UNIVERSIDAD
DE GRANADA

Aprendizaje Automático. Complejidad de H y Modelos Lineales

Ruido y complejidad, Algoritmo de aprendizaje del Perceptrón, Regresión Logística

Ricardo Ruiz Fernández de Alba

Escuela Técnica Ingeniería Informática y Matemáticas

DECSAI

Universidad de Granada

16 de abril de 2022

Índice general

1

Sobre la complejidad de H y el ruido

En este capítulo, trataremos la dificultad que introduce la aparición de ruido en las etiquetas a la hora de elegir la clase de funciones más adecuadas.

1.1 | Dibujar gráficas de nubes de puntos simuladas

1.1.1 | Uniformemente distribuidos

Considere $N = 50$, $\dim = 2$, $rango = [-50, 50]$ **con** `simula_unif(N, dim, rango)`

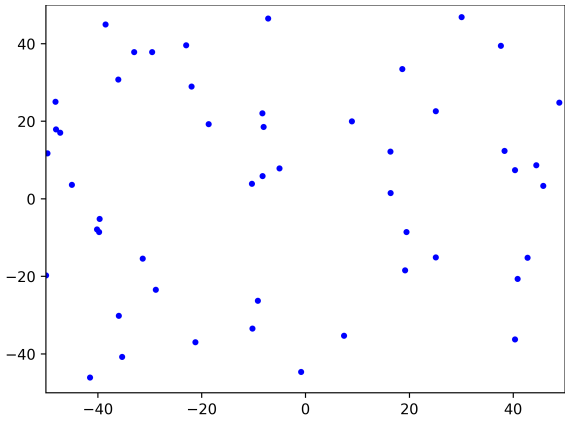


Figura 1.1: Gráfica de nube de puntos uniformemente distribuidos

1.1.2 | Siguiendo distribución gaussiana de media 0 varianza dada

Considere $N = 50$**,** $dim = 2$ **y** $sigma = [5, 7]$ **con** `simula_gauss(N, dim, sigma)`

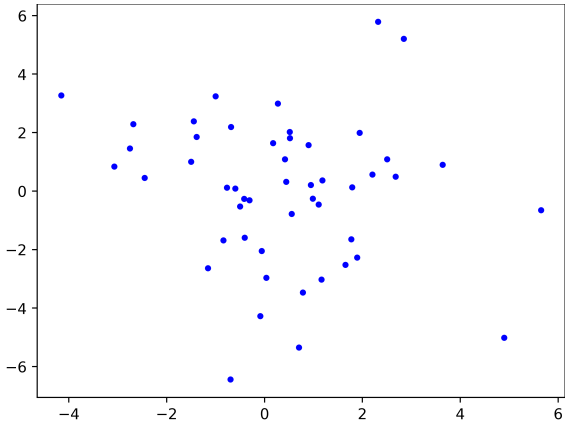


Figura 1.2: Gráfica de nube de puntos en distribución gaussiana.

Al seguir una distribución normal de media cero y varianza sigma, los puntos se acumulan en $\left[-\sqrt{5}, \sqrt{5}\right] \times \left[-\sqrt{7}, \sqrt{7}\right] \approx [-2.2, 2.2] \times [-2.6, 2.6]$

1.2 | Ejercicio 2

Vamos a valorar 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 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 cada punto con respecto a la recta simulada con `simula_recta()`.

1.2.1 | Dibujo de puntos con etiqueta y recta usada

Dibujamos un gráfico 2D con los puntos clasificados por etiquetas junto con la recta usada para etiquetar.

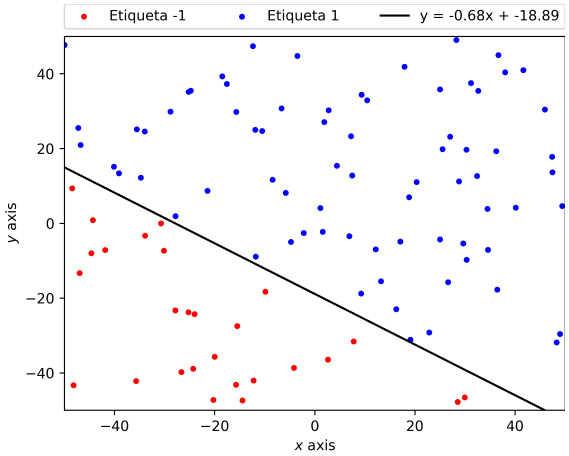


Figura 1.3: Etiquetado de puntos uniformemente distribuidos según recta.

Es claro que si usamos una recta para etiquetar los puntos en dos clases, estos datos está bien clasificados por esta recta.

1.2.2 | Añadir ruido aleatorio

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

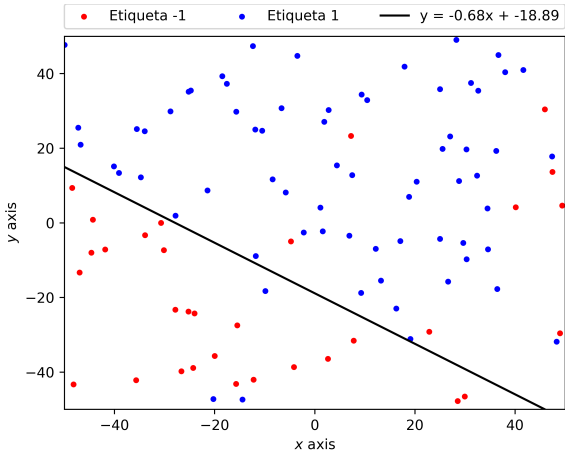


Figura 1.4: Nube de puntos anterior con 10 % de ruido en cada etiqueta.

En efecto, 3 puntos con etiqueta -1 (rojos) ahora tienen etiqueta 1 (son azules). Esto es el 10 % del total (27) redondeado.

1.2.3 | Otras fronteras de clasificación

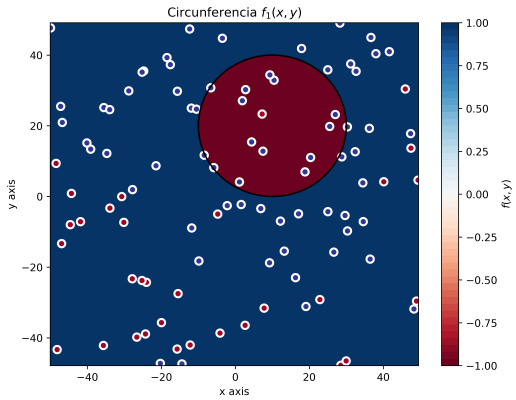
Supongamos ahora que las siguientes funciones (f_1, f_2, f_3, f_4) definen la frontera de clasificación de los puntos de la muestra en lugar de una recta.

Visualizar el etiquetado generado en el apartado 2b junto con la gráfica 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. Argumente si estas funciones más complejas son mejores clasificadores que la función lineal. Observe las gráficas y diga qué consecuencias extrae sobre la influencia de la modificación de etiquetas en el proceso de aprendizaje. Explique el razonamiento.

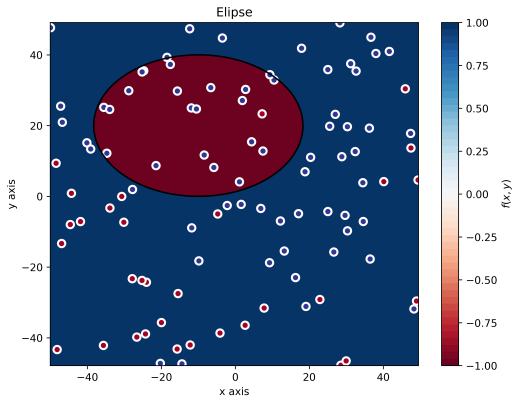
Usando la función `plot_datos_cuad` proporcionada en la plantilla de código, podemos visualizar y comparar las regiones positivas y negativas (azul y rojo respectivamente) que define la frontera dada por $f_i(x, y) = 0$ con $i = 1, 2, 3, 4$.

1.2.3.1 | Circunferencia y Elipse

- $f_1(x, y) = (x - 10)^2 + (y - 20)^2 - 400$
- $f_2(x, y) = \frac{(x+10)^2}{2} + (y - 20)^2 - 400$



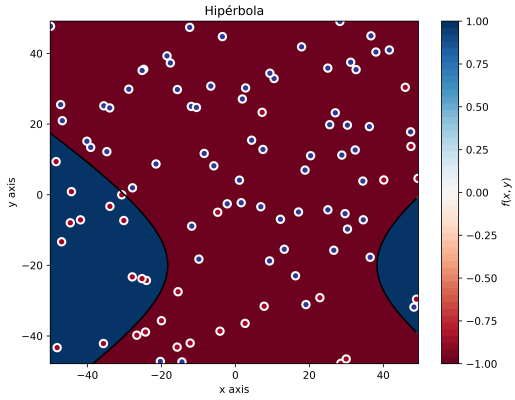
(a) Muestra clasificada por circunferencia f_1



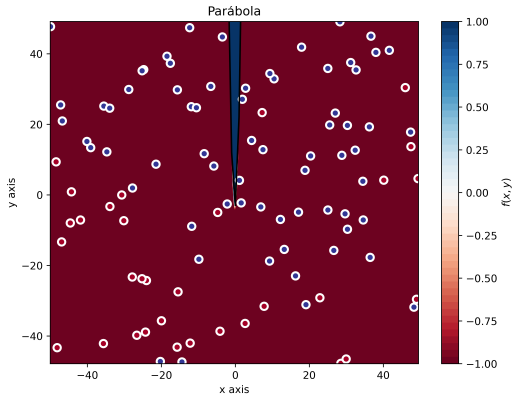
(b) Muestra clasificada por elipse f_2

1.2.3.2 | Hipérbola y Parábola

- $f_3(x,y) = \frac{(x-10)^2}{2} - (y+20)^2 - 400$
- $f_4(x,y) = y - 20x^2 - 5x + 3$



(a) Muestra clasificada por hipérbola f_3



(b) Muestra clasificada por parábola f_4

Si calculamos el error de clasificación en cada uno de los casos obtenemos:

f_i	Error de clasificación (%)
Recta f	10 %
Circunferencia f_1	44 %
Elipse f_2	52 %
Hipérbola f_3	81 %
Parábola f_4	68 %

Podemos confirmar lo que se podía intuir por las grafícas obtenidas. A pesar de ser funciones más complejas que la lineal, no son mejores clasificadores.

En cuanto a la influencia del ruido, no es posible obtener un mejor clasificador que la recta f. Esto se debe a que el ruido ha sido generado aleatoriamente y que es la propia recta f la que se ha usado para etiquetar. Así, el 10 % es cota inferior del de error de clasificación para distinto $\mathcal{H}$.

De hecho, para funciones demasiado complejas es probable que tengamos un problema de **sobreajuste** (overfitting) impidiendo una buena generalización (alto valor de E_{out}).

2

Modelos Lineales

2.1 | Algoritmo de aprendizaje del Perceptrón (PLA)

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.

2.1.1 | Ejecutar el algoritmo PLA con los datos empleados en el apartado 2a del ejercicio 1.

Inicializar el algoritmo con:

- el vector cero y,
- con vectores de números aleatorios en $[0, 1]$ (10 veces).

Anotar el número medio de iteraciones necesarias en ambos para converger.

Se deben mostrar en una tabla cada uno de los pesos iniciales empleados, los finales (obtenidos tras el proceso de entrenamiento), y el porcentaje de error de clasificación. Valorar el resultado relacionando el punto de inicio con el número de iteraciones.

Vector inicial	Iteraciones	Coeficientes w	Error de clasificación
[0.000, 0.000, 0.000]	75	[661.00, 23.20, 32.39]	0 %
[0.574, 0.349, 0.057]	257	[1115.57, 43.48, 62.12]	0 %
[0.229, 0.664, 0.497]	43	[464.23, 15.39, 23.75]	0 %
[0.519, 0.175, 0.571]	231	[1078.52, 39.47, 53.76]	0 %
[0.997, 0.817, 0.594]	71	[664.00, 23.15, 31.90]	0 %
[0.976, 0.902, 0.596]	76	[661.98, 24.90, 36.20]	0 %
[0.032, 0.094, 0.065]	59	[558.03, 19.36, 29.71]	0 %
[0.452, 0.375, 0.975]	274	[1145.45, 40.28, 60.81]	0 %
[0.168, 0.973, 0.767]	235	[1089.17, 39.45, 53.53]	0 %
[0.824, 0.633, 0.669]	257	[1148.82, 39.90, 60.95]	0 %
[0.477, 0.013, 0.353]	74	[673.48, 22.59, 31.35]	0 %
Promedio	157.7		0 %

2.1.2 | Hacer lo mismo usando los datos del apartado 2b del ejercicio 1.

¿Observa algún comportamiento diferente? En caso afirmativo diga cuál y las razones para que ello ocurra.

Vector inicial	Iteraciones	Coeficientes w	Error de clasificación
[0.000, 0.000, 0.000]	5000	[339.00, 24.81, 55.86]	24 %
[0.492, 0.730, 0.469]	5000	[348.49, 17.77, 42.43]	24 %
[0.457, 0.138, 0.011]	5000	[349.46, 16.66, 44.13]	24 %
[0.758, 0.320, 0.984]	5000	[339.76, 21.61, 56.29]	22 %
[0.220, 0.339, 0.524]	5000	[336.22, 17.72, 43.85]	25 %
[0.755, 0.464, 0.125]	5000	[336.75, 17.32, 53.11]	27 %
[0.313, 0.505, 0.674]	5000	[339.31, 15.10, 26.87]	20 %
[0.770, 0.130, 0.023]	5000	[333.77, 27.72, 49.69]	20 %
[0.519, 0.810, 0.013]	5000	[343.52, 12.09, 38.81]	22 %
[0.672, 0.687, 0.449]	5000	[331.67, 12.87, 21.73]	19 %
[0.915, 0.644, 0.005]	5000	[341.91, 21.10, 56.13]	22 %
Promedio	5000		22.5 %

2.2 | Regresión Logística (RL).

En este ejercicio emplearemos nuestra propia función objetivo f y un conjunto de datos D para ver cómo funciona regresión logística. Consideraremos d = 2 para que los datos

sean fácilmente visualizables, y emplearemos $X = [0,2] \times [0,2]$ con probabilidad uniforme de elegir cada $x \in X$. Elegir una línea en el plano que pase por X como la frontera que separa la región en donde y toma valores $+1$ y -1.

Para ello, seleccionar dos puntos aleatorios de X y calcular la línea que pasa por ambos.

Impleméntese RL con Gradiente Descendente Estocástico (SGD) del siguiente modo:

- 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. Recuérdese que una época es un pase completo a través de los N ejemplos de nuestro conjunto de datos.
- Aplicar una permutación aleatoria de $\{1, 2, ..., N\}$ a los índices de los datos, antes de usarlos en cada época del algoritmo.

A continuación, empleando la implementación anterior, realícese el siguiente experimento:

- Seleccione $N = 100$ puntos aleatorios $\{x_n\}$ de X y evalúe las respuestas $\{y_n\}$ de todos ellos respecto de la frontera elegida.
- Ejecute RL para encontrar la función solución g, y evalúe el error E_{out} usando para ello una nueva muestra de datos (> 999). Se debe escoger experimentalmente tanto el learning rate (tasa de aprendizaje η) como el tamaño de batch.
- Repita el experimento 100 veces, y calcule los valores promedio de E_{out}, de porcentaje de error de clasificación, y de épocas necesarias para converger.

Bibliografía