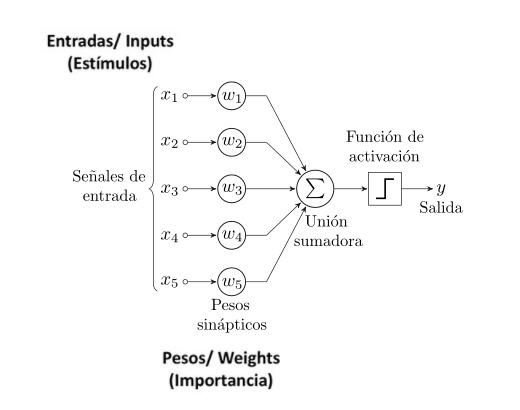
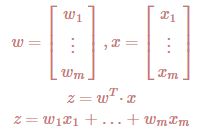
**Informe del Perceptrón**

El **Perceptrón** simple, también conocido una red neuronal de una sola capa (Single-Layer Neural Network), es un algoritmo de clasificación binaria creado por Frank Rosenblatt a partir del modelo neuronal de Warren McCulloch y Walter Pitts desarrollado en 1943.



La neurona recibe impulsos externos (x) que son considerados con distinta importancia o peso (w) en una función de activación (z). Si el estímulo agregado sobrepasa cierto umbral (θ), la neurona se activa.

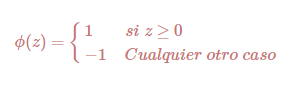
Matemáticamente, definimos x como el vector de estímulos y w como el vector de pesos, ambos m dimensiones, y z como la función de activación.



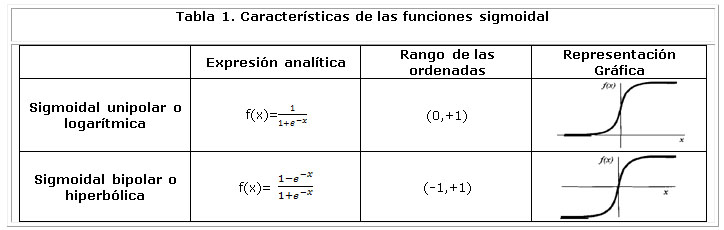
El perceptrón ϕ(z) se considera activo cuando su valor es mayor o igual al umbral θ o inactivo en cualquier otro caso. Formalmente esta es una [función escalón](https://es.wikipedia.org/wiki/Funci%C3%B3n_escal%C3%B3n_de_Heaviside) puede ser escrita de la siguiente forma:



Si incorporamos θ a la expresión, definiendo w0=−θ y x0=1 podemos escribir z = w0x0 + w1x1 + w2x3 + … + wmxm. Tal que:



Entonces es aquí donde elegimos nuestra función de activación como una sigmoide pero rebajada en 0.5, esto es, como una función escalón pero menos brusca en el momento de sus cambios.



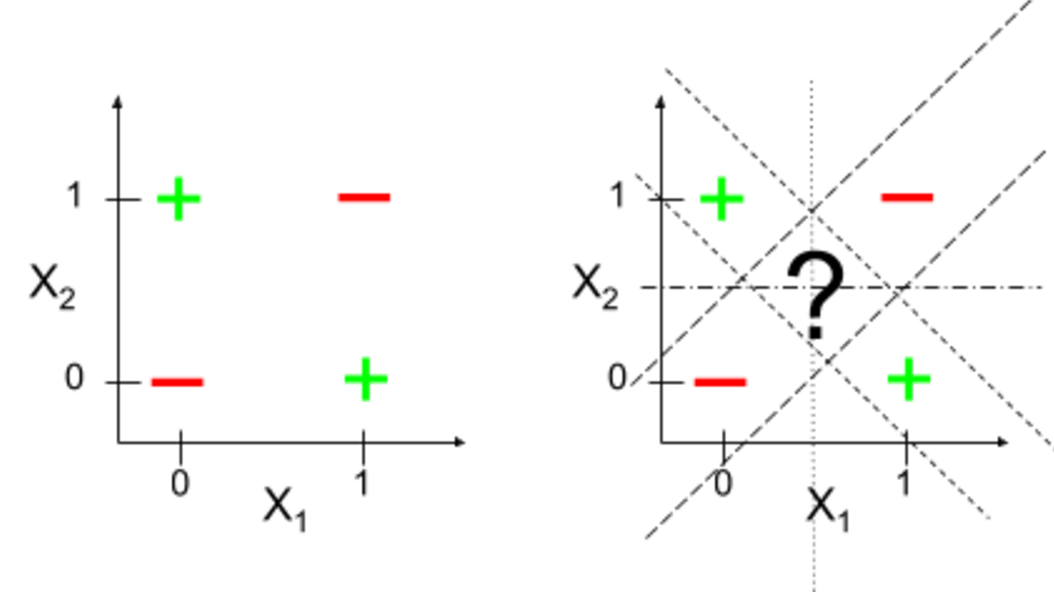
Es decir, pasamos de una función sigmoidal unipolar a una sigmoidal polar, debido a que nuestras clases están divididas en dos etiquetas, -1 y 1.

Posteriormente, implementamos el funcionamiento del perceptrón, básicamente es:

1. Se tiene un sistema de entrada normalizado, y un vector de salida de dos clases, una es -1 y otra 1.
2. Un vector de pesos es multiplicado con el producto punto con el vector de característica de una muestra de nosotros, esto lo hacemos pasar por la función de activación, este valor lo vamos a considerar como un valor predicho y como tenemos el vector de salida, o en especial para este dato conocemos si ha sido etiquetado como de la clase -1 o 1, entonces al hacer la multiplicación consideramos si fue clasificado correcta o incorrectamente, tal que, su representación es la de:

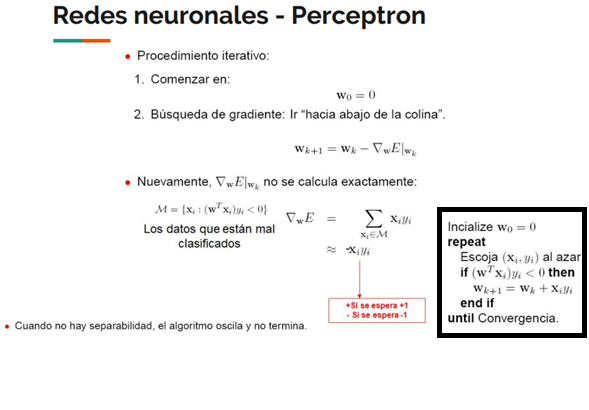
|  |  |  |
| --- | --- | --- |
| **WTXi (= Yp)** | **Yi = etiqueta** | **Yp Yi** |
| -1 | -1 | **+**( = >0) |
| -1 | 1 | **-**(= <0) |
| 1 | -1 | **-**(= <0) |
| 1 | 1 | **+**( = >0) |

Esto es idéntico a la multiplicación entre signos, y semejante a una puerta XOR:



1. Posteriormente el criterio de error del perceptrón es modificar el valor del peso +, - Xi, según lo esperado.

Entonces procedemos a implementar el siguiente código:

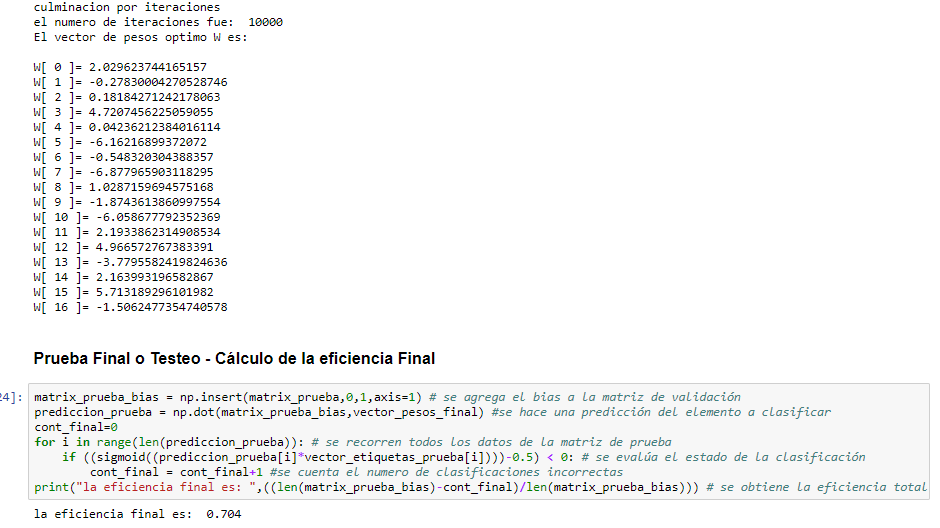


En especial el cuadro seleccionado, este se encuentra en un círculo while en nuestro código, después del subtítulo: “Implementando algoritmo Perceptrón para entrenamiento e Implementando validación según eficiencia”.

De hecho, nuestro código consta de los siguientes pasos:

1. Obteniendo y normalizando dato.
2. Obtenemos la matriz de salida (Valores: Clase1: -1, Clase2: 1 ; [-1,1]).
3. Dividir la matriz de datos de entrada para entrenamiento, validación y prueba.
4. Definiendo funciones necesarias para implementar el algoritmo perceptrón.
5. Implementando algoritmo Perceptrón para entrenamiento e Implementando validación según eficiencia.
6. Probando entrenamiento y validación.
7. Prueba Final o Testeo - Cálculo de la eficiencia Final.
8. Visualización del resultado.

En la forma en la cual culminamos nuestro procedimiento, puede ser entonces por cantidad de iteraciones o por superar un porcentaje de validación, para 10000 iteraciones entonces se logró tal porcentaje de eficiencia:



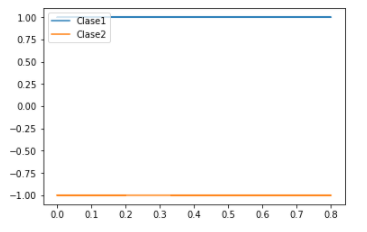
Como su culminación no fue realizada por llegar a una eficiencia mínima, se elige el peso que obtuvo la mejor eficiencia:



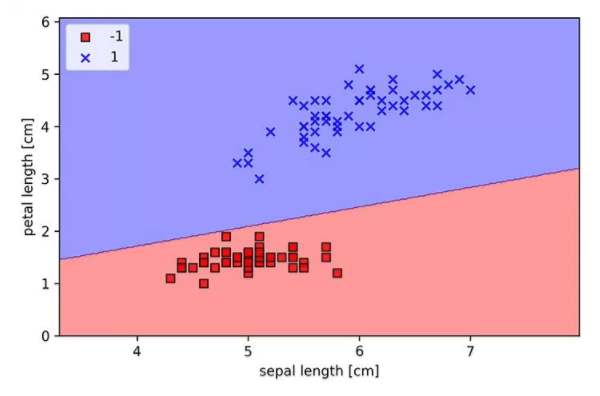
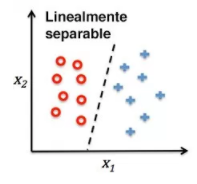
El cálculo de esta eficiencia es simple:

Contamos la cantidad de veces en que se corrigieron los pesos, pues en este algoritmo o procedimiento solo se corrigen los pesos cuando se esperaba que fuese de una clase y su valor fue el opuesto o la otra clase, entonces si para 20000 datos, se equivocó en 10000 entonces quiere decir que la cantidad de datos menos los errores sobre la cantidad total, nos dará un valor entre 0 y 1, referente a porcentaje, para nuestro caso, el perceptrón simple es capaz de clasificar cuales son las letras de nuestros apellidos con una eficiencia del 70.4% (Recordemos que esta fue la eficiencia en los datos de prueba puesto que en los datos de entrenamiento y validación fue de un 72.1%)

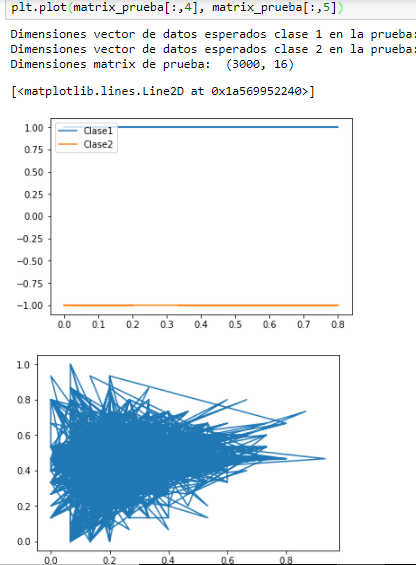
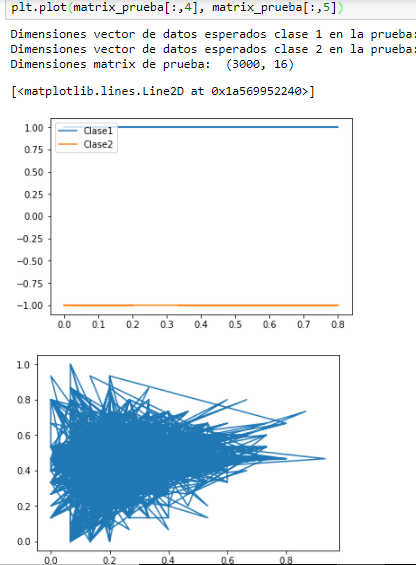
Se espera entonces que al graficar las etiquetas se obtenga:



Y al graficar las entradas con respecto a ellas mismas se pueda obtener lo siguiente:



En nuestro caso no se puede visualizar en un plano, es decir, que no es cortado por una línea recta sino por algún plano en el espacio:



Las principales limitaciones del perceptrón son que sirve únicamente para problemas linealmente separables y que sean de dos clases. Hablando vulgarmente, esto quiere decir que el perceptrón sólo lo podemos usar cuando el problema sea distinguir entre una de dos posibles clases y, que, trazando una línea, plano o hiperplano en un plano o hiperplano, se puedan separar perfectamente estas dos clases.

Resumiendo, se puede decir que el perceptrón fue diseñado para tratar con clases linealmente separables utilizando una función discriminante lineal para crear una frontera de decisión.

El perceptrón es un modelo simple, reduccionista y con muchas limitaciones, pero es un excelente ejemplo introductorio a los clasificadores y a las redes neuronales.