Taller de Modelado II

LUIS ADRIAN MARTINEZ PEREZ

December 2024

1 Backpropagation AND

A continuación vamos a implementar el método de retropropagación en un Perceptrón Simple para modelar la compuerta lógica AND. Esta práctica se realizó usando el software *LibreOffice CAL* que es un tipo de *"excel"* para *linux.* Usamos el *solver* para programación no lineal: **libreoffice-nplsolver** y trabajamos con la siguiente versión de *LibreOffice CAL:*

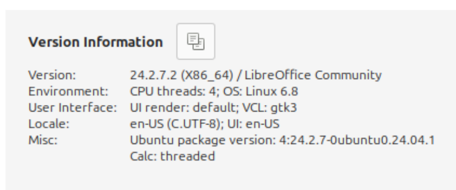

Figure 1: Versión de *LibreOffice CAL* usada en esta práctica.

Un Perceptrón Simple es un modelo neuronal unidireccional, compuesto por dos capas, una de entrada y otra de salida. En la siguiente figura se describe la estructura del modelo que tiene una capa de entrada con dos neuronas y una neurona de salida:

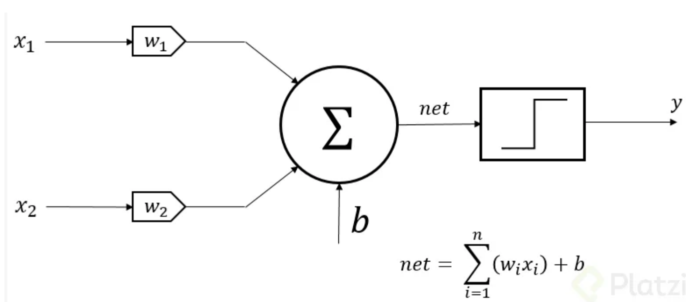

Figure 2: Estructura de un Perceptrón Simple con una capa de entrada y una salida.

Observemos que el conjunto de entrenamiento para la compuerta AND es el siguiente conjunto:

$$\mathcal{E} = \{\vec{x} = (x_1, x_2) : x_1, x_2 \in \{0, 1\}\}.$$

Este conjunto tiene 4 valores como podemos verlo en la siguiente tabla:

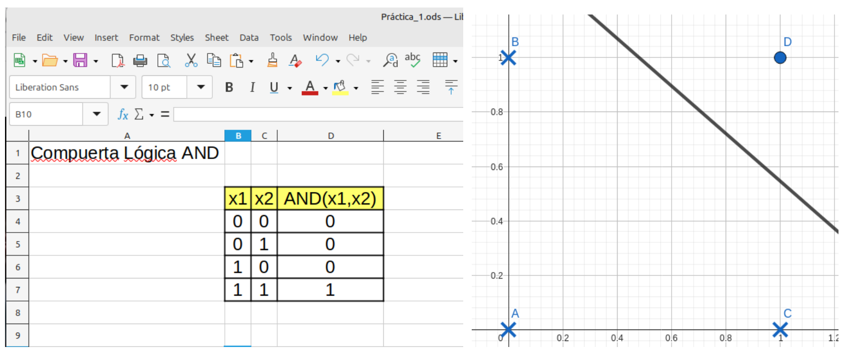

Figure 3: Tabla de verdad de la compuerta AND y gráfica de los puntos a separar.

En la gráfica anterior vemos representado el problema de forma geométrica y que cosnsiste en estimar la línea recta que separe los puntos A, B y C del punto D.

Sea $\vec{x} \in \mathcal{E}$ un vector de entrenamiento y sea $\vec{w} = (w_1, w_2, b)$ el vector de pesos cualesquiera. Definamos

$$z = z(\vec{x}, \vec{w}) = \vec{w} \cdot \vec{x} + b = w_1 x_1 + w_2 x_2 + b.$$

Sea

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

la función de activación sigmoidal junto con la siguiente función de pérdida

$$L(z) = (y - y_r)^2$$

en donde $y = \sigma(z)$ y y_r correspende al valor en la tabla de verdad.

Con base en el método de retropropagación vamos a estimar los valores de los pesos (w_1, w_2, b) que minimicen la función de pérdida L.

1.1 Calcular el error total

Sea $\vec{x}_i \in \mathcal{E}$, entonces el *error total* de la función de pérdida con respecto a $\vec{w}$ se define como

$$L_T(z_i) = \sum_{i=1}^{4} (y_i - y_{ri})^2$$

con $y_i = \sigma(z_i)$.

1.2 Minimizar L_T

Para calcular los valores (w_1, w_2, b) que minimicen la función de pérdida comenzamos con un valor inicial arbritario $\vec{w}^0 = (w_1^0, w_2^0, b^0)$ y calculamos la pérdida total $L_T(\vec{w}^0, \vec{x})$.

Posteriormente calculamos las derivadas parciales usando la regla de la cadena

w1	w2	b	x1	x2	z	y=σ(z)	yr	L
1	3	-2	0	0	-2	0.119203	0	0.014209
			0	1	1	0.731059	0	0.534447
			1	0	-1	0.268941	0	0.072329
			1	1	2	0.880797	1	0.014209
							L_T=	0.6351948

Figure 4: En nuestra hoja de excel comenzamos con el vector arbitrario de pesos $\vec{w}^0 = (1, 3, -2)$

- $\frac{\partial L_T}{\partial w_1} = \frac{\partial L_T}{\partial y}\frac{\partial y}{\partial z}\frac{\partial z}{\partial w_1} = 2(y - y_r)\frac{e^z}{(1+e^z)^2}x_1$
- $\frac{\partial L_T}{\partial w_2} = \frac{\partial L_T}{\partial y}\frac{\partial y}{\partial z}\frac{\partial z}{\partial w_2} = 2(y - y_r)\frac{e^z}{(1+e^z)^2}x_2$
- $\frac{\partial L_T}{\partial b} = \frac{\partial L_T}{\partial y}\frac{\partial y}{\partial z}\frac{\partial z}{\partial b} = 2(y - y_r)\frac{e^z}{(1+e^z)^2}$

∂L/∂y	∂y/∂z	∂z/∂w1	∂z/∂w2	∂z/∂b	∂L/∂w1	∂L/∂w2	∂L/∂b
0.238406	0.104994	0	0	1	0	0	0.025031
1.462117	0.196612	0	1	1	0	0.28747	0.28747
0.537883	0.196612	1	0	1	0.105754	0	0.105754
-0.238406	0.104994	1	1	1	-0.02503	-0.02503	-0.02503
				Prom	0.020181	0.06561	0.098306

Figure 5: En la hoja de excel definimos las derivadas parciales para la retropropagación

y definimos el vector $\vec{w}^1 = \vec{w}^0 - \Delta$, en donde

$$\Delta = k\left(\frac{\partial L_T}{\partial w_1}, \frac{\partial L_T}{\partial w_2}, \frac{\partial L_T}{\partial b}\right)$$

y k es la tasa de aprendizaje. Entonces, calculamos la pérdida total $L_T(\vec{w}^1, \vec{x})$ y si ésta está por debajo de una tolerancia T detenemos el proceso y obtenemos

w1	w2	b	x1	x2	z	y=σ(z)	yr	L	Class		Softmax
1.32	2.53	-2.86	0	0	-3	0.054227	0	0.002941	0	1.055724	0.1807
			0	1	0	0.417348	0	0.174179	0	1.517931	0.2599
			1	0	-2	0.177292	0	0.031433	0	1.19398	0.2044
			1	1	1	0.729157	1	0.073356	1	2.073332	0.355
								0.28190832		5.840967	

Figure 6: Este es el resultado en la tercera iteración con una tasa de aprendizaje $k = 10$

que el mínimo es el vector $\vec{w}^1$. En otro caso, iteramos una vez más y así sucesivamente hasta obtener un error menor que cierta tolerancia $T > 0$.

En la figura 6 observamos los resultados de la tercera iteración y que reporta un error de 0.28190832. Por otra parte, en la columna de la derecha vemos la definición la función indicadora con un umbral de 0.6 tal que $Class(y) = 1$ si $y > 0.6$ y 0 en otro caso. Finalmente, los datos se pueden transformar en una distribución de probabilidad de 0 a 1 con una suma de 1 con la función *softmax*.

Para finalizar usemos el *solver* (SCO Evolutionary Algorithm). En nuestro caso, minimizamos la función de pérdida L sujeto a la restrición de $-2 \leq b \leq -1$ y obtuvimos los resultados que se muestran en la gráfica 7 Después de 72

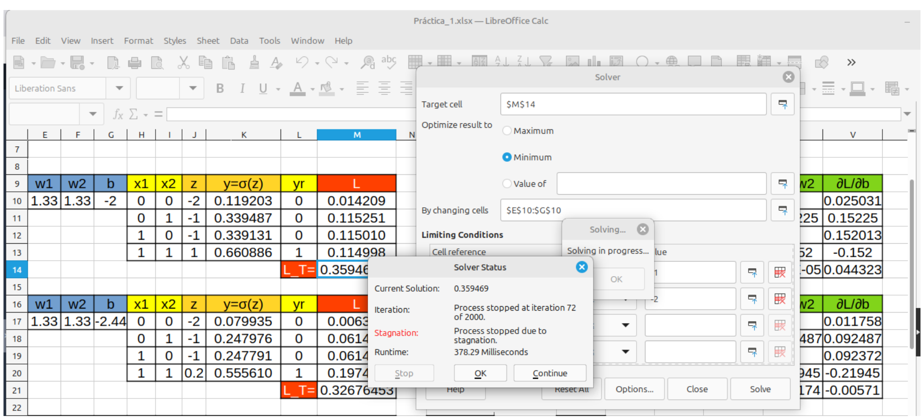

Figure 7: Resultados obtenidos al usar SCO Evolutionary Algorithm del *solver* para problemas no lineales.

iteraciones y con un error de $L = 0.359469$ obtenemos los siguientes valores:

- $w_1 = 1.33313183781849$
- $w_2 = 1.33310890791518$
- $b = -1.99999983335168$

Para ver la hoja de cálculo usada en esta práctica consulta esta liga:

Hoja de Cálculo para AND

2 Iris

2.1 Introducción

El desarrollo de esta práctica, que tiene como objetivo implementar el método de retropropagación en un Perceptón de Capa Múltiple **CPM**$(4, 3, 1)$ que consiste de una capa de entrada, una capa oculta y una capa de salida con $4, 3$ y 1 neuronas para la clasificación de las flores *iris* de Fisher se realiza usando el software *LibreOffice CAL* donde usamos la versión del *solver* para programación no lineal: **libreoffice-nplsolver**.

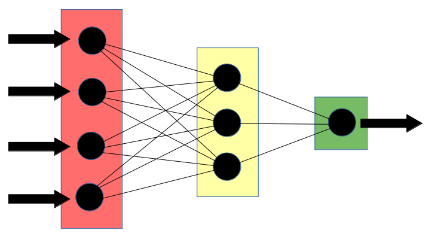

Figure 8: Arquitectura del perceptrón **CPM**$(4, 3, 1)$

Definamos el conjunto de vectores de entrenamiento para nuestro modelo

$$\mathcal{E} = \{(x_1, x_2, x_3, x_4) | x_i \in R \text{ para cada } i = 1, 2, 3, 4\}$$

La capa de entrada usa los vectores de entrenamiento y produce las entradas de las neuronas de la capa oculta que finalmente generan las entradas para la neurona de la capa de salida.

Sea $\vec{x} \in \mathcal{E}$ un vector de entrenamiento cualesquiera. Dado el diseño propuesto en la imagen de la figura 8 debemos, pasada la primera capa, obtener los valores z_1, z_2 y z_3 que son las entradas de las neuronas de la capa oculta.

Sea el vector de pesos $\vec{w}_k = (w_{k1}, w_{k2}, w_{k3}, w_{k4}, b_k)$. Para $k = 1, 2, 3$ definamos

$$z_k = \sum_{i=1}^{4} w_{ki} x_i + b_k$$

La salida de estas 3 neuronas, y que son las entradas para la neurona que tiene la capa de salida del modelo, están filtradas por la función de activación sigmoidal

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Tomemos $y_k = \sigma(z_k)$ con $i = 1, 2, 3$ las entradas de la última capa y definamos la entrada de la capa de salida

$$\zeta = \sum_{i=1}^{3} \omega_k y_k + \beta$$

A este valor lo filtraremos usando de nuevo la función sigmoidal lo que implica que la salida será de la forma

$$\mathrm{y} = \Sigma(\zeta) = \frac{1}{1 + e^{-\zeta}}$$

Calculamos el error con la función de pérdida

$$L = (\mathrm{y} - \mathrm{y}_{real})^2$$

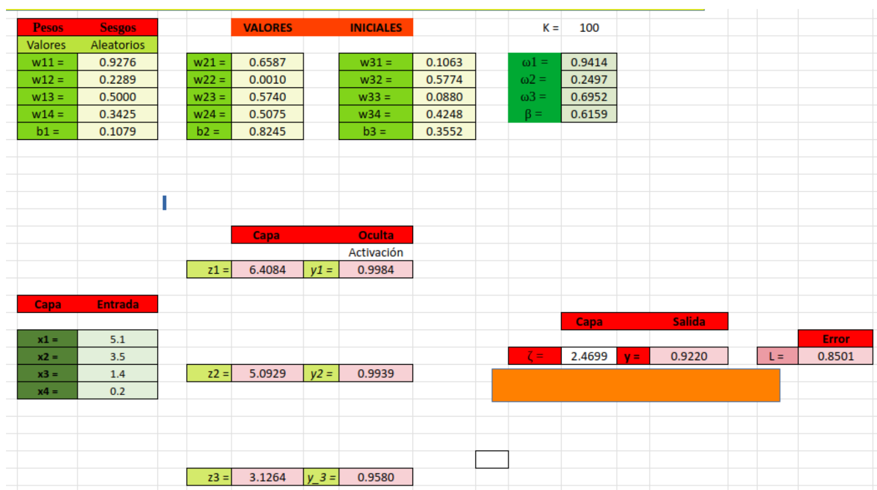

Figure 9: En amarillo se resaltan los valores iniciales de $\vec{w}^0$. Dichos valores se generan de manera aleatoria. En gris se resalta el calculo del error para un sólo dato de entrenamiento. Se usa una tasa de aprendizaje de 100.

Hagamos la implementación en excel de este modelo para clasificar las flores Iris. Partimos de la hoja de cálculo que descargamos con los datos correspondientes a las longitudes y anchuras de los sepálos y los pétalos de 219 flores clasificadas como: *setosa*, *versicolor* y *virginica*. En primer lugar, codifiquemos estas clases como

- $setosa \to 0$,
- $versicolor \to 1$,
- $virginica \to 2$.

En la siguiente figura se observan las primeras entradas de los cálculos del error para los primeros vectores de entrenamiento:

Clasificacion	x1 =	x2 =	x3 =	x4 =	z1 =	y1 =	z2 =	y2 =	z3 =	y3 =	ζ =	y =	L =
0	5.1	3.5	1.4	0.2	6.4084	0.9984	5.0929	0.9939	3.1264	0.9580	2.4699064809	0.9220050400168570	0.85009329
0	4.9	3.0	1.4	0.2	6.1084	0.9978	4.9606	0.9930	2.8164	0.9436	2.4591330860	0.9212267755784810	0.84865877
0	4.7	3.2	1.3	0.2	5.9187	0.9973	4.7717	0.9916	2.9018	0.9479	2.4613849773	0.9213900359069800	0.84895960
0	4.6	3.1	1.5	0.2	5.9030	0.9973	4.8205	0.9920	2.8511	0.9454	2.4596624382	0.9212651810476910	0.84872953
0	5.0	3.6	1.4	0.2	6.3385	0.9982	5.0271	0.9935	3.1735	0.9598	2.4709824393	0.9220823789344380	0.85023591
0	5.4	3.9	1.7	0.4	6.9967	0.9991	5.5646	0.9962	3.5006	0.9707	2.4800202793	0.9227292437930470	0.85142926
0	4.6	3.4	1.4	0.3	5.9560	0.9974	4.8142	0.9920	3.0580	0.9551	2.4665609663	0.9217641183027440	0.84964909
0	5.0	3.4	1.5	0.2	6.3427	0.9982	5.0843	0.9938	3.0668	0.9555	2.4680748264	0.9218732206830270	0.84985024
0	4.4	2.9	1.4	0.2	5.6217	0.9964	4.6311	0.9904	2.7055	0.9374	2.4528433168	0.9207691285953940	0.84781579
0	4.9	3.1	1.5	0.1	6.1471	0.9979	4.9674	0.9931	2.8405	0.9448	2.4601052059	0.9212972915007860	0.84878870
0	5.4	3.7	1.5	0.2	6.7825	0.9989	5.3481	0.9953	3.2825	0.9638	2.4748029509	0.9223564269399780	0.85074138
0	4.8	3.4	1.6	0.2	6.2072	0.9980	5.0100	0.9934	3.0544	0.9550	2.4673472713	0.9218208039097180	0.84975359
[illegible]	4.8	3.0	1.4	0.1	5.9814	0.9975	4.8440	0.9922	2.7633	0.9407	2.4566242947	0.9210445250988580	0.84832302
[illegible]	4.3	3.0	1.1	0.1	5.3676	0.9954	4.3424	0.9872	2.6838	0.9361	2.4501723925	0.9205740566324460	0.84745659
0	5.8	4.0	1.2	0.2	7.0722	0.9992	5.4397	0.9957	3.4719	0.9699	2.4793810831	0.9226836567902490	0.85134513
0	5.7	4.4	1.5	0.4	7.2895	0.9993	5.6480	0.9965	3.8036	0.9782	2.4855219751	0.9231206034445180	0.85215165
0	5.4	3.9	1.3	0.4	6.7967	0.9989	5.3350	0.9952	3.4654	0.9697	2.4788775083	0.9226477248578830	0.85127882
0	5.1	3.5	1.4	0.3	6.4426	0.9984	5.1436	0.9942	3.1689	0.9596	2.4711997822	0.9220979928277000	0.85026471

Figure 10: Lista con los primeros calculos del error para los vectores de entrenamiento.

Ahora calculemos las derivadas parciales necesarias para poder implementar el método de retropropagación

- $\frac{\partial L}{\partial \mathrm{y}} = 2(\mathrm{y} - \mathrm{y}_{real})$.
- $\frac{\partial \mathrm{y}}{\partial \zeta} = \mathrm{y}(1 - \mathrm{y})$.
- $\frac{\partial \zeta}{\partial \omega_k} = y_k$.
- $\frac{\partial \zeta}{\partial \beta} = 1$.
- $\frac{\partial \zeta}{\partial y_k} = \omega_k$.
- $\frac{\partial y_k}{\partial z_k} = y_k(1 - y_k)$.
- $\frac{\partial z_k}{\partial w_{ki}} = x_i$.
- $\frac{\partial z_k}{\partial b_k} = 1$.

Finalmente podemos expresar los pesos y sesgos para $i = 1, 2, 3, 4$ y $k = 1, 2, 3$ como

- $\frac{\partial L}{\partial \omega_k} = \frac{\partial L}{\partial \mathrm{y}} \cdot \frac{\partial \mathrm{y}}{\partial \zeta} \cdot \frac{\partial \zeta}{\partial \omega_k}$.
- $\frac{\partial L}{\partial \beta} = \frac{\partial L}{\partial \mathrm{y}} \cdot \frac{\partial \mathrm{y}}{\partial \zeta} \cdot \frac{\partial \zeta}{\partial \beta}$.
- $\frac{\partial L}{\partial w_{ki}} = \frac{\partial L}{\partial \mathrm{y}} \cdot \frac{\partial \mathrm{y}}{\partial \zeta} \cdot \frac{\partial \zeta}{\partial y_k} \cdot \frac{\partial y_k}{\partial z_k} \cdot \frac{\partial z_k}{\partial w_{ki}}$.
- $\frac{\partial L}{\partial b_k} = \frac{\partial L}{\partial \mathrm{y}} \cdot \frac{\partial \mathrm{y}}{\partial \zeta} \cdot \frac{\partial \zeta}{\partial y_k} \cdot \frac{\partial y_k}{\partial z_k} \cdot \frac{\partial z_k}{\partial b_k}$.

Nuestro objetivo, como en la primera parte de esta práctica, consistirá en encontrar el valor del vector de pesos que minimice L. Comenzamos con un valor inicial arbitrario $\vec{w}^0$. Los pesos w_{ki}^0 se pueden inicializar con valores pequeños

∂L/∂y	∂y/∂ζ	∂ζ/∂ω1	∂ζ/∂ω2	∂ζ/∂ω3	∂ζ/∂β
1.844010	0.071912	0.998355	0.993897	0.957968	1.000000

∂ζ/∂y1	∂ζ/∂y2	∂ζ/∂y3	∂y1/∂z1	∂y2/∂z2	∂y3/∂z3
0.941405	0.249674	0.695219	0.001642	0.006066	0.040265

∂zk/∂wk_1	∂zk/∂wk_2	∂zk/∂wk_3	∂zk/∂wk_4	∂zk/∂bk
5.100000	3.500000	1.400000	0.200000	1.000000

∂L/∂ω1	∂L/∂ω2	∂L/∂ω3	∂L/∂β	
0.132388	0.131797	0.127032	0.132606	
∂L/∂ω11	∂L/∂ω12	∂L/∂ω13	∂L/∂ω14	∂L/∂b1
0.001046	0.000718	0.000287	0.000041	0.000205
∂L/∂ω21	∂L/∂ω22	∂L/∂ω23	∂L/∂ω24	∂L/∂b2
0.001024	0.000703	0.000281	0.000040	0.000201
∂L/∂ω31	∂L/∂ω32	∂L/∂ω33	∂L/∂ω34	∂L/∂b3
0.018932	0.012992	0.005197	0.000742	0.003712

Figure 11: Estimación de las derivadas parciales para la retropropagación.

aleatorios para evitar que todas las neuronas aprendan lo mismo.

Si el error cumple la condición de que $L > T$ para cierta tolerancia dada $T > 0$, concluimos que $\vec{w}^0$ es el vector que minimiza la función de pérdida. En otro caso, construimos el vector $\vec{w}^1 = \vec{w}^0 - \Delta$, en donde

$$\Delta = k(\frac{\partial L_T}{\partial \omega_1}, \frac{\partial L_T}{\partial \omega_2}, \frac{\partial L_T}{\partial \beta}, \frac{\partial L_T}{\partial w_{11}}, \frac{\partial L_T}{\partial w_{12}}, \frac{\partial L_T}{\partial w_{13}}, \frac{\partial L_T}{\partial w_{14}}, \frac{\partial L_T}{\partial b_1} \dots)$$

en donde los términos $\frac{\partial L_T}{\partial \omega_k}$ y $\frac{\partial L_T}{\partial w_{ki}}$ se calculan utilizando las derivadas parciales obtenidas en la retropropagación, como se explicó anteriormente. Este proceso se repite de forma iterativa hasta que el error sea suficientemente pequeño y el modelo converja.

En nuestro caso, usamos el *solver* para minimizar la función de pérdida y obtenemos después de 25 iteraciones con un error cuadrático de 195.77694380 lo que da por resultado los valores que se ven en la siguiente gráfica:

Pesos	Sesgos		VALORES				
Valores	Aleatorios						
w11 =	0.8901	w21 =	1.6581	w31 =	2.8852	ω1 =	-0.3279
w12 =	0.0888	w22 =	-0.0038	w32 =	0.0575	ω2 =	-0.9681
w13 =	2.2888	w23 =	1.6079	w33 =	0.8718	ω3 =	-1.1581
w14 =	1.8288	w24 =	1.6226	w34 =	1.2733	β =	-1.5592
b1 =	2.3760	b2 =	0.2430	b3 =	1.3947		

Figure 12: Resultados obtenidos al usar SCO Evolutionary Algorithm del *solver* para problemas no lineales.

Clasificacion	x1 =	x2 =	x3 =	x4 =	z1 =	y1 =	z2 =	y2 =	z3 =	y3 =	ζ =	y =	L =	CLASS		SOFTMAX
0	5.1	3.5	1.4	0.2	10.7965	1.0000	11.2612	1.0000	17.7853	1.0000	-4.0132309003	0.0177540007540588	0.00031520	0	1.017913	0.008333
0	4.9	3.0	1.4	0.2	10.5741	1.0000	10.9315	1.0000	17.1796	1.0000	-4.0132243476	0.0177541150246531	0.00031521	0	1.017913	0.008333
0	4.7	3.2	1.3	0.2	10.1849	1.0000	10.4384	1.0000	16.5268	1.0000	-4.0132092854	0.0177543776957756	0.00031522	0	1.017913	0.008333
0	4.6	3.1	1.5	0.2	10.5448	1.0000	10.5945	1.0000	16.4069	1.0000	-4.0132171131	0.0177542411872745	0.00031521	0	1.017913	0.008333
0	5.0	3.6	1.4	0.2	10.7164	1.0000	11.0950	1.0000	17.5026	1.0000	-4.0132280820	0.0177540499013526	0.00031521	0	1.017913	0.008333
0	5.4	3.9	1.7	0.4	12.1515	1.0000	12.5640	1.0000	19.1901	1.0000	-4.0132449643	0.0177537554962729	0.00031520	0	1.017912	0.008333
0	4.6	3.4	1.4	0.3	10.5254	1.0000	10.5948	1.0000	16.4643	1.0000	-4.0132169569	0.0177542439114222	0.00031521	0	1.017913	0.008333
0	5.0	3.4	1.5	0.2	10.9275	1.0000	11.2566	1.0000	17.5782	1.0000	-4.0132316613	0.0177539874818551	0.00031520	0	1.017913	0.008333
0	4.4	2.9	1.4	0.2	10.1201	1.0000	10.1029	1.0000	15.7312	1.0000	-4.0131970629	0.0177545908470199	0.00031523	0	1.017913	0.008333
0	4.9	3.1	1.5	0.1	10.6290	1.0000	10.9297	1.0000	17.1452	1.0000	-4.0132247617	0.0177541078038727	0.00031521	0	1.017913	0.008333
0	5.4	3.7	1.5	0.2	11.3102	1.0000	11.9187	1.0000	18.7496	1.0000	-4.0132396092	0.0177538488819457	0.00031520	0	1.017912	0.008333
0	4.8	3.4	1.6	0.2	10.9783	1.0000	11.0858	1.0000	17.0884	1.0000	-4.0132296060	0.0177540233248426	0.00031521	0	1.017913	0.008333
0	4.8	3.0	1.4	0.1	10.3022	1.0000	10.6035	1.0000	16.7637	1.0000	-4.0132149852	0.0177542782965507	0.00031521	0	1.017913	0.008333

Figure 13: Se muestran los primeros resultados de la clasificación.

La hoja de cálculo usada para esta parte de la práctica se puede ver en:

Hoja de Cálculo para Iris

Finalmente, con base en los resultados obtenidos podemos observar que se puede separar y por lo tanto clasificar el conjunto de datos ya que la salida toma los valores $0, 1, 2$. Sin embargo, como puede observarse en la hoja hay algunos errores de clasificación en el caso Iris-versicolor donde se obtuvieron dos clasificaciones incorrectas.