

شناسيي آماري الگو

تمرين هاي سري يك

فرهاد دليراني
۹۶۱۳۱۱۲۵

dalirani@aut.ac.ir
dalirani.1373@gmail.com

ساختار درختی پوشه‌ی ارسال شده:

[./Exercise1.pdf](#)

[./computer exercises 1:](#)

computerExercise1.py extraPart.png

[./computer exercises 2:](#)

computerExercise1.py computerExercise2.py

[./computer exercises 3:](#)

computerExercise1.py computerExercise3.py

[./computer exercises 4:](#)

computerExercise1.py computerExercise4.py

[./computer exercises 5:](#)

computerExercise1.py computerExercise5.py

[./computer exercises 6:](#)

computerExercise6.py Figure_1.png Figure_2.png Figure_3.png

[./problem1:](#)

Problem1-1.png Problem1-2.png program1.py

[./problem2:](#)

Problem2-1.png problem2-2.png problem3-3.png program2.py

[./problem6:](#)

problem6.m

تمام کدها با پایتون 3.6 نوشته شده‌اند. البته کدی دوخطی برای problem 6 نوشته ام که برای نوشتن آن از متلب استفاده کرده ام.

همچنین از پکیج‌های زیر استفاده کرده ام:

- numpy

- pylab

- matplotlib

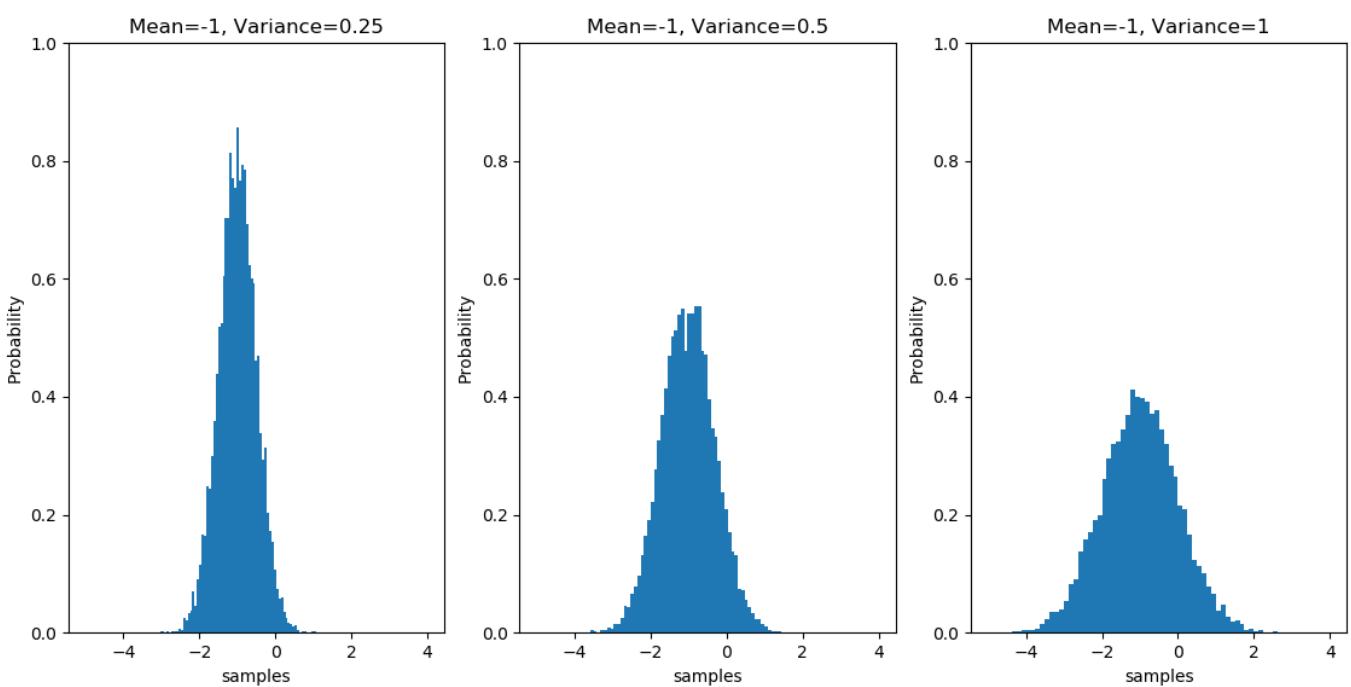
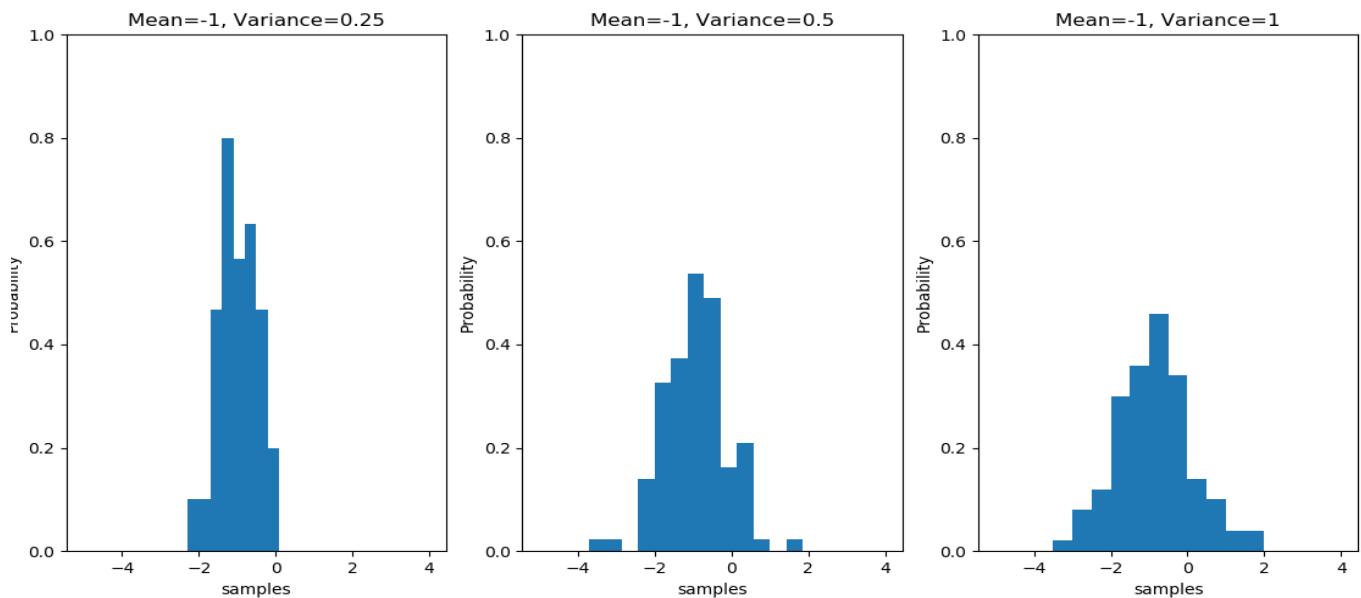
البته برای راحتی در نصب پایتون 3.6 و پکیج‌های مربوط به دیتاساینس که numpy، matplotlib و pylab هم جزیی از آن پکیج‌ها هستند از Anaconda 5.0.0 استفاده کرده‌ام که همه‌ی موارد گفته شده را بدون دردرس و سختی نصب می‌کند. تنها کافی است آن را از دانلود کنید و Installer باقی کار را انجام می‌دهد.
<https://www.anaconda.com/download>

زبان برنامه نویسی: پایتون 3.6
پکیج‌ها: پکیج‌های گفته شده را برای راحتی در نصب با Anaconda نصب کردم.
ورژن Anaconda من: Anaconda 5.0.0 For Linux Installer که البته همین ورژن برای سایر سیستم عامل‌ها هم موجود است.

محیط برنامه نویسی: pyCharm Community Edition
سیستم عامل: Linux – البته همان طور که خودتان اطلاع دارید سیستم عامل مهم نیست، برای ارایه‌ی اطلاعات بیشتر به سیستم عامل اشاره کردم. همین طور کدهایم را در ویندوز تست کردم و مشکلی نداشتند.

سوال ۱:

کد پایتونی که برای این سوال نوشته ام در پوشه‌ی problem1 قرار دارد. در دو شکل زیر خروجی کد را مشاهده می‌کنید. در شکل زیر Histogram ها برای 100 نمونه، همان طور که سوال خواسته است رسم شده‌اند ولی برای نتیجه گیری بهتر برای 10000 نمونه هم Histogram ها را تولید کرده‌ام که در شکل دوم قابل مشاهده است:

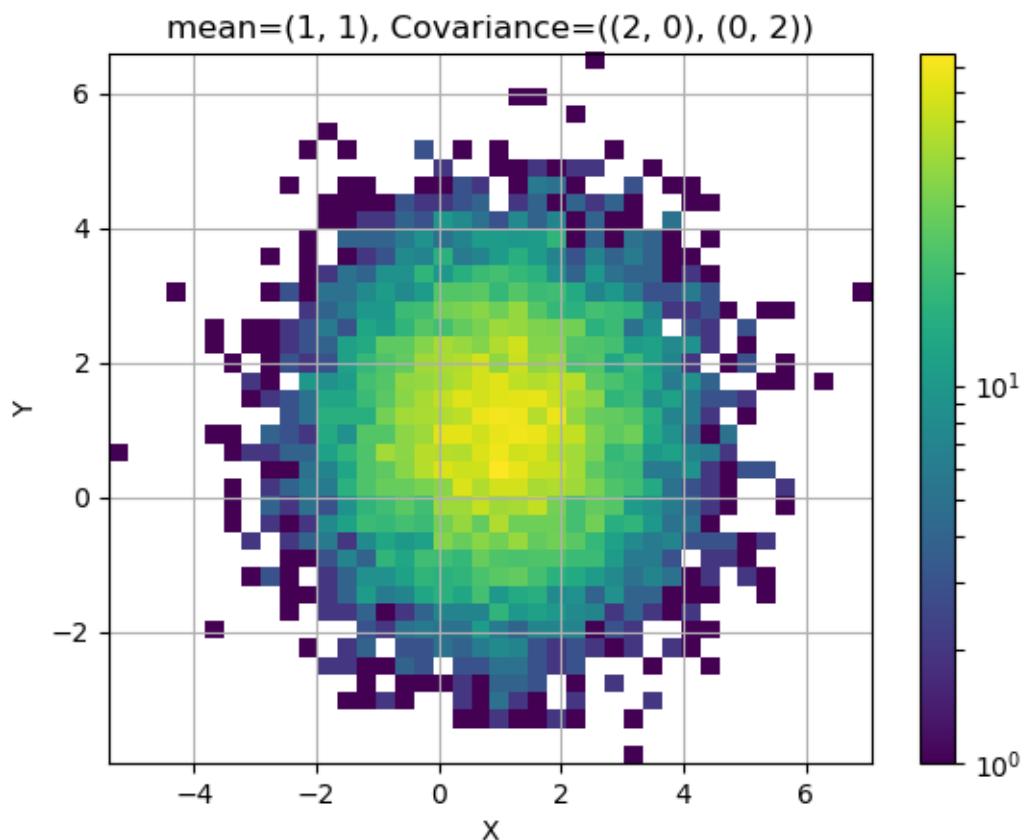


همان طور که مشاهده می شود هر چه واریانس زیاد تر شده است ارتفاع Histogram کم تر شده است و همین طور پهنهای آن بیشتر شده است.

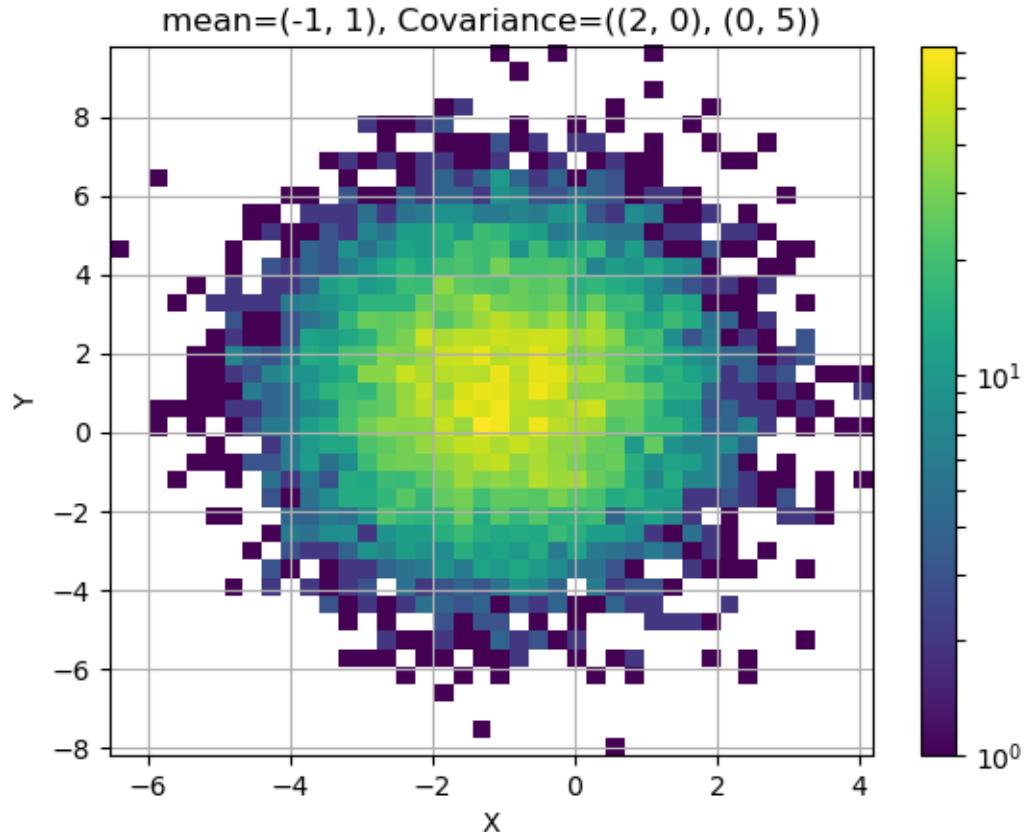
سوال ۲:

کد پایتونی که برای سوال دو نوشته‌ام همراه شکل‌های خروجی برنامه، در پوشه‌ی problem2 قرار دارد. در زیر سه شکل مشاهده می‌کنید که به ترتیب مربوط به بخش‌های a, b, c سوال است، از آنجایی که ماتریس Covariance دو در دو است برای نمایش Histogram، احتمال هر Bar را با colorbar نشان داده ام تا به خوبی قابل درک باشد. در ابتدا Histogram‌های سه بعدی با استفاده از تابع bar3d رسم کردم ولی به خوبی این روش اطلاعات را نشان نمی‌داد.

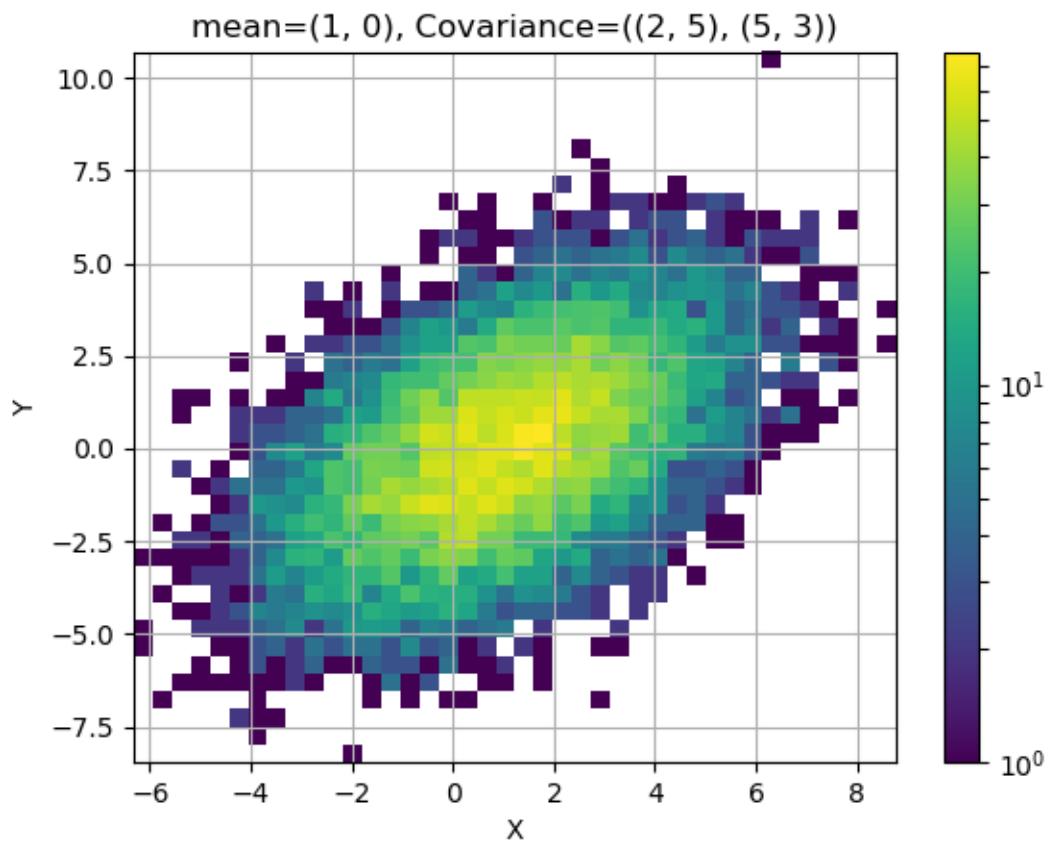
(a)



(b)



(c)



سوال ۳:

پاسخ سوال سه را در تصویر زیر مشاهده می کنید:

۳)

$$x = \emptyset, 1, 2$$

$$P(X = \emptyset) = \emptyset \cdot 25$$

$$P(X = 1) = \emptyset \cdot 5$$

$$P(X = 2) = \emptyset \cdot 25$$

a. $E[X] = \sum_{x=\emptyset}^2 x P(x) = \emptyset * \emptyset \cdot 25 + 1 * \emptyset \cdot 5 + 2 * \emptyset \cdot 25 = 1$

b. $E[X^2] = \sum_{x=\emptyset}^2 x^2 P(x) = (\emptyset)^2 * \emptyset \cdot 25 + (1)^2 * \emptyset \cdot 5 + (2)^2 * \emptyset \cdot 25 = 1.5$

c. $\text{Var}(X) = E[X^2] - (E[X])^2 = 1.5 - 1 = \emptyset \cdot 5$

d. $g(x) = 3x + 2$

$$E[3x + 2] = 3E[X] + 2 = 3 * 1 + 2 = 5$$

$$\begin{aligned} E[(3x + 2)^2] &= E[9x^2 + 12x + 4] = 9E[X^2] + 12E[X] + 4 \\ &= 9 * 1.5 + 12 * 1 + 4 = 29.5 \end{aligned}$$

$$\text{Var}(3x + 2) = E[(3x + 2)^2] - (E[3x + 2])^2 = 29.5 - 5^2 = 4.5$$

سوال ۴:

پاسخ سوال چهار را در تصویر زیر مشاهده می کنید.

$$4) \quad \forall i, j \quad i \neq j \quad \text{Corr}(x_i, x_j) = \phi \Rightarrow \text{Cov}(x_i, x_j) = \phi$$

$$\forall i \quad E[x_i] = \phi$$

$$\text{Var}(x_i) = 1$$

$$\begin{aligned} a) \quad \text{Corr}(X_1 + X_2, X_2 + X_3) &= \frac{\text{Cov}(X_1 + X_2, X_2 + X_3)}{\sqrt{\text{Var}(X_1 + X_2)} * \text{Var}(X_2 + X_3)} \\ &= \frac{\text{Cov}(X_1 + X_2, X_2) + \text{Cov}(X_1 + X_2, X_3)}{\sqrt{(\text{Var}(X_1) + \text{Var}(X_2)) * (\text{Var}(X_2) + \text{Var}(X_3))}} \\ &= \frac{\cancel{\text{Cov}(X_1, X_2)} + \cancel{\text{Cov}(X_2, X_2)} + \cancel{\text{Cov}(X_1, X_3)} + \cancel{\text{Cov}(X_2, X_3)}}{\sqrt{(\text{Var}(X_1) + \text{Var}(X_2)) * (\text{Var}(X_2) + \text{Var}(X_3))}} \\ &= \frac{1}{\sqrt{(1+1)(1+1)}} = \frac{1}{\sqrt{4}} = \frac{1}{2} \end{aligned}$$

$$\begin{aligned} b) \quad \text{Corr}(X_1 + X_2, X_3 + X_4) &= \frac{\text{Cov}(X_1 + X_2, X_3 + X_4)}{\sqrt{\text{Var}(X_1 + X_2)} * \text{Var}(X_3 + X_4)} \\ &= \frac{\text{Cov}(X_1 + X_2, X_3) + \text{Cov}(X_1 + X_2, X_4)}{\sqrt{(\text{Var}(X_1) + \text{Var}(X_2)) * (\text{Var}(X_3) + \text{Var}(X_4))}} \\ &= \frac{\cancel{\text{Cov}(X_1, X_3)} + \cancel{\text{Cov}(X_2, X_3)} + \cancel{\text{Cov}(X_1, X_4)} + \cancel{\text{Cov}(X_2, X_4)}}{\sqrt{(\text{Var}(X_1) + \text{Var}(X_2)) * (\text{Var}(X_3) + \text{Var}(X_4))}} = \phi \end{aligned}$$

سوال ۵:

پاسخ سوال پنج را در تصویر زیر مشاهده می کنید:

• اگر A یک ماتریس با دو eigenvalue متفاوت باشد ماتریس قطری D اینگونه تعریف می شود:

$$D = \begin{bmatrix} \lambda & 0 \\ 0 & \mu \end{bmatrix}$$

• اگر $M \times N$ دو ماتریس باشند: $\text{tr}(M \cdot N) = \text{tr}(N \cdot M)$

• هر ماتریس A را که دارای eigenvector های مستقل باشد را می توان قطری کرد

$$A = P \cdot D \cdot P^{-1}$$

ماتریس قطری که درجه های جوی تحلیل آن هستند.

ستون های ماتریس P همان eigenvectors هاست

eigen value

$$\text{tr}(A) = \lambda + \mu \quad \text{اینست}$$

$$\text{tr}(A) = \text{tr}(PDP^{-1}) = \text{tr}(P(DP^{-1}))$$

$$= \text{tr}((DP^{-1})P) = \text{tr}(D(P^{-1}P)) = \text{tr}(DI)$$

$$= \text{tr}(D) = \text{tr}\left(\begin{bmatrix} \lambda & 0 \\ 0 & \mu \end{bmatrix}\right) = \lambda + \mu$$

$$\det(A) = \lambda \cdot \mu \quad \text{اینست}$$

$$P(\lambda) = |\lambda I - A| = (\lambda - \lambda_1) \cdots (\lambda - \lambda_n) \quad \text{یعنی } P(\lambda) \text{ است:}$$

$$P(0) = |0I - A| = (-\lambda_1) \cdots (0 - \lambda_n)$$

$$(-1)^n \cdot |A| = (-1)^n \lambda_1 \cdot \lambda_2 \cdot \cdots \cdot \lambda_n$$

$$\Rightarrow |A| = \lambda_1 \cdot \lambda_2 \cdot \cdots \cdot \lambda_n$$

برای حالت سوال ۵ باید $\lambda_1, \lambda_2, \lambda_3$ عبارت خواسته شد اینست و شود

$$|A| = \lambda_1 \cdot \lambda_2 \cdot \lambda_3$$

سوال ۶:

پاسخ به سوال ۶ را در ۴ تصویر زیر مشاهده می‌کنید:

6)

$$A = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 2 & 2 \\ 0 & 9 & -5 \end{bmatrix}$$

a) $\det(A - I\lambda) = \emptyset$

$$\det \left(\begin{bmatrix} 4-\lambda & 0 & 0 \\ 0 & 2-\lambda & 2 \\ 0 & 9 & -5-\lambda \end{bmatrix} \right) = \emptyset \quad \xrightarrow{\text{Row operation}}$$

$$\det \left(\begin{bmatrix} 4-\lambda & 0 & 0 \\ 0 & 2-\lambda & 2 \\ 0 & 9 + \frac{-9}{2-\lambda}(2-\lambda) & -5-\lambda + \frac{9}{2-\lambda}(2) \end{bmatrix} \right) = \emptyset$$

$$\det \left(\begin{bmatrix} 4-\lambda & 0 & 0 \\ 0 & 2-\lambda & 2 \\ 0 & 0 & -5-\lambda - \frac{18}{2-\lambda} \end{bmatrix} \right) = \emptyset \quad \xrightarrow{\text{Row operation}}$$

$$\det \left(\begin{bmatrix} 4-\lambda & 0 & 0 \\ 0 & 2-\lambda & 0 \\ 0 & 0 & -5-\lambda - \frac{18}{2-\lambda} \end{bmatrix} \right) = \emptyset$$

$$\rightarrow (4-\lambda)(2-\lambda)\left(-5-\lambda - \frac{18}{2-\lambda}\right) = \emptyset$$

$$\rightarrow (4-\lambda)(-10 - 2\lambda + 5\lambda + \lambda^2 - 18) = \emptyset$$

$$\rightarrow (4-\lambda)(\lambda^2 + 3\lambda - 28) = 0 \rightarrow (4-\lambda)(\lambda+7)(\lambda-4) = 0$$

$$\rightarrow \boxed{\lambda = 4, 4, -7}$$

$$x = 4$$

$$\begin{bmatrix} 4 & \phi & \phi \\ \phi & 2 & 2 \\ \phi & 9 & -5 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = 4 \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

$$\begin{bmatrix} 4 & -4 & 0 \\ 0 & 2 & -4 \\ 0 & 9 & -4 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -2 & 2 \\ 0 & 9 & -9 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\xrightarrow{\left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & -2 & 2 & 0 \\ 0 & 9 & -9 & 0 \end{array} \right] \xrightarrow{\text{Row operation}} \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & -2 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right]} \rightarrow$$

$$\rightarrow \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \rightarrow V = \begin{bmatrix} \theta_1 \\ \theta_3 \\ \theta_3 \end{bmatrix}$$

متغيرات eigen value و eigen vector مترافقانه می باشند

$$\lambda = \underline{\Delta - 7}$$

$$\lambda = \boxed{A - I} \quad \left[\begin{array}{ccc|c} 4 & 1 & 1 & 1 \\ 0 & 2 & 2 & 2 \\ 0 & 0 & -5 & -5 \end{array} \right] \left[\begin{array}{c} \theta_1 \\ \theta_2 \\ \theta_3 \end{array} \right] = \left[\begin{array}{c} 1 \\ 2 \\ -5 \end{array} \right]$$

$$\rightarrow \begin{bmatrix} 4+7 & 0 & 0 \\ 0 & 2+7 & 2 \\ 0 & 7 & -5+7 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} 1 & 0 & 2 \\ 0 & 9 & 2 \\ 0 & 9 & 12 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\rightarrow \left[\begin{array}{ccc|c} 0 & 1 & 2 & \theta_3 \\ 0 & 0 & 9+2 & 0 \\ 0 & 0 & 9 & 0 \\ 0 & 0 & 2 & 0 \end{array} \right] \xrightarrow{\text{Row operation}} \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 9 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right] \rightarrow V = \begin{bmatrix} \theta \\ -\frac{2}{9}\theta_3 \\ \theta_3 \end{bmatrix}$$

است و عکس دو آن ها بر حسب عنصر سوم می شود $\theta_3 = \frac{2}{9}$ $\theta_2 = -\frac{2}{9}$ eigen value معلمات eigen vector می باشد - مرا برای در راهی است که عنصر اول آن متعادل صفر

با استفاده از مطلب اعداد زیر به دست آمده

eigen values : -7 , 4 , 4

eigen vectors :

$$\begin{bmatrix} 0.00000 & 0.00000 & 1.00 \\ 0.21693 & -0.70711 & 0.00 \\ -9.97619 & -0.70711 & 0.00 \end{bmatrix}$$

$$\lambda = -7 \quad \lambda = 4 \quad \lambda = 4$$

در هنگام حل سوال و دیگر متوسطه می شویم که eigen vector

داریم . یعنی در این سوال باید $\lambda = 7$ ، $\lambda = 4$ ، $\lambda = -7$ یک بردار مشخص

با عنوان eigen vector باشد که پارامتری هستند و

و با جایگزین کردن عدددهای مختلف در پارامترها و کنواری های متفاوتی به دست آید.

ولی مطلب آن ها را پارامتری حساب نمی کند بلکه پارامترها را با عدد جایگزین می کند

و همین طور آن ها را زیال می کند تا طولشان برابر یک شود.

$\lambda = 4, 4, -7$ دست آوردم eigen vectors و eigen values که من به دست آوردم

$$v^e = \begin{bmatrix} \theta_1 \\ \theta_3 \\ \theta_3 \end{bmatrix} \quad v^d = \begin{bmatrix} 1 \\ -\frac{2}{9}\theta_3 \\ \theta_3 \end{bmatrix}$$

b) هر ماتریس A را که دارای eigen vector های مستقل باشد را می توان قطری سازی (diagonalize) کرد و آن را به صورت $A = PDP^{-1}$ نوشت که D ماتریس قطری است که علایق های آن P همان eigen vector ها هستند و سینون همای eigen value

$$A = PDP^{-1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 5 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ 0 & 5 \end{bmatrix}$$

$$\rightarrow A = \begin{bmatrix} 2 & 3 \\ 0 & 5 \end{bmatrix}$$

سوال ۷:

سوال ۷ - قسمت Q1: پاسخ به این سوال را در در دو تصویر زیر مشاهده می‌کنید:

۷)

Q1)a) E9.7) $p(\text{error} | x) = \min [P(\omega_1 | x), P(\omega_2 | x)]$

E9.5) $P(\text{error}) = \int_{-\infty}^{+\infty} P(\text{error}, x) dx = \int_{-\infty}^{+\infty} P(\text{error} | x) P(x) dx$

if $x \in \omega_1 \rightarrow p(\text{error} | x) = \min [P(\omega_1 | x), P(\omega_2 | x)]$

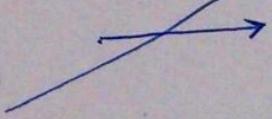
$\xrightarrow{P(\omega_1 | x) > P(\omega_2 | x)} p(\text{error} | x) = P(\omega_2 | x) .$

$P(\omega_1 | x) + P(\omega_2 | x) = 1 \rightarrow P(\omega_2 | x) = 1 - P(\omega_1 | x)$

$\xrightarrow{P(\omega_1 | x) > P(\omega_2 | x)} P(\omega_1 | x) > 1 - P(\omega_1 | x) \rightarrow \underline{2P(\omega_1 | x) > 1} .$

$P(\omega_2 | x) \geq P(\omega_2 | x) = P(\text{error} | x)$

$2P(\omega_1 | x) > 1$



$\rightarrow 2P(\omega_1 | x)P(\omega_2 | x) \Rightarrow P(\omega_2 | x) = P(\text{error} | x)$

$\rightarrow \int 2P(\omega_1 | x)P(\omega_2 | x) dx > \int P(\omega_2 | x) dx = \int P(\text{error} | x) dx$

$\rightarrow \int 2P(\omega_1 | x)P(\omega_2 | x) dx > \int P(\text{error} | x) dx$

$\underbrace{\sim P(\text{error} | x)}$ حبایی برای $2P(\omega_1 | x)P(\omega_2 | x)$ < نتیجه

Q1)b)

در قسمت قبل همین سوال نشان دادیم اگر $x \in \omega_1$ باشد

$$2p(\omega_1|x) > 1 \rightarrow p(\omega_1|x) > \frac{1}{2} \quad \cancel{\text{در نتیجه}}$$

$$\alpha p(\omega_1|x) > \frac{\alpha}{2} \quad \text{در نتیجه}$$

در نتیجه در قسمت قبل حاکم نشان دادیم $2p(\omega_1|x)p(\omega_2|x)$ بزرگتر از $p(\text{error}|x)$ است. دیگر نی تواین همین حیثی را نشان دهیم زیرا دو عبارت زیر را داریم

$$p(\omega_2|x) \geq p(\omega_2|x) = p(\text{error}|x)$$

$$\alpha p(\omega_1|x) > \frac{\alpha}{2}$$

و می توان گفت

$$\alpha p(\omega_1|x)p(\omega_2|x) > p(\omega_2|x) = p(\text{error}|x)$$

Q1)c)

$$p(\omega_1|x) + p(\omega_2|x) = 1$$

$$\therefore p(\text{error}|x) = p(\omega_2|x) \quad \forall x \in \omega_1 \quad \text{از}$$

$$p(\omega_2|x) \leq p(\omega_2|x)$$

$$p(\omega_2|x) \leq p(\text{error}|x)$$

$$\rightarrow$$

$$\underbrace{\omega \leq p(\omega_1|x) \leq 1}_{\rightarrow}$$

$$p(\omega_1|x)p(\omega_2|x) \leq p(\text{error}|x)$$

$$\int p(\omega_1|x)p(\omega_2|x)dx \leq \int p(\text{error}|x)dx$$

برای $\omega \in \omega_2$ هم بھی ترتیب عملی کنیم.

Q1)d)

در قسمت قبل چون $p(\omega_1|x) \leq 1$ بود تو انتیم نشان دهیم از $p(\omega_2|x) \leq p(\text{error}|x)$ می توان ب

$$p(\omega_1|x)p(\omega_2|x) \leq p(\text{error}|x)$$

چون $\beta > 1$ است می توان گفت

$$\beta p(\omega_1|x)p(\omega_2|x) \leq p(\text{error}|x)$$

$$p(\omega_2|x) \leq p(\text{error}|x)$$

سوال ۷ قسمت Q2:

پاسخ به این قسمت را در تصویر زیر مشاهده می‌کنید:

۷) Q2)

a) $P(\omega_1) + P(\omega_2) = 1 \rightarrow P(\omega_2) = 1 - P(\omega_1)$

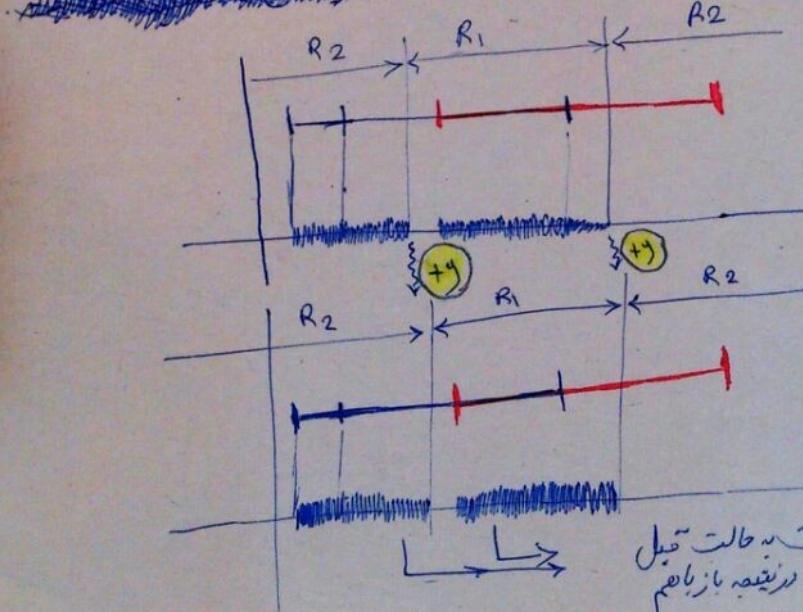
$$R(P(\omega_1)) = P(\omega_2) \int_{R_1}^{\omega_2} \rho(x|\omega_2) dx + P(\omega_1) \int_{R_2}^{\omega_1} \rho(x|\omega_1) dx$$

$$= (1 - P(\omega_1)) \int_{R_1}^{\omega_2} \rho(x|\omega_2) dx + P(\omega_1) \int_{R_2}^{\omega_1} \rho(x|\omega_1) dx$$

$$\xrightarrow[\text{مسنونه}\atop{\text{میانه}}]{\text{minimization}} \frac{d}{dP(\omega_1)} R(P(\omega_1)) = - \int_{R_1}^{\omega_2} \rho(x|\omega_2) dx + \int_{R_2}^{\omega_1} \rho(x|\omega_1) dx = 0$$

$$\xrightarrow[\text{مسنونه}\atop{\text{میانه}}]{} \int_{R_1}^{\omega_2} \rho(x|\omega_2) dx = \int_{R_2}^{\omega_1} \rho(x|\omega_1) dx$$

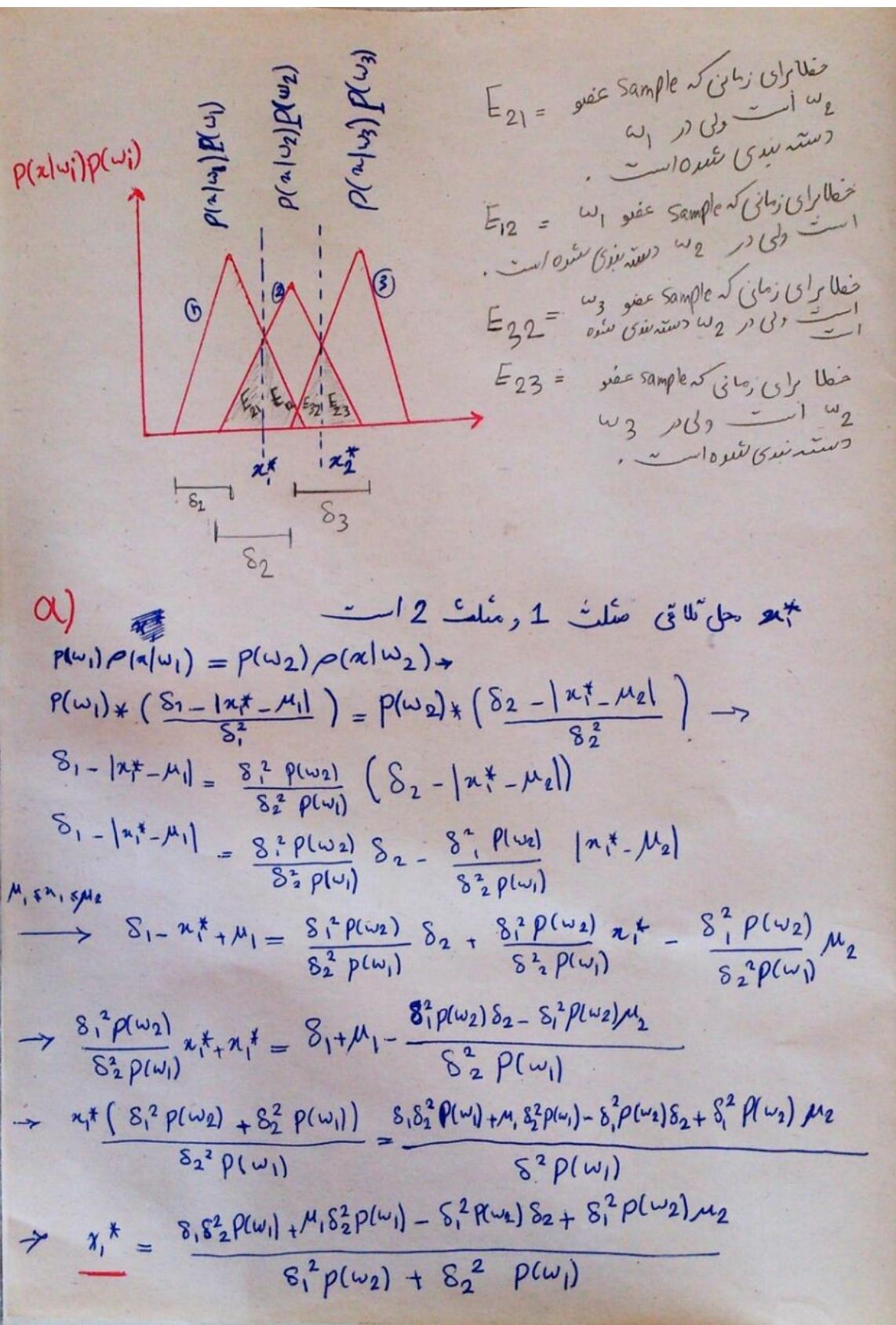
جواب unique زیر توجه نشود که در دو حالت مختلف فاصله‌ها همسو نشوند از



هر دو ناحیه نسبت به حالت قبل و اغذار شده در نظر نمایم باز باقی ماند

سوال ۷ قسمت Q4:

در شش تصویر زیر، پاسخ به Q4 را مشاهده می‌کنید:



برای x_2^* هم مانند x_1^* کنیم

$$p(\omega_2) + p(\alpha|\omega_2) = p(\omega_3) + p(\alpha|\omega_3)$$

$$\rightarrow p(\omega_2) \left(\frac{\delta_2 - (\mu_2^* - \mu_2)}{\delta_2^2} \right) = p(\omega_3) \left(\frac{\delta_3 - (\mu_2^* - \mu_3)}{\delta_3^2} \right)$$

محاسبات مانند x_1^* است متفاوت با این حالتغیراتی کنند

$$x_2^* = \frac{\delta_2 \delta_3^2 p(\omega_2) + \mu_2 \delta_3^2 p(\omega_2) - \delta_2^2 p(\omega_3) \delta_3 + \delta_2^2 p(\omega_3) \mu_3}{\delta_2^2 p(\omega_3) + \delta_3^2 p(\omega_2)}$$

b)

همان طور که در تقویری که در شکل قسمت a نیز بینید

چهار خطای E_{12}, E_{21}, E_{23} و E_{32} را توضیع داده ام و روی شکل

مشخص کردم.

$$E_{\text{total}} = E_{12} + E_{21} + E_{23} + E_{32}$$

$$\begin{aligned} E_{12} &= \int_{x_1^*}^{\mu_1 + \delta_1} p(\omega_1) \rho(x|w_1) dx = \int_{x_1^*}^{\mu_1 + \delta_1} p(\omega_1) \left(\frac{\delta_1 - |x_1^* - \mu_1|}{\delta_1^2} \right) dx \\ &= \int_{x_1^*}^{\mu_1 + \delta_1} p(\omega_1) \left(\frac{\delta_1 + \mu_1 - x_1^*}{\delta_1^2} \right) dx = p(\omega_1) \left(\frac{\delta_1 + \mu_1 - x_1^*}{\delta_1^2} \right) \int_{x_1^*}^{\mu_1 + \delta_1} dx \\ &= \frac{p(\omega_1)}{\delta_1^2} (\delta_1 + \mu_1 - x_1^*) * (\delta_1 + \mu_1 - x_1^*) = \frac{p(\omega_1) (\delta_1 + \mu_1 - x_1^*)^2}{\delta_1^2} \\ \rightarrow E_{12} &= \frac{p(\omega_1) (\delta_1 + \mu_1 - x_1^*)^2}{\delta_1^2} \end{aligned}$$

به من ترتیب مقدار E_{23}, E_{32}, E_{21} را حسابی کنید که زیر

مقدار آنها را مسأله دیگر کنید:

$$E_{21} = \frac{p(\omega_2) (\mu_2 - \delta_2 + x_1^*)^2}{\delta_2^2}$$

$$E_{23} = \frac{p(\omega_2) (\mu_2 - \delta_2 - x_2^*)^2}{\delta_2^2}$$

$$\begin{aligned} E_{32} &= \frac{p(\omega_3) (-\mu_3 + \delta_3 + x_2^*)^2}{\delta_3^2} = \\ &= \frac{(1 - p(\omega_1) - p(\omega_2)) (-\mu_3 + \delta_3 + x_2^*)^2}{\delta_3^2} \end{aligned}$$

$$p(\omega_3) + p(\omega_2) + p(\omega_1) = 1$$

$$E = E_{21} + E_{12} + E_{23} + E_{32}$$

$$= \frac{P(\omega_2) (\delta_2 - \mu_2 + x_1^*)^2}{\delta_2^2} + \frac{P(\omega_1) (\delta_1 + \mu_1 - x_1^*)^2}{\delta_1^2}$$

$$+ \frac{P(\omega_2) (\mu_2 - \delta_2 - x_2^*)^2}{\delta_2^2} + \frac{(1-P(\omega_2) - P(\omega_1)) (-\mu_3 + \delta_3 + x_2^*)^2}{\delta_3^2}$$

برای تعیین مسئلتی که باید $P(\omega_2)$ و $P(\omega_1)$ نسبت به E را \min_{max} کنیم.

و مساوی صفر باشد:

$$\frac{\partial E}{\partial P(\omega_1)} = 0$$

$$\rightarrow \frac{(\delta_1 + \mu_1 - x_1^*)^2}{\delta_1^2} - \frac{P(\omega_1) (-\mu_3 + \delta_3 + x_2^*)^2}{\delta_3^2} = 0$$

$$\rightarrow \frac{(\delta_1 + \mu_1 - x_1^*)^2}{\delta_1^2} = \frac{(-\mu_3 + \delta_3 + x_2^*)^2}{\delta_3^2}$$

$$\rightarrow \boxed{\frac{\delta_1 + \mu_1 - x_1^*}{\delta_1} = \frac{-\mu_3 + \delta_3 + x_2^*}{\delta_3}}$$

$$\frac{\partial E}{\partial P(\omega_2)} = 0 \rightarrow \frac{(\delta_2 - \mu_2 + x_1^*)^2}{\delta_2^2} + \frac{(\mu_2 - \delta_2 - x_2^*)^2}{\delta_2^2} - \frac{(-\mu_3 + \delta_3 + x_2^*)^2}{\delta_3^2} = 0$$

$$\rightarrow \boxed{\frac{(-\mu_3 + \delta_3 + x_2^*)^2}{\delta_3^2} = \frac{(\delta_2 - \mu_2 + x_1^*)^2}{\delta_2^2} + \frac{(\mu_2 - \delta_2 - x_2^*)^2}{\delta_2^2}}$$

دو کام مرز زنگ دو معادله داشتند که در صورت حل کردن آن مورث پارامتری برای این مسئله برسی شود.

C)

از دو مدل در کار تجزیه بخش ب استفاده کنیم

μ_1	S_1	μ_2	S_2	μ_3	S_3
\emptyset	1	$\frac{1}{2}$	$\frac{1}{2}$	1	1

$$\begin{aligned}
 \textcircled{1} \quad & \frac{\mu_1 + S_1 - x_1^*}{S_1} = \frac{-\mu_3 + S_3 + x_2^*}{S_3} \\
 \rightarrow & \frac{0+1-x_1^*}{1} = \frac{-1+1+x_2^*}{1} \rightarrow 1-x_1^* = x_2^* \\
 \textcircled{2} \quad & \frac{(-\mu_3 + S_3 + x_2^*)^2}{S_3^2} = \frac{(S_2 - \mu_2 + x_2^*)^2}{S_2^2} + \frac{(\mu_2 - S_2 - x_2^*)^2}{S_2^2} \\
 \rightarrow & \frac{(1-1+x_2^*)^2}{1^2} = \frac{(\frac{1}{2}-\frac{1}{2}+x_2^*)^2}{(\frac{1}{2})^2} + \frac{(\frac{1}{2}+\frac{1}{2}-x_2^*)^2}{(\frac{1}{2})^2} \\
 (\alpha_2^*)^2 &= 4(x_2^*)^2 + 4(1-x_2^*)^2 \\
 \rightarrow (\alpha_2^*)^2 &= 4 \underbrace{(1-x_2^*)^2}_{\{ } + 4(1-x_2^*)^2 \\
 \rightarrow (\alpha_2^*)^2 &= 8(1-x_2^*)^2 \\
 \rightarrow x_2^* &= 2\sqrt{2}(1-x_2^*) \rightarrow x_2^* = \frac{2\sqrt{2}}{1+2\sqrt{2}} \\
 \rightarrow & \boxed{x_2^* = \phi.738796121} \\
 & \boxed{x_1^* = \phi.261203875}
 \end{aligned}$$

4)

E. با توجه به اعلامات عتمت ایگز برسن می‌کند:

از عصمت ط کوئیل کا واداریم :

$$E = \frac{p(\omega_2)(s_2 - \mu_2 + x_1^*)^2}{s_2^2} + \frac{p(\omega_1)(s_1 + \mu_1 - x_1^*)^2}{s_1^2}$$

$$+ \frac{p(\omega_2)(\mu_2 - s_2 - x_2^*)^2}{s_2^2} + \frac{(1-p(\omega_2) - p(\omega_1))(-\mu_3 + s_3 + x_2^*)^2}{s_3^2}$$

اطلاعات غسته را بازگردانی کنید

$$\rightarrow E = \frac{P(\omega_2) \left(\frac{1}{2} - \frac{1}{2} + 0.2612 \right)^2}{\left(\frac{1}{2} \right)^2} + \frac{P(\omega_1) \left(1 - 0 - 0.2612 \right)^2}{\left(1 \right)^2}$$

$$+ \frac{P(\omega_2) \left(\frac{1}{2} - \frac{1}{2} \cos \theta \right)^2 - 0.7387}{\left(\frac{1}{2} \right)^2}$$

$$+ \frac{P(\omega_3)(-1 + 1 + 0.7387)^2}{(1)^2}$$

$$\rightarrow E = 0.2729 p(w_2) + 0.5458 p(w_1) + p(w_2) * \phi.5456 \\ + 0.5456 p(w_3)$$

$$E = 0.5458_p(w_1) + 0.8185_p(w_2) + 0.5456_p(w_3)$$

Bayes risk بحر اسے با برترین minimax risk

پروژه‌ی برنامه‌نویسی :Q1 – Computer Project

کدهای نوشته شده برای این سوال در فایل computerExercise1.py، در پوشه‌ی computer exercises 1 قرار دارد.
در این فایل ۴ تابع زیر موجود است که به خوبی کامنت‌گذاری شده‌اند.

بخش (a)

در این بخش پروژه‌ی برنامه‌نویسی اول، خواسته‌شده است که تابع‌ای برای تولید تعدادی نمونه، از توزیع نرمال چند متغیر نوشته شود. کد پایتون کامنت‌گذاری شده را در تصویر زیر مشاهده می‌کنید:

```
# Computer exercise 1-a
#####
def d_normal_distribution(mu, covMat, n):
    """
    D-dimensional normal distribution function,
    it draws samples from a D-dimensional normal distribution.

    :param
        (mu)mean vector,
        (covMat)covariance matrix,
        (n)number of samples that should be drawn.
    :return
        each entry out[i,j,...,:] is an D-dimensional value drawn from normal distribution.

    An example of this function:
        d_normal_distribution(mu=[0,0],covMat=[[1, 0], [0, 100]], n=10)
    """
    import numpy as np
    # Use multivariate normal distribution of numpy package
    return np.random.multivariate_normal(mean=mu, cov=covMat, size=n, check_valid='ignore')
```

در بخش doc-string تابع، ورودی‌ها و خروجی‌ها توضیح داده شده‌اند. برای انتخاب نمونه‌ها از توزیع نرمال، از تابع numpy پکیج multivariate_normal استفاده شده است. در تصویر زیر یک نمونه از اجرای تابع را می‌بینید:

```
External Libraries
169
170     # Test Computer exercise 1-a
171     print(d_normal_distribution(mu=[0,2,0],covMat=[[1, 0, 2],[6,0, 100],[6,50, 100]], n=10))
172
173 draw_decision_b...
computerExercise1
/home/bat/anaconda3/bin/python "/home/bat/Dropbox/codes/pythonCode/patternRecognition/computer exercises 1/computerExercise1.py"
[[ 2.18715803 14.73684144 9.62296628]
 [-0.20934715 3.87190818 1.31697558]
 [ 1.70171061 6.51871136 14.19097881]
 [-0.16350204 14.5250572 14.00759132]
 [-0.70592681 11.75947086 7.46215264]
 [ 0.49449156 -6.93110844 -9.95540211]
 [ 0.28455753 8.26241457 11.30585865]
 [-1.72847366 -3.26347622 9.12906775]
 [-0.74176941 6.2527306 7.9728971 ]
 [ 0.41444117 -1.37853021 -0.46952145]]
```

(b) بخش

در این بخش خواسته شده است که یک تابع برای محاسبه discriminant function با توجه به ورودی های خواسته شده نوشته شود، ولی من به جای دریافت یک ورودی تابع را طوری نوشتہ ام که تابع می تواند هر تعداد ورودی را دریافت کند و حاصل discriminant function را برای همه آن هارا در یک لیست برگرداند. در زیر کد مربوط به این بخش را مشاهده می کنید. کد کاملاً کامنت گذاری شده است و ورودی ها و خروجی ها در آن شرح داده شده اند:

```
test.py x computerExercise1.py x
22 ##### discriminant function #####
23 # Computer exercise 1-b
24
25 def discriminant_function(mu, covariance, priorProbability, sampleVector):
26     """
27     This function gets an vector of samples and calculates
28     value of discriminant function for each.
29
30     :param mu: a vector of means (list, couple, ...)
31     :param covariance: covariance matrix (list, couple, ...)
32     :param sampleVector: vectors of samples (list, couple, ...)
33     :param priorProbability: prior probability (list, couple, ...)
34     :return: A vector of discriminant of each sample in the sample vector (np.array)
35     """
36     import numpy as np
37
38     # Convert inputs to numpy.matrix()
39     Mu = np.matrix(mu)
40     Cov = np.matrix(covariance)
41     inputVector = np.matrix(sampleVector)
```

```
test.py x computerExercise1.py x
45 # Dimension of given normal distribution
46 d = Cov.shape[0]
47
48 # Inverse of covariance matrix
49 invCov = np.linalg.inv(Cov)
50
51 # The part of discriminant function which is equal for different input samples
52 discriminant_value_part2 = -(d/2)*np.log(2*np.pi)-\
53     (1/2)*np.log(np.linalg.det(Cov))+np.log(priorProbability)
54
55 # Output
56 outputVector = []
57
58 # Calculating value of discriminant function for all input samples
59 for i in range(inputVector.shape[0]):
60     # Calculating value of discriminant function for input sample i
61     sampleOutput = (-(1/2) * (inputVector[i] - Mu) * invCov * (inputVector[i] - Mu).transpose())+\n62         discriminant_value_part2
63     outputVector.append(sampleOutput)
64
65 #return an array of discriminant function
66 return np.array(outputVector)
```

در کد بالا از پکیج numpy جهت جمع و ضرب ماتریس ها، محاسبه دترمینان و معکوس ماتریس ها استفاده شده است. تابع discriminant function را در شکل زیر مشاهده می کنید:

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^t \boldsymbol{\Sigma}_i^{-1}(\mathbf{x} - \boldsymbol{\mu}_i) - \frac{d}{2} \ln 2\pi - \frac{1}{2} \ln |\boldsymbol{\Sigma}_i| + \ln P(\omega_i). \quad (47)$$

به عنوان مثال در تصویر زیر یک نمونه از فراخوانی و خروجی تابع را مشاهده می کنید:

```

173 # Test Computer exercise 1-b
174 mu_1 = [0.1, 0.1]
175 cov_1 = [[1.1, 0.3], [0.2, 0.8]]
176 priori_1 = 0.5
177 print(discriminant_function(mu_1, cov_1, priori_1, [[0.1,0.1], [5,5]]))
178

computerExercise1
/home/bat/anaconda3/bin/python "/home/bat/Dropbox/codes/pythonCode/patternRecognition/computer exercises 1/computerExercise1.py"
[[[-2.43179878]]]
[[-22.92814024]]]

```

بخش (c)

این بخش از سوال، تابع ای برای پیدا کردن فاصله اقلیدوسی (Euclidean distance) بین دو نقطه را خواسته است. فاصله اقلیدوسی دو نقطه به شکل زیر محاسبه می شود:

$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

که کد نوشته شده و حاوی کامنت را در شکل زیر مشاهده می کنید.(ورودی ها و خروجی ها در تصوی رشرح داده شده اند):

```

#####
# Computer exercise 1-c
#####
def euclidean_distance(x, y):
    """
    This function calculates euclidean distance between two
    arbitrary points.
    :param x: a point that is a list or couple like [x1, x2, ..., xn]
    :param y: a point that is a list or couple like [y1, y2, ..., yn]
    :return:
    """
    import numpy as np
    from math import sqrt

    # Convert point to numpy array
    x_array = np.array(x)
    y_array = np.array(y)

    return sqrt(np.dot(x_array-y_array, x_array-y_array))
#####

```

در شکل زیر یک نمونه از فراخوانی این تابع را بر روی یک ورودی را می بینید:

```
179 # Test Computer exercise 1-c
180 print(euclidean_distance([1, 2, 3, 4],[1, 2, 4, 3]))
181
/home/bat/anaconda3/bin/python "/home/bat/Dropbox/codes/pythonCode/patternRecognition/computerExercise1.py
1.4142135623730951
```

(d) بخش

این بخش، یک تابع برای محاسبه فاصله Mahalanobis یک نقطه تا یک توزیع نرمال را خواسته است. فاصله Mahalanobis را اینگونه محاسبه می کنند:

$$D_M(\vec{x}) = \sqrt{(\vec{x} - \vec{\mu})^T S^{-1} (\vec{x} - \vec{\mu})}$$

که μ توزیع نرمال است و S ماتریس Co-variance است. کد کامنت گذاری شده این بخش را در تصویر زیر می بینید:

```
test.py x computerExercise1.py x
 87
 88 ##### Computer exercise 1-d #####
 89 # Computer exercise 1-d
 90 #####
 91 def mahalanobis_distance(mu, covariance, x):
 92     """
 93         This function calculates Mahalanobis distance between
 94         a point and mean of a normal distribution
 95         :param mu: mean of the normal distribution
 96         :param covariance: covariance of the normal distribution
 97         :param x: input point that we want to find the distance of it
 98         :return: a scalar value that indicate Mahalanobis distance
 99     """
100    import numpy as np
101    from math import sqrt
102
103    # Convert inputs to numpy.matrix()
104    Mu = np.matrix(mu)
105    Cov = np.matrix(covariance)
106    point = np.matrix(x)
107
108    # Calculate and return Mahalanobis distance
109    return sqrt((point - Mu) * np.linalg.inv(Cov) * (point - Mu).transpose())
mahalanobis... 
```

ورودی ها و خروجی ها در تصویر بالا شرح داده شده اند. برای نشان دادن درست کار کردن این تابع، می توانید به یک نمونه از فراخوانی تابع در شکل زیر توجه کنید:

The screenshot shows a Jupyter Notebook interface with two main sections: a file tree on the left and a code editor/terminal on the right.

File Tree:

- ~lock.Exercsie1.odt#
- Exercsie1.odt
- test.py
- External Libraries

Code Editor/Terminal:

```
182 # Test Computer exercise 1-d
183 mu0 = [3, 3]
184 cov0 = [[1.1, 0.3], [0.3, 1.9]]
185 x = [1.0, 2.2]
186 # Result should be: 1.9162463307205573
187 print(mahalanobis_distance(mu0, cov0, x))
```

Run computerExercise1

```
/home/bat/anaconda3/bin/python "/home/bat/Dropbox/codes/pythonCode/patternRecognit:
1.9162463307205573
```

برای پاسخگویی به قسمت‌های مختلف این سوال به تابع‌های زیر نیاز است :

تابع covariance_matrix : این تابع یک سری نمونه تصادفی ایجاد شده را می‌گیرد و ماتریس covariance آن را محاسبه می‌کند. همان طور که در شکل زیر توضیح داده شده است ورودی samples یک لیست یا هر چیز لیست مانندی است که هر ستون آن feature است و هر سطر آن یک observation و نمونه است. ورودی‌ها، خروجی و نحوه محاسبه ماتریس covariance در کامنت‌های کد توضیح داده شده اند.

```

60 ######
61 # Computer exercise 2- needed functions
62 #####
63 def covariance_matrix(samples):
64     """
65     This function gets some samples and return its covariance
66     :param samples: each col represents a variable,
67     with observations in the row
68     :return: Corresponding covariance matrix
69     """
70     import numpy as np
71     samplesArray = np.asarray(samples)
72     samplesArray = np.transpose(samplesArray)
73
74     # Cov function calculates covariance it this way:
75     # X_new(k) = X(k) - mu
76     # B = [X_new(1), X_new(2), ..., X_new(n)]
77     # Cov = (1/(n-1))*B*transpose(B)
78     return np.cov(samplesArray).tolist()
79

```

تابع mean_vector ورودی مانند تابع بالا می‌گیرد و میانگین هر feature را حساب می‌کند.

```

81 def mean_vector(X):
82     """
83     :param X: each col represents a variable,
84     with observations in the row
85     :return: return a list that contains mean of each col
86     """
87     import numpy as np
88     means = np.mean(X, axis=0)
89     return means.tolist()
90

```

تابع `mean_and_covariance` : بعد از نوشتتن تابع هایی برای محاسبه ماتریس covariance و میانگین هر feature تابع ای نوشتم که observation ها را برای داده هایی متعلق به یک کلاس را می گیرد و با استفاده از دو تابع ای قبلی که در بالا توضیح دادم میانگین و ماتریس کواریانس داده ها را پیدا می کند و مشخصات توزیع نرمال را به صورت یک dictionary باز می گرداند که کلید means وکتور میانگین است و کلید covariance ماتریس کواریانس اشاره می کند. در شکل زیر کد کامنت گذاری شده تابع مورد بحث را مشاهده می کنید:

```

92     def mean_and_covariance(samples):
93         """
94             This function gets some samples and then it finds
95             corresponding Covariance and mean of their normal distribution.
96             :param samples: samples.each col represents a variable,
97                 with observations in the row
98             :return: a dictionary with keys: {'means': , 'covariance'}
99             """
100            dic = {}
101            dic['means'] = mean_vector(samples)
102            dic['covariance'] = covariance_matrix(samples)
103            return dic
104

```

تابع `dischotomizer` : این تابع یک sample و مشخصات توزیع های نرمال کلاس های مختلف را که توسط تابع قبل تولید شده اند را می گیرد. تعداد توزیع های نرمال داده شده به تابع می تواند بیش از دو توزیع نرمال هم باشد. سپس با استفاده از discriminant_function مسیله هی یک برنامه نویسی، مقدار $(x)g$ را برای تمام توزیع های نرمال حساب می کند و اینگونه مشخص می کند sample ورودی متعلق به کدام است و کلاسی را که $(x)g$ آن ماکزیمم است را به عنوان پیشビینی برای ورودی x مشخص می کند. در شکل زیر کد این تابع که حاوی کامنت هم هست را مشاهده می کنید:

```

94     def dichotomizer( listOfnormalDis, x):
95         """
96             Labeling sample x by a label which has maximum g
97             :param listOfnormalDis: is list that contains dictionaries,
98                 each dictionary contains these keys: {'means', 'covariance', 'name', 'prior'} that are
99                 properties of a normal distribution
100            :param x: an input which this function determine it belongs to normal distribution
101                one or two
102            :return: If x belongs to normal distribution y output is
103                like: (x, y['name'])
104                """
105
106        # Calculate value of discriminant function for sample x and each normal distribution
107        g = []
108        for normalDist_i in listOfnormalDis:
109            if normalDist_i['prior'] != 0:
110                g.append(discriminant_function(mu=normalDist_i['means'], covariance=normalDist_i['covariance'],
111                                                priorProbability=normalDist_i['prior'], sampleVector=x))
112            else:
113                g.append(float('-inf'))
114
115        # Determine sample x belongs to which class
116        return (x, listOfnormalDis[g.index(max(g))]['name'])

```

تابع دیگری که نیاز داریم، تابع empirical_training_error است که دو ورودی می‌گیرد. همان طور که در کامنت‌های شکل زیر مشاهده می‌کنید ورودی اول دیکشنری از کلاس‌ها است که هر کلاس در آن جداگانه مشخص شده‌اند. ورودی دوم sample‌ها برچسب‌هایی هستند که طراحی کرده‌ایم آنها را تولید کرده است. تابع تعداد برچسب‌های اشتباه را با توجه به برچسب درست می‌شمارد و درصد خطأ را مشخص می‌کند. در زیر کد این تابع را می‌بینید:

```

def empirical_training_error(labeledSamples, predictedSampled):
    """
    This function counts number of misclassified samples
    :param labeledSamples: is a dictionary that contains n keys w1, w2, ..., wn
        each wi is assigned to a list of samples
    :param predictedSampled: corresponding prediction for each labeled sample,
        It's a list likes this: [ [sample1, predicted class], ..., [sample2, predicted class]
    :return: percentage of misclassified samples
    """

    # check predicted label to sample i is True or not
    misclassified = 0
    for i in predictedSampled:
        # Check samples exist where prediction says or not
        if i[0] not in labeledSamples[i[1]]:
            misclassified = misclassified + 1

    return misclassified / len(predictByclassifier) * 100

```

تابع دیگری که برای پاسخ گویی به سوال ها نیاز داریم تابع ای است که Bhattacharyya bound را محاسبه کند.
برای این کار تابع زیر را نوشته ام:

```
148     def bhattacharyya_bound(normalDist1, normalDist2):
149         """
150             This measure shows similarities between two
151             distribution.
152             :param normalDist1: is dictionary that contains
153                 these keys {'means', 'covariance', 'name', 'prior'}
154             :param normalDist2: is dictionary that contains
155                 these keys {'means', 'covariance', 'name', 'prior'}
156             :return: return a scalar
157         """
158         import numpy as np
159
160         # Convert data to proper form
161         mean1 = np.asmatrix(normalDist1['means'])
162         mean2 = np.asmatrix(normalDist2['means'])
163         cov1 = np.asmatrix(normalDist1['covariance'])
164         cov2 = np.asmatrix(normalDist2['covariance'])
165
166         # Average of two covariances
167         covAve = (cov1 + cov2) / 2
168
168         # Subtract of means
169         meanSub = mean2 - mean1
170
171         # Calculate the bound
172         bound = (np.sqrt(normalDist1['prior'] * normalDist2['prior'])) * (
173             np.exp(-1 * ((1/8) * (meanSub * np.linalg.inv(covAve) * np.transpose(meanSub)) +
174                         (1/2) * np.log(
175                             np.linalg.det(covAve)/np.sqrt(np.linalg.det(cov1) *
176                                             np.linalg.det(cov2))
177                         ))))
178
179         #return bound
180         return bound.tolist()[0][0]
```

این تابع دو دیکشنری به عنوان ورودی می‌گیرد که هر دیکشنری ورودی شامل کلیدهای covariance, prior, means و name است که مشخصات یک توزیع تصادفی نرمال است. و طبق تابع ریاضی باهاتچرایی، بازد خطا را محاسبه می‌کند.

اکنون که تابع های مورد نیاز برای سوال دو را معرفی کردم به بخش های مختلف سوال پاسخ می دهم.

بخش b و a)

برای این دو بخش ابتدا دیتاهای را کمی تغییر می دهیم زیرا این بخش گفته است فقط از feature x_1 استفاده شود و مقدار priori ها را هم سمت می کنیم. و بعد از آن تابع های بالا را طبق کد شکل زیر فرا می خوانیم تا ابتدا توزیع های نرمال را پیدا کنیم و بعد از آن برچسب داده ها را مشخص کنیم و بعد از در نهایت خطای خطا را بیابیم. در کد کامنت گذاری شده ای شکل زیر مرحلی را که گفتم را مشاهده می کنید:

```
computerExercise2.py x
183 ######
184 # Computer exercise 2- b|
185 #####
186 from copy import deepcopy
187
188 # Prepare data for computer exercise 2-b
189 # Make a copy of classes and just keep feature x1
190 # of samples.
191 useFirstNfeature = 1
192
193 classesFx1 = deepcopy(classes)
194 for key in classesFx1:
195     # Iterate all samples of class key
196     for index, element in enumerate(classesFx1[key]):
197         # just keep first useFirstNfeature
198         classesFx1[key][index] = element[0:useFirstNfeature]
199
200 # Make a copy of data and just keep feature x1
201 dataFx1 = deepcopy(data)
202 for index, x in enumerate(dataFx1):
203     dataFx1[index] = x[0:useFirstNfeature]
204 # Because of classifying w1 and w2, just use w1 and w2 samples
205 dataFx1 = dataFx1[0:20]
206
207 # Priorities of classes
208 priorities = {'w1':0.5, 'w2':0.5, 'w3':0}
209
210 # Find normal distribution of each class
```

```

209
210     # Find normal distribution of each class
211     classOrder = {}
212     normalDists = []
213     for key in classesFx1:
214         # Find a normal distribution for class key
215         normalDists.append(_mean_and_covariance(classesFx1[key]))
216         # Set class name of normal distribution
217         normalDists[len(normalDists)-1]['name'] = key
218         # Set class prior of normal distribution
219         normalDists[len(normalDists)-1]['prior'] = priories[key]
220         classOrder[key] = len(normalDists)-1
221
222     # Find label of each class
223     predictByclassifier = []
224     for x in dataFx1:
225         # Determine class of unlabeled data x
226         # and add the result of classifier to a list
227         predictByclassifier.append(_dichotomizer(normalDists,x))
228
229     # Find empirical training error
230     print("Computer exercises 2-b:")
231     print("Error: ",
232           empirical_training_error(classesFx1,predictByclassifier))
233

```

خروجی اجرای کد بالا را در تصویر زیر مشاهده می‌کنید:

```

/home/bat/anaconda3/bin/python "/home/bat/Dropbox/codes/pyth
Computer exercises 2-b:
Error:  30.0

```

(C) بخش

این بخش Bhattacharyya bound classifier را برای بخش a و b سوال خواسته است. به همین منظور در ادامه می‌کد شکل بالا تابع محاسبه‌ی باند را بدین شکل فرمای خوانیم:

```

235     #####
236     # Computer exercise 2- c
237     #####
238     print('Bhattacharyya bound for w1 & w2:{:f}\n'.format(
239           bhattacharyya_bound(normalDists[classOrder['w1']], normalDists[classOrder['w2']])))
240

```

و حاصل اجرای کد بالا را در تصویر زیر مشاهده می‌کنید:

Bhattacharyya bound for w1 & w2: 0.473999

(بخش d)

این بخش خواسته است کارهای بخش b, a و c را دوباره انجام دهیم ولی به جای ویژگی اول از ویژگی اول و دوم استفاده کنیم.

برای این کار کافی است متغیر firstNfeature را برابر دو قرار دهیم تا فقط دو ویژگی اول داده‌ها را نگه دارد و تابع ها را دقیقاً مانند قسمت قبل فرا بخوانیم:

```
241 #####  
242 # Computer exercise 2- d  
243 #####  
244 from copy import deepcopy  
245  
246 # Prepare data for computer exercise 2-d  
247 # Make a copy of classes and just keep features x1,x2  
248 # of samples.  
249 useFirstNfeature = 2  
250 |  
251 classesFx1 = deepcopy(classes)  
252 for key in classesFx1:  
253     # Iterate all samples of class key  
254     for index, element in enumerate(classesFx1[key]):  
255         #just keep feature 1  
256         classesFx1[key][index] = element[0:useFirstNfeature]  
257  
258 # Make a copy of data and just keep feature x1  
259 dataFx1 = deepcopy(data)  
260 for index, x in enumerate(dataFx1):  
261     dataFx1[index] = x[0:useFirstNfeature]  
262 # Because of classifying w1 and w2, just use w1 and w2 samples  
263 dataFx1 = dataFx1[0:20]  
264  
265 # Priorities of classes  
266 priorities = {'w1': 0.5, 'w2': 0.5, 'w3': 0}
```

```

267 # Find normal distribution of each class
268 classOrder = {}
269 normalDists = []
270 for key in classesFx1:
271     # Find a normal distribution for class key
272     normalDists.append(mean_and_covariance(classesFx1[key]))
273     # Set class name of normal distribution
274     normalDists[len(normalDists)-1]['name'] = key
275     # Set class prior of normal distribution
276     normalDists[len(normalDists)-1]['prior'] = priorities[key]
277     classOrder[key] = len(normalDists)-1
278
279 # Find label of each class
280 predictByclassifier = []
281 for x in dataFx1:
282     # Determine class of unlabeled data x
283     # and add the result of classifier to a list
284     predictByclassifier.append(dichotomizer(normalDists,x))
285
286 # Find empirical training error
287 print("Computer exercises 2-d:")
288 print("Error: ", empirical_training_error(classesFx1,predictByclassifier))
289
290 print('Bhattacharyya bound for w1 & w2:{:f}\n'.format(
291     bhattacharyya_bound(normalDists[classOrder['w1']], normalDists[classOrder['w2']])))

```

این قسمت دقیقا مانند بخش firstNfeature تغییر کرده است. حاصل اجرای کد بالا را در تصویر زیر مشاهده می کنید:

```

Computer exercises 2-d:
Error: 45.0
Bhattacharyya bound for w1 & w2:0.460466

```

بخش e)

این بخش از سوال برخلاف بخش های قبل خواسته است از سه ویژگی sample ها برای دسته بندی استفاده کنیم. کد این بخش هم دقیقا مثل بخش قبل است با این تفاوت که مقدار متغیر firstNfeature را برابر ۳ قرار می دهیم تا از تمام feature ها استفاده شود. در شکل زیر این کامنت این بخش را می بینید:

```
295 #####  
296 # Computer exercise 2- e  
297 #####  
298 from copy import deepcopy  
299  
300 #####  
301 # Prepare data for computer exercise 2-e  
302 # Make a copy of classes keep features x1,x2,x3  
303 # of samples.  
304 useFirstNfeature = 3  
305  
306 classesFx1 = deepcopy(classes)  
307 for key in classesFx1:  
308     # Iterate all samples of class key  
309     for index, element in enumerate(classesFx1[key]):  
310         #just keep feature 1  
311         classesFx1[key][index] = element[0:useFirstNfeature]  
312  
313 # Make a copy of data and just keep feature x1  
314 dataFx1 = deepcopy(data)  
315 for index, x in enumerate(dataFx1):  
316     dataFx1[index] = x[0:useFirstNfeature]  
317 # Because of classifying w1 and w2, just use w1 and w2 samples  
318 dataFx1 = dataFx1[0:20]  
319  
320 # Priorities of classes  
321 priorities = {'w1':0.5, 'w2':0.5, 'w3':0}  
322  
323 # Find normal distribution of each class  
classOrder = {}
```

```

324     normalDists = []
325     for key in classesFx1:
326         # Find a normal distribution for class key
327         normalDists.append(_mean_and_covariance(classesFx1[key]))
328         # Set class name of normal distribution
329         normalDists[len(normalDists)-1]['name'] = key
330         # Set class prior of normal distribution
331         normalDists[len(normalDists)-1]['prior'] = priories[key]
332         classOrder[key] = len(normalDists)-1
333
334     # Find label of each class
335     predictByclassifier = []
336     for x in dataFx1:
337         # Determine class of unlabeled data x
338         # and add the result of classifier to a list
339         predictByclassifier.append(_dichotomizer(normalDists,x))
340
341     # Find empirical training error
342     print("Computer exercises 2-e:")
343     print("Error: ", empirical_training_error(classesFx1,predictByclassifier))

```

حاصل اجرای کد بالا را در تصویر زیر مشاهده می کنید:

```

Computer exercises 2-e:
Error: 15.0
Bhattacharyya bound for w1 & w2: 0.411926

```

(f) بخش

همان طور که در نتایج مشاهده شد مقدار خطای دو ویژگی از یک ویژگی بیشتر شد و این نشان می دهد الزاماً زیاد کردن تعداد ویژگی ها خطای کمتر نمی کند.

نتایج بخش های مختلف سوال برنامه نویسی ۲ را در شکل زیر مشاهده می کنید:

Computer exercises 2-b:

Error: 30.0

Bhattacharyya bound for w1 & w2: 0.473999

Computer exercises 2-d:

Error: 45.0

Bhattacharyya bound for w1 & w2: 0.460466

Computer exercises 2-e:

Error: 15.0

Bhattacharyya bound for w1 & w2: 0.411926

پروژه‌ی برنامه‌نویسی :Q3 – Computer Project

این بخش دقیقاً مانند بخش قبل است با این تفاوت که باید priori کلاس 3 w_3 را برابر 0.5 کنیم و priori کلاس 2 را صفر کنیم. همین طور باید بجای sample های کلاس یک و دو، باید سمپل های کلاس یک و سه را به classifier بدھیم از آن‌ها برای سنجش ارور استفاده کنیم. کدهای این بخش در پوشه 3 computer exercises و در فایل computerExercise3.py قرار دارد.

تابع‌هایی که در ابتدای این فایل قرار دارند دقیقاً تابع‌هایی هستند که در فایل پروژه برنامه نویسی دو بود به همین دلیل آن‌ها را دوباره توضیح نمی‌دهم. فقط اینجا داده‌های متفاوتی را طبق سوال به تابع‌ها می‌دهیم. در تصویر زیر قسمت متفاوت کد سوال دو و سه را می‌بینید که در اینجا داده‌های کلاس یک و سه برای میزان خطابه تابع‌ها داده شده‌اند. همه چیز دیگر دقیقاً مانند سوال پروژه‌ی ۲ است.

```
173 #####  
174 # Computer exercise 3  
175 #####  
176 from copy import deepcopy  
177  
178 printFeatures = ['X1', 'X1, X2', 'X1, X2, X3']  
179  
180 # Use X1 feature then X1,X2 then X1,X2,X3  
181 for useFirstNfeature in range(1,4):  
182     # Prepare data  
183     # Make a copy of classes and just keep wanted features  
184     # of samples.  
185     classesFx1 = deepcopy(classes)  
186     for key in classesFx1:  
187         # Iterate all samples of class key  
188         for index, element in enumerate(classesFx1[key]):  
189             # just keep first useFirstNfeature  
190             classesFx1[key][index] = element[0:useFirstNfeature]  
191  
192     # Make a copy of data and just keep wanted features  
193     dataFx1 = deepcopy(data)  
194     for index, x in enumerate(dataFx1):  
195         dataFx1[index] = x[0:useFirstNfeature]  
196  
197     # Because of classifying w1 and w3, just use w1 and w3 samples  
198     dataFx1 = dataFx1[0:10] + dataFx1[20:30]  
199  
200     # Priorities of classes
```

```

201 priorities = {'w1': 0.5, 'w2': 0, 'w3': 0.5}
202
203 # Find normal distribution of each class
204 classOrder = {}
205 normalDists = []
206 for key in classesFx1:
207     # Find a normal distribution for class key
208     normalDists.append(mean_and_covariance(classesFx1[key]))
209     # Set class name of normal distribution
210     normalDists[len(normalDists)-1]['name'] = key
211     # Set class prior of normal distribution
212     normalDists[len(normalDists)-1]['prior'] = priorities[key]
213     classOrder[key] = len(normalDists)-1
214
215 # Find label of each sample
216 predictByclassifier = []
217 for x in dataFx1:
218     # Determine class of unlabeled data x
219     # and add the result of classifier to a list
220     predictByclassifier.append(dichotomizer(normalDists, x))
221
222 # Find empirical training error
223 print("Computer exercises 3 (Features {}):".format(printFeatures[useFirstNfeature-1]))
224 print("Error: ", empirical_training_error(classesFx1,predictByclassifier))
225
226 print('Bhattacharyya bound for w1 & w3:{:f}\n'.format(
227     bhattacharyya_bound(normalDists[classOrder['w1']], normalDists[classOrder['w3']])))
228

```

خروجی اجرای کد بالا را در تصویر زیر مشاهده می‌کنید:

```

Run computerExercise3
/home/bat/anaconda3/bin/python "/home/bat/Dropbox/codes
Computer exercises 3 (Features X1):
Error: 25.0
Bhattacharyya bound for w1 & w3:0.399433

Computer exercises 3 (Features X1, X2):
Error: 20.0
Bhattacharyya bound for w1 & w3:0.329193

Computer exercises 3 (Features X1, X2, X3):
Error: 15.0
Bhattacharyya bound for w1 & w3:0.299343

```

پروژه‌ی برنامه‌نویسی Q4 – Computer Project

کدهای این سوال در پوشه 4 computer exercises و در فایل computerExercise4.py است، این قسمت دقیقا مانند پروژه‌ی برنامه‌نویسی سه است و تنها تفاوت با سوال قبل آن است که priori کلاس یک صفر است و priri کلاس دو و سه برابر نیم است. همچنین داده‌های کلاس دو و سه را برای سنجیدن میزان خطا استفاده می‌کنیم. در ابتدای کد تابع‌ها دقیقاً تابع‌های تمرین‌های برنامه‌نویسی دو هستند و چون در قسمت دو آن‌ها را توضیح داده ام اینجا به آن‌ها نمی‌پردازم. در شکل زیر قسمتی از کد را می‌بنید که همانند سوال برنامه‌نویسی سه داده‌ها را به تابع‌ها می‌دهیم:

```
computerExercise3.py x computerExercise4.py x
173 #####
174 # Computer exercise 4
175 #####
176 from copy import deepcopy
177
178 printFeatures = ['X1', 'X1, X2', 'X1, X2, X3']
179
180 # Use X1 feature then X1,X2 then X1,X2,X3
181 for useFirstNfeature in range(1,4):
182     # Prepare data
183     # Make a copy of classes and just keep wanted features
184     # of samples.
185     classesFx1 = deepcopy(classes)
186     for key in classesFx1:
187         # Iterate all samples of class key
188         for index, element in enumerate(classesFx1[key]):
189             # just keep first useFirstNfeature
190             classesFx1[key][index] = element[0:useFirstNfeature]
191
192     # Make a copy of data and just keep wanted features
193     dataFx1 = deepcopy(data)
194     for index, x in enumerate(dataFx1):
195         dataFx1[index] = x[0:useFirstNfeature]
196
197     # Because of classifying w1 and w3, just use w1 and w3 samples
198     dataFx1 = dataFx1[10:30]
199
200     # Priorities of classes
```

```

200     # Priorities of classes
201     priorities = {'w1': 0, 'w2': 0.5, 'w3': 0.5}
202
203     # Find normal distribution of each class
204     classOrder = {}
205     normalDists = []
206     for key in classesFx1:
207         # Find a normal distribution for class key
208         normalDists.append(_mean_and_covariance(classesFx1[key]))
209         # Set class name of normal distribution
210         normalDists[len(normalDists)-1]['name'] = key
211         # Set class prior of normal distribution
212         normalDists[len(normalDists)-1]['prior'] = priorities[key]
213         classOrder[key] = len(normalDists)-1
214
215     # Find label of each sample
216     predictByclassifier = []
217     for x in dataFx1:
218         # Determine class of unlabeled data x
219         # and add the result of classifier to a list
220         predictByclassifier.append(_dichotomizer(normalDists, x))
221
222     # Find empirical training error
223     print("Computer exercises 4 (Features {}):".format(printFeatures[useFirstNfeature-1]))
224     print("Error: ", empirical_training_error(classesFx1,predictByclassifier))
225
226
227     print('Bhattacharyya bound for w1 & w4:{:f}\n'.format(
228
229     bhattacharyya_bound(normalDists[classOrder['w2']], normalDists[classOrder['w3']])))

```

خروجی اجرای کد بالا در شکل زیر مشاهده می‌کنید:

```

/home/bat/anaconda3/bin/python "/home/bat/Dropbox/codes/p
Computer exercises 4 (Features X1):
Error: 30.0
Bhattacharyya bound for w1 & w4:0.394902

Computer exercises 4 (Features X1, X2):
Error: 20.0
Bhattacharyya bound for w1 & w4:0.302994

Computer exercises 4 (Features X1, X2, X3):
Error: 15.0
Bhattacharyya bound for w1 & w4:0.224482

```

پروژه‌ی برنامه‌نویسی :Q5 – Computer Project

کدهای مربوط به این سوال در پوشه‌ی computer exercises 5 در فایل computerExercise5.py قرار دارند. در نوشتن کدهای مربوط به این سوال از کدهای مسیله‌ی برنامه‌نویسی اول و دوم استفاده شده است و به دلیل اینکه آن‌ها را قبل توضیح داده‌ام دیگر به آن‌ها نمی‌پردازم و فقط به قسمت‌های جدید کد می‌پردازم.

(a) بخش (a)

این بخش فاصله نقطه‌هایی را که سوال داده است را، تا مرکز توزیع نرمال سه کلاس داده شده در سوال دو را می‌خواهد. کد کامنت شده‌ی این سوال را در شکل زیر مشاهده می‌کنید:

```
173 #####  
174 # Computer exercise 5 - part a  
175 #####  
176 # Points that are given in computer exercise 5 for part a,b and c  
177 points = [[1, 2, 1], [5, 3, 2], [0, 0, 0], [1, 0, 0]]  
178  
179 # Calculate normal distribution of classes  
180 classOrder = {}  
181 normalDists = []  
182 for key in classes:  
183     # Find a normal distribution for class key  
184     normalDists.append(_mean_and_covariance(classes[key]))  
185     # Set class name of normal distribution  
186     normalDists[len(normalDists)-1]['name'] = key  
187     classOrder[key] = len(normalDists)-1  
188  
189 # Calculate mahalanobis distance of each point to all normal distributions  
190 for index, point in enumerate(points):  
191     for classPosition in ['w1', 'w2', 'w3']:  
192         print("Mahalanobis Dist,Point {0} and normal dis {1}: ".format(index+1, classPosition),  
193             end='')  
194         print(mahalanobis_distance(normalDists[classOrder[classPosition]]['means'],  
195             normalDists[classOrder[classPosition]]['covriance'],  
196             point))  
197
```

در این قسمت با استفاده از sample های هر کلاس مشخصات توزیع نرمال آن‌ها را به دست آورده‌ام و سپس با استفاده از تابع mahalanobios فاصله‌ی مورد نظر هر نقطه تا تمام کلاس‌ها را به دست آورده‌ام. خروجی این قسمت از کد را در شکل زیر مشاهده می‌کنید:

```

/home/bat/anaconda3/bin/python "/home/bat/Dropbox/codes/pythonCode/|
Mahalanobis Dist,Point 1 and normal dis w1: 1.0149706211958083
Mahalanobis Dist,Point 1 and normal dis w2: 0.8580511954300318
Mahalanobis Dist,Point 1 and normal dis w3: 2.674757036806055
Mahalanobis Dist,Point 2 and normal dis w1: 1.5571382110034355
Mahalanobis Dist,Point 2 and normal dis w2: 1.7556806886533693
Mahalanobis Dist,Point 2 and normal dis w3: 0.6470090140927858
Mahalanobis Dist,Point 3 and normal dis w1: 0.4899615415693
Mahalanobis Dist,Point 3 and normal dis w2: 0.26843241115322625
Mahalanobis Dist,Point 3 and normal dis w3: 2.2415013714929657
Mahalanobis Dist,Point 4 and normal dis w1: 0.48723675868718175
Mahalanobis Dist,Point 4 and normal dis w2: 0.45183435215254303
Mahalanobis Dist,Point 4 and normal dis w3: 1.4623364016581062

```

بخش b و C

این بخش خواسته است که classifier طراحی شده نقاط داده شده را با دو سری priori متفاوت پیدا کنیم. این بخش شبیه به سوال‌های برنامه نویسی دو، سه و چهار است. قطعه کدی را که این کار را انجام می‌دهد را در تصویر زیر مشاهده می‌کنید:

```

199 #####
200 # Computer exercise 5 - part b,c
201 #####
202 # Priorities of classes for part b & c computer exercise 5
203 listOfPriorities = [{ 'w1': 1/3, 'w2': 1/3, 'w3': 1/3}, { 'w1': 0.8, 'w2': 0.1, 'w3': 0.1}]
204
205 # Calculate normal distribution of classes
206 for priories in listOfPriorities:
207     # Find normal distribution of each class
208     classOrder = {}
209     normalDists = []
210     for key in classes:
211         # Find a normal distribution for class key
212         normalDists.append(_mean_and_covariance(classes[key]))
213         # Set class name of normal distribution
214         normalDists[len(normalDists)-1]['name'] = key
215         # Set class prior of normal distribution
216         normalDists[len(normalDists)-1]['prior'] = priories[key]
217         classOrder[key] = len(normalDists)-1
218
219     # Find label of each sample
220     predictByclassifier = []
221     for x in points:
222         # Determine class of unlabeled data x
223         # and add the result of classifier to a list
224         predictByclassifier.append(_dichotomizer(normalDists, x))
225
226     # Print predicted labels of points
227     print(" ")
228     print("Label of points when P(w1)={0:.4f},P(w2)={1:.4f},P(w3)={2:.4f}".format(
229         priories['w1'], priories['w2'], priories['w3'])

```

```
230
231    })
232    for point in predictByclassifier:
233        print(point)
```

همانند سوال های قبل با توجه به sample های هر کلاس توزیع نرمال هر کلاس پیدا شده است. بعد نقطه ها توسه طتابع dichotomizer برچسب خورده اند. و این کار برای هر دو سری priori انجام شده است. خروجی کد بالا در شکل زیر می بینید:

Label of points when $P(w1)=0.3333, P(w2)=0.3333, P(w3)=0.3333$

```
([1, 2, 1], 'w2')
([5, 3, 2], 'w3')
([0, 0, 0], 'w1')
([1, 0, 0], 'w1')
```

Label of points when $P(w1)=0.8000, P(w2)=0.1000, P(w3)=0.1000$

```
([1, 2, 1], 'w1')
([5, 3, 2], 'w1')
([0, 0, 0], 'w1')
([1, 0, 0], 'w1')
```

پروژه‌ی برنامه‌نویسی :Q6 – Computer Project

کدهای مربوط به این تمرین در پوشه‌ی «computer exercises 6» قرار دارند.

(a) بخش

این بخش یک تابع را که n عدد از توزیع Uniform(x lower, x higher) را بیرون بکشد، را خواسته است. در شکل زیر کد مربوط به این بخش که کامنت‌گذاری هم شده است را مشاهده می‌کنید:

```
computerExercise6.py x test.py x
1 #####
2 # Computer exercise 6-a
3 #####
4 def uniform_distribution(x_l, x_u, n):
5     """
6         This function draw n samples from uniform distribution
7         uniform(x_l, x_u)
8         :param x_l: lower bound of uniform distribution
9         :param x_up: upper bound of uniform distribution
10        :param n: number of samples that should be drawn
11        :return: return a list which contains n samples
12    """
13    import numpy as np
14
15    # Generate and return n different discrete uniform random sample
16    return np.random.random_integers(x_l, x_u, n).tolist()
```

ورودی‌ها و خروجی در تصویر بالا توضیح داده شده اند. از تابع numpy random_integers پکیج جهت تولید اعداد تصادفی گستته از توزیع یکنواخت استفاده شده است.
یک نمونه از اجرای این تابع را در شکل زیر مشاهده می‌کنید:

```
# Computer exercise 6-a
print( uniform_distribution(0, 5, 10))
[0, 5, 4, 4, 1, 5, 5, 4, 2, 5]
```

(b) بخش

این بخش، سوال خواسته است تابعی برای تولید سه عدد صحیح تصادفی با شرایط زیر نوشته شود.

$$-100 \leq x_l < x_r \leq 100 ; 0 < n < 1000$$

در زیر کد این تابع را که حاوی کامنت هم هست را مشاهده می کنید:

```
18 #####  
19 # Computer exercise 6-b  
20 #####  
21 def choose_bounds_and_size():  
22     """  
23         This function chooses bounds of discrete uniform  
24         distribution and number of samples.  
25         -100 <= lower bound of uniform distribution < upper bound of uniform distribution <= 100  
26  
27     :return: it returns a list that contains three number  
28     [lower bound of uniform distribution,  
29      upper bound of uniform distribution,  
30      number of samples]  
31     """  
32     import numpy as np  
33  
34     #choose lower and upper bound of uniform distribution  
35     while True:  
36         x_l, x_u = np.random.random_integers(-100, 100, 2)  
37         if x_l > x_u:  
38             x_l, x_u = x_u, x_l  
39         if x_l != x_u:  
40             break  
41  
42     # choose number of samples  
43     n = np.random.random_integers(1, 1000)  
44  
45     #return lower bound, upper bound, size  
46     return [x_l, x_u, n]
```

ورودی و خروجی ها در بخش Doc String توضیح داده شده‌اند. در شکل زیر یک نمونه از اجرای این تابع را مشاهده می‌کنید.

```
131 # Computer exercise 6-b  
132 print(choose_bounds_and_size())  
133  
computerExercise6  
/home/bat/anaconda3/bin/python "/home/bat/Dropbox/codes/pythonCode/pat  
[-100, 60, 925]
```

بخش (C ، D ، E)

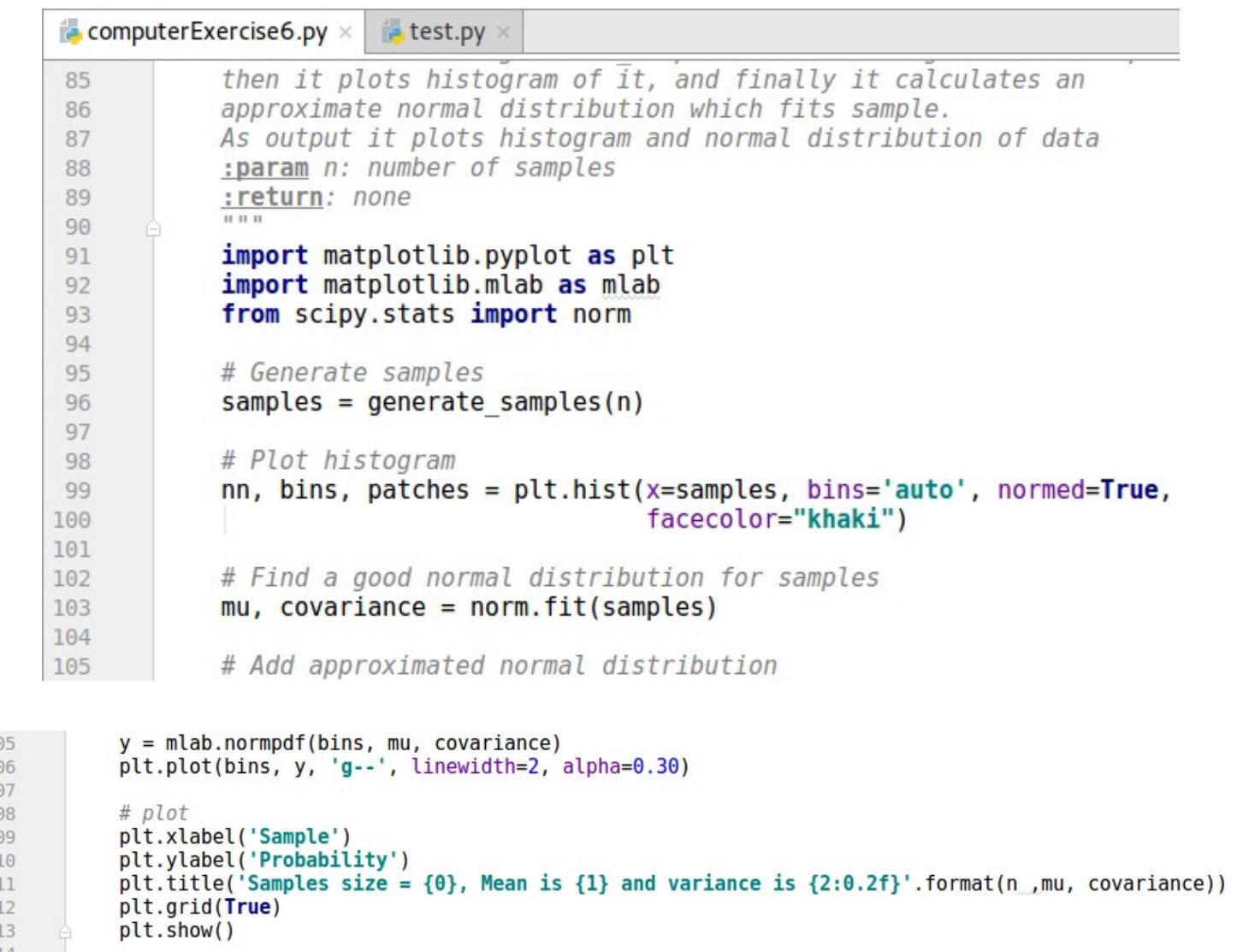
این سه بخش را به دلیل شباهت زیاد، با هم توضیح می دهم. برای این سه بخش 2 تابع نوشته ام که با فراخوانی آن ها با آرگمن های مختلف پاسخ بخش های C,D و E را به دست می آوریم.

تابع generate samples همان طور که در تصویر زیر مشاهده می کنید با استفاده از دو تابع بخش a و b همین سوال بارها اقدام به تولید اعداد تصادفی می کند و هر بار میانگین آن ها را نگه می دارد. به تعداد آرگومان ورودی این کار را تکرار می کند در نهایت میانگین هایی را که نگه داشته است را return می کند.

```
52     def generate_samples(n):
53         """
54             This function gets an scalar input which determine
55             number of samples and it uses choose_bounds_and_size()
56             to determine uniform distribution bound and num of samples
57             that it should draw from the distribution each time,
58             In each iteration it generates some samples then it calculates
59             mean of samples.
60             :param n: number of iterations
61             :return: return n samples
62             """
63             # Output
64             out = []
65
66             # Randomly determine lower and upper bound of
67             # Uniform discrete function and the number of samples
68             x_l, x_u, size = choose_bounds_and_size()
69
70             # Do n iteration
71             for i in range(n):
72
72
73                 # Draw samples from uniform distribution (x_l, x_u)
74                 # and add average of them to output list
75                 samples = uniform_distribution(x_l, x_u, size)
76                 out.append(sum(samples)/len(samples))
77
78             # Return n samples
79             return out
```

تابع دومی که برای این سه بخش نوشته ام، تابع plot_histogram_and_norm_for_samples است که از تابع generate_samples استفاده می کند و به کمک آن به اندازه ای آرگمان ورودی، عدد تصادفی تولید

می کند و بعد از آن Histogram اعداد تصادفی را می کشد و همین طور mean و Variance داده ها را پیدا می کند و منحنی توزیع نرمال را بر روی histogram رس ه می کند. در تصویر زیر کد این تابع را مشاهده می کنید:



```

computerExercise6.py x test.py x
 85     then it plots histogram of it, and finally it calculates an
 86     approximate normal distribution which fits sample.
 87     As output it plots histogram and normal distribution of data
 88     :param n: number of samples
 89     :return: none
 90     """
 91     import matplotlib.pyplot as plt
 92     import matplotlib.mlab as mlab
 93     from scipy.stats import norm
 94
 95     # Generate samples
 96     samples = generate_samples(n)
 97
 98     # Plot histogram
 99     nn, bins, patches = plt.hist(x=samples, bins='auto', normed=True,
100                                 facecolor="khaki")
101
102     # Find a good normal distribution for samples
103     mu, covariance = norm.fit(samples)
104
105     # Add approximated normal distribution
106
107     y = mlab.normpdf(bins, mu, covariance)
108     plt.plot(bins, y, 'g--', linewidth=2, alpha=0.30)
109
110     # plot
111     plt.xlabel('Sample')
112     plt.ylabel('Probability')
113     plt.title('Samples size = {0}, Mean is {1} and variance is {2:0.2f}'.format(n, mu, covariance))
114     plt.grid(True)
115     plt.show()

```

این تابع، تابع generate_samples بخش قبل را فرا می خواند و n عدد تصادفی که به شیوه‌ای که در سوال شرح داده شده است را دریافت می کند. و بعد با استفاده از تابع hist پکیج pylab هیستوگرام داده‌ها را رس ه می کند و همین طور با استفاده از تابع fit مشخصات توزیع توزیع نرمال داده ها را به دست می آورد و بر روی histogram منحنی نرمال داده‌ها را رس می کند.

برای اینکه برای اعداد خواسته شده در قسمت های e,d,c هیستوگرام و توزیع نرمال را به دست بیارویم تابع بالا را

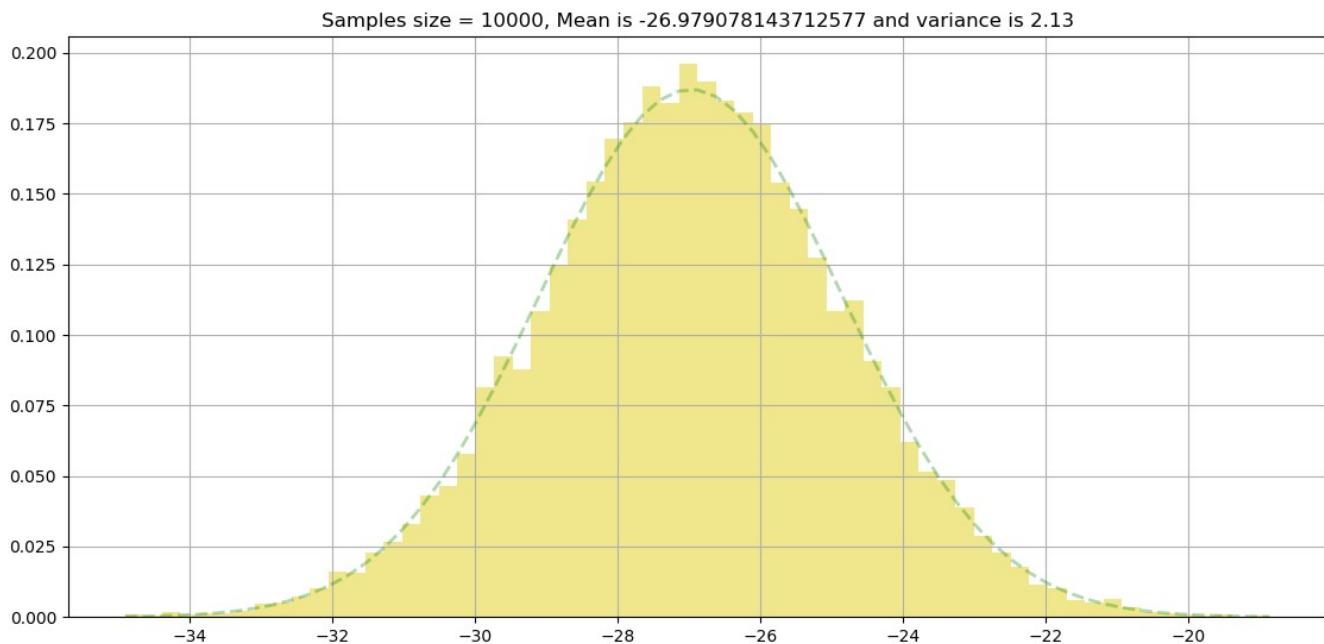
برای آن اعداد به شکل زیر فرا می خوانیم:

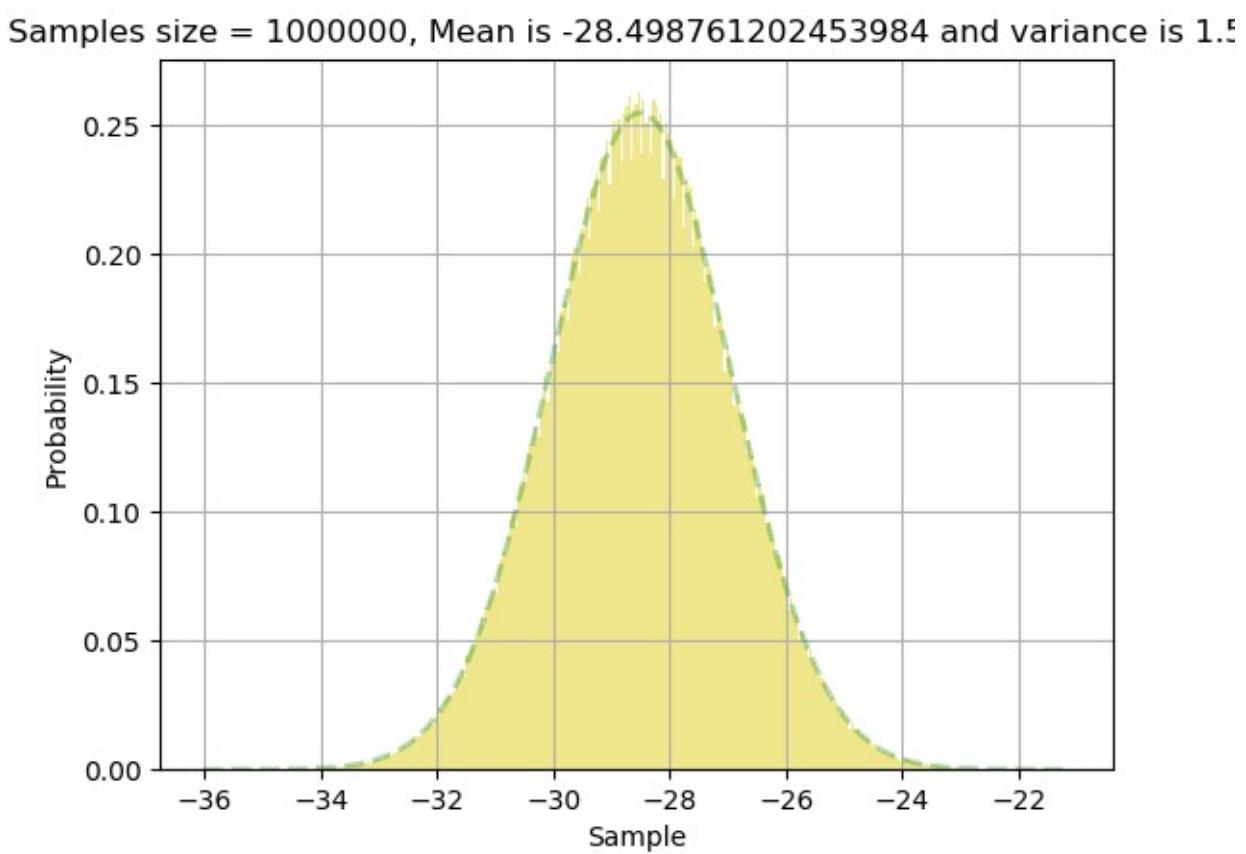
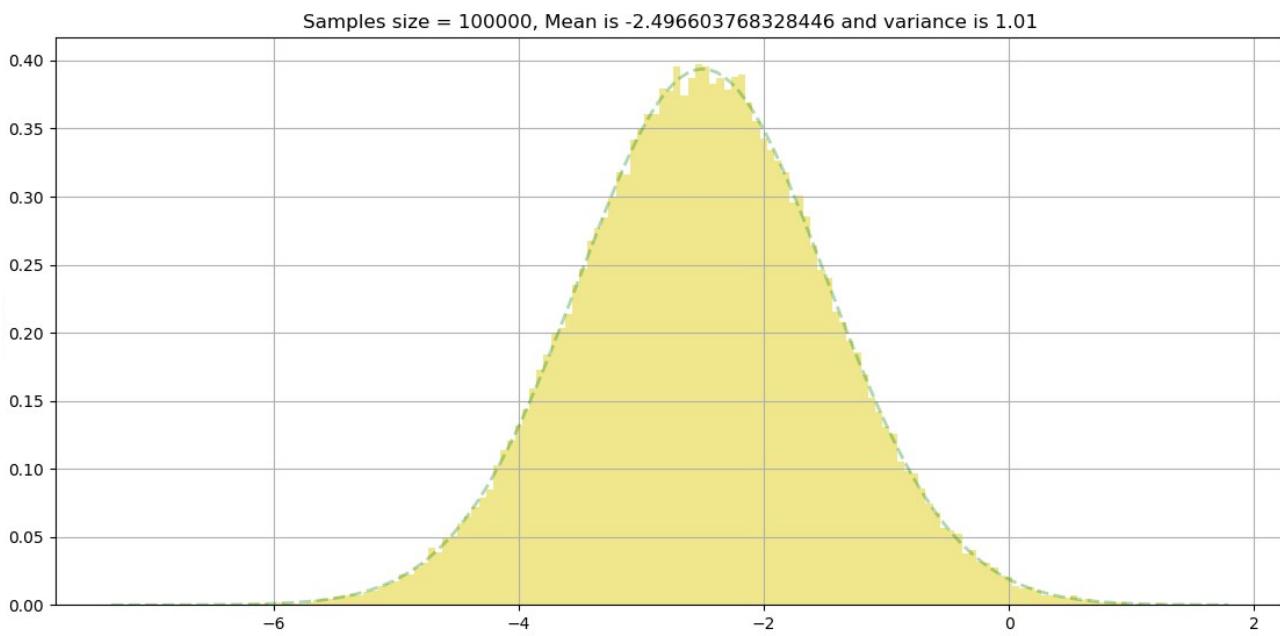
```
# Computer exercise 6-c,d,e
from multiprocessing import Process

processes = []
for i in [10000, 100000, 1000000]:
    processes.append(Process(target=plot_histogram_and_norm_for_samples, args=[i]))

for p in processes:
    p.start()
for p in processes:
    p.join()
```

برای افزایش سرعت در کد بالا چند پروسس ایجاد کرده ام. بعد از فراخوانی سه تصویر رسم می شود که آن ها را در زیر می بینید. مشخصات منحنی نرمال در تصویرها نمایش داده شده اند:





از آنجایی که بازه‌های توزیع یکنواخت تصادفی انتخاب می‌شوند در سه تصویر بالا بازه‌هایی که انتخاب شده‌اند برابر نیستند.

طبق اصل Irwin–Hall distribution در آمار و احتمال مجموع یک سری توزیع یکنواخت Independent and identically distributed یک توزیع نرمال می‌شود. که در بالا همین مسیله را مشاهده می‌کنید. همین طور هر چه تعداد Iteration ها بیشتر شده است هیستوگرام به بردار نرمال محاسبه شده نزدیک تر شده است.

