Uptc
Universidad Pedagógica y Tecnológica de Colombia

Ingeniería de Sistemas y Computación

Taller No 1 - Inteligencia Computacional

Conceptos de Inteligencia y Redes Neuronales

David Felipe Huertas Flechas, German Andrés Torres Cánchala, Lina Alejandra Melo Patiño.

Punto 1: Leyendo... ando

Resumen capítulo 1: Introducción y Antecedentes de la Inteligencia Artificial

Este capítulo nos dio un punto de partida para entender la Inteligencia Artificial, presentando los personajes, la historia contribuciones, aplicaciones, las herramientas que dieron la base de esta ciencia, etc.

El padre de la Inteligencia Artificial es Alan Turing y McCarthy, Marvin Minsky, Nathaniel Rochester, Claude Shannon, Herbert Simon y Allen Newell dieron la base para lo que hoy conocemos como IA. Actualmente aún no se tiene una definición exacta de que es ya que existen varios tipos de esta. Otras ciencias como la filosofía, matemáticas, psicología, computación, lingüística, economía y la neurociencia han contribuido con conocimientos para dar cabida a esta ciencia.

Nota: Existen otras ciencias que han aportado conocimiento, pero no se mencionan en el libro como lo es, la sociología y su enfoque de lo que es éticamente correcto.

En el libro nos mencionan dos tipos de representaciones:

1. Simbólicas: basadas en un número finito de primitivas y de reglas para la manipulación de símbolos.

2. Sub-simbólico: se caracteriza por crear sistemas con capacidad de aprendizaje. Éste se puede obtener a nivel de individuo imitando el cerebro (Redes Neuronales), a nivel de especie, imitando la evolución.

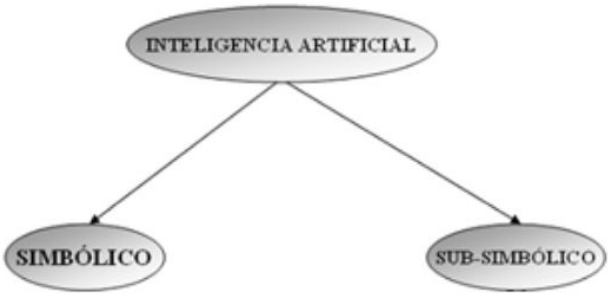


Figura 1: Clasificación tipos de inteligencia Artificial. [3]

Actualmente existe cuatro tipos de sistemas de inteligencia:

1. Sistemas que Piensan como Humanos
   Tratan de emular el pensamiento humano; por ejemplo, las redes neuronales artificiales.
2. Sistemas que Actúan como Humanos
   Tratan de actuar como humanos; es decir, imitan el comportamiento humano; por ejemplo, la robótica.
3. Sistemas que Piensan Racionalmente
   Tratan de imitar o emular el pensamiento lógico racional del ser humano; por ejemplo, los sistemas expertos.
4. Sistemas actuantes racionales
   Tratan de emular de forma racional el comportamiento humano; por ejemplo, los agentes inteligentes.

Figura 2: Clasificación de sistemas [4].

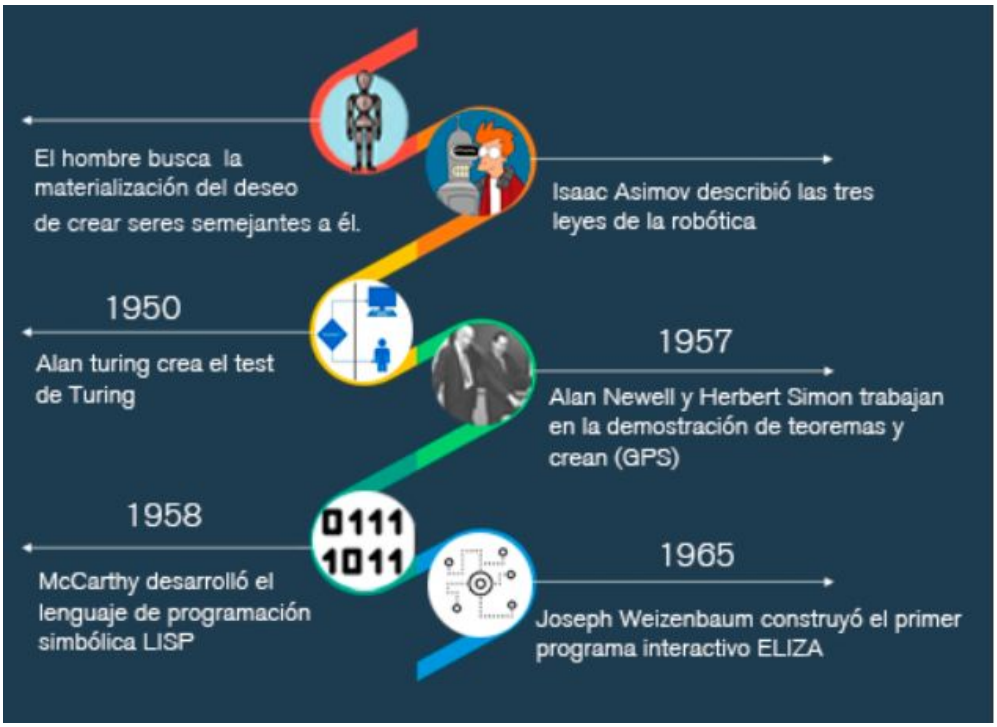


Figura 3: Histórico de la IA.

La prueba de Turing intenta ofrecer una definición de Inteligencia Artificial que se pueda evaluar mediante un test en el que se establece si se sostuvo una charla con una máquina o una persona. Para aprobar es necesario que la computadora tenga vista y robótica.

Algunas de las herramientas derivadas de la inteligencia artificial:

- Lenguajes de Programación.
- Aplicaciones y Sistemas Expertos.
- Ambientes de desarrollo (Shells).

El matemático inglés Alan Turing en 1950, estableció una prueba que determinaba si una maquina o inteligencia era realmente inteligente, para ello la maquina se comunicaba con una persona en un cuarto separado, y esta debía engañar al evaluador haciéndolo creer que se está comunicando con otra persona, de esta manera se podía evaluar la inteligencia si el número de errores se acerba a los errores cometidos por una persona. Actualmente para pasar la prueba la inteligencia artificial debe procesar un lenguaje natural, representar el conocimiento, razonar automáticamente y auto aprender.

Durante esta época se desarrollaron varios sistemas de inteligencia artificial lingüísticos, también conocidos como chats bots, seguido de esto se trabajó en inteligencias artificiales que pudieran resolver problemas de la vida cotidiana, dando origen a los sistemas expertos. En los años 80 se

desarrollaron lenguajes especializados en la inteligencia artificial como LISP o Prolog, así como programas especializados en resolver problemas lógicos en juegos, como el ajedrez o las damas.

Entre las múltiples definiciones de la inteligencia artificial, hoy en día se tienen los sistemas que piensan como humanos, enfocándose en el estudio de la mente humana, también los sistemas que actúan como humanos, las acciones y decisiones de este tipo de inteligencia deben ser similares a las de un ser humano que se encuentra en sus mismas condiciones, lo que llevaría a realizar la prueba de Turing, así como la capacidad de Procesamiento del Lenguaje Natural, Representación del Conocimiento, Razonamiento, Aprendizaje; Los sistemas que piensan racionalmente, indica que las leyes del pensamiento se fundamentan en la lógica y en la elección de la mejor opción.

Como se puede observar la inteligencia artificial tiene varios tópicos, como lo pueden ser minería de datos, el reconocimiento de patrones, el reconocimiento de imágenes, sistemas expertos, algoritmos genéticos, entre otros.[1]

Capítulo 5: Introducción al Aprendizaje

Actualmente podemos acceder a información que no este ubicada físicamente cerca de nosotros y procesar grandes cantidades de datos, gracias a los avances tecnológicos y las redes computacionales. Debido a esto aparece el concepto de aprendizaje en el ámbito de la informática, también conocido como aprendizaje de máquina, el cual hace referencia a programas que tienen como objetivo optimizar los parámetros por medio de datos de entrenamiento. Existen así los modelos inductivos, cuando realizan predicciones y descriptivos cuando generan conocimientos.

El aprendizaje automático o de máquina, usa la estadística para la construcción de modelos matemáticos, haciendo posible las inferencias a partir de una muestra. La computación es parte fundamental del entrenamiento para la implementar algoritmos de optimización eficientes, tareas de almacenamiento y procesamiento de grandes volúmenes de datos.

Extraer información de grandes volúmenes de datos es una aplicación del aprendizaje automático, también conocido como minería de datos, las aplicaciones de esta se han extendido a áreas como el estudio del comportamiento de consumo de los clientes de un supermercado. Existen aplicaciones para el diagnostico médico, en telecomunicaciones, los patrones de llamadas son analizados para optimizar y maximizar la calidad del servicio.

[1] Julio Cesar Ponce Gallegos, Aurora Torres Soto, et al. Inteligencia Artificial.1ra. Iniciativa Latinoamericana de Libros de Texto Abiertos (LATln), 2014.

Existe dos clasificaciones para los algoritmos de aprendizaje automática, el supervisado y no supervisado. Siendo el supervisado el que corresponde a una situación en que una variable de salida, cuantitativa o cualitativa, que se desee predecir basado en un conjunto de características.

El aprendizaje no supervisado corresponde a la situación en que existe un conjunto de datos que contiene diversas características de determinados individuos, sin que ninguna sea considera una variable de salida que se desee predecir, para esto se establece un problema un modelo que permite relacionar la variable de salida con las características, luego teniendo en cuenta un conjunto de datos de entrenamiento, el cual se observan valores para la variable de salida y las características para determinados casos o individuos. Usando datos que se ajusten a los parámetros es posible predecir la variable de salida. Dicho proceso se denomina aprendizaje supervisado, al ser un proceso guiado por la variable de salida.

El aprendizaje no supervisado es la situación en la cual un conjunto de datos contiene diversas características, pero ninguna de ellas es una variable de salida. La tarea del aprendizaje en este caso es describir la organización de los datos, su agrupamiento y las asociaciones.

El problema más usual de los algoritmos de aprendizaje supervisado es entorno a su clasificación. Una de las estrategias de aprendizaje es la regresión en los datos de entrenamiento, en el cual se implementa un clasificador, poniendo una línea recta en el plano y dando valores de 1 a los valores de entrada que están sobre la línea y de 0 a aquellos que están debajo de la línea. Sin embargo, esta estrategia da como resultado una mala clasificación de muchos datos, si el problema no es lineal.

También existe el algoritmo los k vecinos más cercanos, el cual no hace suposiciones acerca de la relación entre variables de entrada y de salida, en vez de ello estima el valor de la variable de salida en función de los vecinos más cercanos (k) en relación con las variables de entrada.

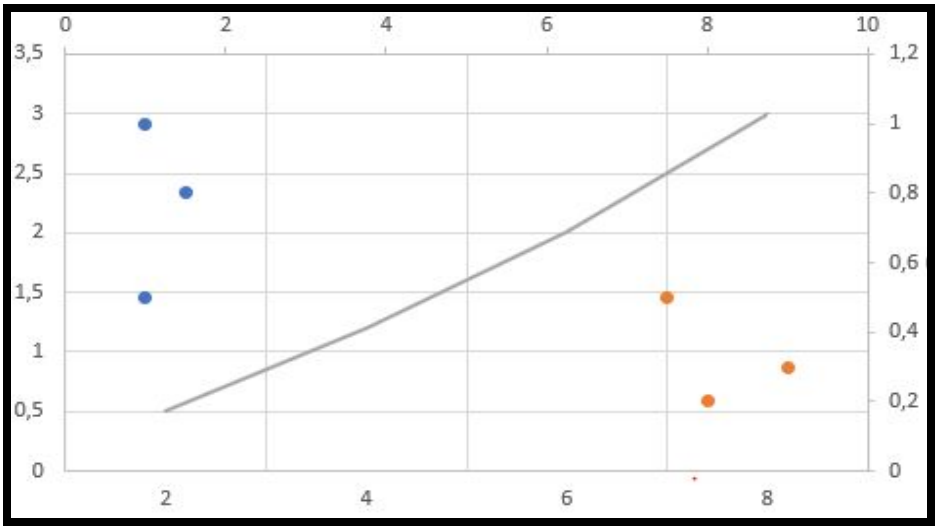
Teniendo en cuenta los perceptrones.

En el siguiente enlace encontrará todos los recursos (código, datos y videos) de todos los problemas del taller:

https://drive.google.com/drive/folders/1iEirvHgD3ZdklPncwdXK-AIry_h2YiQZ?usp=sharing

CUADRO DE INFORMACIÓN	
Lenguaje de programación	Java
Sistema Operativo	Windows 10 Pro-64 bits
Procesador del computador	Intel® Core ™ i5-6200U CPU 2.30GHz (4CPUs), ~2.4GHz
Memoria del computador	8192MB RAM
Placa del computador	ASUSTek COMPUTER INC X555UB

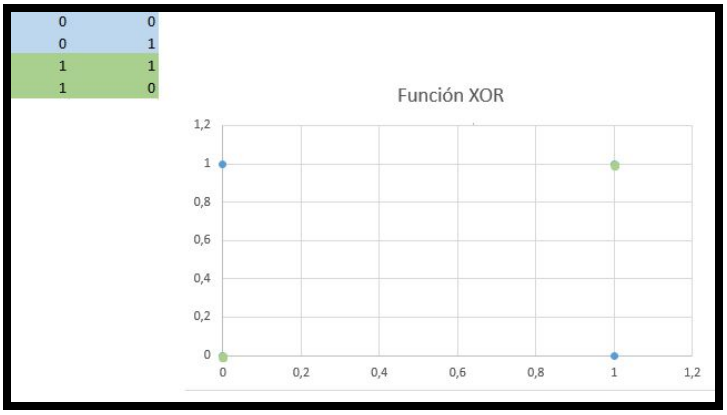
3.1 Siendo la línea gris un perceptrón simple con función de activación f(x)= $w1\ x1 + w2x2 + \cdots +$ wnxn, donde los puntos menores a la función son los naranjas y se encuentran por debajo de la línea y los mayores son los puntos azules por encimas de la misma.



3.2 La salida de una función XOR está dada por la tabla:

X_1	X_2	X_1 XOR X_2
0	0	0
0	1	1
1	0	1
1	1	0

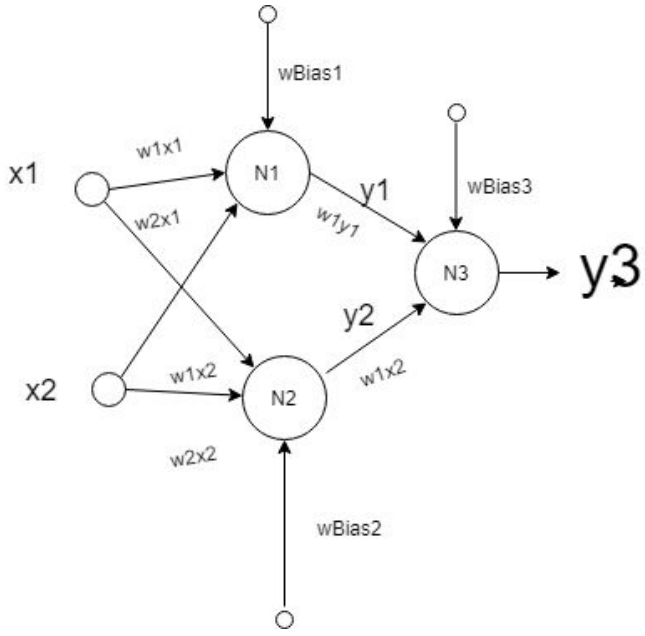
3.2.1 Demostrar que un perceptrón no tiene la capacidad de entender la función anteriormente citada.



Teniendo en cuenta que un perceptrón tiene la capacidad de resolver un problema con un conjunto de puntos linealmente separables, y no hay forma de separar estos conjuntos de forma lineal, tendrían que haber 2 líneas rectas para separarlos o una parábola pero no se puede implementar la parábola con lo visto hasta el momento en el curso, por ende se implementó la solución con 2 rectas.

3.2.2. Resolver el problema de aprendizaje mediante una red neuronal.

Para resolver el problema se plantea un perceptrón con que corresponde al siguiente esquema de salida:



Para la simulación todos los pesos fueron inicializados con un valor aleatorio entre o y 1, y un factor de aprendizaje de 0,5.

Inicialmente tenemos el arreglo definido en el código:

```
public static void main(String args[]){
 //
    double [][] arregloXOR = {
                        {0, 0, 0},
                        {0, 1, 1},
                        {1, 0, 1},
                        {1, 1, 0}
                        };
```

Se utiliza la función sumatoria para la neurona 1 y la neurona 2.

$$y = \sum_{i=1}^{n} wixi + (1 \cdot wbias)$$

```
//Calcula la Salida de las Neuronas de la capa oculta
y1 = (arregloXOR[fila][0] * w1X1) + (arregloXOR[fila][1] * w1X2) + (1 * wBias1);
y2 = (arregloXOR[fila][0] * w2X1) + (arregloXOR[fila][1] * w2X2) + (1 * wBias2);
```

Se utiliza la función de activación para la neurona 1 y la neurona 2 que es una función sigmoide.

$$f(y) = \frac{1}{1 + e^{-y}}$$

```
//Implementa la funcion de activacion de activación
y1 = 1.0/(1 + Math.pow(Math.E, (-1) * y1));
y2 = 1.0/(1 + Math.pow(Math.E, (-1) * y2));
```

Con esta información se encuentra la función de activación y la función sumatoria de la neurona 3.

```
//Calcula la salida de la neurona de salida
y3 = (y1 * w1Y1) + (y2 * w1Y2) + (1 * wBias3);
//Implementa la funcion Sigmoide
y3 = 1.0/(1 + Math.pow(Math.E, (-1) * y3));
```

Se realiza back propagación 1000 veces para que la neurona adquiera aprendizaje y se obtenga unas salidas más específicas.

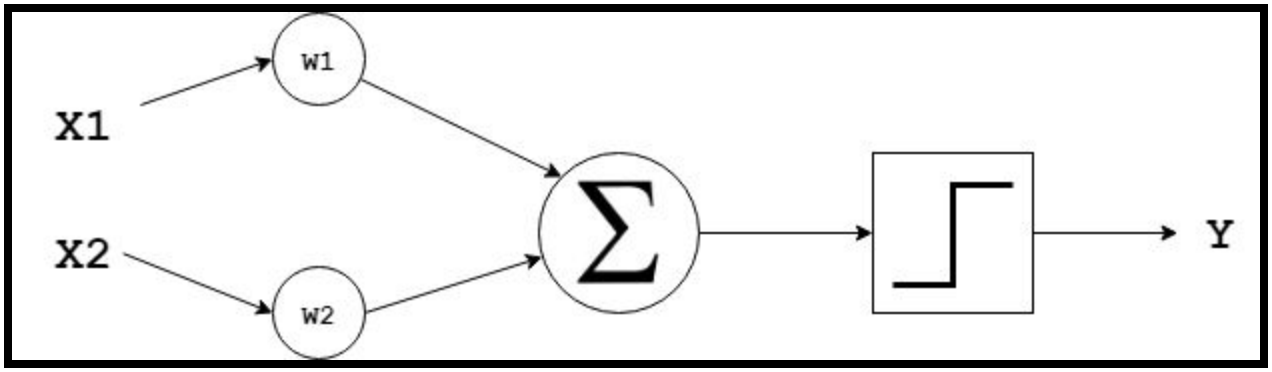
```
while(iteraciones<=1000){

    //FORWARD PROPAGATION
    //Calcula la Salida de las Neuronas de la capa oculta
    y1 = (arregloXOR[fila][0] * w1X1) + (arregloXOR[fila][1] * w1X2) + (1 * wBias1);
    y2 = (arregloXOR[fila][0] * w2X1) + (arregloXOR[fila][1] * w2X2) + (1 * wBias2);

    //Implementa la funcion de activacion de activación
    y1 = 1.0/(1 + Math.pow(Math.E, (-1) * y1));
    y2 = 1.0/(1 + Math.pow(Math.E, (-1) * y2));

    //Calcula la salida de la neurona de salida
    y3 = (y1 * w1Y1) + (y2 * w1Y2) + (1 * wBias3);
    //Implementa la funcion Sigmoide
    y3 = 1.0/(1 + Math.pow(Math.E, (-1) * y3));

    //BACKPROPAGATION
    //Calcula el error de la neurona de Salida
    errorDelta3=(y3 * (1 - y3))*(arregloXOR[fila][2] - y3);

    //Ajusta los pesos de la neurona de salida
    w1Y1 = w1Y1 + (factorAprendizaje*errorDelta3 * y1);
    w1Y2 = w1Y2 + (factorAprendizaje*errorDelta3 * y2);
    wBias3 = wBias3 + (errorDelta3);

    //Calcula el error de las neuronas de capa oculta
    errorDelta1=(y1 * (1 - y1)) * errorDelta3 - w1Y1;
    errorDelta2=(y2 * (1 - y2)) * errorDelta3 - w1Y2;

    //Ajusta los pesos de las neuronas de la capa oculta (Neurona 1)
    w1X1 = w1X1 + (factorAprendizaje*errorDelta1 * arregloXOR[fila][0]);
    w1X2 = w1X2 + (factorAprendizaje*errorDelta1 * arregloXOR[fila][1]);
    wBias1 = wBias1 + errorDelta1;
    //Ajusta los pesos de las neuronas de la capa oculta (Neurona 2)
    w2X1 = w2X1 + (factorAprendizaje*errorDelta2 * arregloXOR[fila][0]);
    w2X2 = w2X2 + (factorAprendizaje*errorDelta2 * arregloXOR[fila][1]);
    wBias2 = wBias2 + errorDelta2;
    iteraciones++;
}
```

Finalmente, estas son las salidas que se generaron en el software

```
0       XOR     0       =       0       -->  0.023488824053191154
0       XOR     1       =       1       -->  0.9766619516570674
1       XOR     0       =       1       -->  0.9765984429423775
1       XOR     1       =       0       -->  0.023472630546134655
BUILD SUCCESSFUL (total time: 0 seconds)
```

3.3 Público en un evento social, clasificado por el género.

Para resolver el problema planteado se planteó un perceptrón que corresponde al siguiente esquema de salida.



El cual utiliza la función de activación definida por $(\sum w_i * x_i) + \theta$. Luego procedió a determinar rangos entre cada característica dependiendo de su género.

Genero	Característica	Rango	Valor asignado
Hombre	Peso	>65	0
	Altura	>170	
Mujer	Peso	<=65	1
	Altura	<=170	

Siendo así, se generan veintidós (22) valores para cada variable que se almacenan en un archivo separado por comas que se lee y procesa dentro del código del perceptrón, el cual una vez se ejecuta da los valores para los pesos y el baias.

Nota: Los valores asignados los puede encontrar en el enlace que se adjuntó al inicio del punto 3, el documento se llama “Valores problema género”. Estos valores fueron obtenidos con la ayuda de la herramienta de Excel para obtener número aleatorios con una distribución uniforme; esto debido a que en el curso de Simulación se determinó que la herramienta cumple con su función.

Para la función de activación usamos la siguiente configuración:

$$\{0\ ,\ x<\ 0\ \}y\ \{\ 1\ ,\ x\geq 0\ \}$$

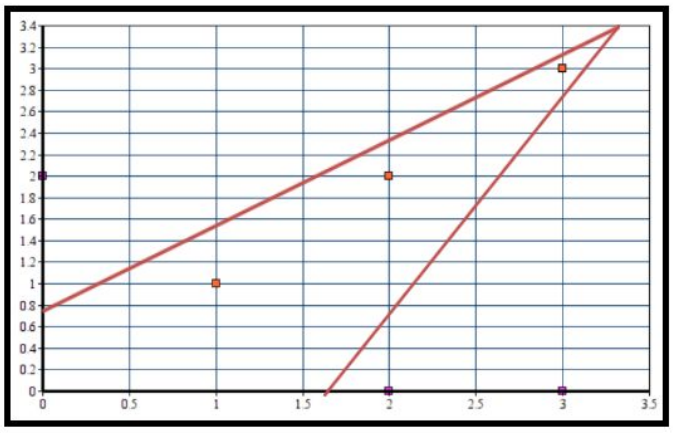
Con un valor de baias inicial de 0.1 y un factor de aprendizaje de 0.02 obtuvimos los siguientes valores:

```
Pesos
W1-0.03549715433426398
W2-0.02938561037033624
Baias
-0.020000000000000063
```

Ya para finalizar este punto, se trató el problema como una compuerta OR y se observó que la cantidad de hombres era mayor y el perceptrón aprendió y nos dio los valores de peso correspondientes para lograr una separación de puntos exitosa.

3.4 Conjuntos de puntos

Como se puede apreciar en el gráfico, los puntos naranjas (Conjunto A) NO puede ser separado por una línea recta de los puntos morados (Conjunto B), es decir que un solo perceptrón no podría dar solución a este problema. Si utilizamos la configuración del punto de la compuerta XOR se podría llegar a una solución.



Igualmente, se intentó realizar el procedimiento con un solo perceptrón y no se llegó a una respuesta.

```
    public static void main(String[] args) {
        //Cargamos los datos a memoria
        Persistence persistence = new Persistence();
        persistence.readCSVInteger();
        // Creamos el perceptron
        Perceptron perceptron = new Perceptron(persistence.getY(), persistence.getInput(), 0);
        //Entrenamos al perceptron
        perceptron.training();
        System.out.println("Pesos");
        System.out.println("W1" + perceptron.getWeight()[0]);
        System.out.println("W2" + perceptron.getWeight()[1]);
        System.out.println("Baias");
        System.out.println(perceptron.baias);
```

```
Console  Problems  Progress  Debug Shell
erminated> Run [Java Application] C:\Program Files\Java\jdk-11.0.3\bin\javaw.exe (6/05/2019, 12:09:50 a. m.)
=> Esperada = 0, Calculada = 1[Error]0
------------------
=> Esperada = 0, Calculada = 1[Error]0
------------------
=> Esperada = 0, Calculada = 1[Error]0
------------------
=> Esperada = 0, Calculada = 1[Error]0
------------------
=> Esperada = 0, Calculada = 1[Error]0
------------------
=> Esperada = 0, Calculada = 1[Error]0
------------------
=> Esperada = 0, Calculada = 1[Error]0
------------------
=> Esperada = 0, Calculada = 1[Error]0
------------------
=> Esperada = 0, Calculada = 1[Error]0
------------------
=> Esperada = 0, Calculada =
```

Programando un perceptrón.

CUADRO DE INFORMACIÓN	
Lenguaje de programación	Java
Sistema Operativo	Windows 10 Pro-64 bits
Procesador del computador	Intel® Core ™ i5-6200U CPU 2.30GHz (4CPUs), ~2.4GHz
Memoria del computador	8192MB RAM
Placa del computador	ASUSTek COMPUTER INC X555UB

Para la generación de un perceptrón que implemente un problema de aplicación, teniendo en cuenta las variables de entrenamiento fijadas. Se toma como ejemplo o caso de aplicación la devolución de un producto o artículo teniendo en cuenta su estado y el número de días después de su compra, lo cual representa las variables que se tienen en el sistema y aquellas que se van a evaluar.

Variables	
x1	Estado
x2	Días desde su compra

Teniendo entonces como la caracterización del estado unos valores dependiendo de una de las tres opciones que hay para el valor del estado de un producto.

Estado	
1	Bueno
2	Aceptable
3	Malo

Teniendo entonces inicialmente los datos sin normalizar y con sus respectivos picos máximos y mínimos.

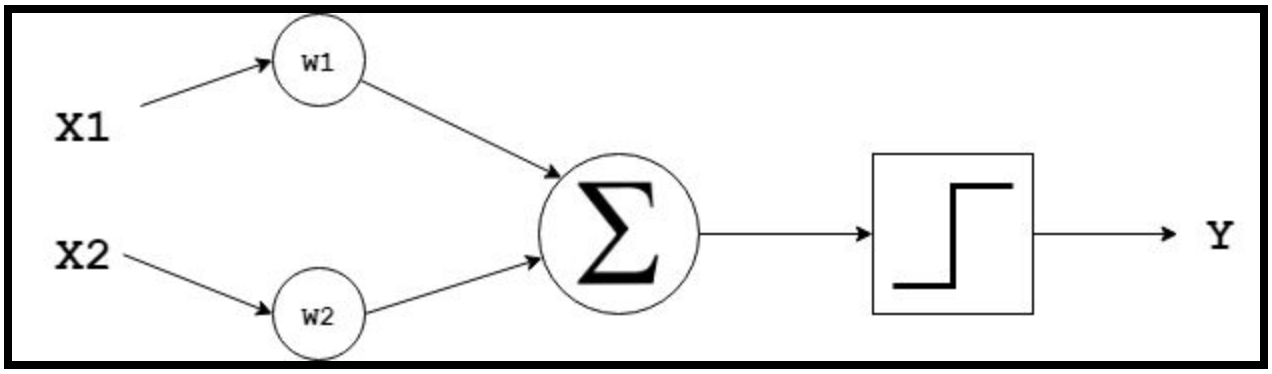
DC		
X1	X2	Y
1	2	V
2	2	V
2	4	V
3	4	F

1	2	MIN
3	4	MAX

A partir de estos datos podemos calcular la matriz de entrenamiento que vamos a utilizar.

ME		
x1	x2	y
0	0	1
0,5	0	1
0,5	1	1
1	1	0

El esquema al que corresponde el perceptrón y la función de activación se presenta a continuación:



Donde la función de activación *hard limit* está dada por:

$$\{0\ ,\ x < 0\}$$

$$\{1\ ,\ x \geq 0\ \}$$

Siendo así, se generan veintidós (22) valores por cada variable que se almacenan en un archivo separado por comas, luego se lee y procesa dentro del código del perceptrón, el cual una vez se ejecuta da los valores para los pesos y el baias.

```
Pesos
W10.8344090146824014
W2-0.020925652298805802
Baias
0.55
```

El comportamiento de este ejemplo de aplicación corresponde a una compuerta AND, al tener que cumplirse ambas condiciones, es decir que el estado sea **Bueno o Excelente** y **el número de días debe estar entre 0 y 10** para que la devolución se pueda realizar.

Conclusiones

- Los comportamientos de las compuertas OR y AND puede reflejarse en muchos ejemplos de aplicación o casos de la vida real y permiten que el perceptrón aprenda adecuadamente el comportamiento que se está representando por medio del modelo matemático y la función de activación *Hard Limit.*

- La función escalonada no da una suficiente flexibilidad en algunos casos, como se puedo ver en el taller, está no logro dar una solución a problemas que parecen sencillo sí se usará otro tipo de función, por ejemplo la función Sigmoide.

- El Perceptrón es un modelo de red neuronal de aprendizaje, síncrono y sin retroalimentación.

- Las aplicaciones que se le pueden dar a un perceptrón están limitadas a problemas separables linealmente como la separación de elementos en varias clases