Aprendizaje Automático: Práctica 2

David Cabezas Berrido

Índice

1. Ejercicio sobre la complejidad de H y el ruido **2**
1.1. Dibujar nubes de puntos . . . 2
1.2. Muestra etiquetada por una recta . . . 2
1.3. Comparación con otras funciones frontera . . . 3

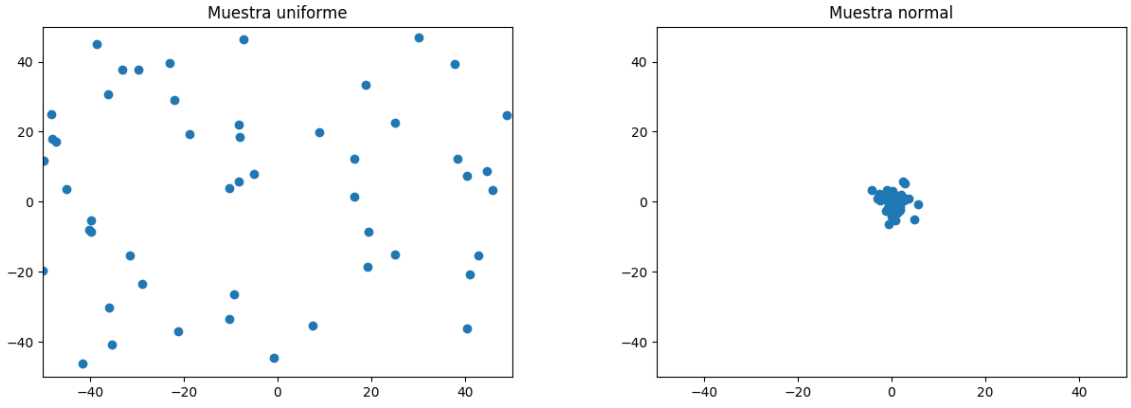
2. Modelos lineales **4**
2.1. Algoritmo Perceptron . . . 4
2.2. Regresión Logística . . . 5

3. Bonus: Clasificación de dígitos **6**

1. Ejercicio sobre la complejidad de H y el ruido

1.1. Dibujar nubes de puntos

Utilizando las funciones `simula_unif` y `simula_gaus`, he generado dos muestras en dimensión 2, de tamaño $N = 50$.
La primera muestra es una uniforme en el cuadrado $[-50, 50] \times [-50, 50]$ y la segunda muestra es una normal de media $(0, 0)$ y varianza $(5, 7)$ (desviación típica $(\sqrt{5}, \sqrt{7})$).
He mantenido fija la escala de los ejes para que se aprecie que los puntos generados por la normal están mucho más concentrados en torno al punto $(0, 0)$, la concentración es ligeramente mayor en el eje de abscisas debido a que tiene menor varianza, pero apenas se aprecia.



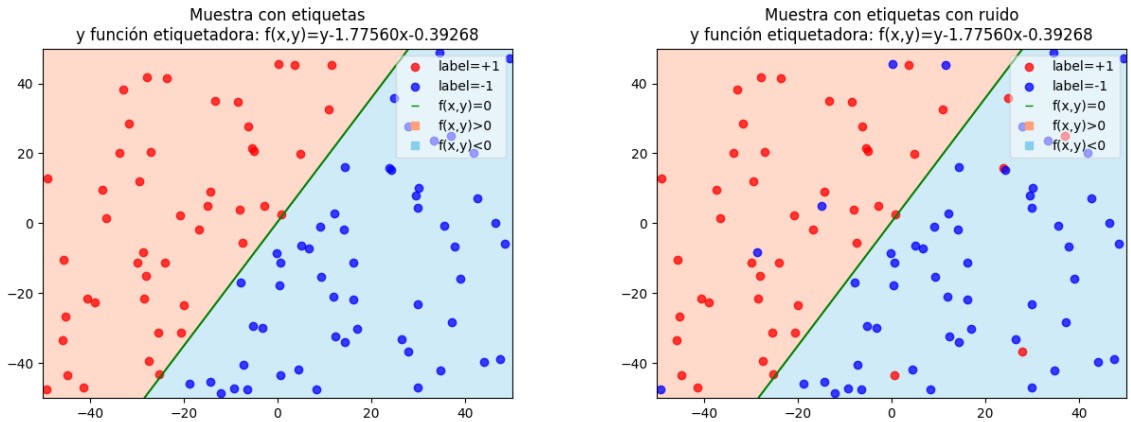
(a) $U\big((-50, 50) \times (-50, 50)\big)$ (b) $N\big((0, 0), (\sqrt{5}, \sqrt{7})\big)$

Figura 1: Muestras uniforme y normal

1.2. Muestra etiquetada por una recta

He generado (con `simula_unif`) una muestra de $N = 100$ puntos del cuadrado $[-50, 50] \times [-50, 50]$ y los he etiquetado con el signo de la distancia a una recta generada por `simula_recta` que corta a dicho cuadrado. Esto quiere decir que si la recta es $y = ax + b$, la etiqueta de un punto (x, y) es el signo de la función $y - ax - b$ (+1 si el punto queda por encima de la recta y -1 si queda por debajo). Obviamente la recta clasifica a la perfección todos los puntos.

A continuación, he añadido ruido a las etiquetas de la muestra (10 % en cada clase). Claramente la recta tiene un 10 % de puntos mal clasificados ahora.



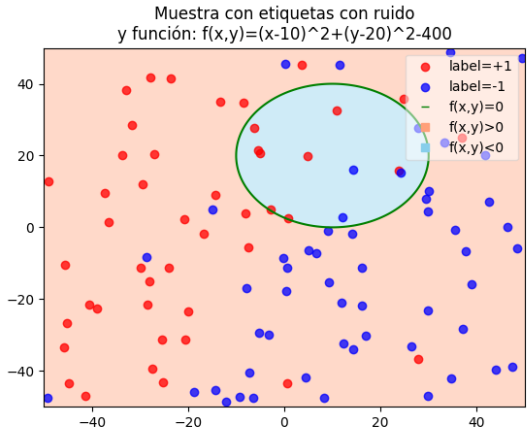
(a) Muestra sin ruido (b) Muestra con ruido

Figura 2: Muestra y recta etiquetadora

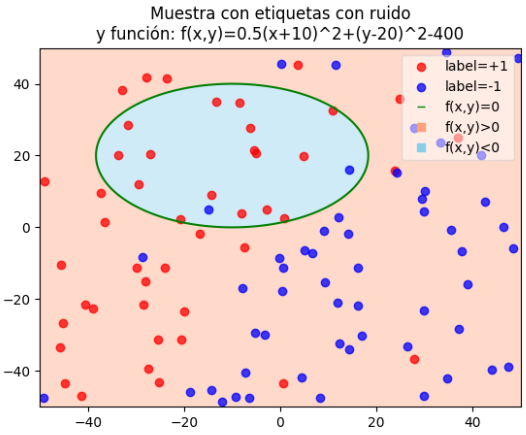
1.3. Comparación con otras funciones frontera

Ahora he representado la misma muestra (con ruido) junto con otras funciones, he calculado también el error que cometen dichas funciones funciones a la hora de clasificar la muestra.

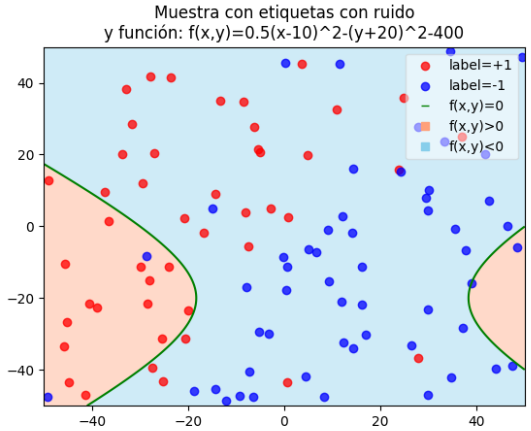
En la siguiente figura se ve que puntos clasifica bien y mal cada función, los bien clasificados son los que están en la región de su color. Incluyo también el porcentaje de puntos mal clasificados para cada función.



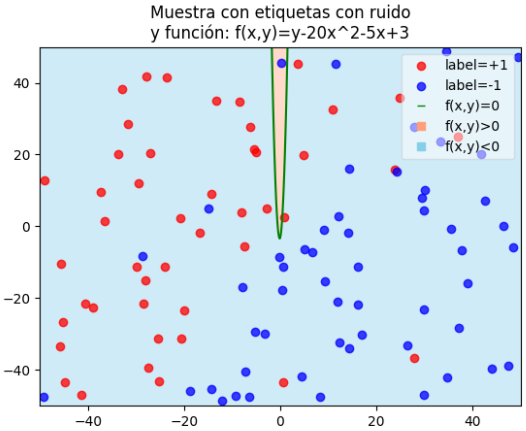
(a) $f(x, y) = (x - 10)^2 + (y - 20)^2 - 400$; Error: 58 %



(b) $f(x, y) = 0.5(x + 10)^2 + (y - 20)^2 - 400$; Error: 68 %



(c) $f(x, y) = 0.5(x - 10)^2 - (y + 20)^2 - 400$; Error: 35 %



(d) $f(x, y) = y - 20x^2 - 5x + 3$; Error: 47 %

Figura 3: Muestra con otras funciones como delimitadores

Ninguna de estas cuatro funciones ha conseguido si quiera acercarse al error del 10 % que presentaba la recta. Esto era de esperar, ya que la recta la he usado para etiquetar la muestra (salvo ese 10 % de ruido) y las funciones estaban prefijadas antes (no han recibido ningún aprendizaje ni ajuste).

Sin embargo, es perfectamente posible que el ruido de la muestra provoque que el mejor ajuste no sea la misma recta que usé para etiquetarla, en este caso se puede producir sobreajuste si la clase H es lo suficientemente compleja como para ajustar el ruido.

Por ejemplo, si ajustase la muestra sin ruido mediante una parábola $y = ax^2 + bx + c$ el peso a obtendría un valor muy cercano a 0. Pero en la muestra con ruido, esto puede no ocurrir, es probable que el valor que se obtenga para a al ajustar la muestra con ruido no sea cercano a 0, lo que ajustaría mejor la muestra (con ruido) pero se alejaría de la función objetivo, dando lugar a sobreajuste.

2. Modelos lineales

2.1. Algoritmo Perceptron

Caso de muestra separable:

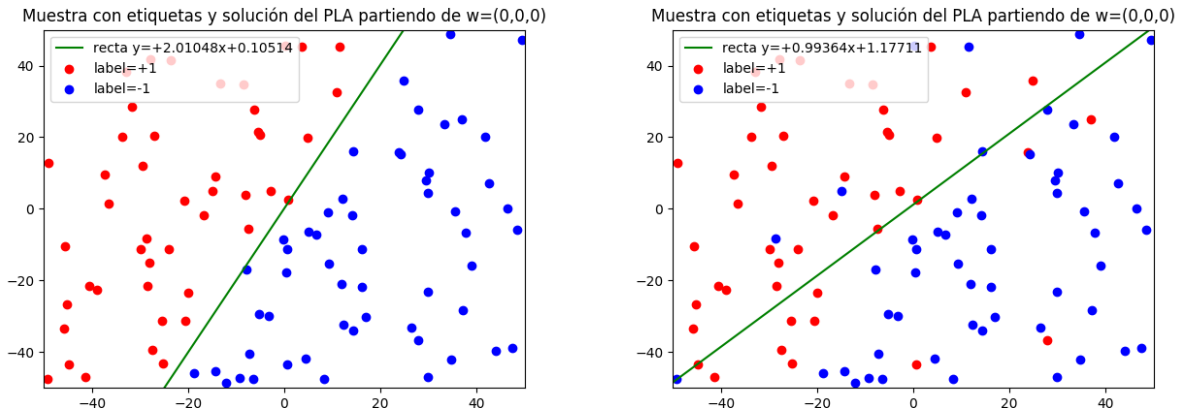
He ejecutado el algoritmo PLA sobre la muestra etiquetada del ejercicio $2a$) de la sección anterior y he anotado el número de iteraciones necesarias para que el algoritmo converja en 10 ejecuciones.

Partiendo del vector 0, en las 10 ejecuciones ha necesitado 2 iteraciones, ya que al partir del mismo punto el comportamiento del algoritmo es exactamente el mismo. La solución obtenida ha sido $w = [-3, -57.36159336, 28.53120771]$.

Luego, he realizado 10 ejecuciones partiendo de vectores de números aleatorios en $[0, 1]$, las iteraciones necesarias en cada ejecución han sido: 2, 3, 3, 2, 3, 2, 3, 2, 3, 3 (media 2.6). Ahora el algoritmo ha tenido comportamientos diferentes, ya que ha partido de un w_{ini} diferente en cada iteración. En este caso el número de iteraciones no presenta más que una ligera variación respecto al vector inicial, pero con otras muestras (variando la semilla) he obtenido resultados muy variados (tanto mejores como peores), luego no podemos concluir nada respecto a la eficiencia del algoritmo, al menos con un w inicial que no dependa de la muestra (probablemente partiendo de un ajuste por regresión lineal necesitaría menos iteraciones para converger).

Caso de muestra no separable:

He realizado el mismo experimento con la muestra etiquetada del ejercicio $2b$) de la sección anterior, que es la misma muestra pero con ruido en las etiquetas, este ruido hace que los datos no sean separables y que el algoritmo no converja. He limitado las iteraciones a 500 para que pare en algún momento y efectivamente ha agotado las 500 iteraciones en todos los casos, tanto partiendo del vector cero como de vectores de valores aleatorios en $[0, 1]$. Partiendo del vector cero, he obtenido la solución (que no separa todos los puntos) $w = [-45, -37.98579548, 38.22893039]$.



(a) Muestra separable

(b) Muestra no separable

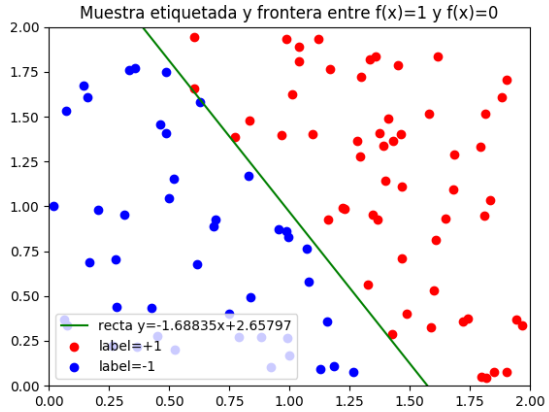
Figura 4: Recta obtenida con PLA partiendo de pesos nulos

2.2. Regresión Logística

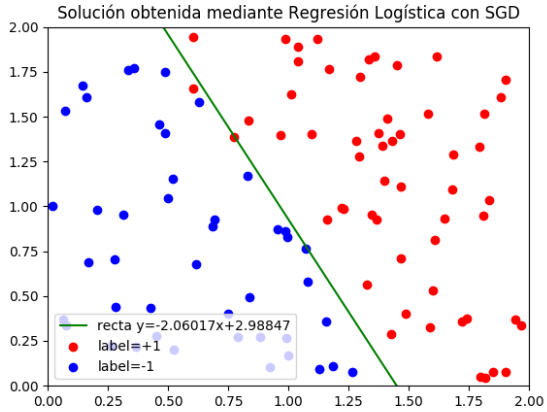
Quiero estimar una función f que asigna la probabilidad (0 ó 1) de que un elemento del cuadrado $\mathcal{X} = [0,2] \times [0,2]$ esté por encima (etiqueta $y = +1$) o por debajo (etiqueta $y = -1$) de una recta que corta a $\mathcal{X}$ (generada con `simula_recta`). Para ello, he elegido una muestra aleatoria de puntos de $\mathcal{X}$ de tamaño $N = 100$ y he aplicado Regresión Logística con Gradiente Descendiente Estocástico partiendo de $w = 0$ y parando cuando $\|w^{(t-1)} - w^{(t)}\| < 0.01$, donde $w^{(t)}$ denota el vector de pesos al final de la época t, considerando una época como un pase completo por los N datos de uno en uno, barajando los datos antes de cada época. Y con learning rate $\eta = 0.01$. Para asegurar que termina, lo he limitado a 5000 épocas.

Tras 490 épocas, se ha cumplido la condición de parada y he obtenido la solución $w = [-9.67809029, 6.67183542, 3.23847583]$, y un error en la muestra $E_{in} = 0.12321116789555302$.

La función g obtenida es de la forma $g(x) = \sigma(w^T x)$ donde $x \in \{1\} \times \mathcal{X} = \{1\} \times [0,2] \times [0,2]$ y σ denota la sigmoide: $\sigma(t) = \frac{e^t}{e^t+1} = \frac{1}{1+e^{-t}} \in (0,1) \quad \forall t \in \mathbb{R}$.



(a) Muestra junto con función objetivo



(b) Datos de entrenamiento junto con solución obtenida

Figura 5: Muestra con funciones objetivo y solución

En la gráfica de la derecha se representa la recta $\sigma(w^T x) = 0.5$, la frontera que separa los puntos $x \in \mathcal{X}$ para los que nuestra función g predice pertenencia a la clase del 8 $\big(g(x) = P(y = +1|x) > 0.5\big)$ de los puntos para los que predice pertenencia a la clase del 4 $\big(g(x) = P(y = +1|x) < 0.5\big)$.

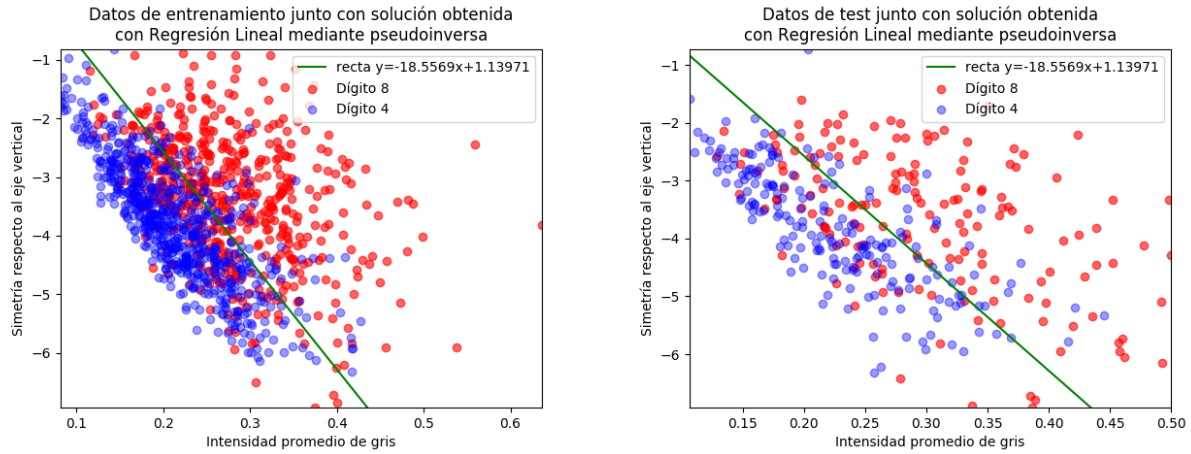
A continuación, he generado una nueva muestra de 1500 puntos nuevos de $\mathcal{X}$ y he calculado el error de Regresión Logística sobre ésta para obtener una estimación del valor de E_{out}. La estimación que he obtenido es $E_{out} \approx 0.12589042241455026$.

3. Bonus: Clasificación de dígitos

Tenemos un conjunto de dígitos manuscritos que queremos clasificar en dos clases, los que corresponden al dígito 4 y lo que corresponden al dígito 8. Para cada datos $\mathbf{x}$ consideramos dos características, x_1 la intensidad promedio de gris y x_2 la simetría respecto al eje vertical. Luego cada dato de la muestra será de la forma $\mathbf{x} = (1, x_1, x_2)$. Las etiquetas son $y = +1$ si $\mathbf{x}$ corresponde al dígito 8 e $y = -1$ si $\mathbf{x}$ corresponde al dígito 4.

Pretendemos aprender una función lineal g que prediga para cada $\mathbf{x}$ su etiqueta y. La función será de la forma $g(\mathbf{x}) = \text{sign}(w^T x)$ para algún vector de pesos w.

Para ello, primero he ajustado un modelo de Regresión Lineal, usando el algoritmo de la pseudoinversa. He obtenido los pesos $w = [-0.50676351, 8.25119739, 0.44464113]$ y los errores de clasificación (proporción de puntos mal clasificados) $E_{in} = 0.22780569514237856$ y $E_{test} = 0.25136612021857924$.

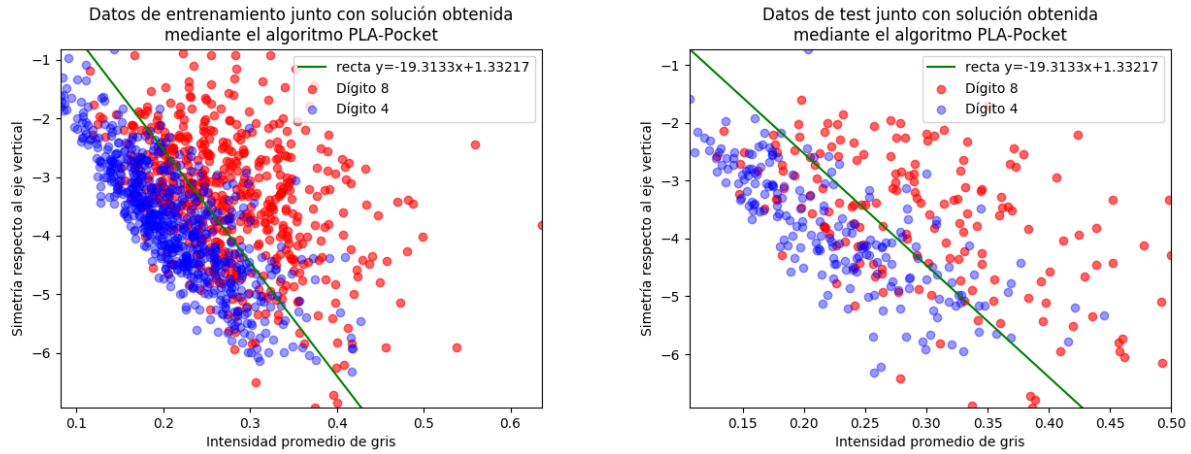


(a) Resultado regresión junto con datos de entrenamiento

(b) Resultado regresión junto con datos de test

Figura 6: Recta obtenida por Regresión Lineal con pseudoinversa

A continuación, he utilizado esa solución como valor inicial para el algoritmo PLA-Pocket con 100 épocas, he obtenido los pesos $w = [-6.50676351, 94.33278003, 4.88432863]$ y los errores de clasificación (proporción de puntos mal clasificados) $E_{in} = 0.22529313232830822$ y $E_{test} = 0.2540983606557377$.



(a) Resultado PLA-Pocket junto con datos de entrenamiento

(b) Resultado PLA-Pocket junto con datos de test

Figura 7: Recta obtenida con PLA-Pocket

Obviamente en PLA-Pocket E_{in} no puede empeorar respecto al inicial, he obtenido un E_{in} ligeramente más bajo y E_{test} ligeramente más alto al aplicar la mejora de PLA-Pocket.

Utilizando la desigualdad de Hoeffding

$$E_{out}(h) \leq E_{in}(h) + \sqrt{\frac{1}{2N} \log \frac{2|\mathcal{H}|}{\delta}} \qquad \text{con probabilidad} \geq 1 - \delta$$

puedo calcular una cota para E_{out} basada en E_{in} con tolerancia 0.05.

El problema es que el conjunto de rectas de $\mathbb{R}^2$ es continuo, así que tenemos que discretizarlo. ¿Cuántas rectas podemos codificar en nuestro ordenador?

La única forma de contemplar todas las posibles rectas con una única representación necesita 3 parámetros $w = (w_0, w_1, w_2)$. Es cierto que al considerar las funciones $h_w(x) = \text{sign}(w^T x)$, dos vectores w, w' para los que existe un escalar positivo k tal que $w = kw'$ dan lugar a la misma función ($h_w = h_{w'}$). Sin embargo, en un ordenador el conjunto de posibles valores k es discreto, luego la cota que estamos hallando es usable.

Como cada parámetro es un escalar que ocupa 64 bits, podemos concluir que $|\mathcal{H}| \leq 2^{3 \cdot 64}$.

En el conjunto de entrenamiento $N = 1194$, luego puedo asegurar con probabilidad mayor o igual que 0.95 que

$$E_{out} \leq 0.22529313232830822 + 0.2393223422 = 0.4646154745$$

El caso de E_{test} es muy diferente. Por una parte sólo tenemos $N = 366$ datos, lo que aumenta la cota. Pero en cambio, la función elegida ya está fijada y no depende de los datos de test (cosa que no pasaba con los de entrenamiento), por tanto puedo tomar $|\mathcal{H}| = 1$ en la desigualdad, obteniendo

$$E_{out} \leq 0.2540983606557377 + 0.07098910343 = 0.3250874641$$

con probabilidad mayor o igual que 0.95. Esta independencia entre la función y los datos hace que la cota basada en E_{test} sea mucho más precisa que la basada en E_{in}.