Práctica 2

Celia Arias Martínez

Ejercicio 1

En este ejercicio vamos a estudiar cómo se comportan determinadas funciones que definen la frontera de clasificación al intentar separar puntos de una muestra que contiene ruido. Para ello utilizaremos las tres funciones proporcionadas en el template: *simula_unif*, *simula_gaus* y *simula_recta*.

Ejercicio 1.1

En este apartado vamos a dibujar dos gráficas de nubes de puntos, cada una con una distribución diferente. Las dos tienen un tamaño de puntos igual a `50` y dimensión `2`.

La primera gráfica de nubes de puntos sigue una distribución uniforme y los puntos están en un rango de $[-50, 50]$.

El código es:

```
def simula_unif(N, dim, rango):
    return np.random.uniform(rango[0],rango[1],(N,dim))

x_1 = simula_unif(50, 2, [-50,50])
plt.scatter(x_1[:,0], x_1[:,1], c ='r')
plt.show()
```

La segunda gráfica de nubes sigue una distribución de gauss de parámetros $nu = 0$ y $sigmaelarray[5, 7]$. Para generarla vemos que llama a la función simula_gaus. Esta función fija la media a cero, y luego llama a la función de np random.normal para que genere el conjunto de puntos siguiendo una distribución normal, y utilizando para cada columna un sigma determinado. En este caso para el eje x utilizamos un sigma igual a 5, y para el eje y un sigma igual a 7.

El código explicado es:

```
def simula_gaus(N, dim, sigma):
    media = 0
    out = np.zeros((N,dim),np.float64)
    for i in range(N):
        # Para cada columna dim se emplea un sigma determinado. Es decir, para
        # la primera columna (eje X) se usará una N(0,sqrt(sigma[0]))
        # y para la segunda (eje Y) N(0,sqrt(sigma[1]))
        out[i,:] = np.random.normal(loc=media, scale=np.sqrt(sigma), size=dim)
```

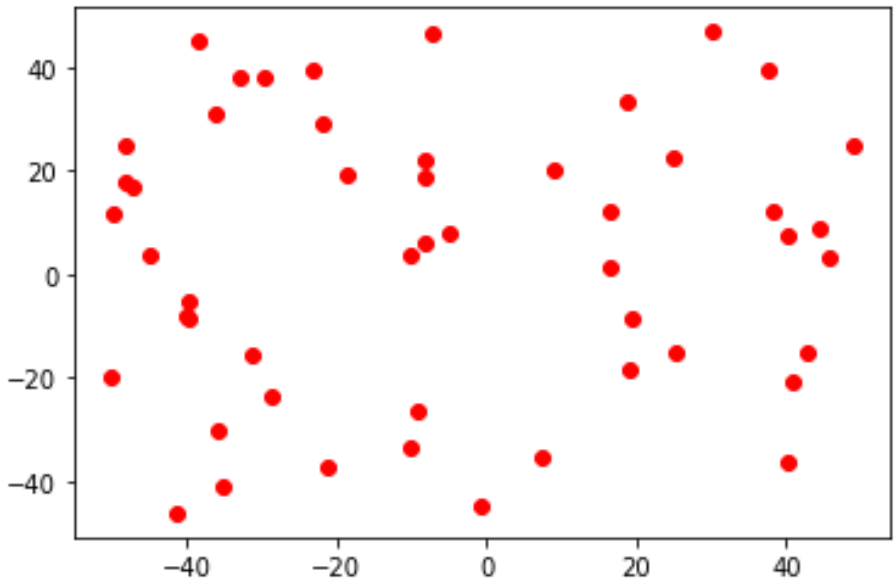


Figure 1: Distribución uniforme

```
    return out
```

Ejercicio 1.1

En este apartado introducimos una función para etiquetar los puntos y un ruido.

Primero dibujamos una nube de puntos que sigue una distribución uniforme, según lo explicado en el apartado anterior. En este caso el tamaño de puntos será de `100`, la dimensión seguirá siendo `2`, y el intervalo será $[-50, 50]$

La recta que usamos para etiquetar los puntos es $f(x, y) = y - a * x - b$ donde la `a` y la `b` las hemos obtenido con la función *simula_recta*.

Dibujamos el gráfico de puntos generado, así como la recta usada para etiquetar.

```
x_3 = simula_unif(100, 2, [-50,50])
etiquetas=[]
a,b = simula_recta([-50,50])
for i in range(0,len(x_3)):
    etiquetas.append(f(x_3[i,0], x_3[i,1], a,b))
etiquetas = np.asarray(etiquetas)
t = np.linspace(min(x_3[:,0]),max(x_3[:,0]), 100)
plt.scatter(x_3[:,0], x_3[:,1], c =etiquetas)
```

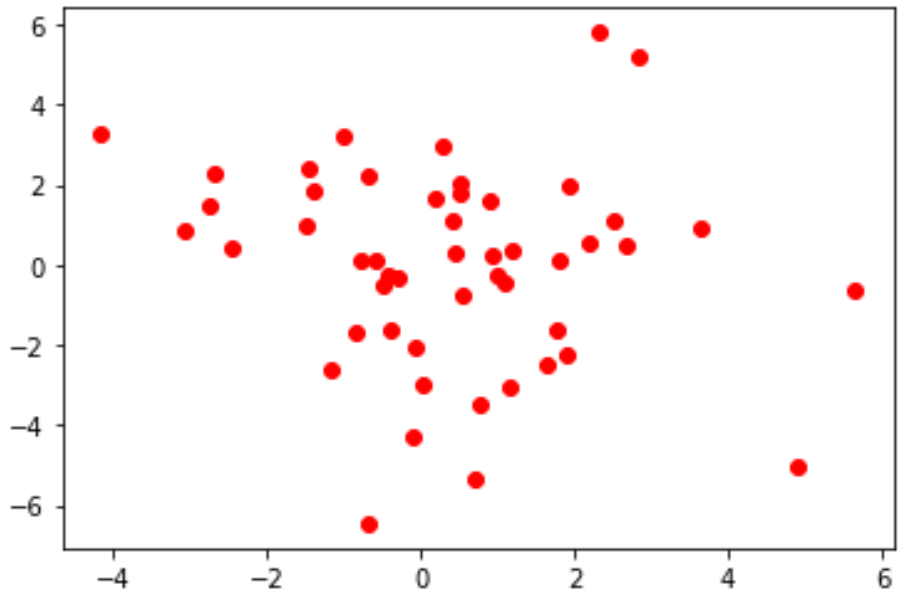


Figure 2: Distribución uniforme

```
plt.plot( t, a*t+b, c = 'red')
plt.show()
```

He definido una función para calcular el porcentaje de puntos mal etiquetados, que en este caso no tiene mucha utilidad porque sabemos que sera del 0%, pero la utilizaremos en los apartados siguientes.

El código es:

```
"""
calculaPorcentaje: calcula la proporción de puntos mal clasificados
        x: muestra de puntos
        y: vector con las etiquetas
        g: función que clasifica los puntos de la muestra

        porcentaje_mal: proporción de puntos mal clasificada
"""
def calculaPorcentaje(x, y, g):

    mal_etiquetadas1 = 0
    for i in range(x[:,0].size):
        etiqueta_real = y[i]
        etiqueta_obtenida = g(x[i,0], x[i,1])
        if (etiqueta_real != etiqueta_obtenida):
```

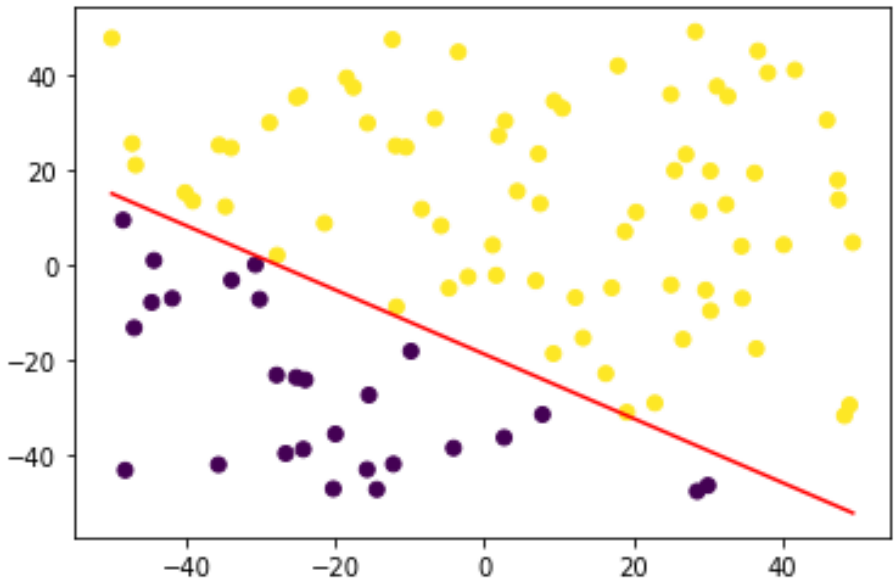


Figure 3: Distribución uniforme 1.2

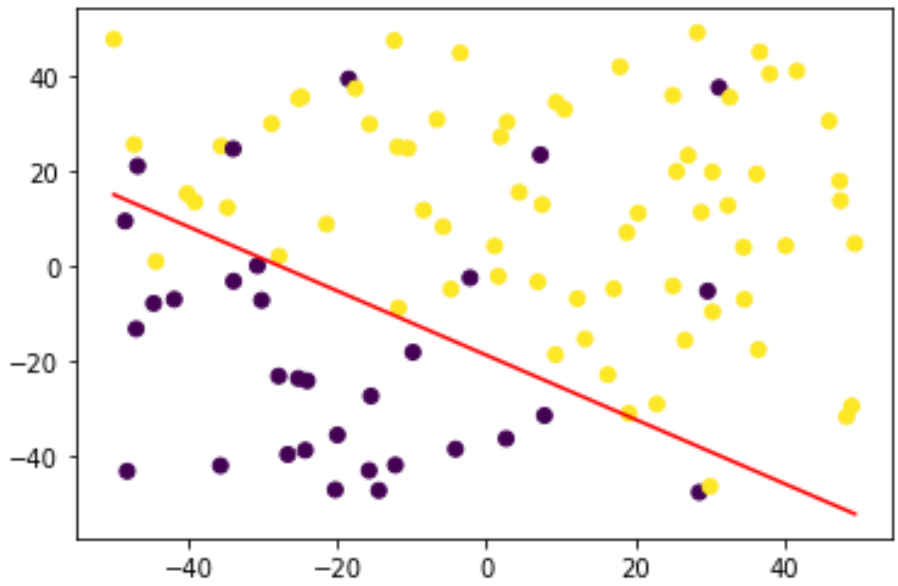
```
            mal_etiquetadas1+=1

    mal_etiquetadas2 = 0
    for i in range(x[:,0].size):
        etiqueta_real = y[i]
        etiqueta_obtenida = -g(x[i,0], x[i,1])
        if (etiqueta_real != etiqueta_obtenida):
            mal_etiquetadas2+=1

    mal_etiquetadas = min(mal_etiquetadas1, mal_etiquetadas2)
    porcentaje_mal = mal_etiquetadas / x[:,0].size

    return porcentaje_mal
```

Esta función lo que hace es, por un lado calcula el valor de una variable llamada `mal_etiquetadas1`, recorriendo todos los vectores de características y viendo si la etiqueta asignada coincide con la real. Por otro lado calcula el valor de `mal_etiquetadas2` de la misma forma, pero lo hacemos porque la función puede separar los puntos dándole un valor por debajo de la función y otro por encima o al revés. Por tanto nos quedamos con el mínimo de estos dos valores. Por último calcula la media y devuelve dicho valor.

Ejercicio 1.2

En este apartado modifico de forma aleatoria un 10% de los valores positivos, y un 10% de los valores negativos. Para eso he creado dos vectores auxiliares, uno con las posiciones de los valores positivos, y uno con las de los negativos. Después he seleccionado aleatoriamente un 10% de números entre 0 y el tamaño del vector de positivos, y esos valores los he utilizado de índices en el vector de positivos, con lo que he obtenido un 10% de los índices de los valores positivos del vector original. Con los valores negativos he hecho lo mismo, he concatenado los dos vectores y he cambiado los valores de los índices obtenidos.

```
positivas = np.where(etiquetas == 1)
negativas = np.where(etiquetas == -1)
positivas = np.asarray(positivas).T
negativas = np.asarray(negativas).T
ind_pos = np.random.choice(len(positivas), int(0.1*len(positivas)), replace = True)
cambiar_signo = positivas[ind_pos,:]
ind_neg = np.random.choice(len(negativas), int(0.1*len(negativas)), replace = True)
cambiar_signo = np.concatenate((cambiar_signo, negativas[ind_neg,:]), axis=0)
for i in range(0, len(cambiar_signo)):
    etiquetas[cambiar_signo[i]]=-etiquetas[cambiar_signo[i]]
```



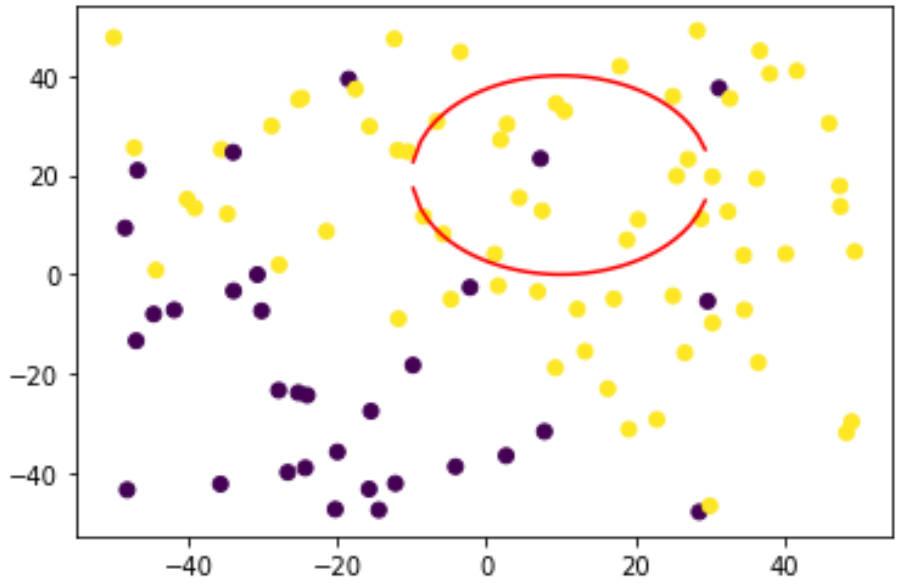
Porcentaje mal etiquetadas: 0.09

Ejercicio 1.3

Tenemos ahora cuatro funciones diferentes, y vamos a ver cómo separan la muestra que tenemos, y cómo se comportan en cuánto al ruido.

Ejercicio 1.3.1

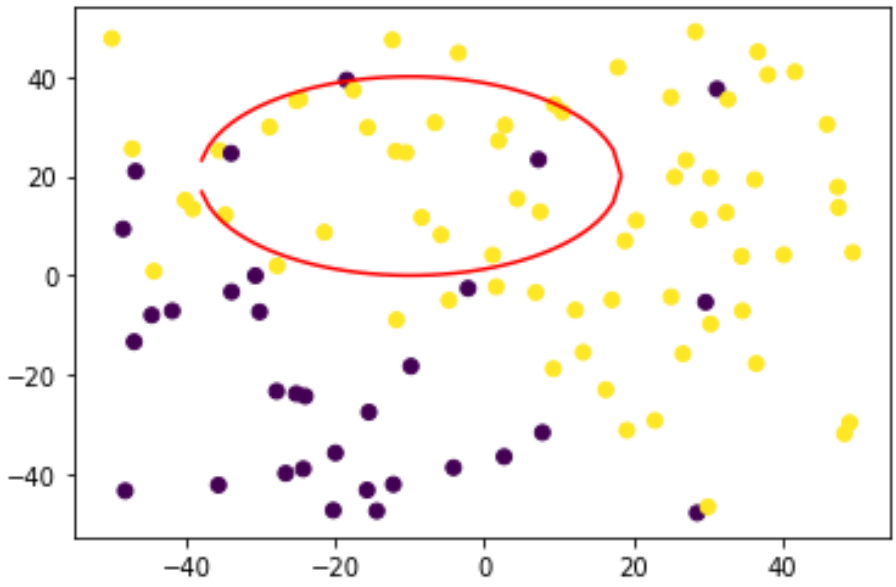
$f(x, y) = (x - 10)^2 + (y - 20)^2 - 400$



Porcentaje mal etiquetadas: 0.44

Ejercicio 1.3.2

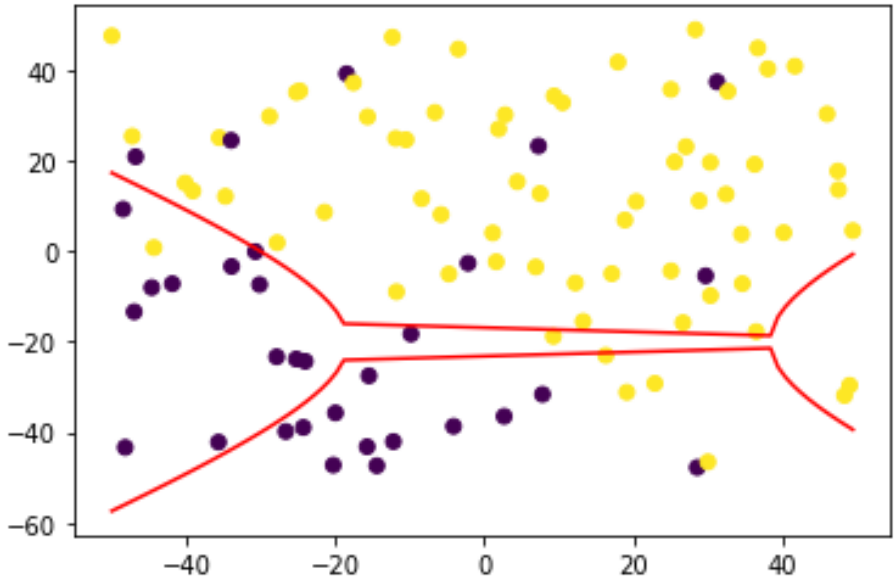
$f(x, y) = 0.5 * (x + 10)^2 + (y - 20)^2 - 400$



Porcentaje mal etiquetadas: 0.5

Ejercicio 1.3.3

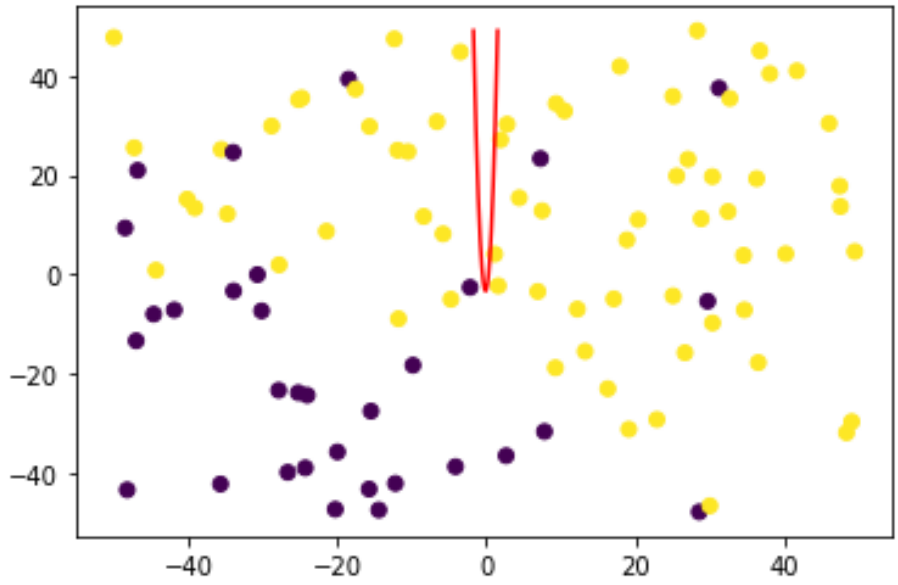
$f(x, y) = 0.5 * (x - 10)^2 - (y + 20)^2 - 40$



Porcentaje mal etiquetadas: 0.23

Ejercicio 1.3.4

$f(x, y) = y - 20 * x^2 - 5 * x + 3$



Porcentaje mal etiquetadas: 0.32