Ejercicio sobre la complejidad de H y el ruido.

En este ejercicio debemos aprender la dificultad que introduce la aparición de ruido en las etiquetas a la hora de elegir la clase de funciones más adecuada. Haremos uso de tres funciones:

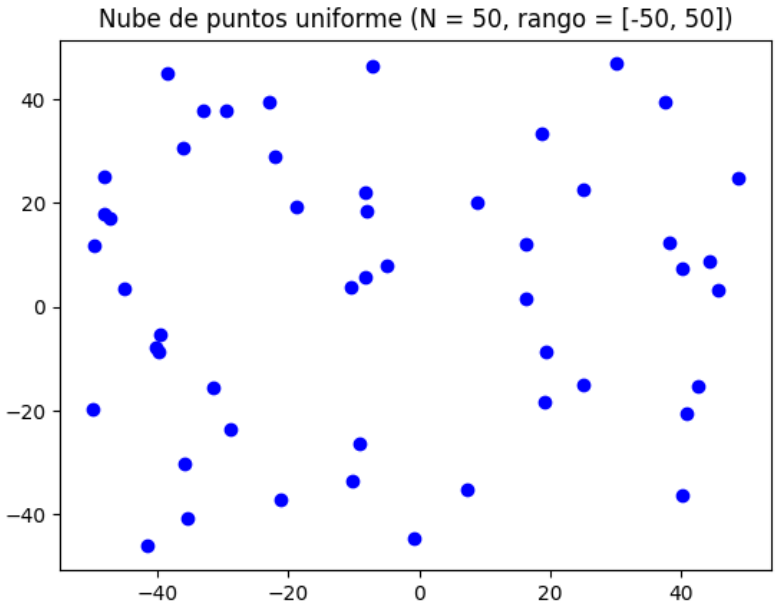
- `simula_unif (N, dim, rango)`, que calcula una lista de N vectores de dimensión dim. Cada vector contiene dim números aleatorios uniformes en el intervalo rango.
- `simula_gaus(N, dim, sigma)`, que calcula una lista de longitud N de vectores de dimen- sión dim, donde cada posición del vector contiene un número aleatorio extraído de una distribucción Gaussiana de media 0 y varianza dada, para cada dimension, por la posición del vector sigma.
- `simula_recta(intervalo)` , que simula de forma aleatoria los parámetros, v = (a, b) de una recta, $y = ax + b$, que corta al cuadrado $[-50, 50] \times [-50, 50]$.

1 Dibujo de las gráficas

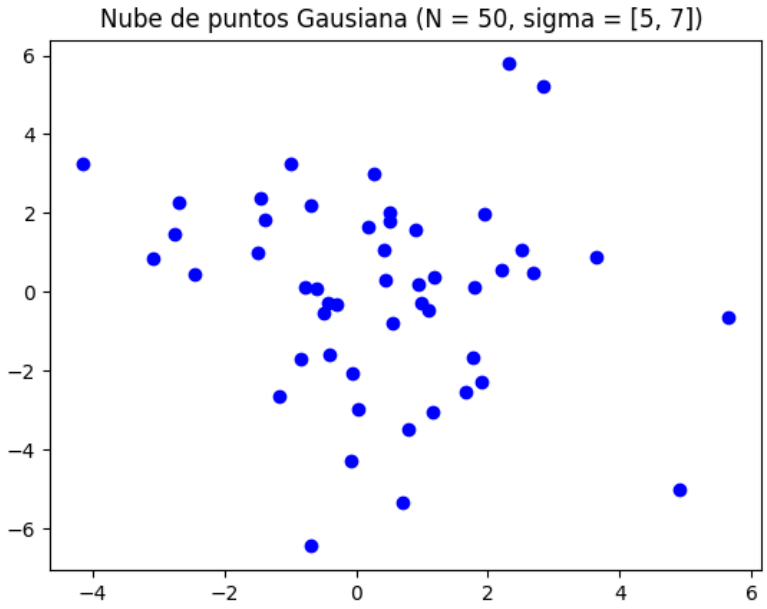
Dibujar gráficas con las nubes de puntos simuladas con las siguientes condiciones:

Para ellos hemos utilizado la función `scatter plot` (introducir código de la función TO-DO)

a) Considere $N = 50, \quad dim = 2, rango = [-50, +50]$ con `simula_unif(N,dim, rango)`.



b) Considere $N = 50, \quad dim = 2, sigma = [5, 7]$ con `simula_gaus(N,dim, sigma)`.



Valoración de la influencia del ruido en la selección de la complejidad de la clase de funciones.

Con ayuda de la función `simula_unif(100, 2, [-50, 50])` generamos una muesta de puntos 2D a los que vamos añadir una etiqueta usando el signo de la función $f(x, y) = y - ax - b$, es decir el signo de la distancia de cada punto a la recta simulada con `simula_recta()`.

Función de muestra de gráficas

Todas estas funciones fueron explicadas en la práctica uno, se utilizan la función `scattered` y `contour` de la librería `matplotlib.pyplot`.

Como único comentario, la función de clasificación $f(x, y)$ es muy fácil de calcular.

Sabemos que $f(x, y) = 0$ es una recta, luego solo habría que calcular dos puntos de ésta (por ejemplo hacer $x_1 = 0 e y_2 = 0$ y resolver la ecuación) y pintar la recta que pasa por ellos.

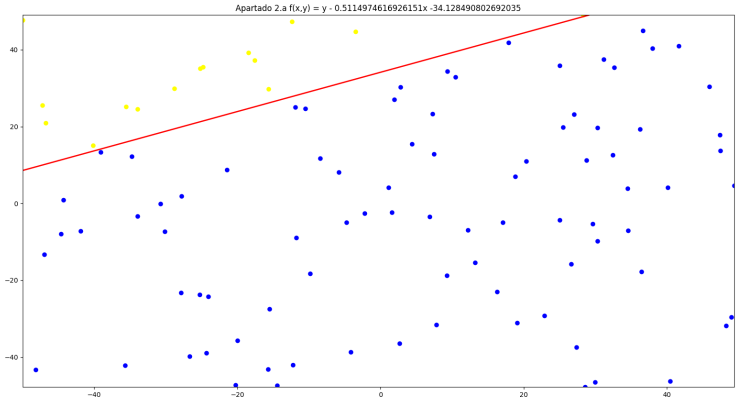
Sin embargo se ha optado por hacer uso de la función `contout` para tener mayor generalidad, ya que esta es capaz de pintar los puntos de la ecuación $g(x, y) = 0$ de $g : \mathbb{R}^2 \longrightarrow \mathbb{R}$

```
def classified_scatter_plot(x,y, function, plot_title, labels, colors):
```

```
'''Dibuja los datos x con sus respectivas etiquetas y
Dibuja la función: function
labels: son las etiquetas posibles que queremos que distinga para colorear,
(todo esto en el mismo gráfico
'''
plt.clf()


for l in labels:
        index = [i for i,v in enumerate(y) if v == l]
        plt.scatter(x[index, 0], x[index, 1], c = colors[l], label = str(l))



## ejes
xmin, xmax = np.min(x[:, 0]), np.max(x[:, 0])
ymin, ymax = np.min(x[:, 1]), np.max(x[:, 1])


## function plot
spacex = np.linspace(xmin,xmax,100)
spacey = np.linspace(ymin,ymax,100)
z = [[ function(i,j) for i in spacex] for j in spacey ]
plt.contour(spacex,spacey, z, 0, colors=['red'],linewidths=2 )

# título
plt.title(plot_title)

plt.show()
```

a) Gráfico 2D

El resultado de dibujar esto es



Podemos observar que como era de esperar el dibujo y la clasificación están ien hechos.