Aprendizaje Automático
Trabajo 2: Programación

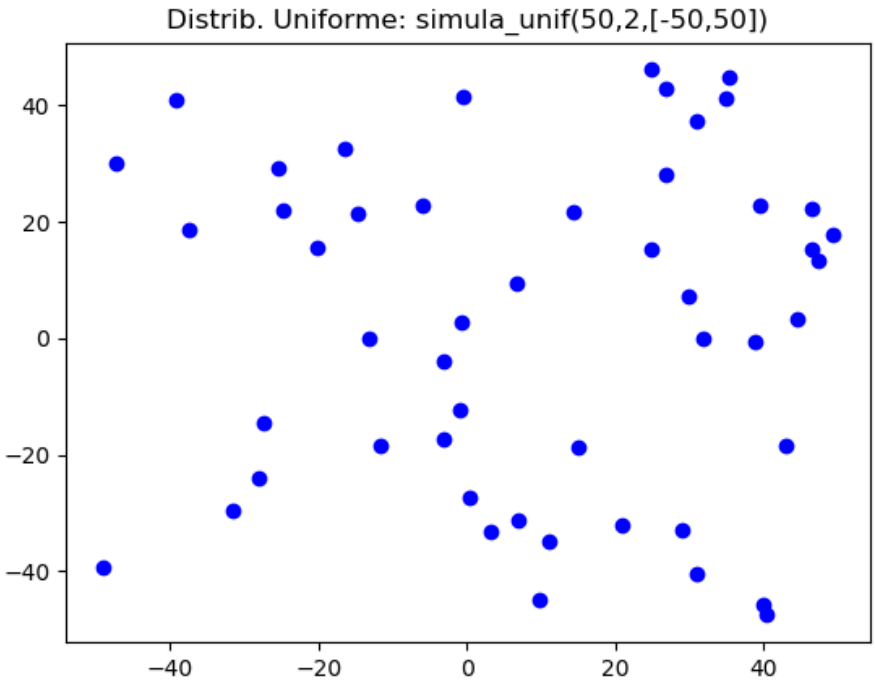
Alfonso García Martínez
alfonsogmw@correo.ugr.es

Abril de 2019

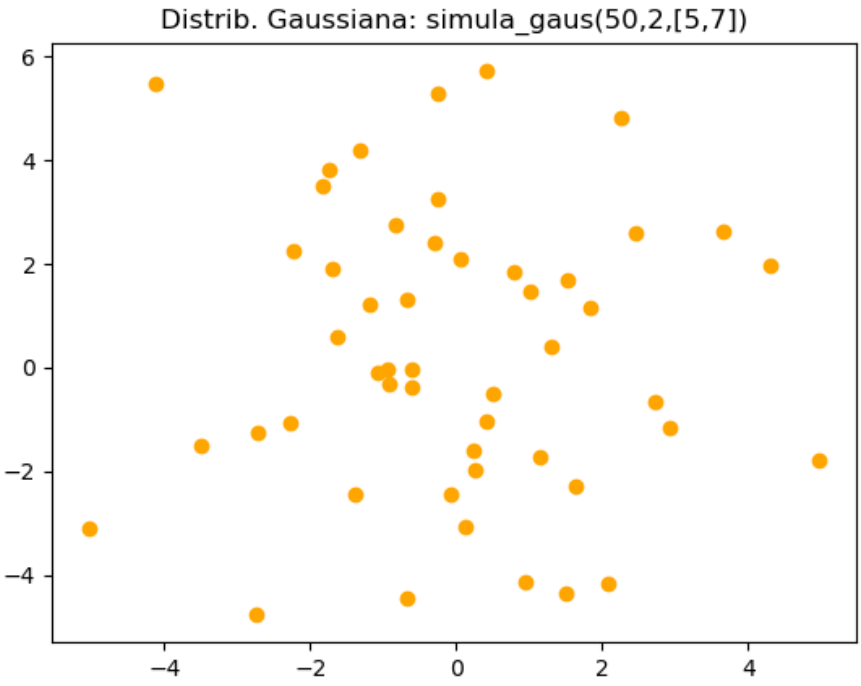
1. Ejercicio sobre la complejidad de H y el ruido

1. Dibujar una gráfica con la nube de puntos de salida correspondiente.

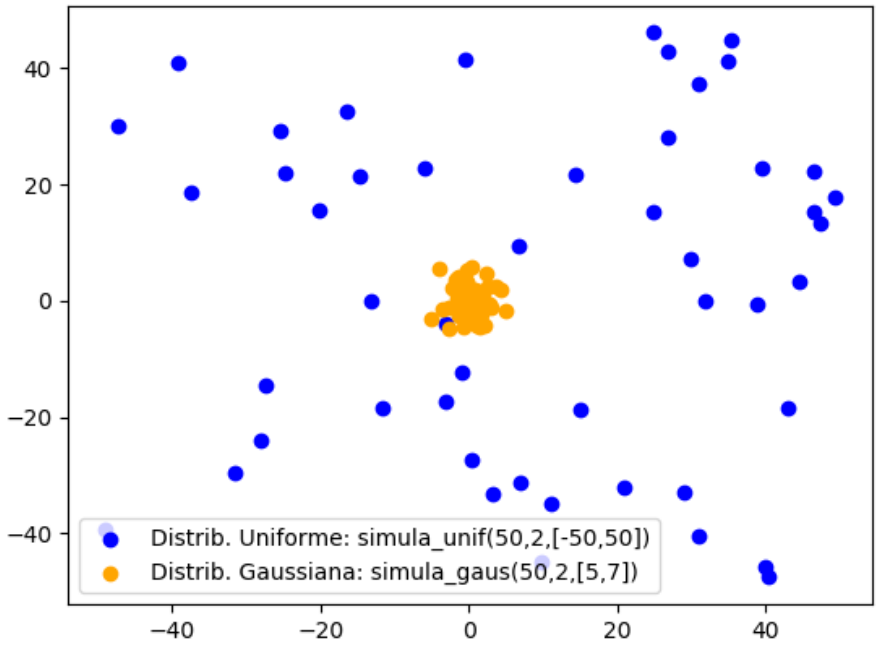
   a) Considere $N = 50$, $dim = 2$, $rango = [-50, +50]$ con $simula\_unif(N, dim, rango)$.



b) Considere $N = 50$, $dim = 2$ y $sigma = [5, 7]$ con $simula\_gaus(N, dim, sigma)$.



Ambas combinadas:



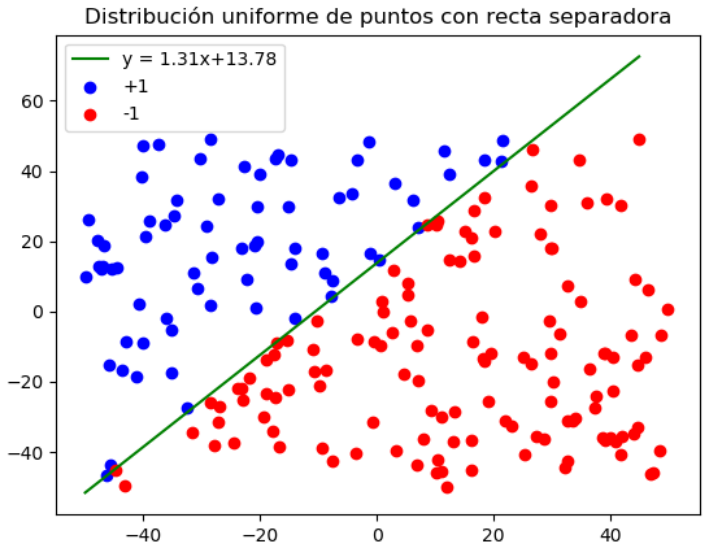
2. Con ayuda de la función $simula\_unif()$ generar una muestra de puntos 2D a los que vamos añadir una etiqueta usando el signo de la función $f(x, y) = y - ax - b$, es decir el signo de la distancia de cada punto a la recta simulada con $simula\_recta()$.

   a) Dibujar una gráfica donde los puntos muestren el resultado de su etiqueta, junto con la recta usada para ello. (Observe que todos los puntos están bien clasificados respecto de la recta).

   Respuesta

   Los coeficientes de la recta aleatoria obtenida $y = ax + b$ son $a = 1{,}307$ y $b = 13{,}776$.

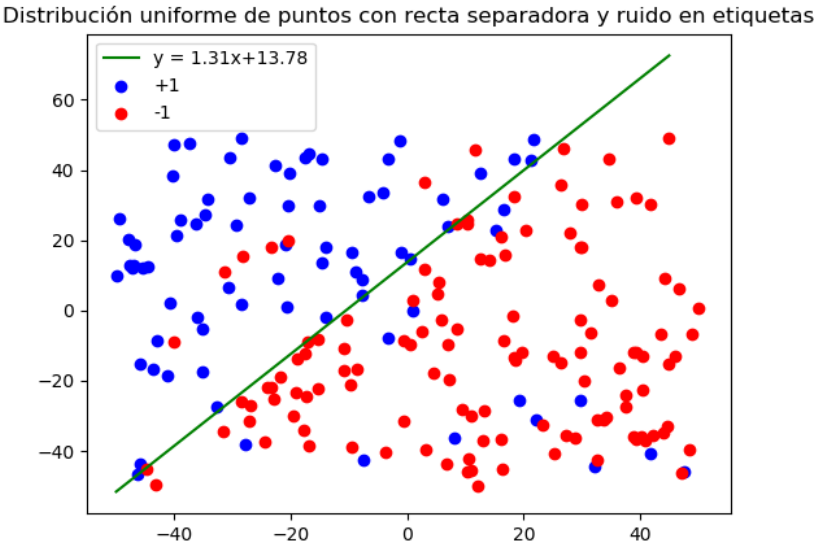
   Efectivamente, si primero generamos los coeficientes de una recta aleatoriamente, y luego generamos las etiquetas de los datos a partir de ella, **la recta debe separar perfectamente todos los puntos**.



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. (Ahora hay puntos mal clasificados respecto de la recta).

Respuesta

Al meter ruido a posteriori tras etiquetar, obviamente tendremos ejemplos mal clasificados (un 10 % tanto para ejemplos positivos como negativos).



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

   - $f(x, y) = (x - 10)^2 + (y - 20)^2 - 400$
   - $f(x, y) = 0,5(x + 10)^2 + (y - 20)^2 - 400$
   - $f(x, y) = 0,5(x - 10)^2 - (y + 20)^2 - 400$
   - $f(x, y) = y - 20x^2 - 5x + 3$

   Visualizar el etiquetado generado en 2*b* junto con cada una de las gráficas de cada una de las funciones. Comparar las formas de las regiones positivas y negativas de estas nuevas funciones con las obtenidas en el

caso de la recta ¿Son estas funciones más complejas mejores clasificadores que la función lineal? ¿En qué ganan a la función lineal? Explicar el razonamiento

Respuesta

Todas estas funciones, al ser de 2 variables, describen superficies curvadas definidas en el espacio. En lugar de pintar las gráficas 3D tal cual, procederemos de forma similar a como se hizo en la práctica 1: mostraremos las curvas que resultan de la intersección con el plano $z = 0$, es decir, las curvas en 2D para las que se cumple que $f(x, y) = 0$.

- Para $f(x, y) = (x - 10)^2 + (y - 20)^2 - 400$

  Cuando $f(x, y) = 0$, tenemos que:

  $$(x - 10)^2 + (y - 20)^2 - 400 = 0$$

  Y despejando y obtenemos:

  $$y = 20 \pm \sqrt{300 + 20x - x^2}$$

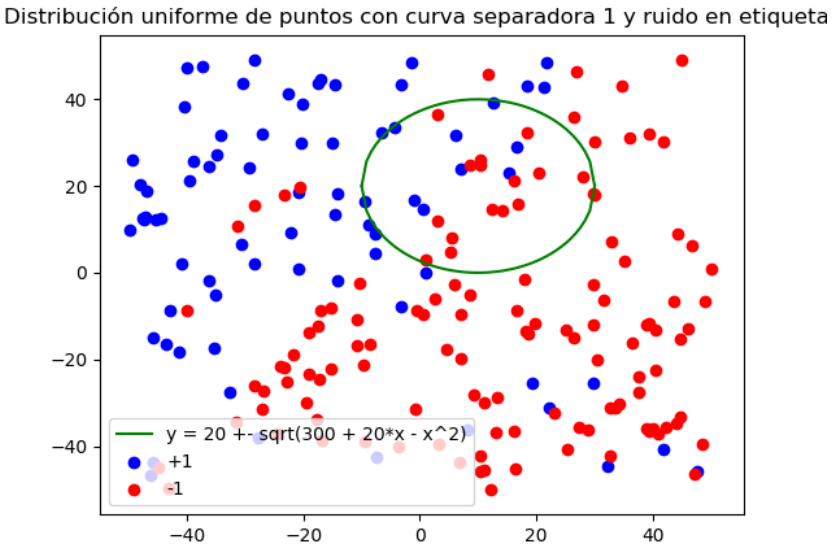
  Recordemos antes de nada que las raíces cuadradas de números negativos no están definidas en $\mathbb{R}$, luego queremos encontrar el intervalo de valores para los cuales el polinomio de segundo grado que hay dentro de la raíz cuadrada $(300+20x-x^2)$ sea positivo.

  Si nos fijamos, podemos ver que el coeficiente principal es negativo, luego la parábola que describe es **cóncava**. Por tanto, si las raíces del polinomio son $x = x_0$ y $x = x_1$, entonces el intervalo de valores para los cuales el polinomio es positivo es $[x_0, x_1]$. Como en este caso las raíces son $x = -10$ y $x = 30$, debemos pintar la curva en el intervalo $[-10, 30]$.

  Una última cosa que debemos tener en cuenta antes de pintar la curva es que la expresión $y = 20 \pm \sqrt{300 + 20x - x^2}$, tal cual, no define una función porque a un valor de y le correspondería más de un valor de x (2 en este caso). Por tanto, hay que separar la expresión en dos diferentes para que se pueda definir una función para cada una de ellas

$$y_1 = 20 + \sqrt{300 + 20x - x^2}$$
$$y_2 = 20 - \sqrt{300 + 20x - x^2}$$

y de esta forma pueda pintarse la curva completa:



- Para $f(x, y) = 0,5(x + 10)^2 + (y - 20)^2 - 400$

  Siguiendo el mismo procedimiento que para la funcion anterior, igualando $f(x, y)$ a 0:

  $$0,5(x + 10)^2 + (y - 20)^2 - 400 = 0$$

  Despejamos y:

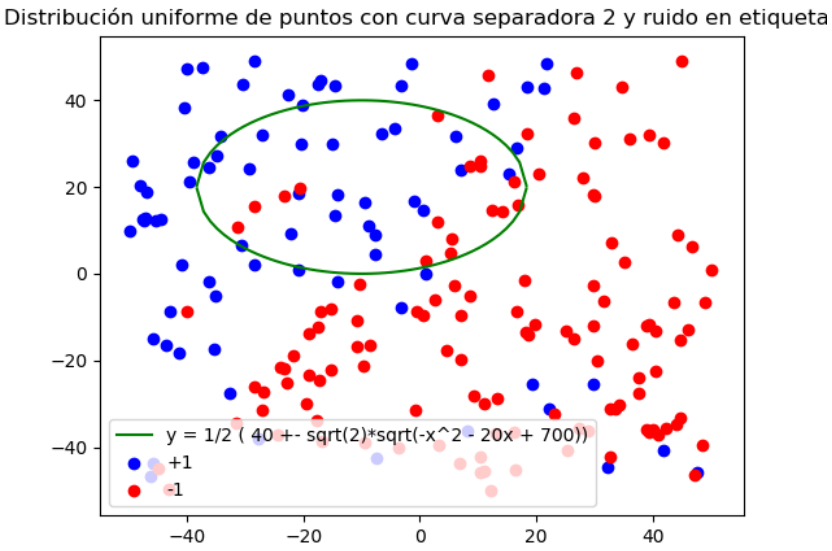
  $$y = 20 \pm \sqrt{-\frac{1}{2}x^2 - 10x + 350} = 20 \pm \sqrt{\frac{1}{2}}\sqrt{-x^2 - 20x + 700} =$$

  $$20 \pm \frac{1}{2}\sqrt{2}\sqrt{-x^2 - 20x + 700} = \frac{1}{2}\left(40 \pm \sqrt{2}\sqrt{-x^2 - 20x + 700}\right)$$

  De nuevo tenemos un polinomio de segundo grado cóncavo dentro de una raíz cuadrada. Sus raíces son $x = -10(1 + 2\sqrt{2}) \approx -38,28$ y $x = 20\sqrt{2} - 10 \approx 18,28$, con lo que debemos pintar la curva en el intervalo $[-38{,}28, 18{,}28]$

  Y de nuevo, hay que volver a *separar* la ecuación:

  $$y_1 = \frac{1}{2}\left(40 + \sqrt{2}\sqrt{-x^2 - 20x + 700}\right)$$
  $$y_2 = \frac{1}{2}\left(40 - \sqrt{2}\sqrt{-x^2 - 20x + 700}\right)$$

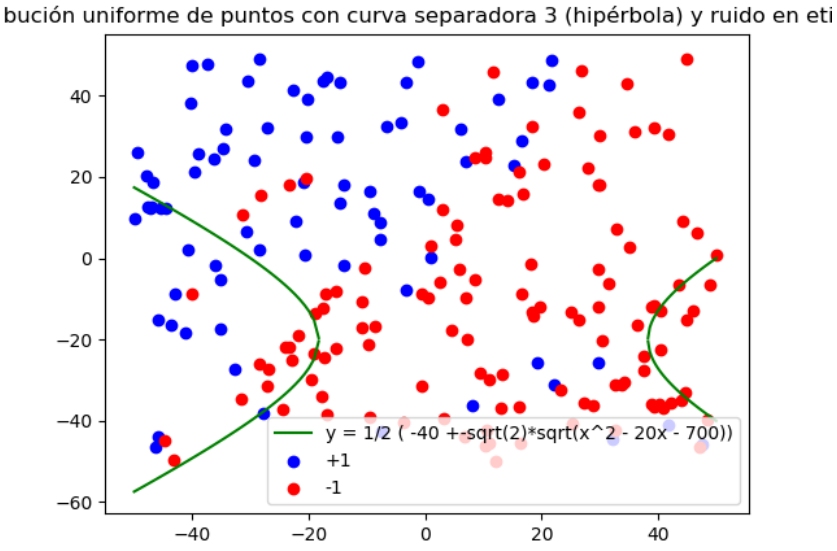

Distribución uniforme de puntos con curva separadora 2 y ruido en etiqueta
y = 1/2 (40 +- sqrt(2)*sqrt(-x^2 - 20x + 700))
+1
-1
40
20
0
−20
−40
−40
−20
0
20
40

- Para $f(x,y) = 0,5(x-10)^2 - (y+20)^2 - 400$
  Repetimos el mismo procedimiento de los dos casos anteriores:

$$0,5(x-10)^2 - (y+20)^2 - 400 = 0$$
$$y = \frac{1}{2}\left(-40 \pm \sqrt{2}\sqrt{x^2 - 20x - 700}\right)$$

  Como esta vez el coeficiente principal del polinomio de la raíz cuadrada es positivo, entonces el polinomio es **convexo**. Como las raíces son $x = 10 - 20\sqrt{2} \approx -18,29$ y $x = 10 + 20\sqrt{2} \approx 38,29$, el intervalo que hay que pintar es $(-\infty, -18,29] \cup [38,29, +\infty)$. Obviamente, no vamos a pintar *hasta el infinito*, sino que acotaremos el intervalo al mismo intervalo en el que hemos definido los puntos, $[-50, 50]$, con lo que nos quedaría el intervalo:

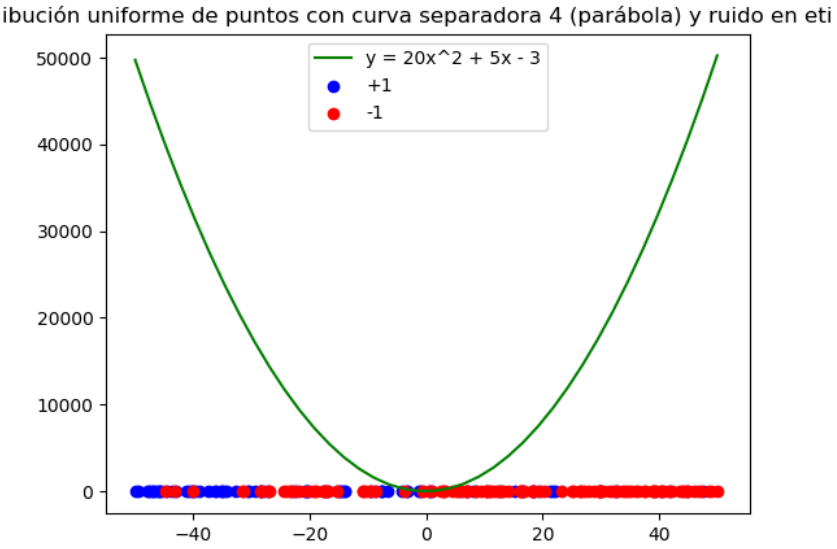
$$((-\infty, -18,29] \cup [38,29, +\infty)) \cap [-50, 50] = [-50, -18,29] \cup [38,29, +50]$$



- Para $f(x, y) = y - 20x^2 - 5x + 3$

$$y - 20x^2 - 5x + 3 = 0$$
$$y = 20x^2 + 5x - 3$$

Aquí no tenemos restricciones sobre el dominio en que debemos mostrar la función, luego en principio utilizaremos el mismo rango en el que se han pintado los puntos: $[-50, 50]$.



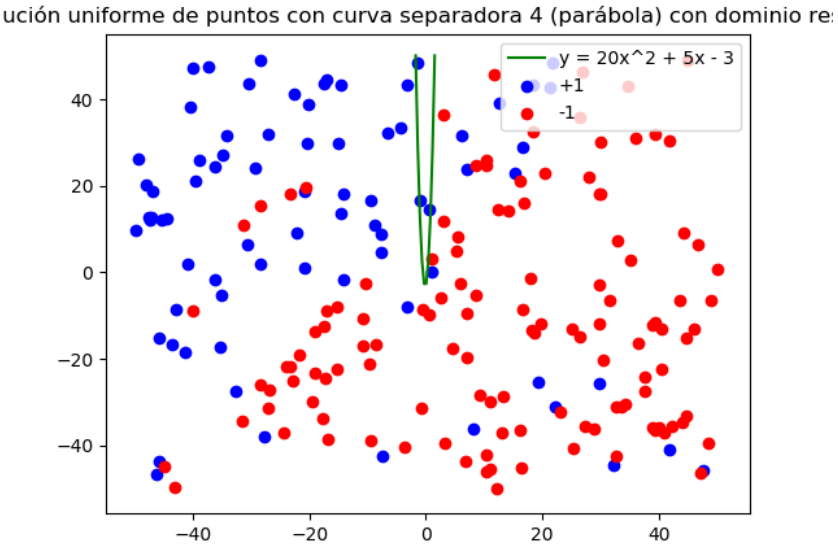
Como vemos, el crecimiento de la parábola es demasiado rápido y sobrepasa fácilmente los puntos. Limitaremos el dominio a los valores de x en los que se alcanza un valor $y = 50$:

$$20x^2 + 5x - 3 = 50$$
$$20x^2 + 5x - 53 = 0$$

Cuyas raíces son

$$x = \frac{-1 - \sqrt{\frac{853}{5}}}{8} \approx -1{,}76$$
$$x = \frac{-1 + \sqrt{\frac{853}{5}}}{8} \approx 1{,}51$$

Luego hay que pintar la cuarta curva en el intervalo $[-1{,}76, 1{,}51]$



Las funciones que hemos usado para intentar *separar* los datos corresponden a una circunferencia, una elipse, una hipérbola y una parábola, en las que aparecen variables elevadas al cuadrado. Efectivamente, estas funciones son más complejas, y ajustando correctamente los coeficientes podrían servir como fronteras de decisión para datos que no sean linealmente separables (o para incluso datos linealmente separables en el caso de la parábola, si fijamos a 0 el coeficiente principal).

Sin embargo, para este conjunto de datos concreto, ninguna de las curvas dibujadas hacen una buena separación. Es cierto que este conjunto no es linealmente separable, pero no lo es porque hemos introducido una pequeña cantidad de ruido aleatorio deliberadamente a un conjunto de datos que originalente sí que lo era, no porque siga un comportamiento cuadrático o siguiendo una curva cónica.

Por otra parte, aunque los datos sí fuesen adecuados para separarlos con alguna de estas funciones, habría que **ajustar los coeficientes** con un algoritmo de aprendizaje.

2. Modelos lineales

1. **Algoritmo Perceptrón:** 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, *max_iter* es el número máximo de iteraciones permitidas y *vini* el valor inicial del vector. La función devuelve los coeficientes del hiperplano.

   a) 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.

   Respuesta:

   Ya conocemos la sencillez del algoritmo PLA: recorremos diversas veces el conjunto de datos, y cada vez que encontremos un dato $\mathbf{x}$ mal clasificado actualizamos el vector de pesos $\mathbf{w}$ de nuestro modelo lineal según la siguiente regla:

   $$\mathbf{w}(t+1) = \mathbf{w}(t) + y\mathbf{x}(t)$$

   Donde t es la iteración actual y y es la verdadera etiqueta de $\mathbf{x}$.

   Nuestra implementación del PLA aceptará los parámetros que se piden en el enunciado, y devolverá como salida no solo el vector de pesos $\mathbf{w}$ de nuestro modelo lineal, sino el número de iteraciones. Para nuestro caso, consideraremos que una iteración corresponde a recorrer el conjunto de datos una vez, ajustando los pesos cada vez que encontremos un dato mal clasificado.

   Cuando queramos visualizar resultados en este problema debemos recordar, como ya ocurrió en la práctica anterior, que nuestro clasificador es un **plano**, y por tanto para visualizarlo satisfactoriamente en 2 dimensiones, hay que hayar su recta intersección con el plano $z = 0$. Dicho de otra forma: si

$$h(x_1, x_2) = w_0 + w_1 x_1 + w_2 x_1$$

es el plano clasificador, debemos calcular la intersección

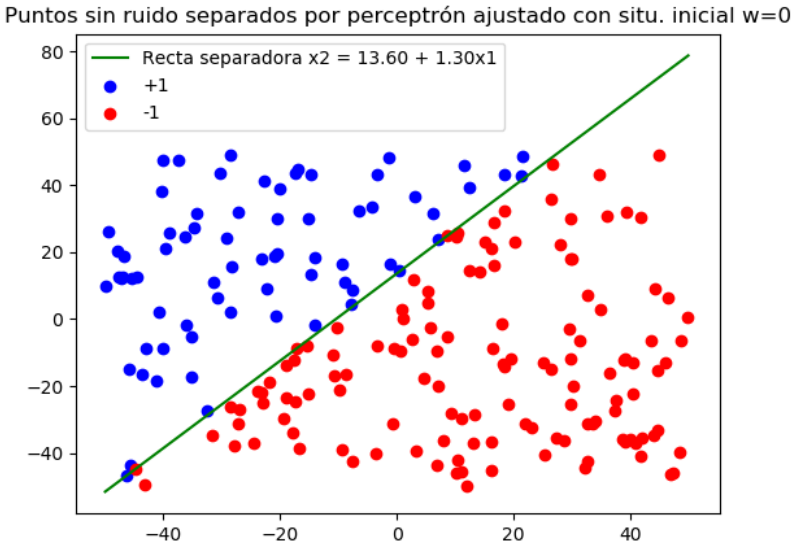
$$h(x_1, x_2) = 0$$
$$w_0 + w_1 x_1 + w_2 x_2 = 0$$

Con lo que podemos conocer x_2 en función de x_1:

$$x_2 = -\frac{w_0}{w_2} - \frac{w_1}{w_2} x_1$$

Como en este caso vamos a trabajar con el conjunto de datos generado en el apartado $2a$, el resultado debería parecerse mucho a la recta dibujada en ese mismo apartado.

Para un vector de pesos inicial de 0 (o sea, $vini = 0$), PLA es capaz de clasificar nuestro conjunto de datos del apartado $2a$ tras un total de **199 iteraciones**.



Ahora volveremos a aplicar el PLA sobre el mismo conjunto de datos otras 10 veces más, con pesos iniciales aleatorios con valores entre 0 y 1. Como estos datos son linealmente separables, siempre se obtendrá una clasificación perfecta siempre y cuando

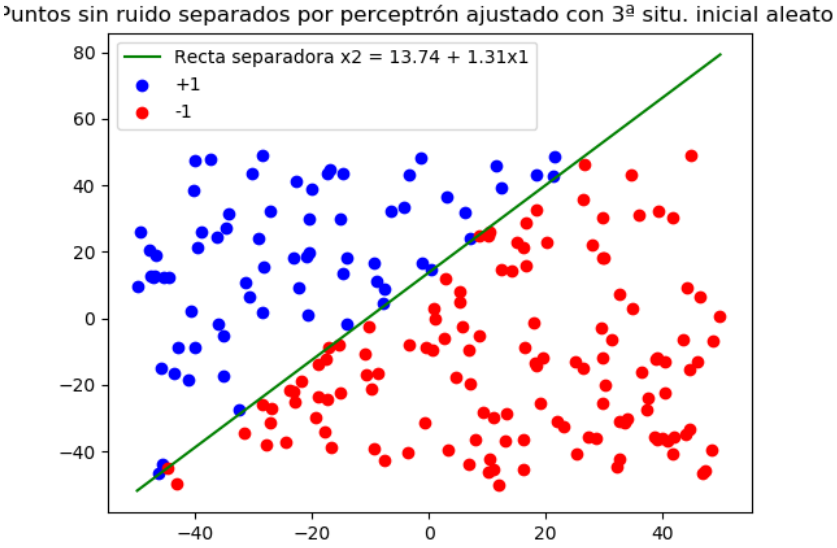
se establezca un número máximo de iteraciones lo suficientemente grande.

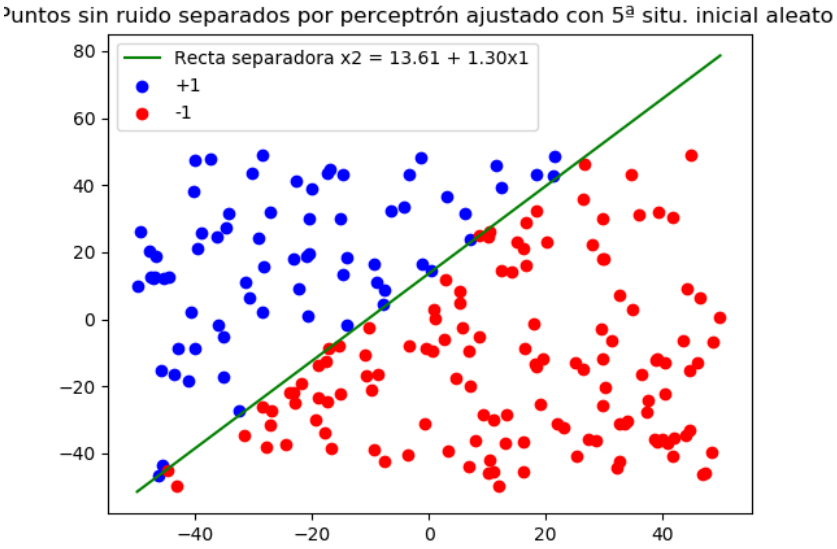
Resultados PLA con datos de $2a$		
Valor inicial	Ajuste obtenido	Iteraciones
(0, 0, 0)	(-1200.0, -115.0, 88.24)	199
(0.54,0.2 ,0.78)	(-1530.46, -146.17,111.89)	357
(0.81,0.71,0.94)	(-1190.19, -113.53, 86.65)	196
(0.85,0.96,0.46)	(-1194.15, -113.9 , 86.9)	186
(0.48,0.59,0.94)	(-1437.52, -137.29,105.27)	306
(0.37,0.6 ,0.61)	(-1458.63, -139.5 ,107.2)	323
(0.56,0.12,0.4)	(-1109.44, -104.82, 80.31)	159
(0.22,0.79,0.76)	(-1363.78, -130.15, 99.81)	280
(0.17,0.72,0.92)	(-1406.83, -134.48,103.2)	300
(0.48,0.91,0.44)	(-1442.52, -137.79,105.87)	318
(0.07,0.36,0.19)	(-1167.93, -110.91, 84.91)	189

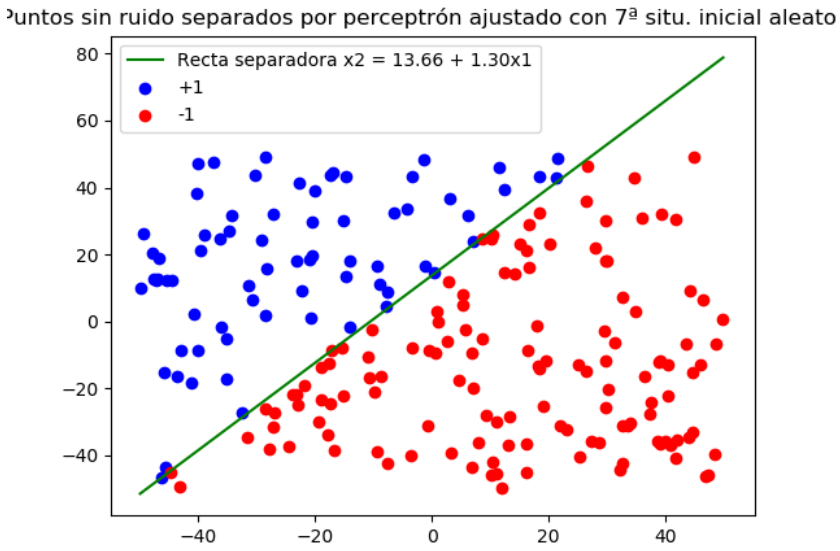
Promedio de iteraciones del PLA con datos de $2a$	
11 ejecuciones	255
10 ejecuciones (sin contar $vini = 0$)	261

Al variar el punto de inicio, **los ajustes son todos distintos en cada caso**, pero todos hacen una separación perfecta de los datos. También se observan variaciones considerables en el número de iteraciones a pesar de haber variado los valores iniciales en términos de décimas. Esto nos da una idea de lo que influye el valor inicial al ajustar un modelo lineal con PLA.

No vamos a pintar los ajustes obtenidos de todas y cada una de las 10 ejecuciones, pero sí podemos pintar 3 de ellas para apreciar que, aunque los ajustes sean distintos, todos consiguen una clasificación perfecta. Cogeremos, por ejemplo, los ajustes de la tercera, la quinta y la séptima ejecución, es decir, aquellas con valores iniciales (0.85,0.96,0.46), (0.37,0.6 ,0.61) y (0.22,0.79,0.76) respectivamente.


Puntos sin ruido separados por perceptrón ajustado con 3ª situ. inicial aleato
Recta separadora x2 = 13.74 + 1.31x1
+1
-1
80
60
40
20
0
−20
−40
−40
−20
0
20
40


Puntos sin ruido separados por perceptrón ajustado con 5ª situ. inicial aleato
Recta separadora x2 = 13.61 + 1.30x1
+1
-1
80
60
40
20
0
−20
−40
−40
−20
0
20
40

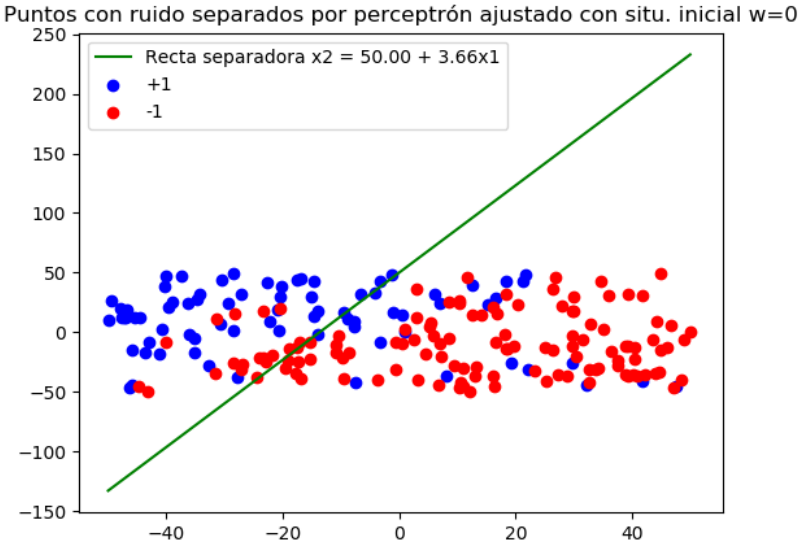

Puntos sin ruido separados por perceptrón ajustado con 7ª situ. inicial aleato
Recta separadora x2 = 13.66 + 1.30x1
+1
-1
80
60
40
20
0
−20
−40
−40
−20
0
20
40

b) 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 cuál y las razones para que ello ocurra.

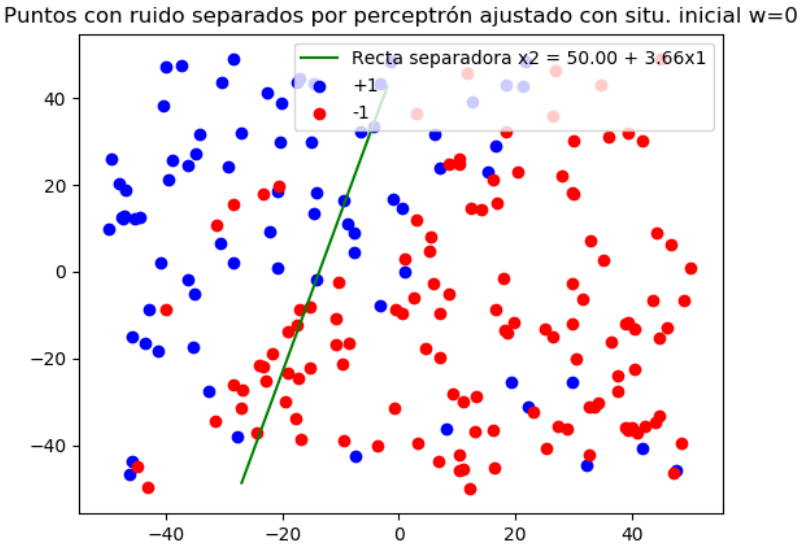
Respuesta:

Ahora hemos metido ruido en los datos, por lo que **ya no son linealmente separables** y el PLA nunca conseguirá una clasificación perfecta.

Por ejemplo, para el caso en que $vini = 0$, el resultado con los datos de 2*b* es muy distinto al obtenido con los de 2*a*. Para empezar, el número de iteraciones que emplea es el mismo que el que hemos fijado con el parámetro max_iter, que es 1000 en nuestro caso. Además, el ajuste obtenido es totalmente distinto:



Restringiendo el intervalo en el que pintamos la resta para que se vea mejor:

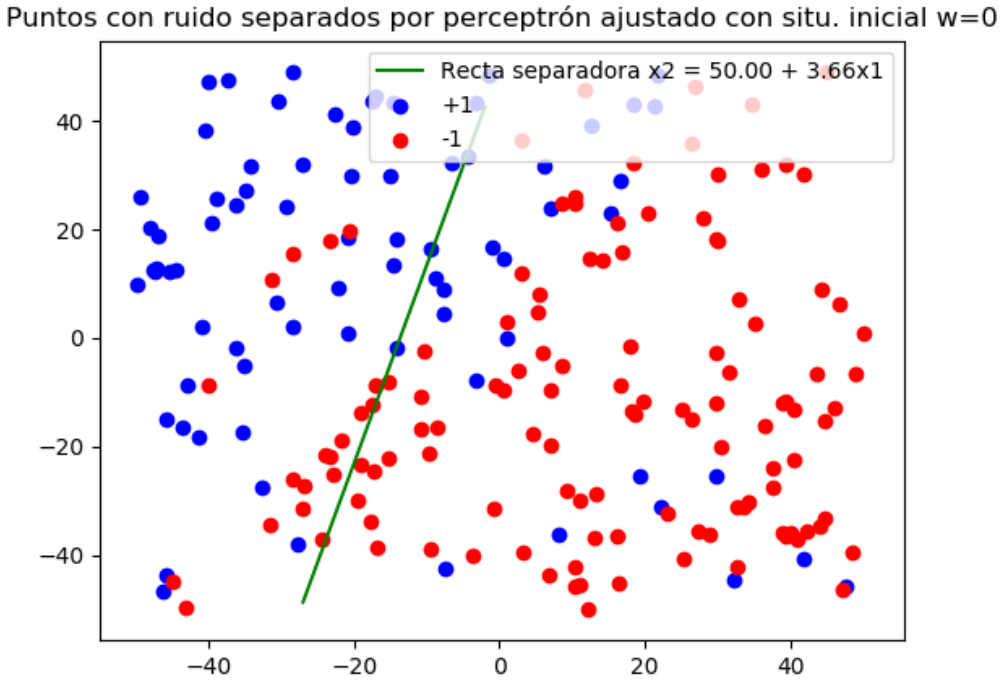


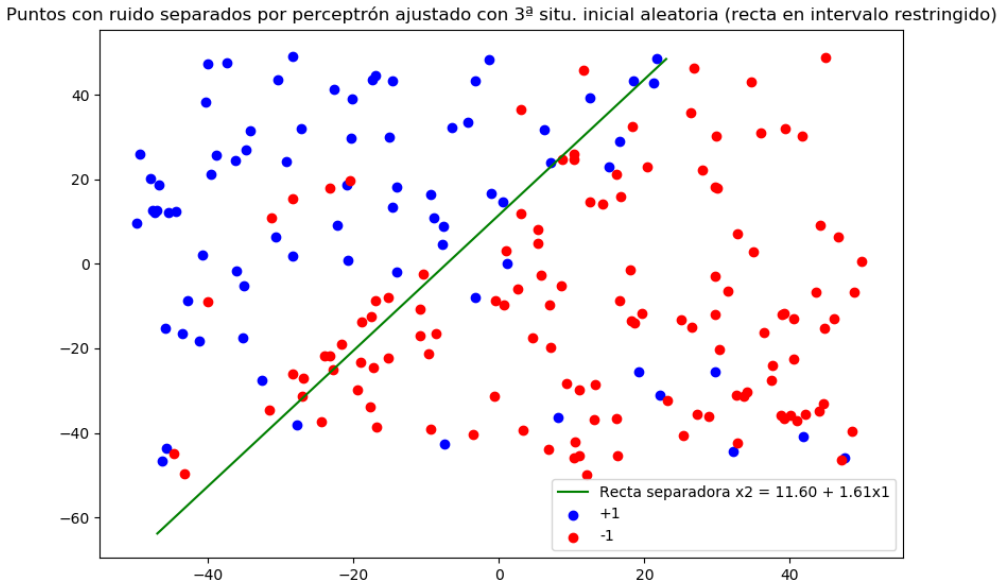
Repitamos ahora las 10 ejecuciones del PLA con exactamente los mismos 10 valores iniciales aleatorios que usamos para los datos de $2a$:

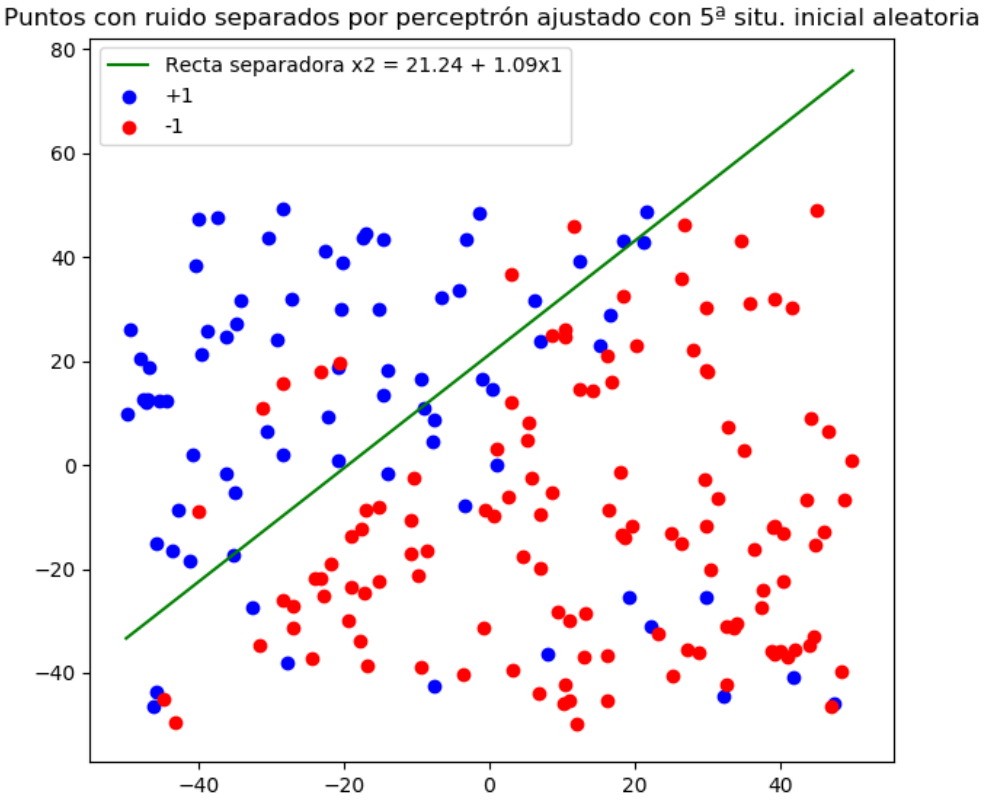
Resultados PLA con datos de $2b$		
Valor inicial	Ajuste obtenido	Iteraciones
(0, 0, 0)	(-321. , -23.48, 6.42)	1000
(0.54,0.2 ,0.78)	(-322.46, -44.72, 23.44)	1000
(0.81,0.71,0.94)	(-318.19, -31.25, 9.49)	1000
(0.85,0.96,0.46)	(-315.15, -43.6 , 27.16)	1000
(0.48,0.59,0.94)	(-313.52, -38.39, 23.56)	1000
(0.37,0.6 ,0.61)	(-314.63, -16.18, 14.81)	1000
(0.56,0.12,0.4)	(-315.44, -41.55, 29.22)	1000
(0.22,0.79,0.76)	(-320.78, -36.86, 10.71)	1000
(0.17,0.72,0.92)	(-314.83, -38.21, 23.43)	1000
(0.48,0.91,0.44)	(-310.52, -39.34, 13.12)	1000
(0.07,0.36,0.19)	(-319.93, -31.95, 7.77)	1000

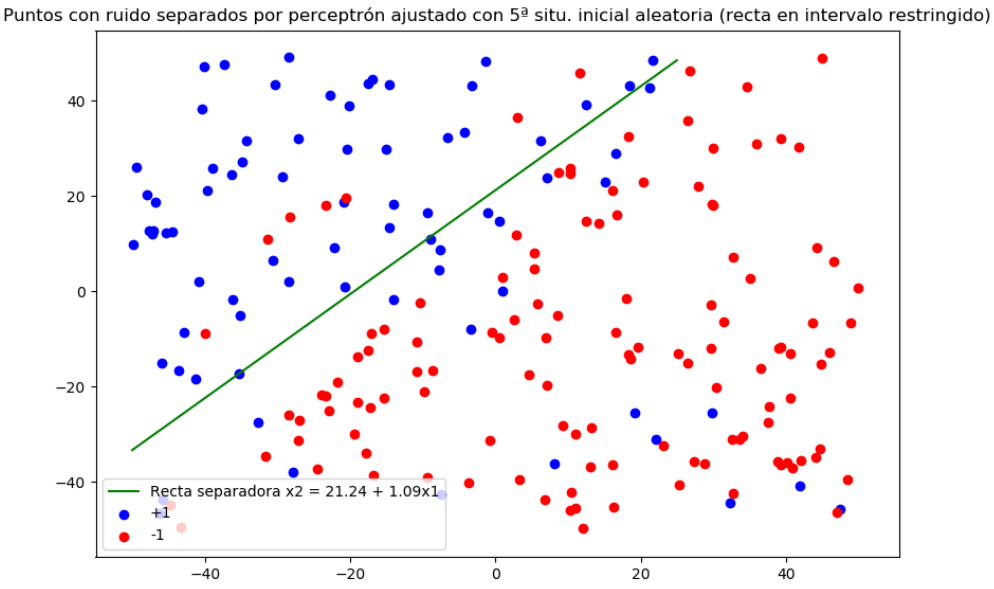
Lo más destacable de estos resultados es que **siempre se llega al máximo número de iteraciones permitido**, que previamente hemos fijado en 1000. Esto se debe a que, como ya sabemos, los datos ya no son linealmente separables y por ello PLA ya no es capaz de finalizar su ejecución encontrando una clasificación perfecta.

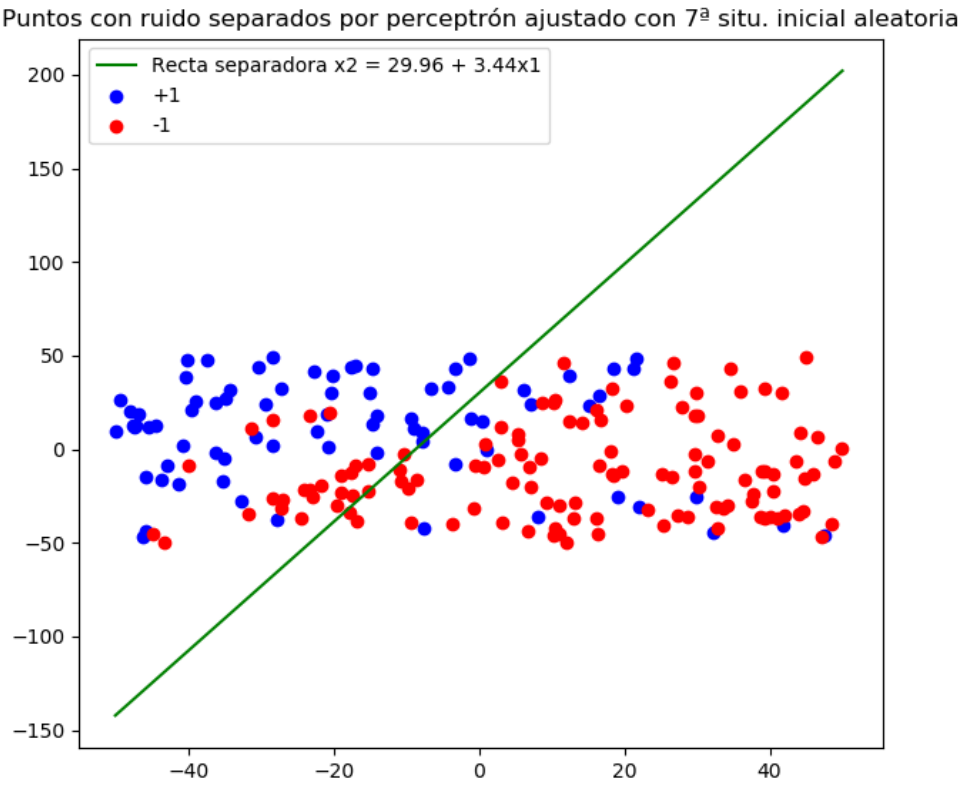
Por otra parte, los ajustes obtenidos son hasta cierto punto parecidos, probablemente porque los valores iniciales no son muy distintos entre sí (recordemos que sus valores se sitúan en el intervalo [0,1]). De la misma forma que se hizo cuando no teníamos ruido, veamos ahora los ajustes obtenidos a partir de los valores iniciales del tercer, quinto y séptimo caso, teniendo en cuenta que hay que reajustar en cada gráfica el rango en el que pintamos la recta para visualizar el ajuste satisfactoriamente:

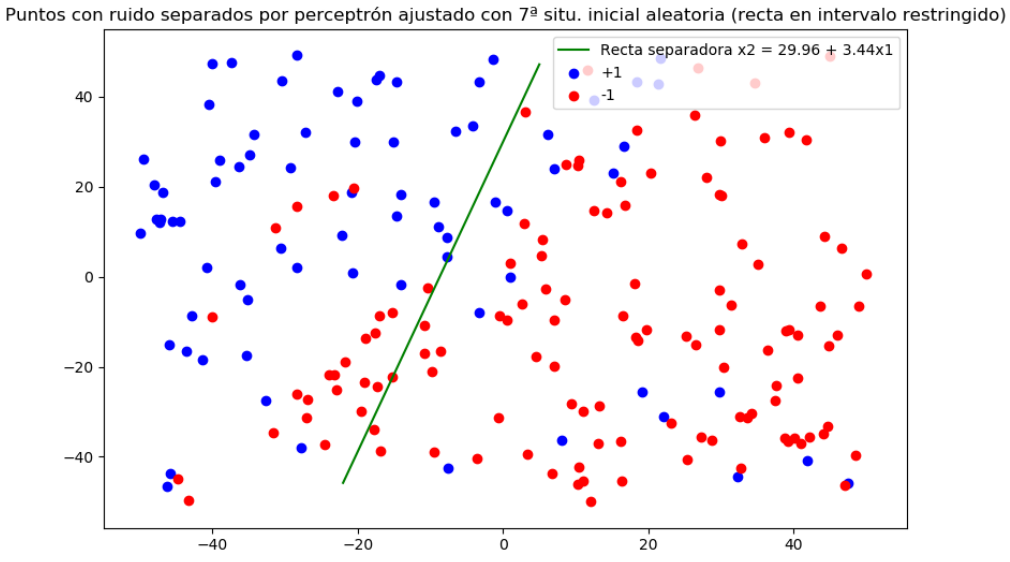

Puntos con ruido separados por perceptrón ajustado con situ. inicial w=0
Recta separadora x2 = 50.00 + 3.66x1
+1
-1
40
20
0
−20
−40
−40
−20
0
20
40


Puntos con ruido separados por perceptrón ajustado con 3ª situ. inicial aleatoria (recta en intervalo restringido)
Recta separadora x2 = 11.60 + 1.61x1
+1
-1
40
20
0
−20
−40
−60
−40
−20
0
20
40


Puntos con ruido separados por perceptrón ajustado con 5ª situ. inicial aleatoria
Recta separadora x2 = 21.24 + 1.09x1
+1
-1
80
60
40
20
0
−20
−40
−40
−20
0
20
40


Puntos con ruido separados por perceptrón ajustado con 5ª situ. inicial aleatoria (recta en intervalo restringido)
Recta separadora x2 = 21.24 + 1.09x1
+1
-1
40
20
0
−20
−40
−40
−20
0
20
40


Puntos con ruido separados por perceptrón ajustado con 7ª situ. inicial aleatoria
Recta separadora x2 = 29.96 + 3.44x1
+1
-1
200
150
100
50
0
−50
−100
−150
−40
−20
0
20
40


Puntos con ruido separados por perceptrón ajustado con 7ª situ. inicial aleatoria (recta en intervalo restringido)
Recta separadora x2 = 29.96 + 3.44x1
+1
-1
40
20
0
−20
−40
−40
−20
0
20
40

2. **Regresión Logística**: En este ejercicio crearemos nuestra propia función objetivo f (una probabilidad en este caso) y nuestro conjunto de datos $\mathcal{D}$ 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 $\mathbf{x}$.

   Consideremos $d = 2$ para que los datos sean visualizables, y sea $\mathcal{X} = [0, 2] \times [0, 2]$ con probabilidad uniforme de elegir cada $\mathbf{x} \in \mathcal{X}$. Elegir una línea en el plano que pase por $\mathcal{X}$ 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 $\{\mathbf{x}_n\}$ de $\mathcal{X}$ y evaluar las respuestas $\{y_n\}$ de todos ellos respecto de la frontera elegida.

   a) 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 $||\mathbf{w}^{(t-1)} - \mathbf{w}^{(t)}|| < 0,01$, 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.
   - Aplicar una permutación aleatoria, $1, 2, \ldots, N$, en el orden de los datos antes de usarlos en cada época del algoritmo.
   - Usar una tasa de aprendizaje de $\eta = 0,01$

   Respuesta:

   La implementación del SGD en este caso es prácticamente igual a la de la práctica 1, salvo que ahora tendremos en cuenta estas condiciones. Como salida, la función devuelve el vector de pesos de nuestro ajuste obtenido y el número de épocas en que ha ejecutado el algoritmo.

   El algoritmo de Regresión Logística no es sino un caso particular del SGD. En el que **el error que pretendemos minimizar es el opuesto del neperiano de la verosimilitud** (*likelihood*) entre el tamaño de la muestra:

$$E_{\text{in}}(\mathbf{w}) = -\frac{1}{N}\ln(\mathcal{L}(\mathbf{w})) = -\frac{1}{N}\ln\left(\prod_{n=1}^{N} P(y_n|\mathbf{x}_n)\right) =$$

$$\frac{1}{N}\sum_{n=1}^{N}\ln\left(\frac{1}{\theta(y_n\mathbf{w}^T\mathbf{x}_n)}\right)=\frac{1}{N}\sum_{n=1}^{N}\ln\left(1+e^{-y_n\mathbf{w}^T\mathbf{x}_n}\right)$$

Efectivamente, esta función es estrictamente decreciente, por lo que podremos minimizarla con SGD si calculamos su gradiente (aplicado a un minibatch, si tenemos en cuenta que estamos usando SGD):

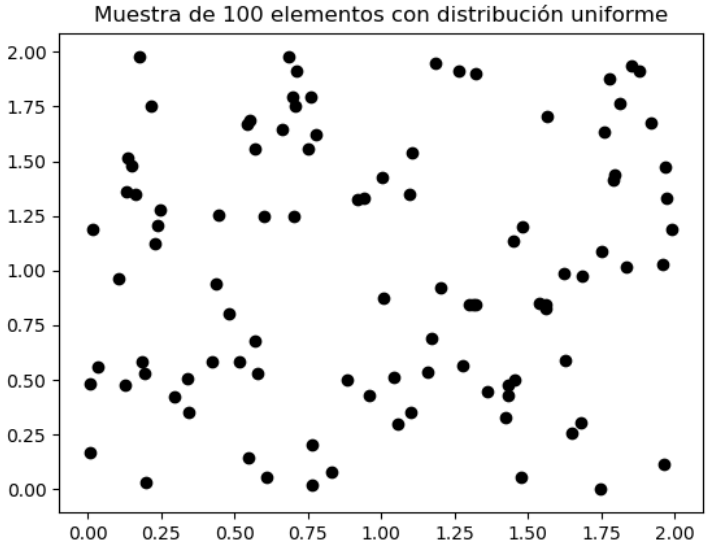
$$\nabla E_{\text{in}}(\mathbf{w})=\frac{1}{N}\sum_{n=1}^{N}\frac{y_n\mathbf{x}_n}{1+e^{y_n\mathbf{w}^T\mathbf{x}_n}}$$

b) 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).

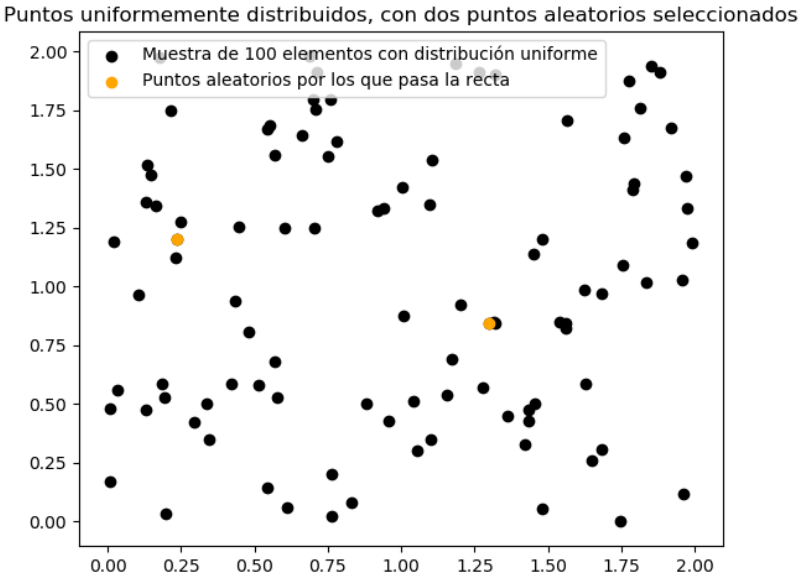
Respuesta:

Para aplicar la regresión logística que hemos implementado, generaremos el conjunto de datos siguiendo los pasos del enunciado:

En primer lugar, generamos la muestra de 100 puntos bidimensionales con distribución uniforme en el cuadrado $[0, 2] \times [0, 2]$



Seleccionamos después dos puntos de la muestra al azar



Y nos quedaría trazar la recta que pase por esos dos puntos para etiquetar los datos. Para ello basta con usar la ecuación recta pendiente: dados dos puntos (x_1, y_1) y (x_2, y_2), la pendiente de la recta sería

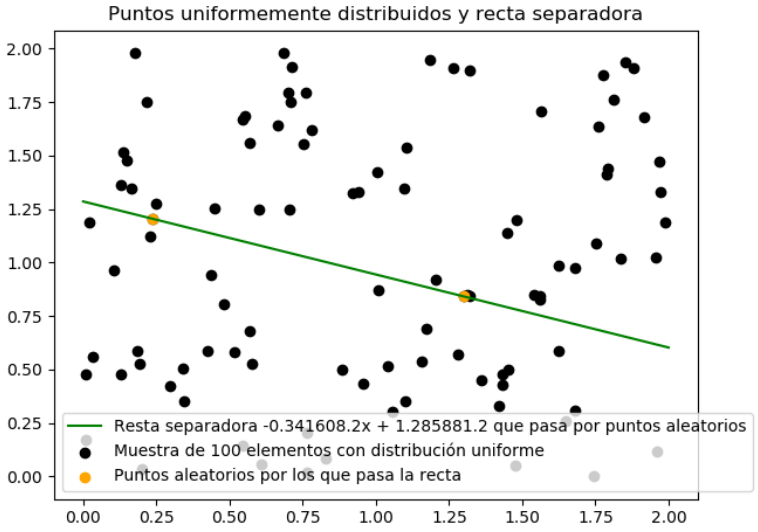
$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

Y la recta que pasa por ambos estaría descrita por la expresión

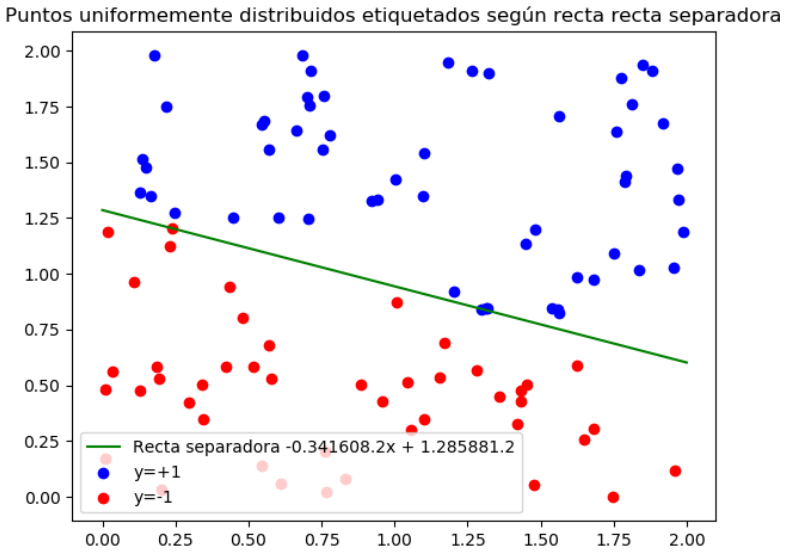
$$y - y_1 = m(x - x_1)$$

De donde despejamos y para obtener los coeficientes a y b de una recta $y = ax + b$:

$$y = mx + (y_1 - mx_1)$$



Finalmente, etiquetamos nuestra muestra según esa misma recta que acabamos de generar

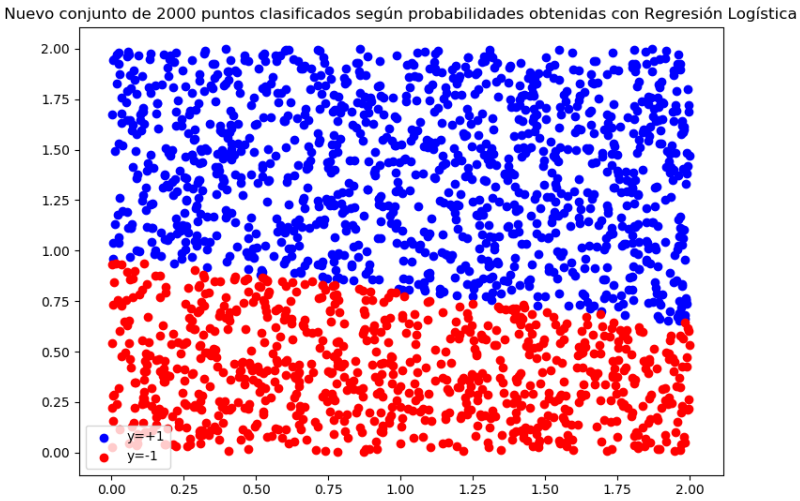


Ahora es el momento de aplicar SGD sobre la muestra, con lo que obtenemos un modelo $\mathbf{w}$ con el que obtenemos un error igual a 0.274, usando la fórmula que hemos descrito en el apartado a).

Si ahora generamos otra muestra uniformemente distribuida en el mismo intervalo de valores, podemos usar la $\mathbf{w}$ que acabamos de obtener para calcular sus probabilidades con la función

$$\theta(\mathbf{w}^{\mathrm{T}}\mathbf{x}) = \frac{e^{\mathbf{w}^{\mathrm{T}}\mathbf{x}}}{1 + e^{\mathbf{w}^{\mathrm{T}}\mathbf{x}}}$$

Si, por ejemplo, asignamos las etiquetas +1 y -1 según la probabilidad de un dato sea mayor o menor (o igual) a una frontera 0.5 respectivamente, obtenemos un error $E_{\text{out}} = 0{,}261$


Nuevo conjunto de 2000 puntos clasificados según probabilidades obtenidas con Regresión Logística
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
y=+1
y=-1
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00