Memoria práctica 2 de Aprendizaje Automático

UNIVERSIDAD DE GRANADA

Antonio Manuel Fresneda Rodríguez
antoniomfr@correo.ugr.es

Índice

Ejercicio sobre la complejidad de H y el ruido

Apartado 1

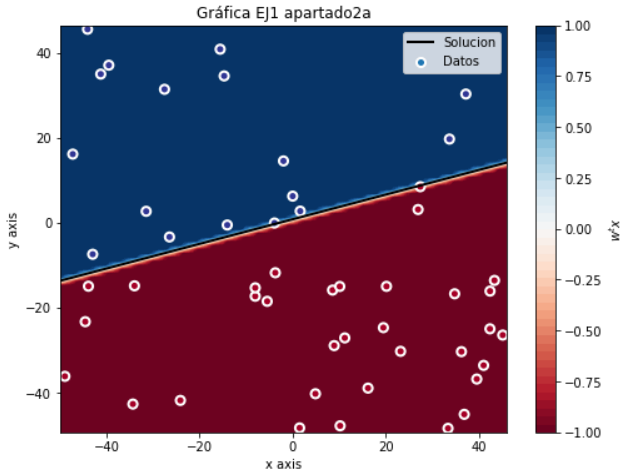
Este apartado del ejercicio esta implementado en la función Apartado1_1a(simulación de la distribución uniforme) y Apartado1_1b (simulación de la distribución Gaussiana)

Apartado 2

Este apartado esta implementado en la función Apartado1_2

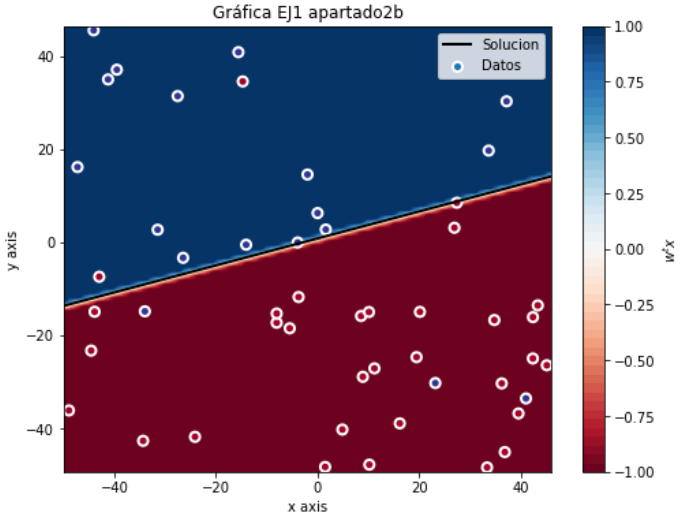
Apartado 2a

En esta imagen vemos como la recta que hemos generado clasifica bien todos los datos.



Apartado 2b

En esta imagen vemos como la recta sigue siendo la misma, pero esta vez al haber cambiado algunas etiquetas, vemos como hay algún punto mal clasificado.



Apartado 3

Este apartado esta implementado en la función Apartado1_3.

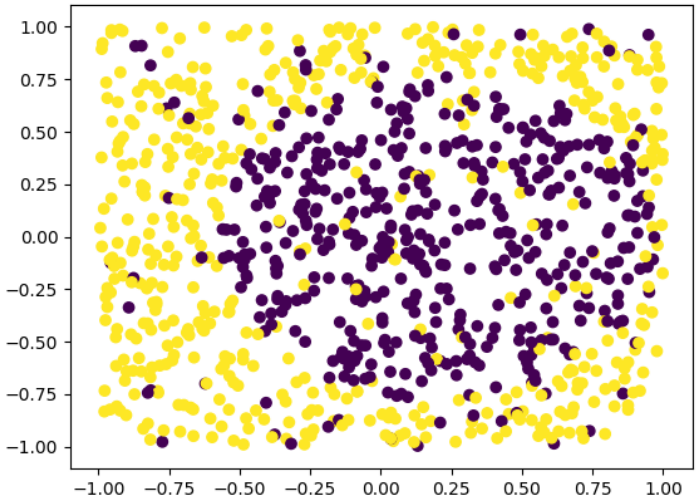
Las funciones están implementadas en:

1. $f(x, y) = (x - 10)^2 + (y - 20)^2 - 400 \rightarrow f1$
2. $f(x, y) = 0{,}5(x + 10)^2 + (y - 20)^2 - 400 \rightarrow f2$
3. $f(x, y) = 0{,}5(x - 10)^2 - (y + 20)^2 - 400 \rightarrow f3$
4. $f(x, y) = y - 20x^2 - 5x + 3 \rightarrow f4$

La conclusión a la que llegamos si vemos las gráficas obtenidas es que son peores clasificadoras para esta muestra que la recta anteriormente planteada.

Esto se debe a que las funciones planteadas son cónicas y la distribución de la muestra no sigue una distribución adecuada para el uso de este tipo de funciones a la hora de clasificar.

Estas funciones ganan a las lineales cuando la muestra sigue una distribución como la que vimos en la practica anterior. Adjunto una imagen de la distribución de dicha muestra.



Si con una muestra con este tipo de distribución utilizásemos alguna de estas funciones, muy probablemente obtendríamos mejores resultados que usando una función lineal.

Modelos lineales

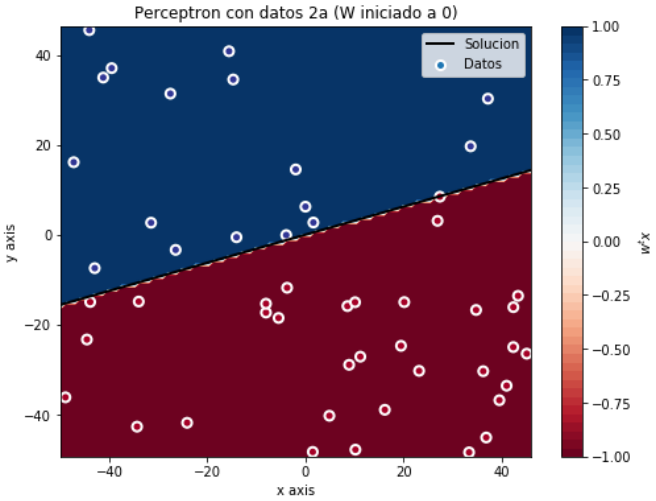
Apartado 1

El algoritmo del Perceptron está implementado en la función PLA. El numero de iteraciones fijado para la ejecución del PLA es de 1000 iteraciones.

La imagen adjunta pertenece a UNA de las diez veces con las que he ejecutado el PLA con vectores aleatorios. Con una es suficiente, puesto que en este caso, la muestra es linealmente separable y el PLA converge hacia un hiperplano que separa perfectamente la muestra.

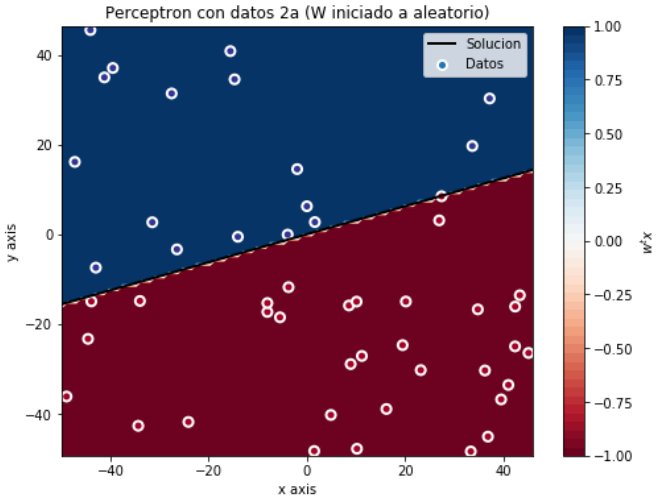
Apartado 1a

Vector iniciado a 0:



Numero de iteraciones=5

Vector iniciado a valores aleatorios:



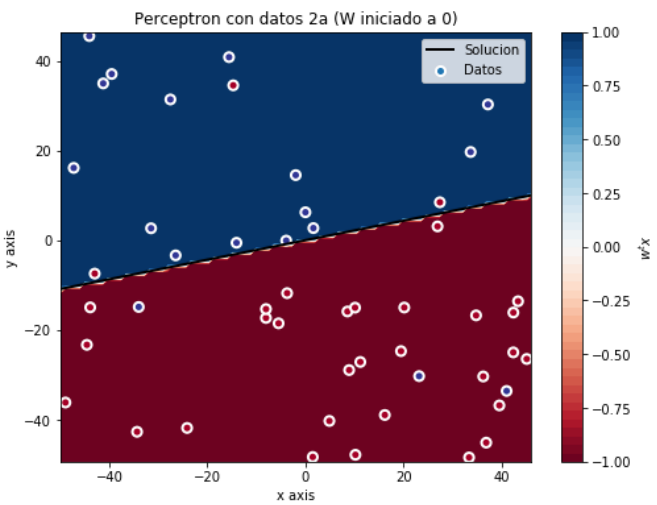
Media de iteraciones=6,9

Como vemos, el algoritmo PLA es capaz de obtener una solución que divide a la muestra perfectamente en 5 iteraciones inicializando el vector a 0. Si iniciamos el vector de forma aleatoria, puede que la solución inicial caiga muy lejos del objetivo por lo que el algoritmo necesitaría un numero más elevado de iteraciones para converger.

Apartado 2a

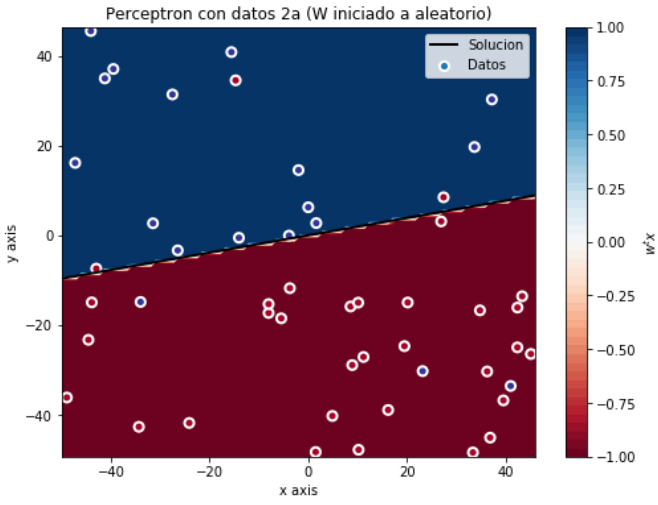
En este caso también es una gráfica proveniente de una de las 10 veces que ejecutamos el PLA con vectores aleatorios.

Vector iniciado a 0:



Numero de iteraciones=1000

Vector iniciado a valores aleatorios:



Media de iteraciones=1000

Como vemos en las imágenes, las soluciones con el vector inicial a cero y con valores aleatorios son parecidas.

En cuanto a la media de iteraciones vemos que todas las ejecuciones llegan al máximo de iteraciones. Esto se debe a que al haber introducido ruido en las etiquetas, hemos hecho que la muestra no sea linealmente separable, por lo que si ejecutamos el PLA con estos datos, el algoritmo va a empezar a ciclar indefinidamente ya que como la muestra no es linealmente separable, no existe un hiperplano que separe completamente toda la muestra según las etiquetas y vuelve a repetir el ciclo indefinidamente (en nuestro caso para pero porque le ponemos un limite de iteraciones).

Por ello, cabe destacar que la solución que vemos en la gráfica puede que no sea la misma si cogemos otra solución de las 10 que obtenemos iniciando el vector con valores aleatorios.

Apartado 2

La regresión logística está implementada en la función RL. El gradiente descendiente estocástico en la función SGD.

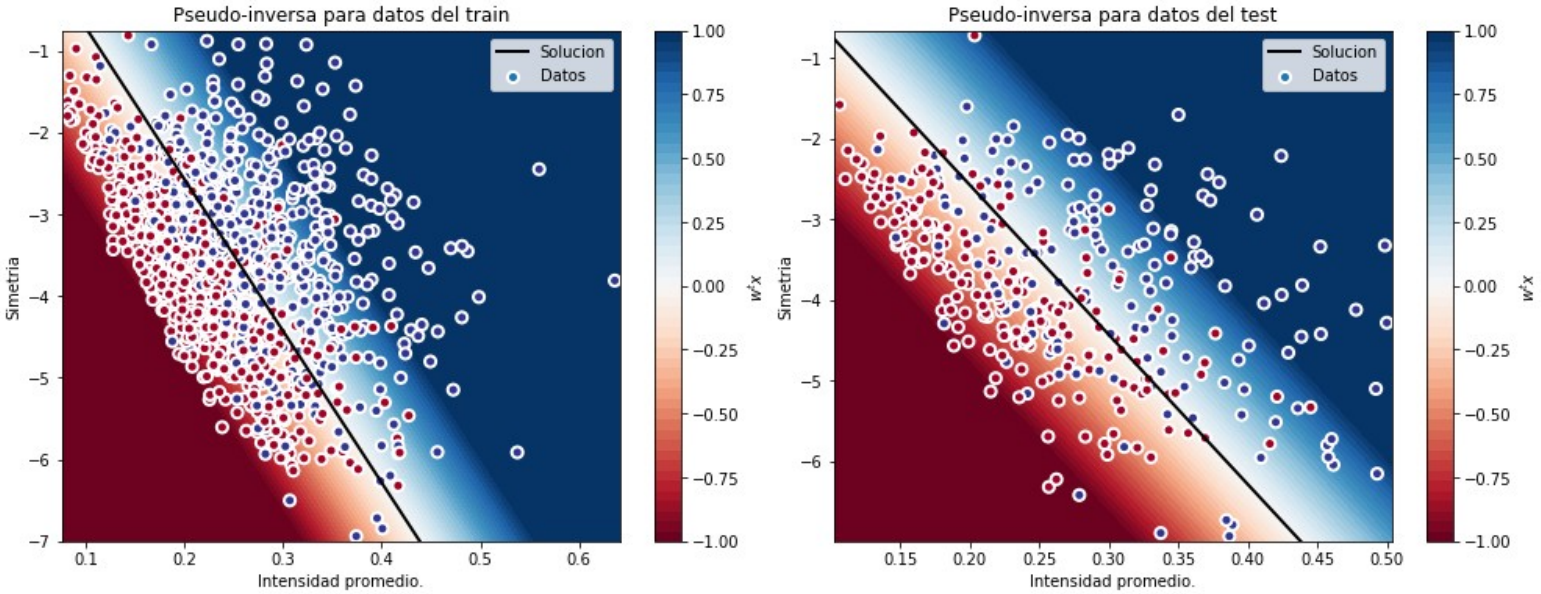
Para generar la muestra he usado la función genera_datos. El tamaño de la muestra de test es de 3000 muestras.

El error dentro de la muestra (Ein) ha sido: 0,01 (1%)

El error fuera de la muestra (Eout) ha sido: 0,17 (17%)

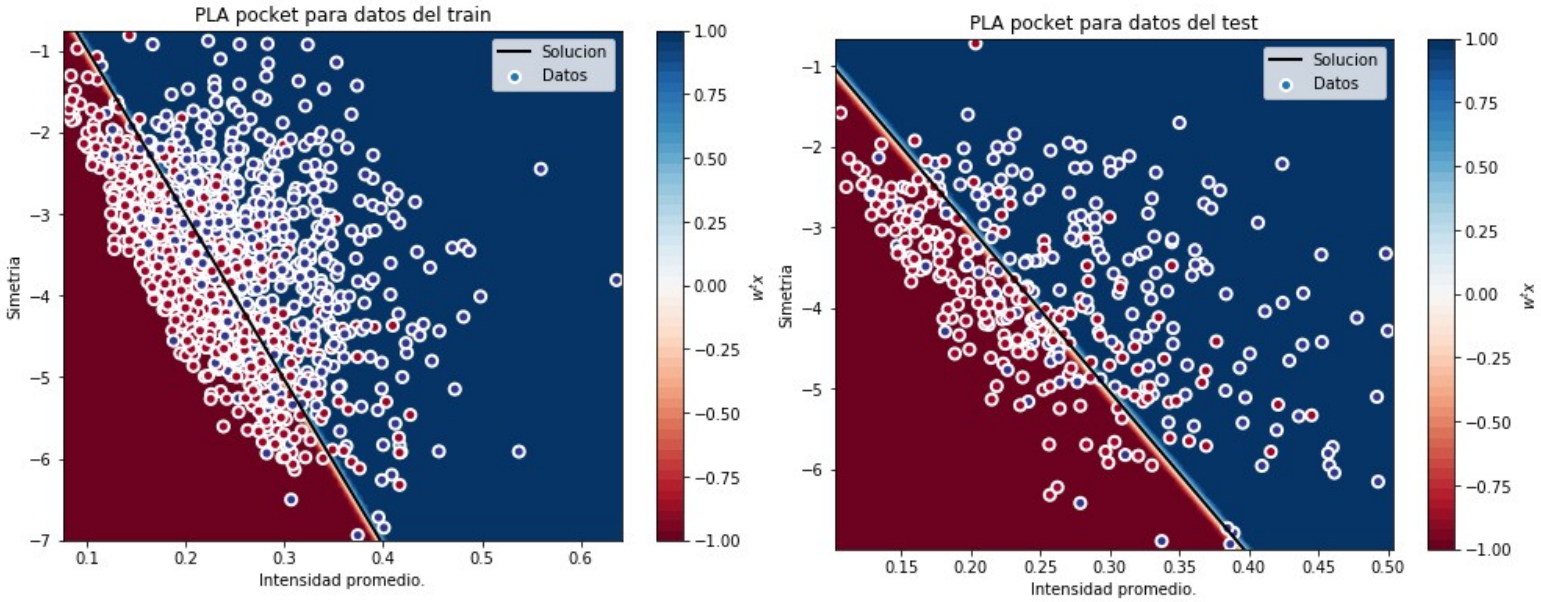
Bonus

Aqui vemos ambas graficas obtenidas con la Pseudo-inversa, tanto en el train como en el test.



El error de clasificación para train es de: 0,23 (23%)

El error de clasificación en el test es de: 0,25 (25%)



Aquí vemos ambas gráficas obtenidas con el PLA-pocket, tanto en el train como en el test.

El error de clasificación para train es de: 0,21 (21%)

El error de clasificación en el test es de: 0,24 (24%)

Considerando la dimensión de Vapnik-Chervonenkis $d_{vc} = 3$ (La dimensión de nuestro problema es 2)

$$E_{out}(h) \leq E_{in}(h) + \sqrt{\left(\frac{d_{vc}\log N - \log \delta}{N}\right)} \rightarrow E_{in}(h) + \sqrt{\left(\frac{3\log 1194 - \log 0{,}05}{1194}\right)} = E_{in}(h) + 0{,}1425$$

La cota del E_{out} para el algoritmo de la pseudo-inversa es: 0.3725

La cota del E_{out} para el algoritmo del PLA-pocket es: 0.325