APRENDIZAJE AUTOMÁTICO

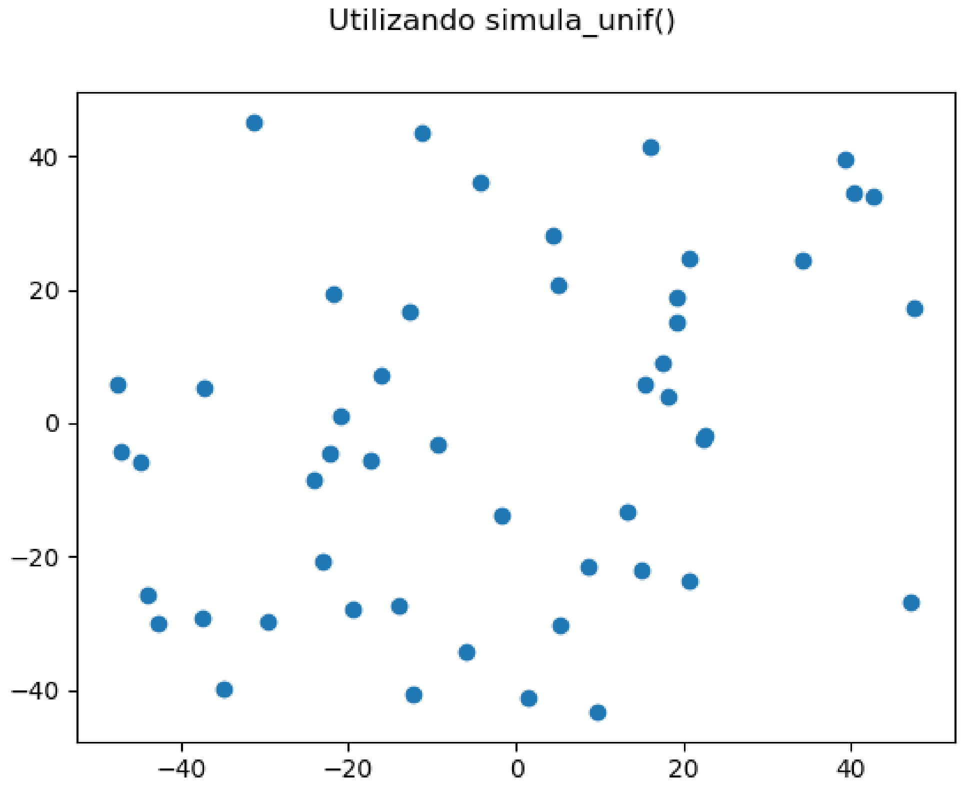
# Trabajo 2: Programación

Iván López Justicia

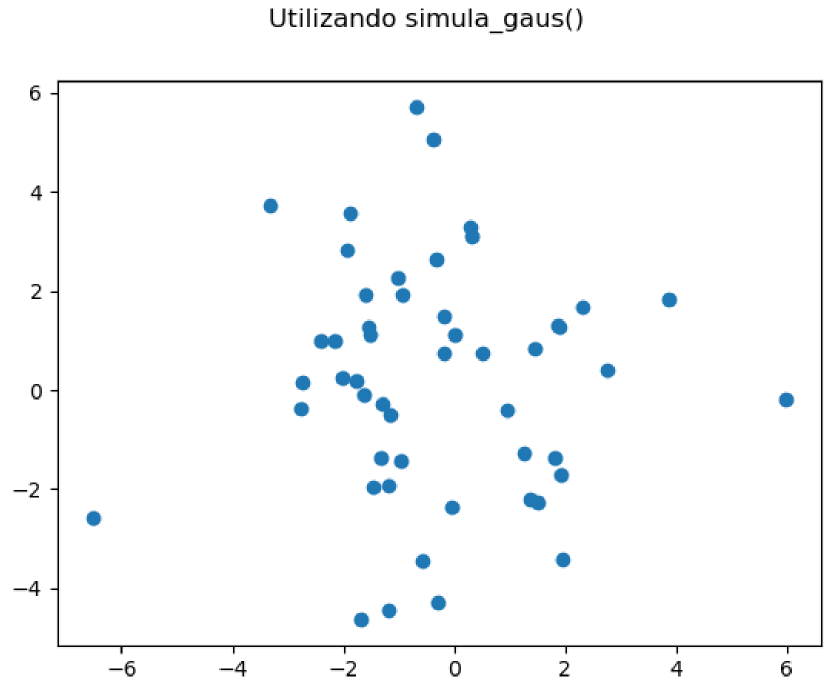
# 1. EJERCICIO SOBRE LA COMPLEJIDAD DE H Y EL RUIDO

**1. (1 punto) Dibujar una gráfica con la nube de puntos de salida correspondiente.**

**a) Considere N = 50, dim = 2, rango = [-50, +50] con simula_unif(N; dim; rango).**

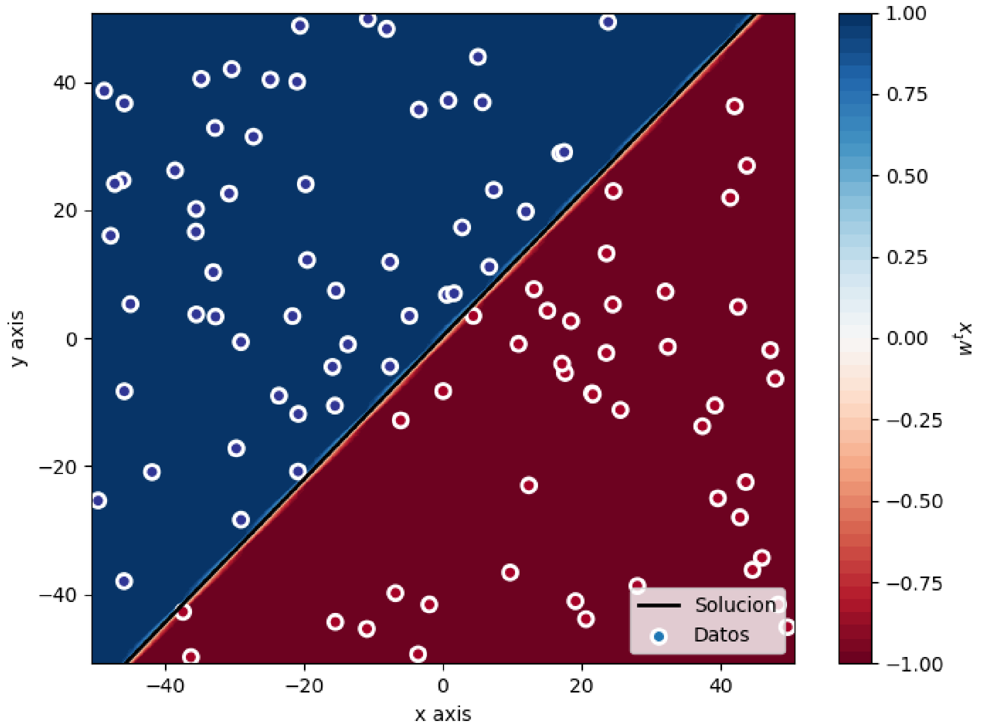


**b) Considere N = 50, dim = 2 y sigma = [5, 7] con simula_gaus(N; dim; sigma).**

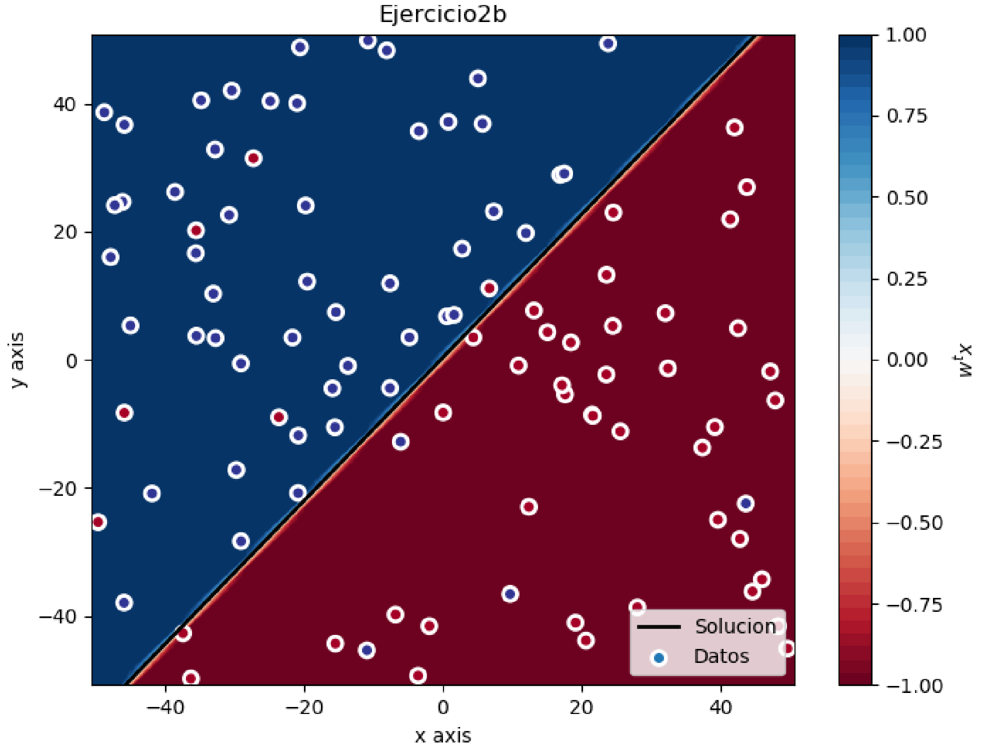



**2. (2 puntos) Con ayuda de la función simula_unif() generar una muestra de puntos 2D a los que vamos añadir una etiqueta usando el signo de la función $f(x,y) = y - ax - b$, es decir el signo de la distancia de cada punto a la recta simulada con simula_recta().**

**a) Dibujar una gráfica donde los puntos muestren el resultado de su etiqueta, junto con la recta usada para ello. (Observe que todos los puntos están bien clasificados respecto de la recta)**



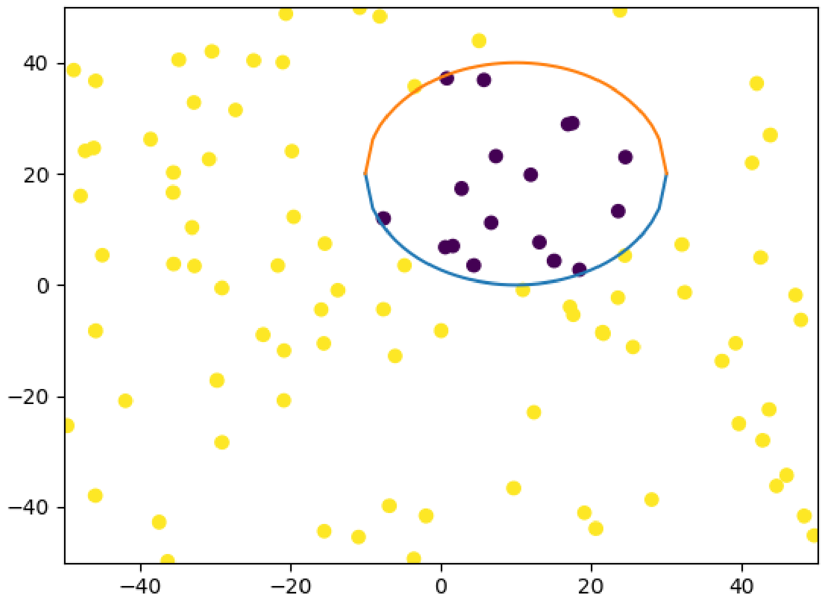
**b) Modifique de forma aleatoria un 10% etiquetas positivas y otro 10% de negativas y guarde los puntos con sus nuevas etiquetas. Dibuje de nuevo la gráfica anterior. (Ahora hay puntos mal clasificados respecto de la recta)**



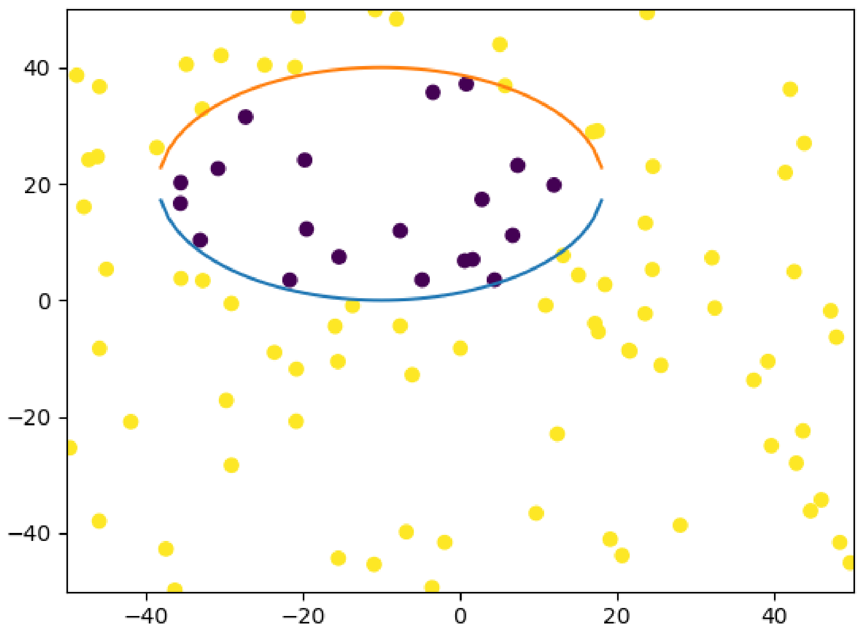

**3. (3 puntos) Supongamos ahora que las siguientes funciones definen la frontera de clasificación de los puntos de la muestra en lugar de una recta**

- $f(x,y) = (x-10)^2 + (y-20)^2 - 400$
- $f(x,y) = 0.5(x-10)^2 + (y-20)^2 - 400$
- $f(x,y) = 0.5(x-10)^2 - (y+20)^2 - 400$
- $f(x,y) = y - 20x^2 - 5x + 3$

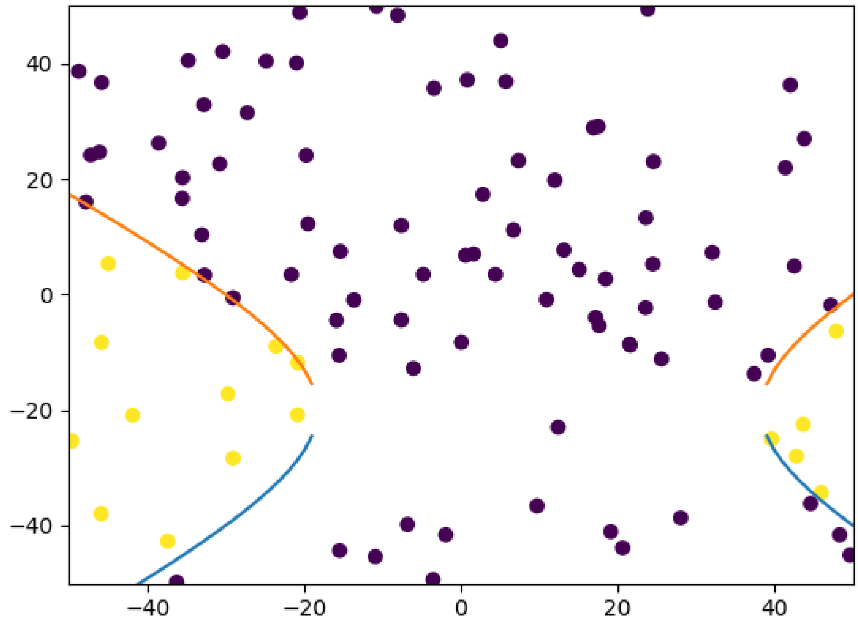
**Visualizar el etiquetado generado en 2b junto con cada una de las gráficas de cada una de las funciones. Comparar las formas de las regiones positivas y negativas de estas nuevas funciones con las obtenidas en el caso de la recta ¿Son estas funciones más complejas mejores clasificadoras que la función lineal? ¿En que ganan a la función lineal? Explicar el razonamiento.**



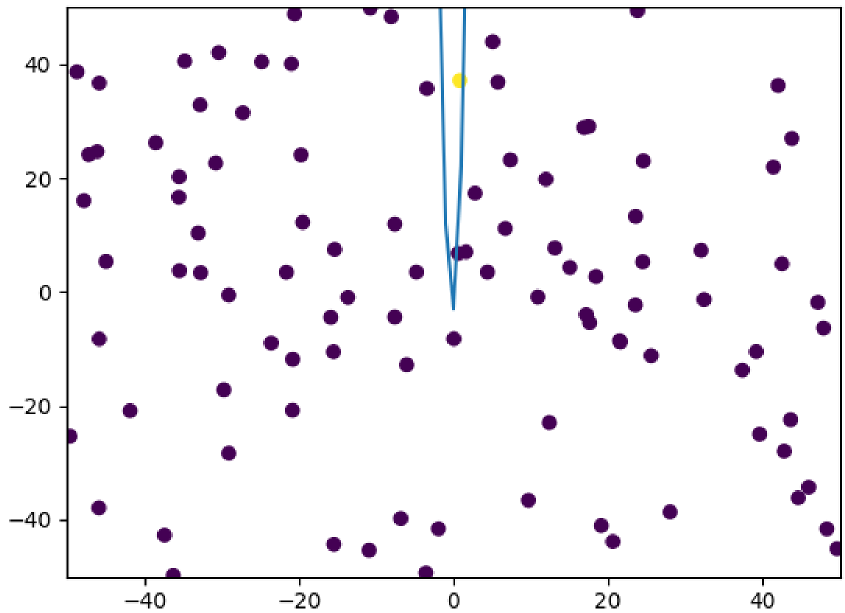

$$f(x, y) = (x - 10)^2 + \ (y - 20)^2 - 400$$




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




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




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

Podemos ver que en la gráfica del ejercicio anterior (apartado 2b) no podemos encontrar ninguna recta que separe de manera lineal las dos clases. En las funciones cuadráticas ocurre lo mismo, excepto en la última, donde hay infinitas rectas que pueden separar en dos clases. Este conjunto de rectas son las que dejan a los puntos por encima de la misma. En este caso se obtendría un cien por cien de acierto en la clasificación.

# 2. MODELOS LINEALES

**1. (3 puntos) Algoritmo Perceptron: Implementar la función ajusta_PLA(datos; label; max_iter; vini) que calcula el hiperplano solución a un problema de clasificación binaria usando el algoritmo PLA. La entrada datos es una matriz donde cada ítem con su etiqueta está representado por una fila de la matriz, label el vector de etiquetas (cada etiqueta es un valor +1 o -1), max_iter es el número máximo de iteraciones permitidas y vini el valor inicial del vector.**
**La función devuelve los coeficientes del hiperplano.**

a) **Ejecutar el algoritmo PLA con los datos simulados en los apartados 2a de la sección.1. Inicializar el algoritmo con: a) el vector cero y, b) con vectores de números aleatorios en [0,1] (10 veces). Anotar el número medio de iteraciones necesarias en ambos para converger. Valorar el resultado relacionando el punto de inicio con el número de iteraciones.**

```
Iteraciones necesarias con etiquetas sin ruido
5
La media de las iteraciones necesitadas es
7.9
```

Con el vector inicial a 0, el algoritmo necesita 5 iteraciones para ajustar la recta y una media de 7.9 iteraciones si el vector inicial es de números aleatorios. Por tanto podemos ver que si el vector está inicializado a 0 tarda menos iteraciones en realizar el ajuste.

b) **Hacer lo mismo que antes usando ahora los datos del apartado 2b de la sección.1. ¿Observa algún comportamiento diferente? En caso afirmativo diga cuál y las razones para que ello ocurra.**

```
Iteraciones necesarias con etiquetas con ruido
1000
La media de las iteraciones necesitadas es
1000.0
```

Si ejecutamos el algoritmo utilizando etiquetas con ruido (las del apartado 2b de la sección 1) podemos ver que necesita el máximo de iteraciones, sean las que sean, ya que no existe ninguna separación lineal que pueda separar los datos. En la prueba se han probado con diferentes valores de iteraciones máximas, siendo el resultado en todas igual.