shumë atributeve ngadalëson konvergjencën e modelit dhe algoritmi bëhet më pak i aftë të bëjë dallimin midis atributeve kuptimplota. Algoritmi kompromentohet kështu në aftësinë e tij për të gjetur korrelacione (lidhmëria midis dy ose më shumë variablave).

Numri i komponenteve
EM zakonisht kërkon që përdoruesi të specifikojë numrin e komponenteve. Në shumicën e rasteve, ky nuk është informacion që përdoruesi mund ta dijë paraprakisht.

Inicializimi i parametrit
Zgjedhja e vlerave të përshtatshme të parametrave fillestarë mund të ketë një efekt të rëndësishëm në cilësinë e modelit. Strategjitë e inicializimit që janë përdorur për EM në përgjithësisht ndikojnë në llogaritjen perfundimtare.

Kalimi nga komponente në grupe
Komponentët e modelit EM shpesh trajtohen si grupe. Kjo qasje mund të jetë mashtruese pasi grupimet kohezive shpesh modelohen nga shumë përbërës. [3]

4. Përdorimi i algoritmit

Algoritmi Expectation-maximization përdoret:

- Në teorinë e gjasave maksimale (maximum-likelihood ML) për rindërtimit të imazhit, kryesisht për të rritur cilësinë e tomografisë me induksion magnetik (MIT). Koordinatat e matjes mund të përdoren direkt për llogaritjen e probabiliteteve të tranzicionit. Prandaj, asnjë informacion nuk humbet në procesin e mbledhjes. [4]

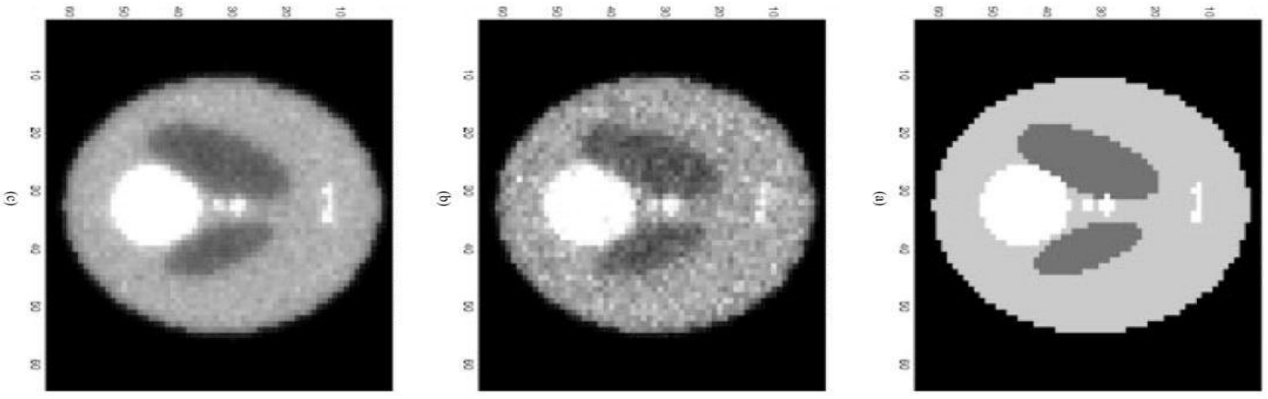

Fig.4. Përdorimi i EM algoritmit për rindertim të imazhit të tomografisë

- Çrregullim të sinjaleve të mbivendosura
- Vlerësimi i modeleve të përzierjes Gaussiane (GMMs)
- Vlerësimi i modeleve të fshehura të Markov-it (HMM)
- Vlerësimi i parametrave për shpërndarjet e përbëra të Dirichlet-ut
- Gjetja e përzierjeve optimale të modeleve fikse [5]

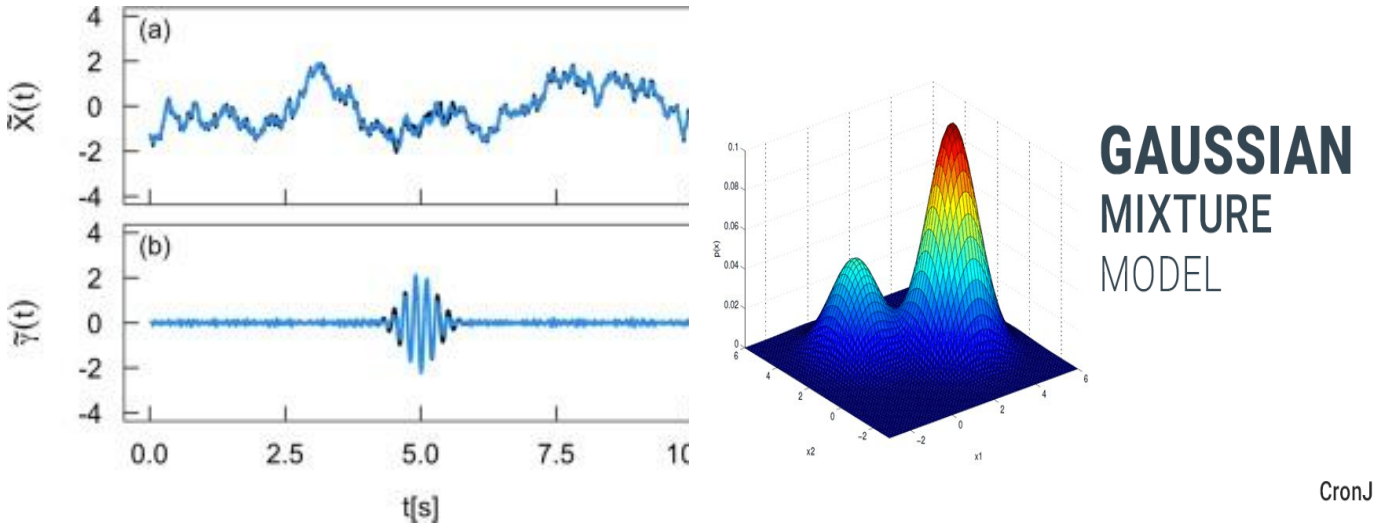

Fig.5. Përdorimi i EM algoritmit tek sinjalet e mbivendosura dhe përzierja Gaussiane

5. Implementimi i algoritmit në Python

Implementimi i algoritmit në Python është bërë me Jupyter Notebook.
Që të ekzekutohet ky algoritëm, së pari duhet t'i importojmë disa librari si në figurën më poshtë:

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style("white")
%matplotlib inline
import numpy as np
from scipy import stats
import pandas as pd
from math import sqrt, log, exp, pi
from random import uniform
```

Fig.6. Libraritë e përdorura në projekt

Fillimisht japim disa vlera me hamendësim për parametrat e modelit me anë të input-it, më pas këto vlera përdoren për gjenerimin e të dhënave të cilat edhe paraqiten në mënyrë grafike:

```
random_seed=36788765
np.random.seed(random_seed)

M1 = input('Jepni vleren per Mean1: ')
S1 = input('Jepni vleren per Standard_dev1: ')
M2 = input('Jepni vleren per Mean2: ')
S2 = input('Jepni vleren per Standard_dev2: ')

if isinstance(M1, str) or isinstance(S1, str) or isinstance(M2, str) or isinstance(S2, str):
    print("Duhet te jepni vetem numra float ose integer!")
    M1 = input('Jepni vleren per Mean1: ')
    S1 = input('Jepni vleren per Standard_dev1: ')
    M2 = input('Jepni vleren per Mean2: ')
    S2 = input('Jepni vleren per Standard_dev2: ')

Mean1 = float(M1)
Standard_dev1=float(S1)
Mean2=float(M2)
Standard_dev2=float(S2)
```

Fig.7. Dhënia e vlerave hyrëse

Siç shihet edhe nga figura, nëse jepet ndonjë vlerë që është string, atëherë del lajmërimi se duhet të shënojmë numër integer ose float. Këtë testim mund ta shohim tek pjesa **Rezultatet**.

```
y1 = np.random.normal(Mean1, Standard_dev1, 1500)
y2 = np.random.normal(Mean2, Standard_dev2, 750)
data=np.append(y1,y2)
```

Fig.8. Gjenerimi i të dhënave nga vlerat hyrëse

```
Min_graph = min(data)
Max_graph = max(data)
x = np.linspace(Min_graph, Max_graph, 2000)

print('Input Gaussian {:}: μ = {:.2}, σ = {:.2}'.format("1", Mean1, Standard_dev1))
print('Input Gaussian {:}: μ = {:.2}, σ = {:.2}'.format("2", Mean2, Standard_dev2))
sns.distplot(data, bins=20, kde=False);
print(data)
```

Fig.9. Marrja e të dhënave dhe vizatimi i grafikonit

Në vazhdim është klasa Gaussian e cila është për Single Gaussian në të cilën bëhet llogaritja e probabilitetit të një data point duke u bazuar në vlerat e dhëna të parametrave, por duke e parë edhe nga rezultati dhe paraqitja grafike, nuk mjafton një Single Gaussian për tu përshtatur me të dhënat.

```
class Gaussian:

    def __init__(self, mu, sigma):

        self.mu = mu
        self.sigma = sigma

    def pdf(self, datum):

        u = (datum - self.mu) / abs(self.sigma)
        y = (1 / (sqrt(2 * pi) * abs(self.sigma))) * exp(-u * u / 2)
        return y

    def __repr__(self):
        return 'Gaussian({0:4.6}, {1:4.6})'.format(self.mu, self.sigma)
```

```
best_single = Gaussian(np.mean(data), np.std(data))
print('Single Gaussian me i mire: μ = {:.2}, σ = {:.2}'.format(best_single.mu, best_single.sigma))

g_single = stats.norm(best_single.mu, best_single.sigma).pdf(x)
sns.distplot(data, bins=20, kde=False, norm_hist=True);
plt.plot(x, g_single, label='single gaussian');
plt.legend();
```

```
Best single Gaussian: μ = 4.4, σ = 7.6
```

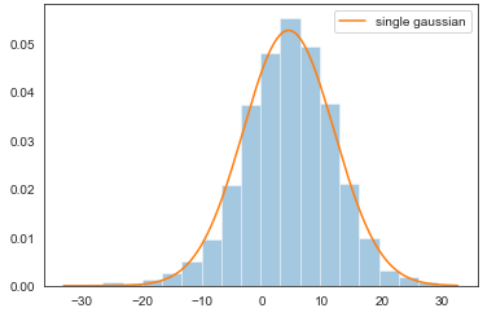

Fig.10. Single Gaussian

Pra, përmes kësaj klase bëhet llogaritja më e mirë e parametrave të dhënë ne fillim për tu përafruar sa më shumë, por për ta arritur këtë qëllim në mënyrën sa më të mirë të mundshmë, tani përdorim dy mixture modele Gausiane.

Tani në klasën e krijuar për këto dy mixture modele Gausiane ekzekutohen hapat E dhe M.
Në hapin E, secila pikë bëhet assign në gaussian 1 ose 2 me një përqindje të caktuar, d.m.th bëhet llogaritja e peshës (weight), probabilitetit dhe bëhet normalizimi i tyre.
Në hapin M, bëhet llogaritja e vlerave të reja të parametrave dhe shpërndarja e re gausiane.

```
def Estep(self):

    self.loglike = 0.
    for datum in self.data:

        wp1 = self.one.pdf(datum) * self.mix
        wp2 = self.two.pdf(datum) * (1. - self.mix)

        den = wp1 + wp2

        wp1 /= den
        wp2 /= den

        self.loglike += log(den)

        yield (wp1, wp2)
```

Fig.11. Hapi Exceptation

```
def Mstep(self, weights):

    (left, rigt) = zip(*weights)
    one_den = sum(left)
    two_den = sum(rigt)

    self.one.mu = sum(w * d  for (w, d) in zip(left, data)) / one_den
    self.two.mu = sum(w * d  for (w, d) in zip(rigt, data)) / two_den

    self.one.sigma = sqrt(sum(w * ((d - self.one.mu) ** 2)
                              for (w, d) in zip(left, data)) / one_den)
    self.two.sigma = sqrt(sum(w * ((d - self.two.mu) ** 2)
                              for (w, d) in zip(rigt, data)) / two_den)

    self.mix = one_den / len(data)
```

Fig.12. Hapi Maximization

Tani, për ta parë algoritmin në aksion, japim një numër të caktuar të iteracioneve dhe fillon të bëhet llogaritja e modelit më të mirë. Me llogaritjen e këtij modeli, algoritmi përcakton vlerat e duhura të parametrave duke krijuar kështu edhe shpërndarjen e re Gausiane me dy mixture modele.

```
n_iterations = 300
n_random_restarts = 4
best_mix = None
best_loglike = float('-inf')
print('Llogaritja e modelit me te mire...\n')
for _ in range(n_random_restarts):
    mix = GaussianMixture_self(data)
    for _ in range(n_iterations):
        try:
            mix.iterate()
            if mix.loglike > best_loglike:
                best_loglike = mix.loglike
                best_mix = mix
        except (ZeroDivisionError, ValueError, RuntimeWarning):
            pass
```

Fig.13. Pjesa kryesore e algoritmit

Në figurën më poshtë është i paraqitur rezultati i fituar në mënyrë grafike:

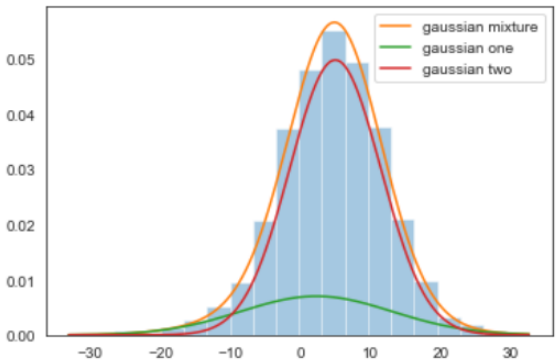

Fig.14. Best Gaussian Mixture Model

6. Implementimi i algoritmit si GUI (Aplikacioni)

Implementimi i algoritmit si GUI është bërë poashtu në Python me Tkinter. Ndërfaqja e UI përmban 4 inpute për parametrat e modelit dhe tre butona: butoni i parë për llogaritjen e Best Single Gaussian, butoni i dytë për llogaritjen e modelit më të mirë Gausian duke përdorur dy mixture modele të tij dhe butoni tjetër për ndaljen e ekzekutimit të programit.

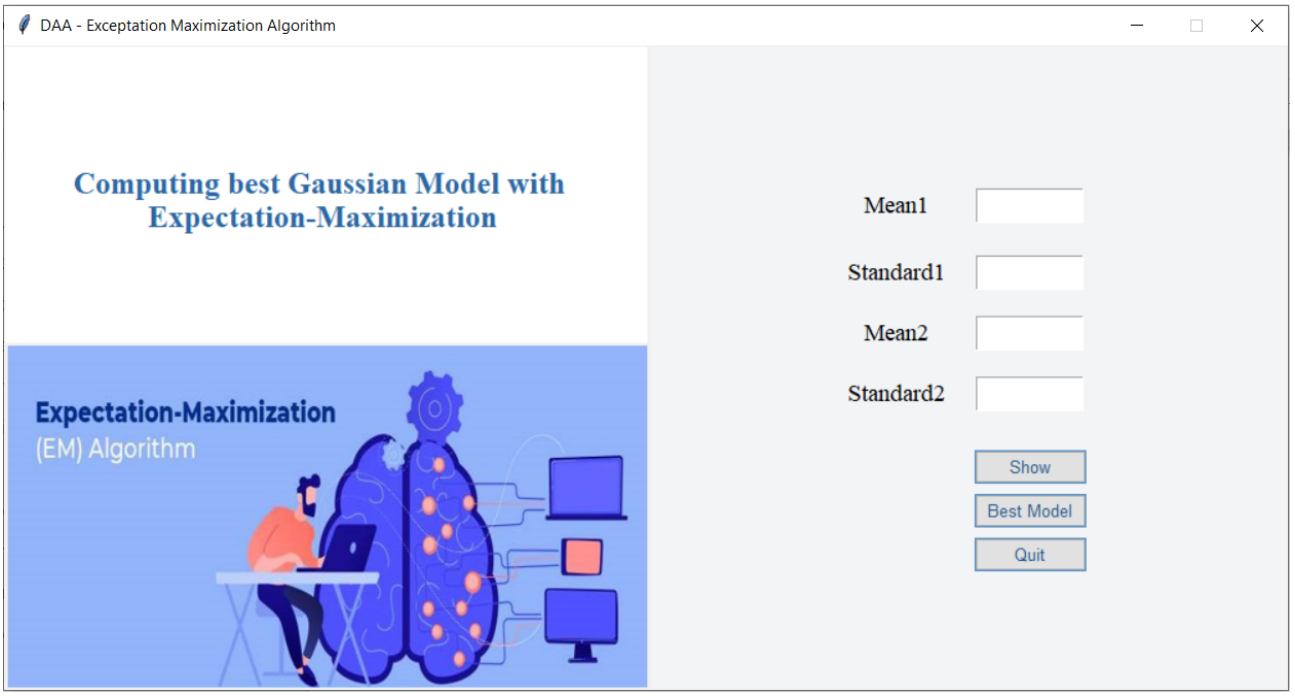

Fig.15. Pamja e aplikacionit

Aplikacionin e kemi zhvilluar në ambientin zhvillimor të Python 3.8.
Më poshtë është e paraqitur një pjesë e kodit të implementuar për krijimin e GUI:

```
17 root = tk.Tk()
18 root.title("DAA - Exceptation Maximization Algorithm")
19 root.geometry("1000x500")
20 root.resizable(False, False)
21
22 panedwindow=tk.PanedWindow(root,orient=HORIZONTAL)
23 panedwindow.pack(fill=BOTH, expand=True)
24
25 frame1 = tk.Frame(panedwindow, width=500, height=100, bg="white", relief=SUNKEN)
26
27 intro = tk.Label(frame1, text="Computing best Gaussian Model with\n Expectation-Maximization"
28                  , font = ('times new roman', 18,'bold'), bg="white", fg="#2B68A3")
29
30 intro.place(x=50,y=90)
31
32 em = Image.open("em.jpg")
33 emImage = em.resize((500, 265))
34 test = ImageTk.PhotoImage(emImage)
35
36 img = tk.Label(frame1,image=test)
37 img.image=test
38
39 img.place(x=0,y=230)
40
```

Fig.16. Një pjesë e shkëputur nga kodi për krijimin e pamjes së aplikacionit

7. Rezultatet

Së pari po paraqesim rezultatet e fituara nga projekti në Jupyter NoteBook dhe më pas edhe në aplikacion.

```
Jepni vleren per Mean1: k
Jepni vleren per Standard_dev1: k
Jepni vleren per Mean2: d
Jepni vleren per Standard_dev2: d
Duhet te jepni vetem numra float ose integer!
Jepni vleren per Mean1: 4
Jepni vleren per Standard_dev1: 4
Jepni vleren per Mean2: 4
Jepni vleren per Standard_dev2: 4
Input Gaussian 1: μ = 4.0, σ = 4.0
Input Gaussian 2: μ = 4.0, σ = 4.0
[ 2.64539892  2.97705513  4.20458457 ...  0.38624525 -1.52993789
  1.86277646]
```

Fig.17. Dhënia e vlerave hyrëse

Në figurën më lart është paraqitur pjesa hyrëse e ekzekutimit. Siç shihet, së pari kemi testuar për vlera string ku si pasojë është paraqitur mesazhi përkatës, pastaj me dhënien e numrave float ose integer vazhdon edhe rrjedha e ekzekutimit.

Më poshtë po paraqesim rezultatin në mënyrë grafike pas llogaritjes së Single Gaussian me klasën Gaussian:

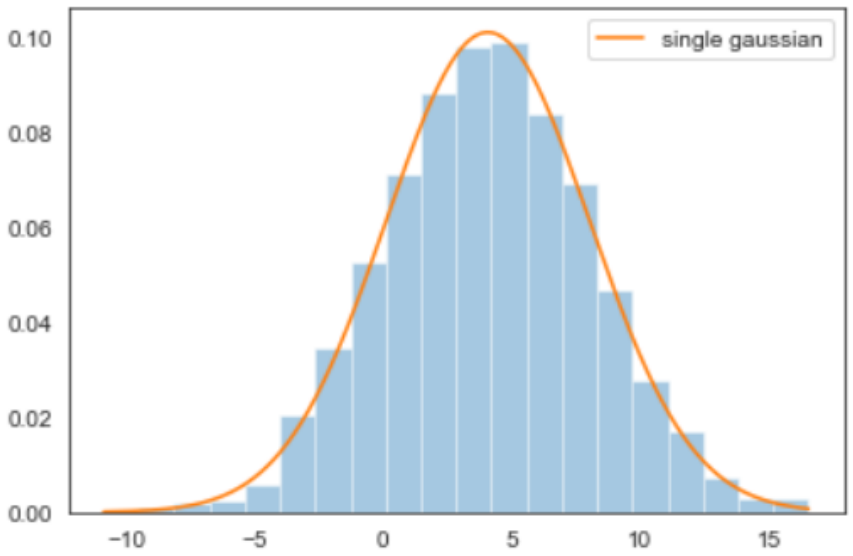

Fig.18. Rezultati grafik për Single Gaussian

Përpos kësaj ne kemi mundësinë që të llogarisim modelin më të mirë Gaussian duke u bazuar në dy modele:

```
Llogaritja e modelit me te mire...

Input Gaussian 1: μ = 4.0, σ = 4.0
Input Gaussian 2: μ = 4.0, σ = 4.0
Gaussian 1: μ = -1.4, σ = 1.5, weight = 0.016
Gaussian 2: μ = 4.2, σ = 3.9, weight = 0.98
```

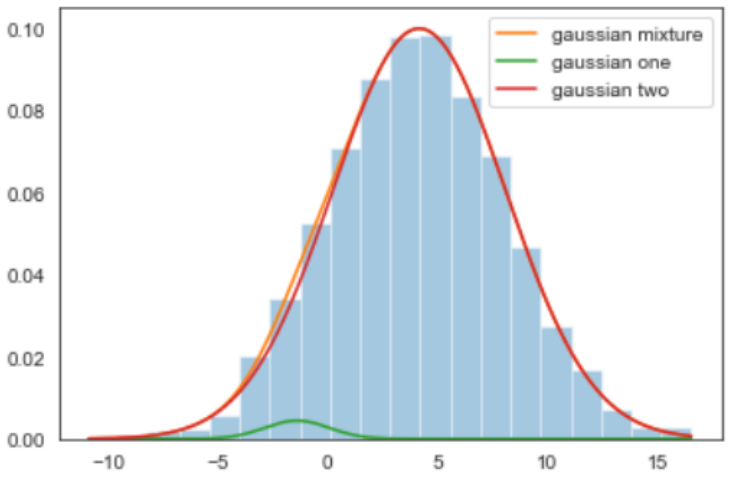

Fig.19. Rezultati grafik për Best Gaussian Mixture Model

Tani po i paraqesim rastet e testimit dhe ekzekutimit në aplikacion:

Fig.20. Rasti i testimit kur nuk jepet asnjë vlerë hyrëse

Në rastin si në figurën më lart nuk kemi dhënë asnjë vlerë hyrëse për të marrë ndonjë rezultat dhe për këtë arsye na del edhe mesazhi që duhet të japim një numër float. Edhe njëjta gjë ndodhë edhe nëse si input japim string:

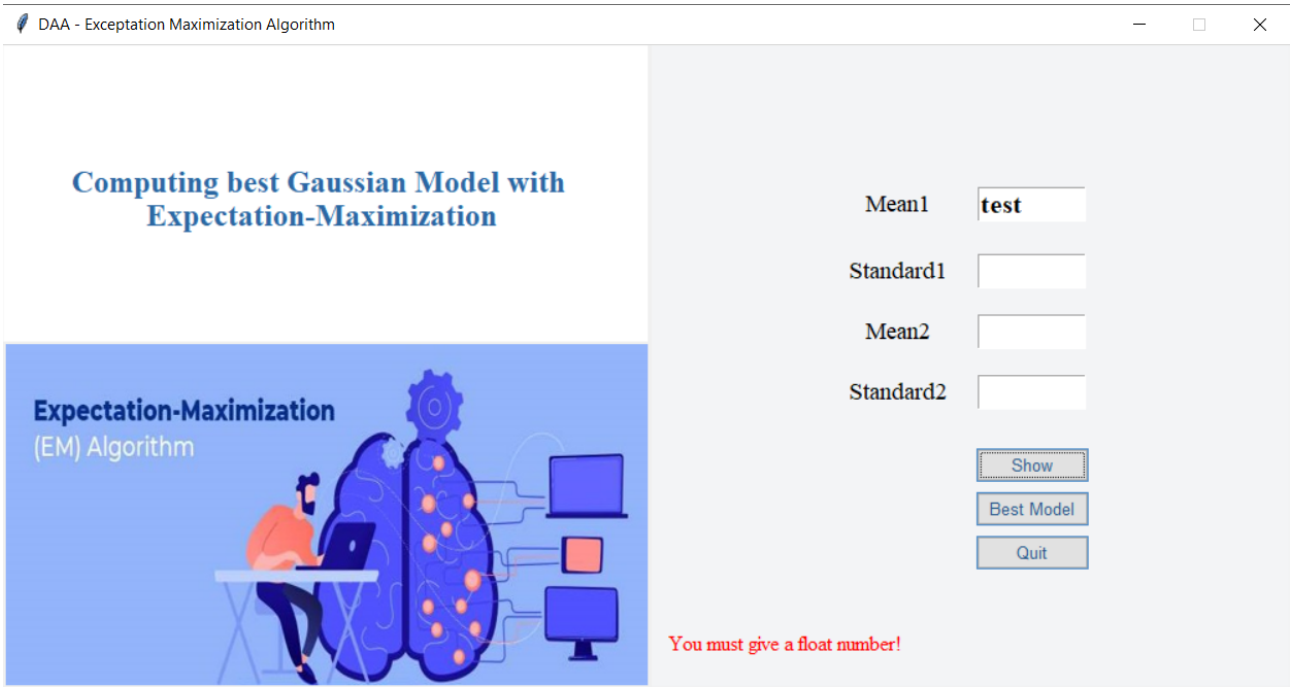

Fig.21. Rasti i testimit ku si input japim një string

Kështu ndodhë kur klikojmë butonin Show (për Single Gaussian) ose Best Model (për gjetjen e modelit më të mirë Gausian).

Tani po shfaqim rezultatet e ekzekutimit kur të dhënat e shënuara janë në rregull:

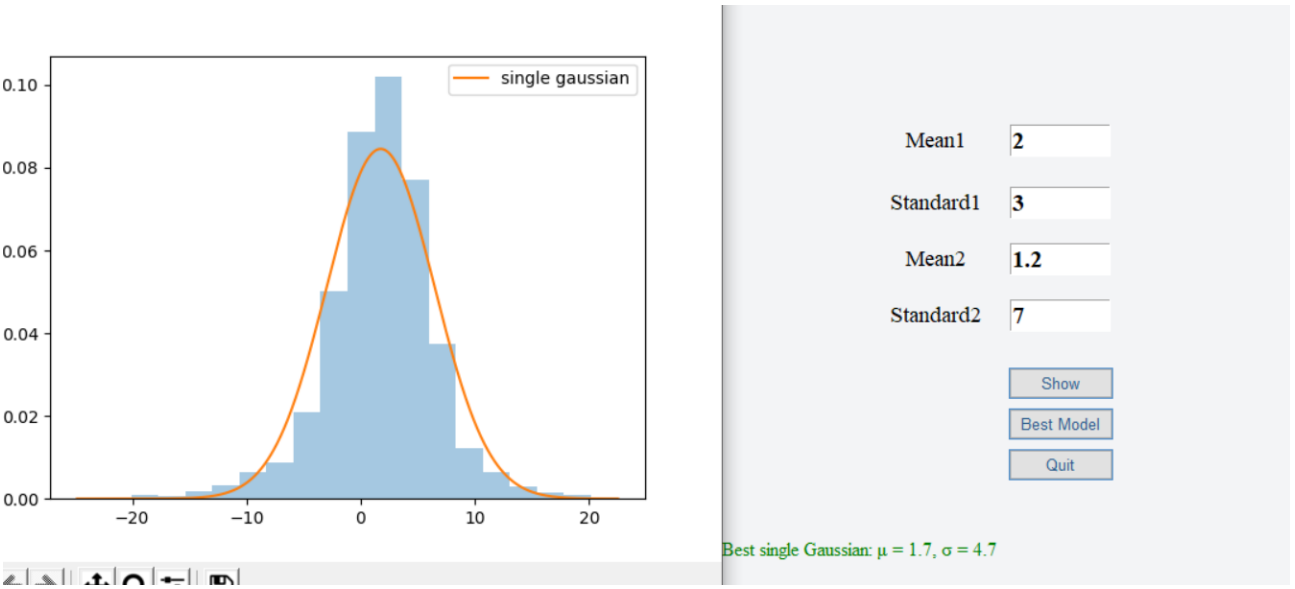

Fig. 22. Grafiku për Single Gaussian

Ky rezultat shfaqet në rastin kur klikojmë butonin Show. Më poshtë është paraqitur rezultati i fituar kur klikojmë butonin Best Model:

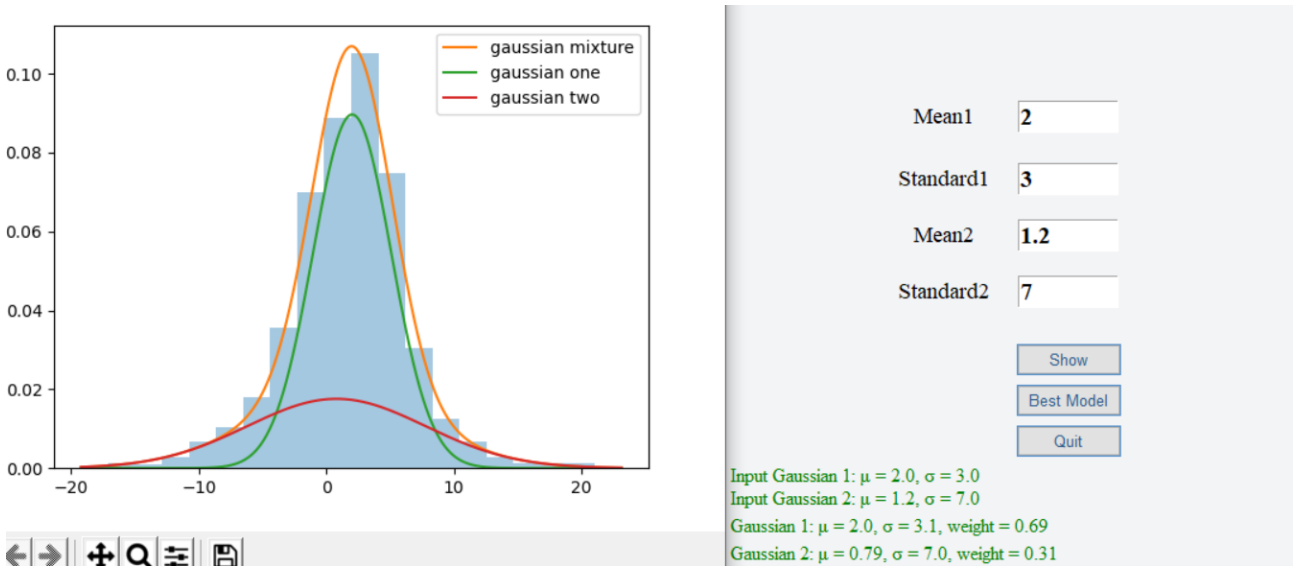

Fig. 23. Grafiku për Best Gaussian Mixture Model

8. Konkluzioni

Nga ekzekutimi i algoritmit më lart, mund të vërejmë disa përparësi por edhe disa të meta. Nëse fillojmë nga të metat mund të përmendim shkallën e madhe të kompleksitetit i cili lehtë vërehet edhe gjatë ekzekutimit. Për vlera të mëdha të iteracioneve vonesa e ekzekutimit të algoritmit vjen duke u rritur gjithnjë e më shumë duke ndikuar në arritjen më të ngadaltë të rezultateve. Por pamë edhe shumë përparësi. P.sh. me këtë algoritëm gjithmonë është e garantuar që gjasa do të rritet pas çdo iterimi. Hapat E dhe M (Expect dhe Maximization) janë shumë të lehtë për probleme të ndryshme në termin e implementimit.

Ne ramë në konkluzion se algoritmi Expectation-Maximization është shumë i përshtatshëm në identifikimin në mënyrë numerike të numrave të komponenteve të një GMM (Gaussian Mixture Model).

9. Referencat

[1]. https://link.springer.com/article/10.1007/s41060-017-0062-1
[2]. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4433949/
[3]. https://docs.oracle.com/database/121/
[4]. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2969844/
[5]. https://www.statisticshowto.com/em-algorithm-expectation-maximization/
[6] https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm
[7] https://www.statisticshowto.com/em-algorithm-expectation-maximization/?fbclid=IwAR2c5QhYXYWMnIL85Wg_3qgm2JaZdcod_X3AVqZsO31pSsRmp3lzha0gVU8