Practica 2

Aprendizaje Automático

José Manuel Pérez Lendínez

Contents

1 EJERCIÓ SOBRE LA COMPLEJIDAD DE H Y EL RUIDO

1.1 Ejercicio 1

Dibujar una grafica con la nube de puntos de salida correspondiente.

1. **Considere N = 50, dim = 2, rango = [-50,+50] con simula_unif(N,dim,rango).**
   A continuación vemos la grafica generada con simula_unif(N,dim,rango).

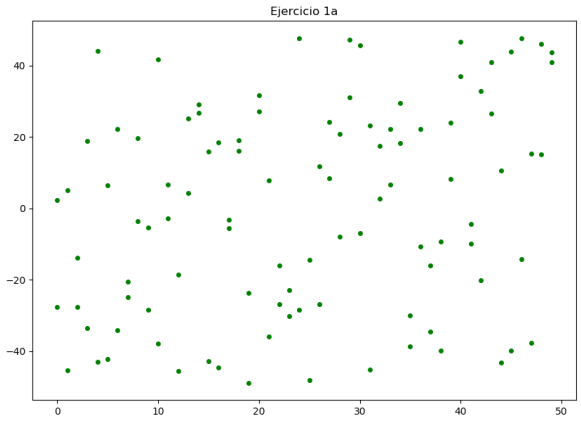

Figure 1:

2. **Considere N = 50, dim = 2, rango = [5,7] con simula_gaus(N,dim,rango).**
   Esta grafica es la generada por simula_gaus(N,dim,rango).

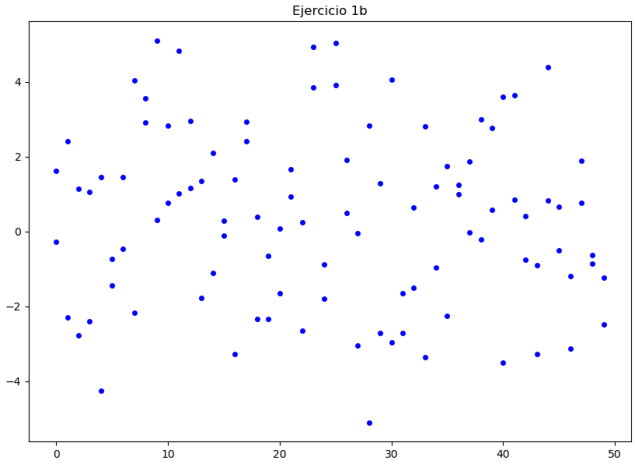

Figure 2:

1.2 Ejercicio 2

Con ayuda de la función simula_unif() generar una muestra de puntos 2D a los que 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()

1. **Dibujar de forma aleatoria los puntos muestren el resultado de su etiqueta, junto con la recta usada para ello. (Observe que todos los puntos están bien clasificados respecto la resta)**

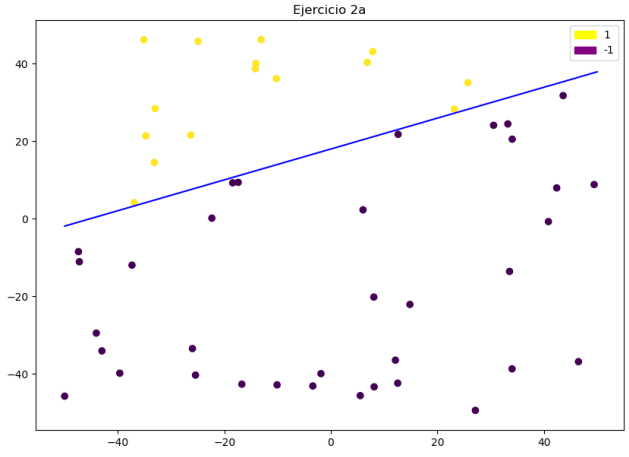

Figure 3:

Como se ve en la grafica los putos amarillos son los que se encuentran por encima de la recta y los morados los inferiores. La recta separa perfectamente los puntos.

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

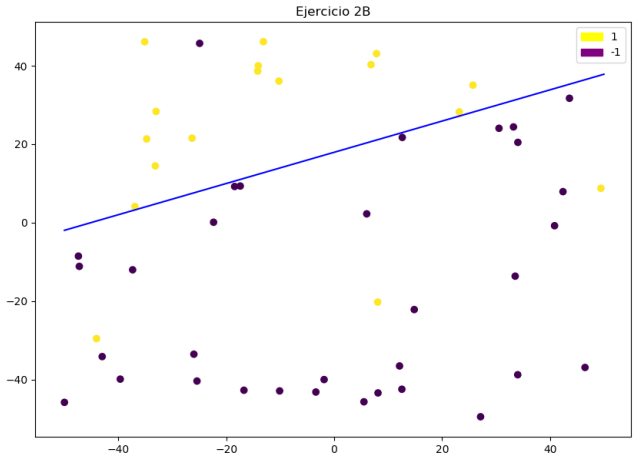

Figure 4:

En este caso no se separan totalmente los puntos al cambiar algunos de signo. Estos puntos se ven claramente al tener el color contrario al que le pertenecería.

1.3 Ejercicio 3

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

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

Visualizar el etiquetado generado en 2.2 junto con cada una de las gráficas de cada una de las funciones. Comparar las formas de las regiones y negativas de estas nuevas funciones con las obtenidas en el caso de la recta. ¿Son Estas funciones mas complejas mejores para

clasificar que la función linea? ¿En que ganan a la función lineal? Explicar el razonamiento.

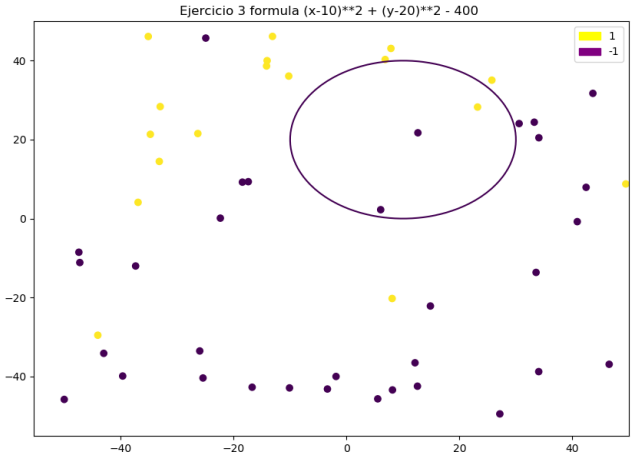

Figure 5:

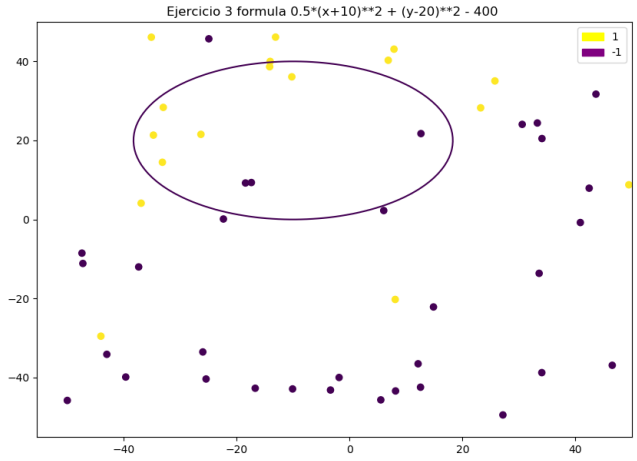

Figure 6:

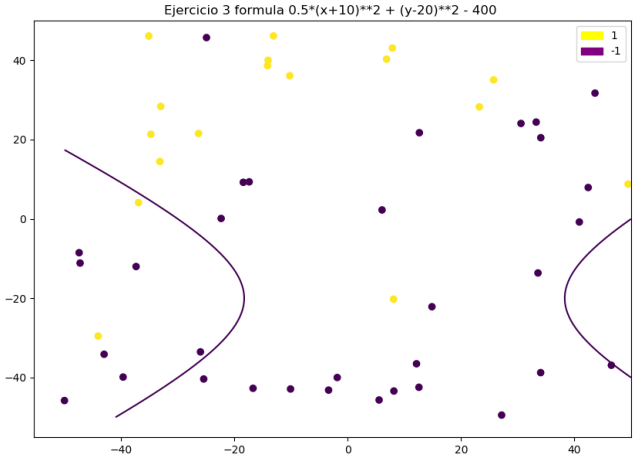

Figure 7:

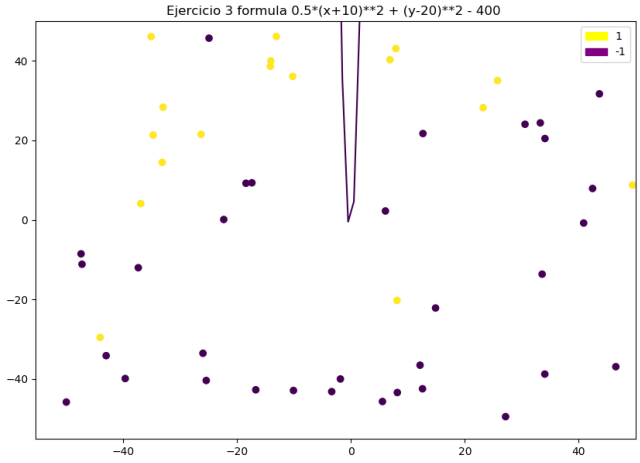

Figure 8:

Si son mas complejas estas funciones pero en este caso como los datos fueron generados para ser separados por un hiperplano estas formulas no se adaptan bien a los datos y los separan mucho peor que el hidroplano.

Con estos datos las funciones propuestas en este ejercicio no mejoran en nada a el hiperplano. Esto nos demuestra que elegir funciones mas complicadas no tiene porque mejorar nuestro modelos. Se tiene que buscar el tipo de función que mas se adapte a nuestros datos.

2 Modelos Lineales

2.1 Ejercicio 1

Implementar la función ajusta_PLA(datos,label,max_iter,vini) que calcula el hiperplano solución 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

1. **Ejecutar el algoritmo PLA con los datos simulados en el apartado 2.1 de la seccion 1. Inicializar el algoritmo con: a) 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 numero de iteraciones.**

   Con estos datos el PLA puede separar los datos perfectamente al ser posible usar un hiperplano para esto.

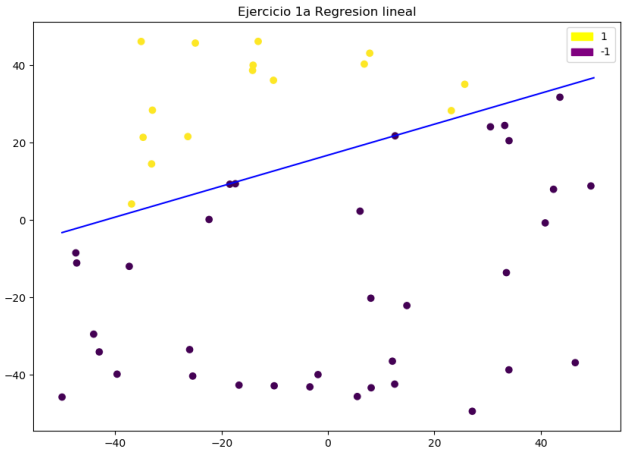

Figure 9:

   Cuando empezamos con el vector de 0 se alcanza en 86 iteraciones y con el vector aleatorio lo alcanzamos en 78.2 de media. Las dos opciones no alcanzan el máximo de iteraciones que en mi caso es 100.

2. **Hacer lo mismo que antes usando ahora los datos del apartado 2.2 de la sección 1. ¿Observa algún comportamiento diferente?**

En caso afirmativo diga cual y las razones para que ello ocurra.

En este caso como se le metió ruido y estos puntos cambiados de signo nos hacían los datos imposible de separar por un hiperplano y por esto no converge y siempre utiliza el máximo de iteraciones que le pasemos al algoritmo.

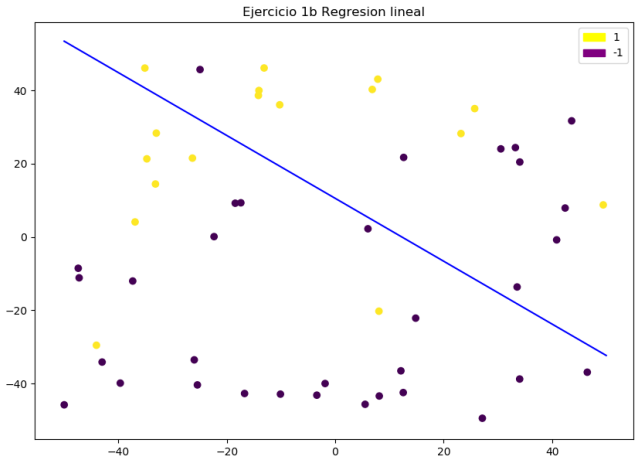

Figure 10:

2.2 Ejercicio 2

En este ejercicio crearemos nuestra propia función objetivo f (una probabilidad en este caso) y nuestro conjunto de datos D para ver como función regresión logística. Suponemos por simplicidad que f es una probabilidad con valores 0/1 y por tanto la etiqueta y es una función determinista de x. Consideramos d = 2 para que los datos sean visualizables, y sea X = [0,2]x[0,2] con probabilidad uniforme de elegir cada x $\in \mathcal{X}$. Elegir una línea en el plano que pase por X como la frontera entre f(x) = 1 (donde y toma valores +1 y f(x) = 0 (donde y toma valores-1), para ello seleccionar dos puntos aleatorios del plano y calcular la línea que pasa por ambos. Seleccionar N = 100 puntos aleatorios $\{x_n\}$ de X y evaluar las respuestas $\{y_n\}$ de todos ellos respecto de la frontera elegida.

Los datos los generaremos con las siguientes lineas.
Obtenemos la recta.
a,b = simula_recta((0,2)).
Generamos los puntos y las etiquetas.
X=simula_unif(100,2,(0,2))
Y = np.sign(X.T[1] - a * X.T[0] -b)

1. **Implementar Regresión Logística con gradiente descendente estocástico bajo las siguientes condiciones:**
   Para el algoritmo del gradiente descendente he utilizado el código de la practica anterior. He modificado la forma de obtener los pesos utilizando la formula de la regresión lineal. $\nabla_w E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{i=0}^{N} -y_i\mathbf{x}_i\sigma\left(-y_i\mathbf{w}^{\text{T}}\mathbf{x}_i\right)$ donde $\sigma(x) = \frac{1}{1+e^{-x}}$

   (a) **Iniciar el vector de pesos con valores 0.**

   (b) **Para el algoritmo cuando $\left\|\mathbf{w}^{(\text{t}-1)} - \mathbf{w}^{(\text{t})}\right\| < 0,01$ donde $\mathbf{W}^{(t)}$ denota el vector de pesos al final de la época t. Una época es un pase completo a través de los N datos.**

   (c) **Aplicar una permutación aleatoria, $1, 2, \ldots, N$, en el orden de los datos antes de usarlos en cada época del algoritmo.**

   (d) **Usar una tasa de aprendizaje de n = 0,01**

   El algoritmo nos da la siguiente grafica.

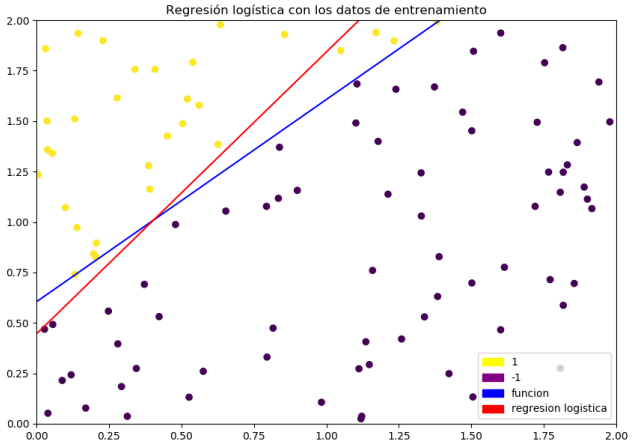

Figure 11:

Como se ve no llega a ajustar del todo bien debido a que fijamos que parar cuando la resta de pesos de menos que 0.01. Para que se ajustara mejor se tendría que reducir ese valor y darle mas numero de iteraciones máximo.

2. **Usar la muestra de datos etiquetada para encontrar nuestra solución g y estimar E_{out} usando para ello un numero suficientemente grande de nuevas muestras** (> 999)

   En mi caso he obtenido una muestra de 3000 puntos con las siguientes lineas.

x_test=simula_unif(3000,2,(0,2))
y_test = np.sign(x_test.T[1] - a * x_test.T[0] -b)

En la grafica muestro en azul la linea de la función utilizada para clasificar los puntos y la linea roja sera nuestra regresión logística.

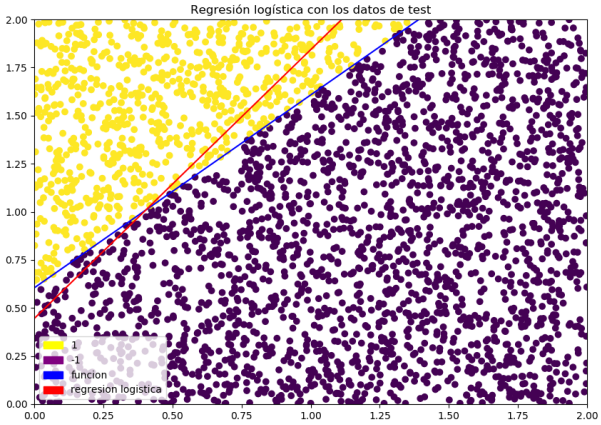

Figure 12:

El error que nos da es de 0.0416

3 Bibliografía

Para dibujar las graficas del ejercicio 3 del primer apartado, he utilizado el código dado en la siguiente pagina:
https://stackoverflow.com/questions/32092899/plot-equation-showing-a-circle