ejercicio11

April 7, 2020

1 Ejercicio 11:

Sea X un vector aleatorio gaussiano cuya media y matriz de covarianza son:

$$\mu_{\underline{X}} = \begin{bmatrix} 1 \\ -2 \end{bmatrix} \qquad C_{\underline{X}} = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$$

1. Obtenga las curvas de nivel $\mathcal{C}_\alpha = \{(x,y) : f_{\underline{X}}(x,y) = \alpha\}$ con $\underline{X} = [x,y]^T$.
2. Grafique en el plano-(x,y) N = 1000 realizaciones del vector $\underline{X}$ junto con las curvas de nivel anteriores.

```
In [116]: # cargamos las bibliotecas de python que vamos a utilizar
          import math as m
          import numpy as np
          import scipy as sp
          import matplotlib.pyplot as plt
```

```
In [117]: # generamos N realizaciones de un vector gaussiano Y de dimensión 2
          N = int(1e4)
          Y = np.random.randn(2,N) # normal estandar
```

```
In [118]: # matriz de covarianza de X
          Cx = np.array([[2,1],[1,1]])
          # vector de la media de X
          mux = np.array([[1],[-2]])
```

Coloreamos las realizaciones de Y para obtener X como vimos en el Ejercicio 10: $X = AY + b$ con $b = \mu_X$ y $A = P\,D^{1/2}\,P^T$, donde P y D salen de la descomposición en autovectores y autovalores de C_X.

Una alternativa es usar la función: `np.random.multivariate_normal(mu,C,N)`.

```
In [119]: L, P = np.linalg.eig(Cx)
          D = np.diag(L)
          Pi = P.transpose()
          A = np.linalg.multi_dot([P,np.sqrt(D),Pi])
          X = np.dot(A,Y) + np.tile(mux,N)
```

1.1 Curvas de nivel

Las curvas de nivel salen de igualar la función de densidad de probabilidad de $\underline{X}$ a una constante α:

$$f_{\underline{X}}(\underline{x}) = \frac{1}{(2\pi)\,|C_{\underline{X}}|^{1/2}} \exp\left\{ -\frac{1}{2}(\underline{x}-\mu_{\underline{X}})^T\, C_{\underline{X}}^{-1}\,(\underline{x}-\mu_{\underline{X}}) \right\} = \alpha$$

Operando vemos que esto corresponde a una forma cuadrática:

$$\alpha' = (\underline{x}-\mu_{\underline{X}})^T\, C_{\underline{X}}^{-1}\,(\underline{x}-\mu_{\underline{X}})$$

Si llamamos: $\underline{x}-\mu_{\underline{X}} = [x_1, x_2]^T$ y $C_{\underline{X}}^{-1} = \begin{bmatrix} a & b \\ b & c \end{bmatrix}$ entonces: $\alpha' = a\,x_1^2 + b\,x_2^2 + 2\,c\,x_1\,x_2$ que es la ecuación de una elipse.

Si desarrollamos un poco más y diagonalizamos la matriz de covarianza:

$$\begin{aligned} \alpha' &= (\underline{x}-\mu_{\underline{X}})^T\, C_{\underline{X}}^{-1}\,(\underline{x}-\mu_{\underline{X}}) \\ &= (\underline{x}-\mu_{\underline{X}})^T\, P\, D^{-1}\, P^T\,(\underline{x}-\mu_{\underline{X}}) \\ &= \left(P^T\,(\underline{x}-\mu_{\underline{X}})\right)^T\, D^{-1}\, \left(P^T\,(\underline{x}-\mu_{\underline{X}})\right) \\ &= \underline{u}^T\, D^{-1}\, \underline{u} \end{aligned}$$

donde $\underline{u} = P^T\,(\underline{x}-\mu_{\underline{X}})$. Dado que P es una matriz ortogonal entonces es una matriz de rotación (conserva la norma) y por lo tanto $\underline{u}$ sale de aplicar un desplazamiento y una rotación a $\underline{x}$. Por otro lado, P es la matriz de autovectores de $C_{\underline{X}}$ y sus columnas corresponden con los autovectores.

De esto último podemos ver que los ejes principales de las curvas de nivel $\mathcal{C}_\alpha$ son los autovectores de $C_{\underline{X}}$ y que están centradas en $\mu_{\underline{X}}$.

```
In [120]: plt.figure()
          # graficamos las realizaciones de X
          plt.plot(X[0,:],X[1,:],'.',alpha = 0.1)
          # autovectores
          v1 = P[:,0] # autovector 1 es la primer columna de P
          v2 = P[:,1] # autovector 2 es la segunda columna de P
          # graficamos los ejes principales
          plt.plot([mux[0],v1[0]+mux[0]],[mux[1],v1[1]+mux[1]],'r')
          plt.plot([mux[0],v2[0]+mux[0]],[mux[1],v2[1]+mux[1]],'g')
          # curvas de nivel (elipses)
          x = np.arange(-4.0, 6.0, 0.1)
          y = np.arange(-6.0, 2.0, 0.1)
          XX, YY = np.meshgrid(x,y)
          Cxi = np.linalg.inv(Cx) # inv(Cx)
          ZZ = Cxi[0,0]*(XX-mux[0])**2 + Cxi[1,1]*(YY-mux[1])**2 + 2*Cxi[0,1]*(XX-mux[0])*(YY-m
          contour = plt.contour(XX, YY, ZZ, levels = [2, 3, 4, 5])
          plt.clabel(contour, inline = True, fontsize = 10)
          # decoraciones
          plt.xlabel('x1')
          plt.ylabel('x2')
```

```
plt.title('Realizaciones de X y curvas de nivel')
plt.grid(True)
```