Aprendizaje Automático

Práctica 2

Curso 2019-2020

Javier Gálvez Obispo
javiergalvez@correo.ugr.es
Grupo 2 Martes 17:30 – 19:30

1. Ejercicio sobre la complejidad de H y el ruido

1. 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)*.

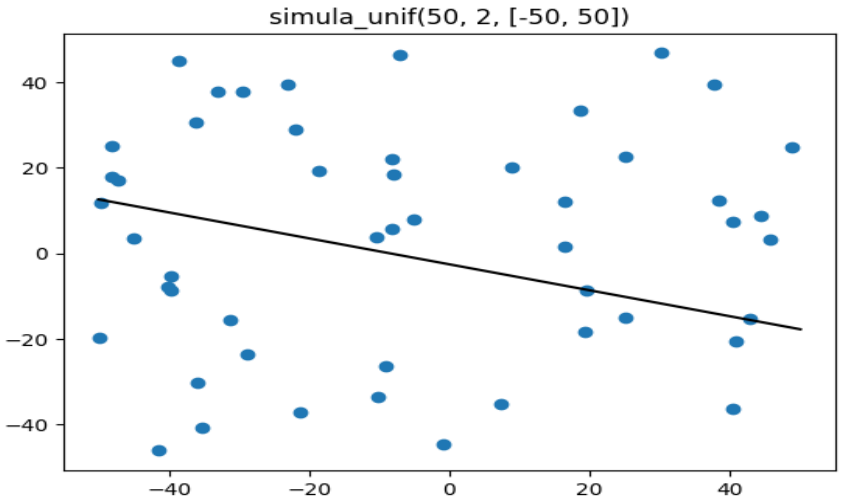

Figure 1: Gráfica simula_unif(N, dim, rango)

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

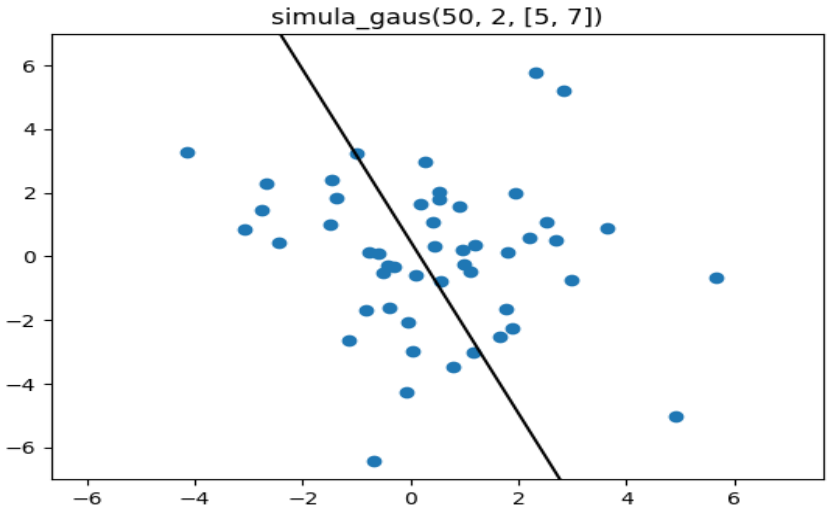

Figure 2: Gráfica simula_gaus(N, dim, rango)

2. Vamos a valorar la influencia del rudio en la seleccion de la complejidad de la clase defunciones. Con ayuda de la función *simula_unif*(100, 2, [−50, 50]) 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.

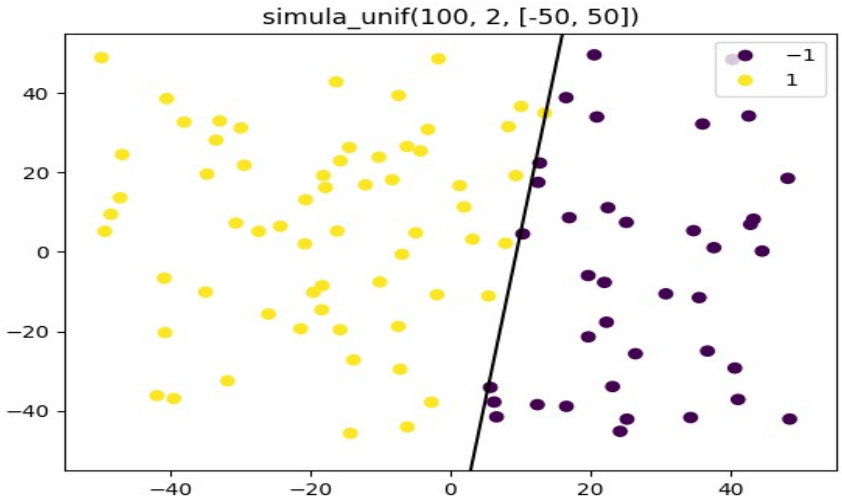

Figure 3: Gráfica simula_unif(100, 2, [-50, 50]) con etiquetas sign(y-ax-b)

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.

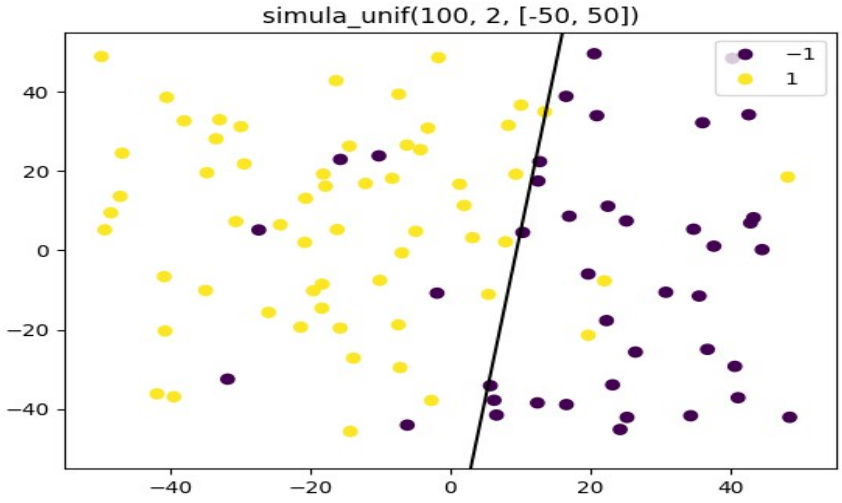

Figure 4: Gráfica simula_unif(100, 2, [-50, 50]) con ruido en las etiquetas

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

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

Visualizar el etiquetado generado en 2*b* junto con cada una de las gráficas de cada una de las funciones. Comparar las regiones positivas y negativas de estas nuevas funciones con las obtenidas en el caso de la recta. ¿Son estas funciones más complejas mejoresclasificadores que la función lineal? Observe las gráficas y diga que consecuenciasextrae sobre la influencia del proceso de modificación de etiquetas en el proceso deaprendizaje. Explicar el razonamiento.

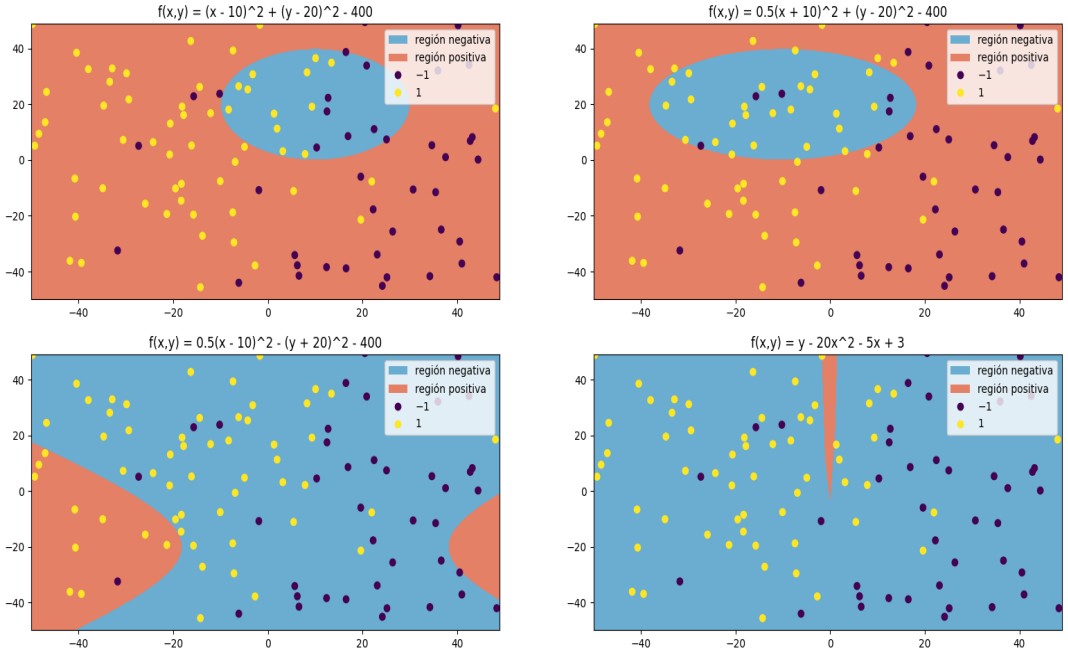

Figure 5: Comparativa de las fronteras para las distintas f(x, y)

Al haber introducido ruido éstas funciones más complejas podrían dar mejor resultado que una recta pero, como podemos observar en la figura de arriba, todas las funciones están muy lejos de hacer una buena separación de los datos. El hecho de haber introducido ruido en los datos no cambia que éstos hayan sido etiquetados según una recta y es por ello que funciones más complejas que una función lineal no obtienen mejores resultados.

2. Modelos lineales

a) **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 item 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 los apartados 2*a* 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.

   Se ha ejecutado PLA con *max_iter* = 200.

   Iteraciones de PLA hasta converger con *vini* = vector cero: 87

   Media de iteraciones para converger con *vini* = vector números aleatorios: 106.7

   Tomando el punto de inicio como un vector de números aleatorios en [0, 1] aumenta el número de iteraciones que tiene que realizar PLA para encontrar una solución, por lo que es importante elegir un buen punto de partida para el algoritmo converja lo más rápido posible.

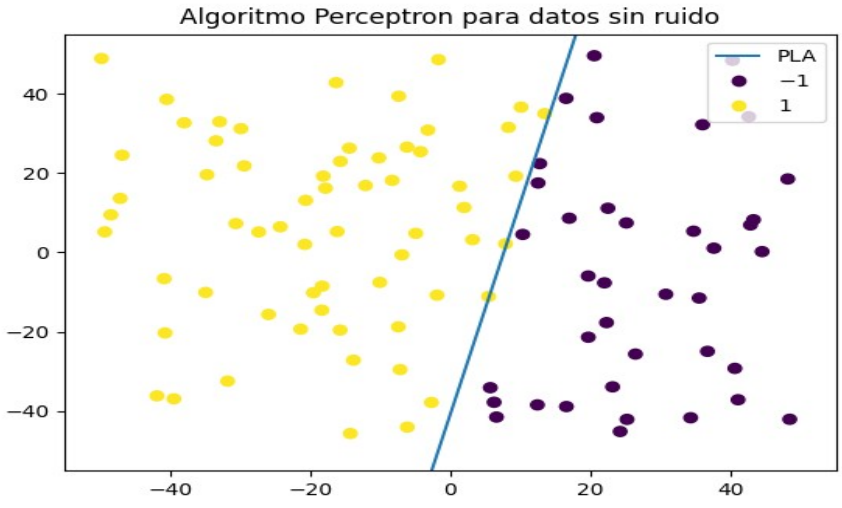

Figure 6: Gráfica resultado de ejecutar PLA con vini = vector cero

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

   Realizamos el mismo experimento pero ahora introducimos ruido en los datos.

   Se ha ejecutado PLA con *max_iter* = 200.

   Iteraciones de PLA hasta converger con *vini* = vector cero: 200

   Media de iteraciones para converger con *vini* = vector números aleatorios: 200

   Al introducir ruido PLA nunca va a encontrar una solución que separe los datos en dos, es imposible trazar una recta que divida los puntos de tal manera, como se puede ver en la *Figura 7*. Por esto, independientemente del punto de partida que se elija, PLA siempre va a realizar el máximo de iteraciones que se le pase como argumento.

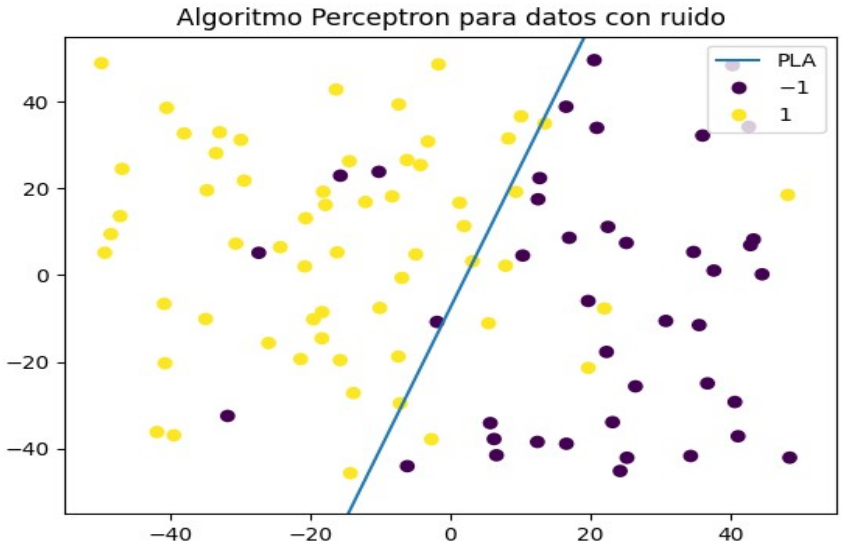

Figure 7: Gráfica resultado de ejecutar PLA con vini = vector cero al introducir ruido en los datos

b) **Regresión Logística:** En este ejercicio crearemos nuestra propia función objetivo *f* (una probabilidad en este caso) y nuestro conjunto de datos *D* para ver cómo funciona regresión logística. Supondremos por simplicidad que *f* es una probabilidad con valores 0/1 y por tanto que la etiqueta *y* es una función determinista de *x*.

Consideremos $d = 2$ para que los datos sean visualizables, y sea $X = [0,2]\times[0,2]$ con probabilidad uniforme de elegir cada $x \in X$. 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.

1) Implementar Regresión Logística (RL) con Gradiente Descendente Estocástico (SGD) bajo las siguientes condiciones:

   - Inicializar el vector de pesos con valores 0.
   - Parar el algoritmo cuando $\|w(t-1)-w(t)\| < 0.01$, donde 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.
   - Aplicar una permutación aleatoria, 1, 2, ..., *N*, en el orden de los datos antes de usarlos en cada época del algoritmo
   - Usar una tasa de aprendizaje de $\eta = 0.01$

   Consultar el código para ver la implementación

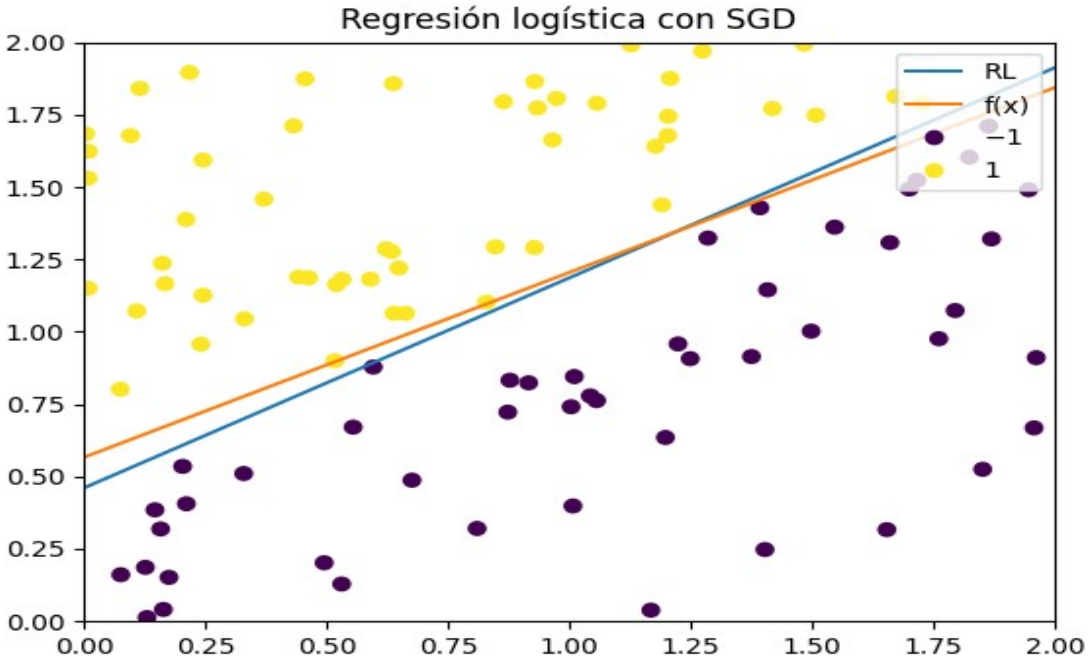

Figure 8: Gráfica resultado de la Regresión Logística con Gradiente Descendente Estocástico

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

   $E_{in} = 0.08844235694294757$

   Para una muestra de tamaño 2000

   $E_{out} = 0.10141306095193754$

3. Bonus

3. **Clasificación de Dígitos.** Considerar el conjunto de datos de los dígitos manuscritos y seleccionar las muestras de los dígitos 4 y 8. Usar los ficheros de entrenamiento (training) y test que se proporcionan. Extraer las características de **intensidad promedio** y **simetría** en la manera que se indicó en el ejercicio 3 del trabajo 1.

a) Plantear un problema de clasificación binaria que considere el conjunto de entrenamiento como datos de entrada para aprender la función *g*.

   Etiquetaremos los dígitos 4 como 1 y los 8 como -1

b) Usar un modelo de Regresión Lineal y aplicar PLA-Pocket como mejora. Responder a las siguientes cuestiones.

   El modelo de Regresión Lineal que vamos a usar es el algoritmo de la pseudoinversa visto en la práctica anterior y el resultado obtenido al aplicarlo a los datos será el valor inicial de **w** para PLA-Pocket.

   1) Generar gráficos separados (en color) de los datos de entrenamiento y test junto con la función estimada.

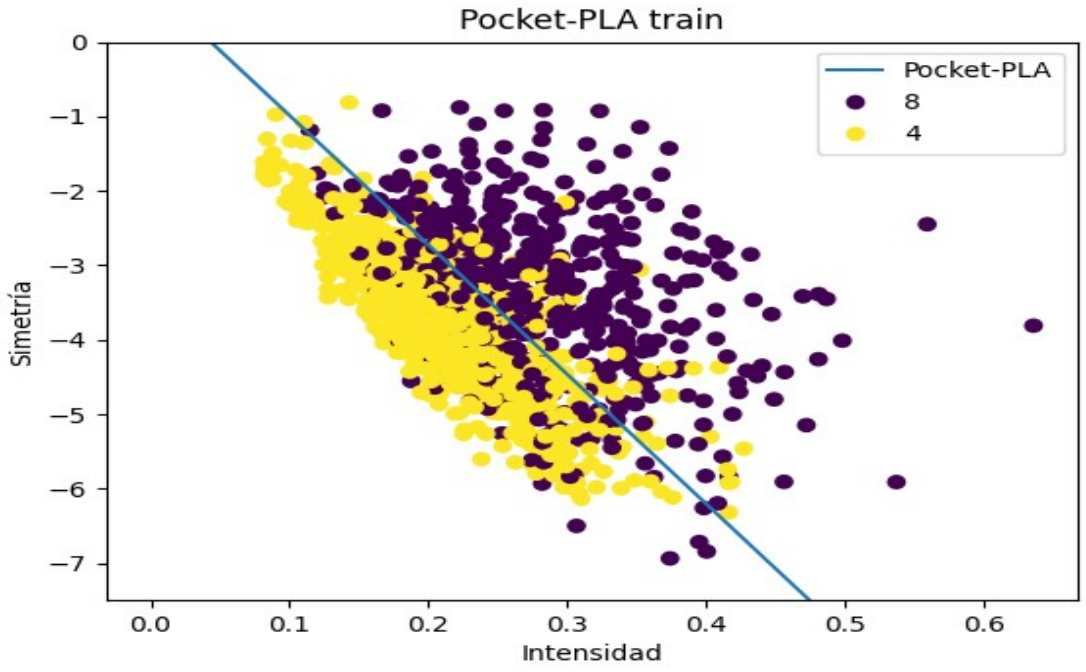

Figure 9: Gráfica de la función estimada de PLA-Pocket junto a los datos de entrenamiento

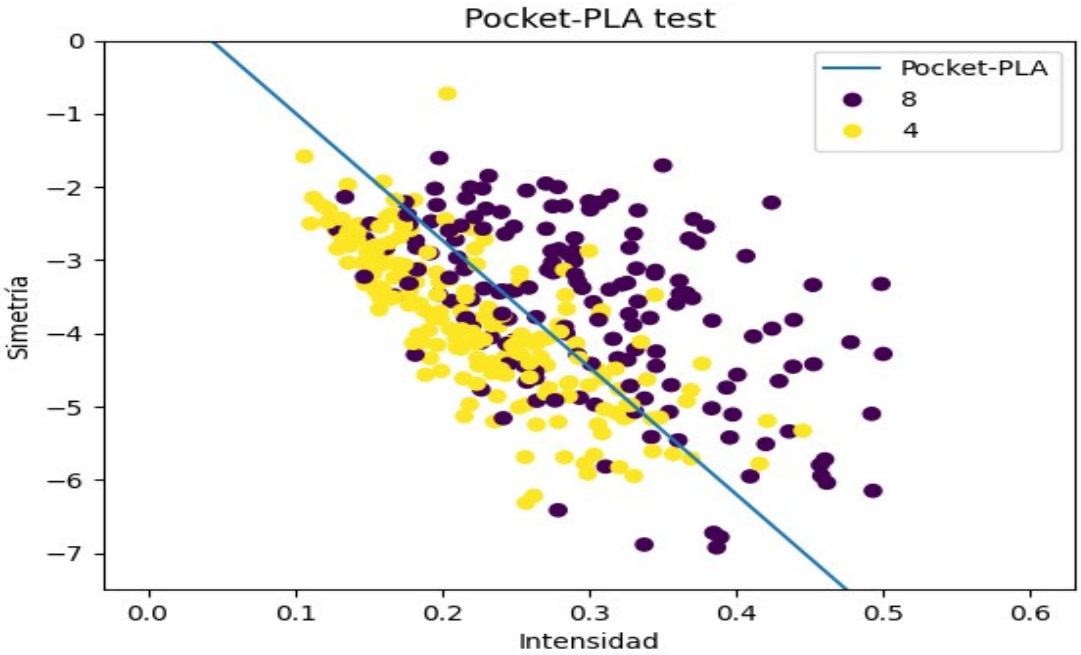

Figure 10: Gráfica de la función estimada de PLA-Pocket junto a los datos de test

2) Calcular E_{in} y E_{test}

   E_{in} = 0.2135678391959799

   E_{test} = 0.2459016393442623

3) Obtener cotas sobre el verdadero valor de E_{out}. Pueden calcularse dos cotas una basada en E_{in} y otra basada en E_{test}. Usar una tolerancia $\delta = 0.05$. ¿Que cota es mejor?

   Calculamos la cota de E_{out} de la siguiente forma:

$$E_{out}(h) \leq E_{in}(h) + \sqrt{\frac{1}{2N}\log\frac{2}{\delta}}$$

   Al utilizar E_{in} para calcular la cota obtenemos

$$E_{out}(h) \leq 0.2528712346006071$$

   y al basarnos en E_{test}

$$E_{out}(h) \leq 0.3168907427768778$$

   Podemos considerar E_{test} como una estimación de E_{out} ya que es una muestra fuera del conjuto de datos utilizados para calcular g y, como podemos comprobar, E_{test} es menor que la cota obtenida a partir de E_{in}. Claramente la cota de E_{in} será mejor que la de E_{test}, como se observa en los resultados.