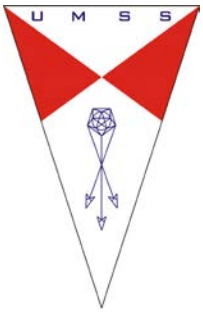
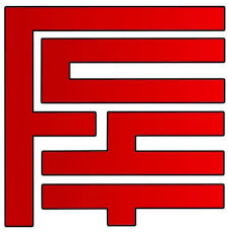


UNIVERSIDAD MAYOR DE SAN SIMÓN

FACULTAD DE CIENCIAS Y TECNOLOGÍA

INFORMATICA - SISTEMAS



INTELIGENCIA ARTIFICIAL I

APRENDIZAJE SUPERVISADO – REDES NEURONALES

RUTINAS PERSONALES PARA LLAMADAS DE EMERGENCIA

ELABORADO POR: Choque Loreño Gerson Egües.

Flores Aserico Noemí.

Rojas Román Joel Rodrigo.

DOCENTE: Msc. Lic. Rodríguez Bilbao Erika Patricia.

PERIODO: I-2019

Cochabamba – Bolivia

INDICE

1 INTRODUCCIÓN

Nuestra vida diaria incluye cientos de hábitos rutinarios, como cepillarnos los dientes, ir a la universidad o guardar los platos, que son solo algunas de las tareas que nuestros cerebros han automatizado hasta el punto de que apenas necesitamos pensar en ellas.

Aunque podemos pensar en cada una de estas rutinas como una tarea única, por lo general se componen de muchas acciones más pequeñas, como levantar nuestro cepillo de dientes, apretar la pasta dental sobre él y luego levantar el cepillo hacia la boca; se podría decir que hay una agrupación de tareas pequeñas para empezar un rutina y que nuestro cerebro llama al cuerpo para realizarlas.

En la actualidad las tendencias informáticas van creciendo cada vez más; por ejemplo, un equipo de investigación del Georgia Tech, en Estados Unidos, ha creado un sistema capaz de analizar imágenes tomadas en primera persona, junto con información contextual relevante, para conocer y predecir las actividades diarias de un individuo. De esta forma se pretende mejorar la manera de entender las rutinas personales y crear herramientas que aprovechen esa información para ofrecer asistencia o ayuda sin necesidad de buscarla.

Este tipo de análisis se basa en métodos de aprendizaje automático profundos basado en aprender representaciones de datos, en el ejemplo anterior seria a partir de imágenes.

En esta práctica se pretende resolver el problema de rutinas personales para llamadas, el análisis que se realiza es aprendiendo el comportamiento del aprendizaje supervisado, utilizando la técnica de redes neuronales existen varios algoritmos, en este caso, el algoritmo de perceptron multicapa; en la práctica se hace uso de representaciones de datos y no así imágenes.

2 OBJETIVO

Presentar la solución al problema de rutinas personales para llamadas aplicando aprendizaje supervisado con el modelo de perceptron multicapa.

3 MARCO TEÓRICO

3.1 APRENDIZAJE SUPERVISADO

El aprendizaje supervisado es el más utilizado.

Incluye algoritmos como regresión lineal y logística, máquinas de vectores de soporte.

Se llama algoritmo supervisado porque el desarrollador actúa como una guía para enseñar al algoritmo las conclusiones a las que debe llegar, es decir la salida del algoritmo ya es conocida; es similar a la forma en que un niño podría aprender de su profesor.

Y que los datos utilizados que tenga para entrenar el algoritmo los tenga etiquetados con las respuestas correctas, por ejemplo una algoritmo de clasificación aprenderá a clasificar animales después de haber sido entrenados con un conjunto de datos de imágenes que están apropiadamente etiquetados con las especies del animal y algunas características de identificación.

Este aprendizaje tiene tres formas de llevarse a cabo:

a) **Por corrección de error.-** Se basa en ajustar los pesos de las conexiones de la red en función de la diferencia entre los valores deseados y obtenidos a la salida según el error realizado. Los algoritmos más conocidos son: perceptron y retropropagación LMS o Multicapa.
b) **Por refuerzo.-**La información facilitada es mínima, estableciendo un límite al indicar si la respuesta de la red es correcta o incorrecta.
c) **Estocástico.-** Se basa en realizar cambios aleatorios en los valores de los pesos y evaluar su efecto a partir del objetivo deseado.

3.1.1 TÉCNICAS

3.1.1.1 REGRESIÓN

Estos algoritmos se usan para predecir valores de salidas basados en algunas características de entradas obtenidas de los datos, a esto el algoritmo construye un algoritmo un modelo basado en las características y los valores de salida en el entrenamiento y este modelo se usa para predecir los valores para nuevos datos.

Los valores de salida en este caso son continuos.

Ejemplos:

- Predecir los precios de la vivienda
- Predecir la cantidad de ingresos que se generan a partir de una nueva campaña de marketing
- Predecir la cantidad de compras
- Predecir si un cliente va a cancelar o no su tarjeta de crédito
- Predecir si un estudiante va a pasar de curso o no.

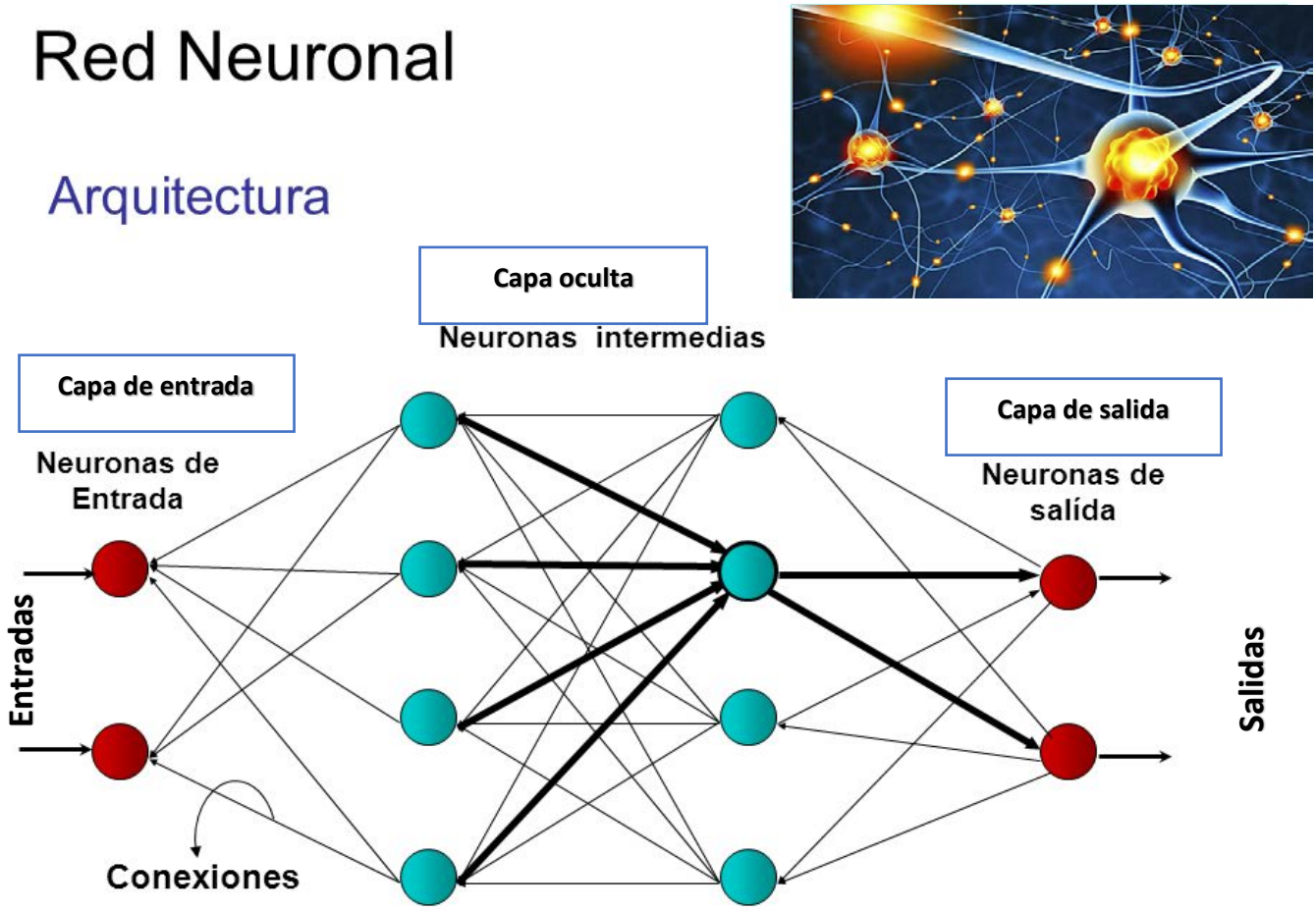
3.1.1.2 REDES NEURONALES

Existen diversas formas de definir a las redes neuronales; desde las definiciones cortas y genéricas hasta las que intentan explicar detalladamente que son las redes neuronales. Por ejemplo:

1) Una nueva forma de computación, inspirada en modelos biológicos.
2) Un modelo matemático compuesto por un gran número de elementos procesales organizados en niveles.

3) Un sistema de computación compuesto por un gran número de elementos simples, elementos de procesos muy interconectados, los cuales procesan información por medio de su estado dinámico como respuesta a entradas externas.
4) Redes neuronales artificiales son redes interconectadas masivamente en paralelo de elementos simples (usualmente adaptativos) y con organización jerárquica, las cuales intentan interactuar con los objetos del mundo real del mismo modo que lo hace el sistema nervioso biológico.

Ejemplo de una red totalmente conectada



a) Capa de entrada**:** Son aquellos nodos que reciben información del exterior
b) Capa oculta: Son aquellos nodos que no tienen ningún contacto con el exterior y solo intercambian información con otros nodos de la red
c) Capa de salida**:** Son aquellos nodos que transmiten la información al exterior

Explicación del funcionamiento de redes neuronales:

En cualquier tipo de red los nodos (neuronas) ocultos se encuentran fuertemente interconectados entre si organizándose por capas ya sea monocapas (si contienen un sola) o multicapas (si contienen más de uno). Estos adquieren un valor de los nodos de entrada que se van modificando en un proceso llamado APRENDIZAJE, existen diferentes algoritmos(en esta práctica se usa un PERCEPTRON MULTICAPA) para demostrar el comportamiento de los nodos ocultos que

dictaran que conexiones serán más o menos significativas de esta manera se toma una decisión sobre que entradas es más importante en base al modelo que se está aplicando; una vez teniendo bien estructurada nuestra red se pasa a la fase de APRENDIZAJE y ENTRANAMIENTO donde por medio de patrones objetivos los valores en los nodos se ajustan de forma iterativa hasta que se demuestren respuestas satisfactorias, es decir que en base a ello la red neuronal sabrá cuando crear, destruir, modificar nodos para dar respuesta óptimas para el conjunto de patrones de entrenamiento. Para este proceso podemos distinguir de tres tipos de APRENDIZAJE SUPERVISADO, aprendizaje no supervisado y aprendizaje hibrido; una vez que la red neuronal esta entrenada puede utilizarse para hacer predicciones o clasificaciones de diversas cosas.

3.2 PERCEPTRON MULTICAPA

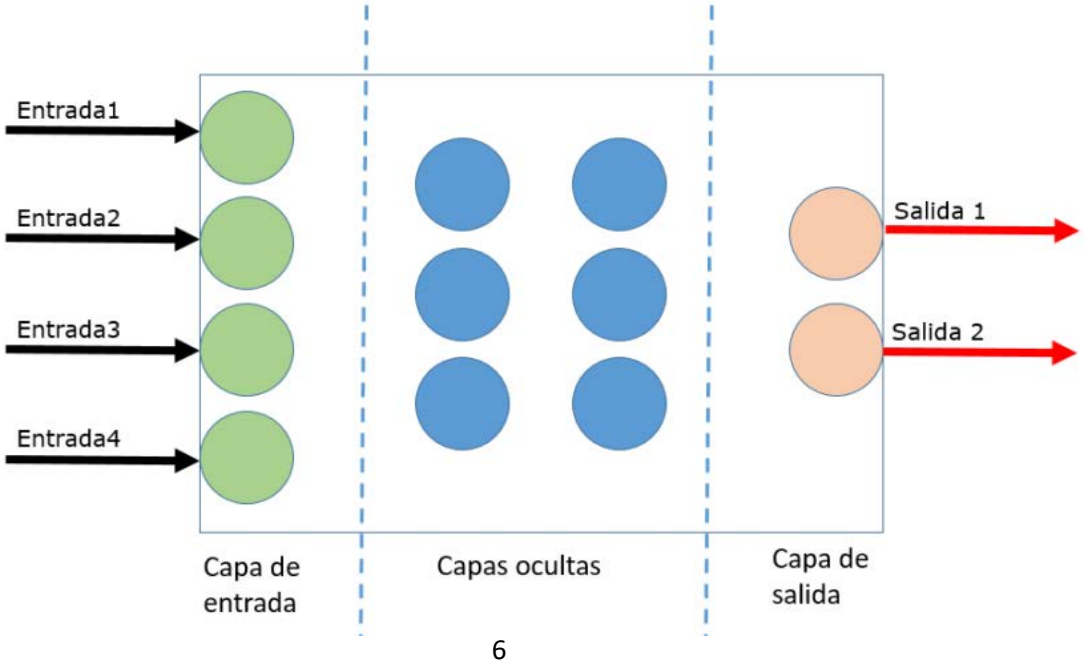
La red neuronal conocida como Perceptron Multicapa es un modelo neuronal con “propagación hacia adelante”, que se caracteriza por su organización en capas de celdas disjuntas, de forma que ninguna salida neuronal constituye una entrada para las neuronas de la misma capa o de capas previas, evitándose así las conexiones “hacia atrás” o “auto recurrentes”(HILERA Y MARTINEZ, 1995).

En el perceptron multicapa se puede diferenciar dos fases:

a) Propagación => en la que se calcula el resultado de salida de la red desde los valores de entrada hacia delante.
b) Aprendizaje => en la que los errores obtenidos a la salida del perceptrón se van propagando hacia atrás **(**backpropagation) con el objetivo de modificar los pesos de las conexiones para que el valor estimado de la red se asemeje cada vez más al real, este aproximación se realiza mediante la función gradiente del error.

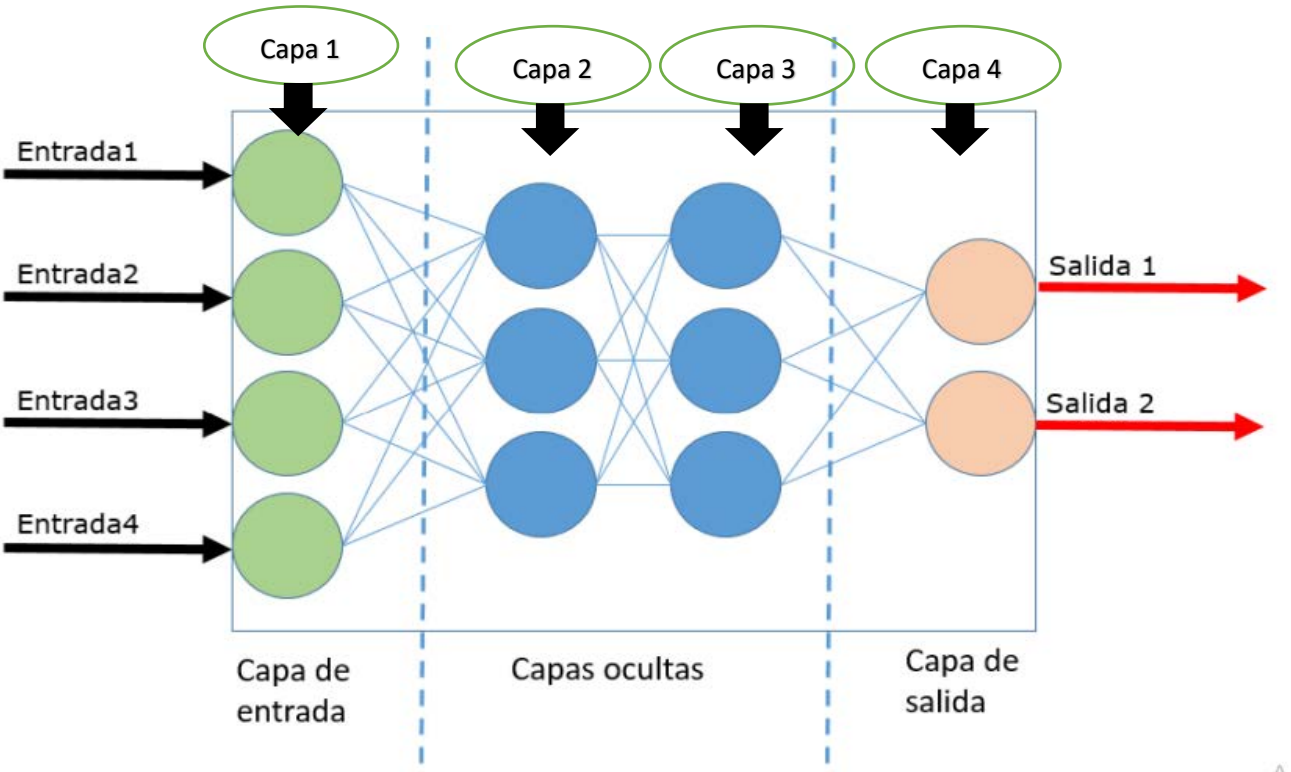
3.2.1 ARQUITECTURA

En la siguiente figura se muestra un ejemplo de un perceptron multicapa



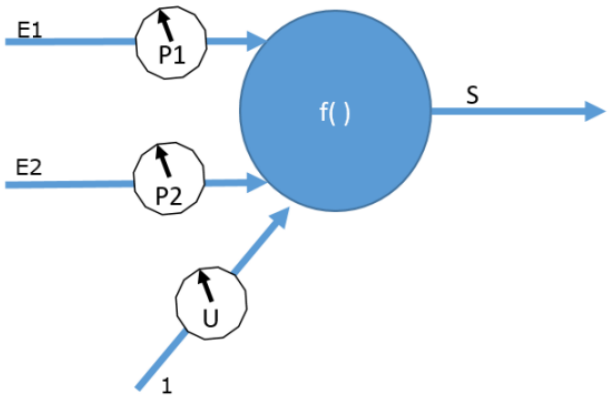
3.2.2 CONEXIONES ENTRE LAS CAPAS DEL PERCEPTRON MULTICAPA

Las neuronas (nodos) de la capa 1 se conectan con las capa 2, las neuronas de la capa 2 se conectan con la capa 3 y así sucesivamente. No se permite conectar neuronas de la capa 1 con neuronas de la capa 4, por ejemplo, ese salto sucede en otros tipos de neuronas en el caso de perceptron multicapa no sucede.



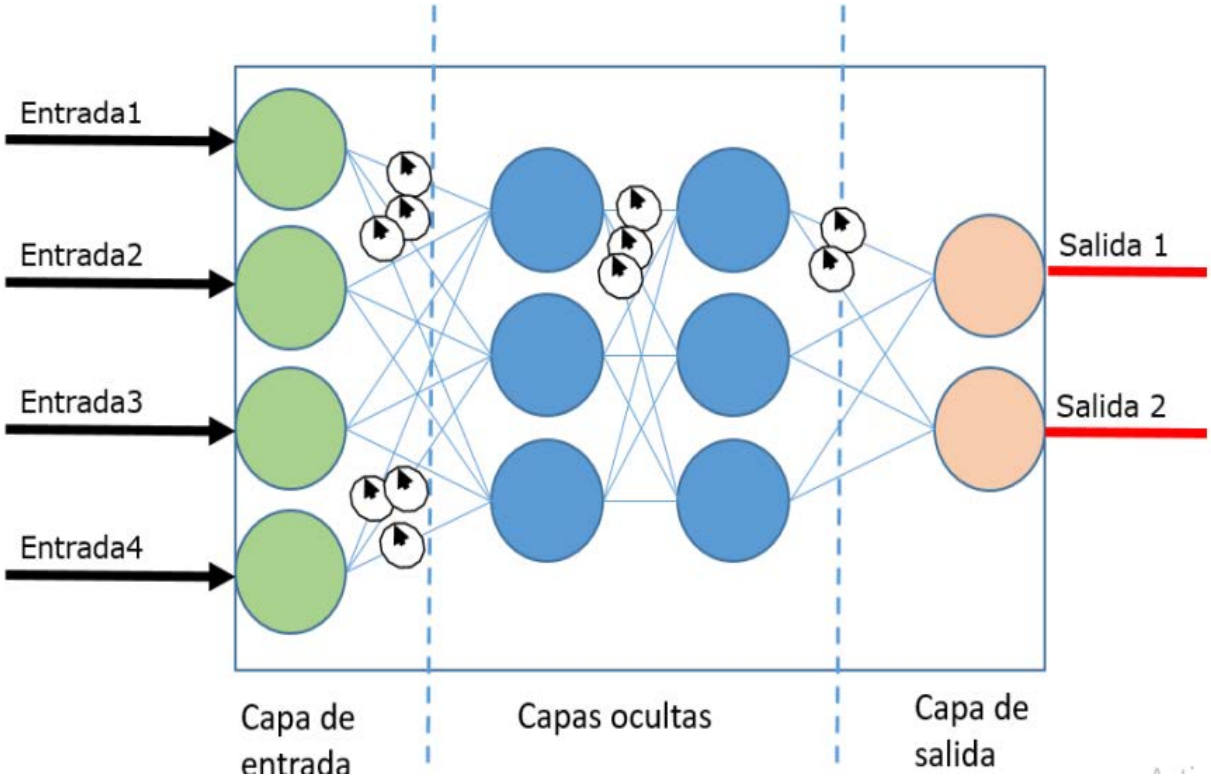
3.2.3 LAS NEURONAS

Se muestra un esquema de cómo es una neurona con dos entradas externas y su salida. En cada entrada hay un peso P1 y P2. Para la entrada interna, que siempre es 1, el peso se llama U.



3.2.4 NOMBRAMIENTO DE LOS PESOS

En el gráfico se dibujan algunos pesos (controles analógicos) y como se podrá dilucidar, el número de estos pesos crece rápidamente a medida que se agregan capas y neuronas.



Un ejemplo: Capa 1 tiene 5 neuronas, capa 2 tiene 4 neuronas, luego el total de conexiones entre Capa 1 y Capa 2 son 5*4=20 conexiones, luego son 20 pesos. Se nombra de la siguiente forma:

$$w^{(\text{capa de donde sale la conexión})}_{\text{neurona inicial,neurona final}}$$

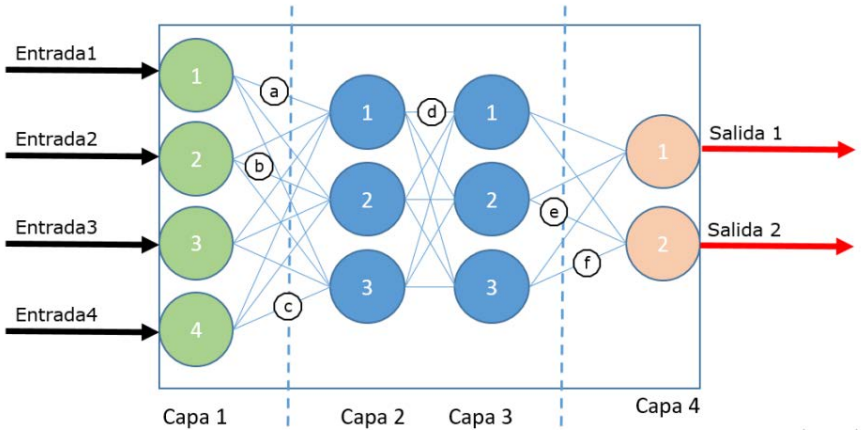
W => Weight(peso)

Capa de donde sale la conexión =>Las capas se enumeran desde 1 que sería en este caso la capa de entrada

Neurona inicial =>De donde parte la conexión

Neurona final =>A donde llega la conexión

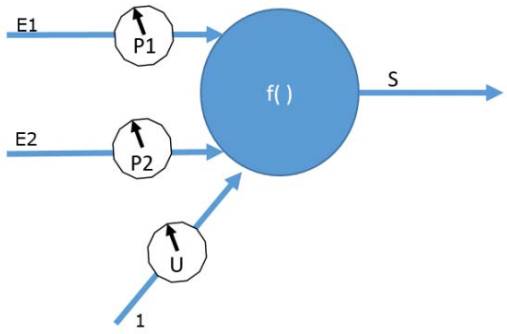
A continuación, se muestra el esquema con cada capa y cada neurona con un número.



En la siguiente tabla se muestra como se nombrarían los pesos que se han puesto en la gráfica.

PESO	NOMBRAMIENTO
a	$W_{1,1}^{(1)}$
b	$W_{2,2}^{(1)}$
c	$W_{4,3}^{(1)}$
d	$W_{1,1}^{(2)}$
e	$W_{2,2}^{(3)}$
f	$W_{3,2}^{(3)}$

3.2.5 LA FUNCIÓN DE ACTIVACIÓN O TRANSFERENCIA DE LA NEURONA



El cálculo de la salida, es: S = f (E1 * P1 + E2 * P2 + 1 * U)

Definición.- La función de activación combina la entrada total a la j-esima neurona o potencial pos-sináptico, obtenido a partir de los estímulos o pesos recibidos (función de propagación), con el estado inicial de la neurona, para producir un nuevo estado de activación de acuerdo con la información recibida. En este problema presentaremos la función sigmoidea

Existen seis tipos de función de activación:

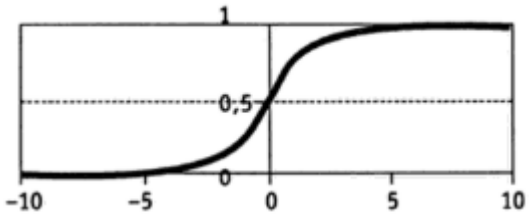
1) La función lineal o identidad
2) La función escalón o signo
3) La función mixta o lineal a tramos
4) La función sigmoidea
5) La función gaussiana
6) La función sinusoidal

Función sigmoide, definida en un determinado intervalo monotónico con limites superiores e inferiores, se caracteriza por presentar una derivada siempre `positiva o igual a 0 en sus límites asintóticos, que toma su valor máximo cuando x=0.

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

Rango= (0,1)

x>0



3.2.6 ALGORITMO DE PROPAGACIÓN HACIA ATRÁS DE ERRORES (BACKPROPAGATION) O RETROPROPAGACION (BP).

Constituye el modelo de aprendizaje de la red perceptron multicapa más utilizado en la práctica, debido a su sencillez y eficacia para la resolución de problemas complejos.

El algoritmo BP está basado en el método de gradiente descendente, que constituye a su vez uno de los métodos de optimización de funciones multivarientes más antiguos y conocidos (CAUCHY, 1847). El gradiente descendente trata de obtener una aproximación lineal de la función de error a través de la expresión:

E (W+W)

$E(W + \Delta W) \approx E(W) + \Delta W' E'(W)$,

De forma que la actualización de pesos viene dada por:

$\Delta w = -\propto E'(W), \propto > 0$

Siendo $\propto$ el tamaño de paso o tasa de aprendizaje, que puede ser constante de tamaño reducido $0 < \propto \leq 1$

4. MARCO REFERENCIAL

El problema rutinas personales para llamadas de emergencia hace referencia a por ejemplo, se tiene a un estudiante que de lunes a viernes realiza una serie de actividades que caen dentro de una rutina semanal, por ejemplo desayuna a la misma hora, va al colegio, retorna del colegio, almuerza, toma unas siesta, va a su instituto, vuelve de su instituto, cena, hace su tarea, y posteriormente va a dormir, todo esto se vuelve en una rutina, con la variante de que puede desayunar distintos tipos de desayunos, almuerzos y cenas, pero al fin y al cabo es una rutina.

En el hipotético caso en que el estudiante salga de su rutina, es señal de que algo no está bien, que bien puede ser una pequeña variante de un día, o quizá esté ocurriendo alguna desgracia, lo que se quiere resolver con este red neuronal es poder identificar la rutina, en este caso específico del estudiante, y poder en base a su rutina hacer una llamada de emergencia en caso de que el estudiante salga de su rutina.

Podemos asociar a un vector la rutina del estudiante por cada día de la siguiente manera:

rutina = [0,1,2,0,4,5,6,7,0,9,1]

rutina [0] = desayunar (Avena)

rutina [1] = ir al colegio

rutina [2] = volver del colegio

rutina [3] = almorzar (Sopa de Maní)

rutina [4] = siesta

rutina [5] = ir al instituto

rutina [6] = volver del instituto

rutina [7] = hacer tarea

rutina [8] = cenar (Majadito)

rutina [9] = ir a dormir

rutina [10] = resultado esperado para esta rutina = 1

De esta manera podemos tener una serie de rutinas en una matriz que represente la rutina de 10 días de la actividad de un estudiante.

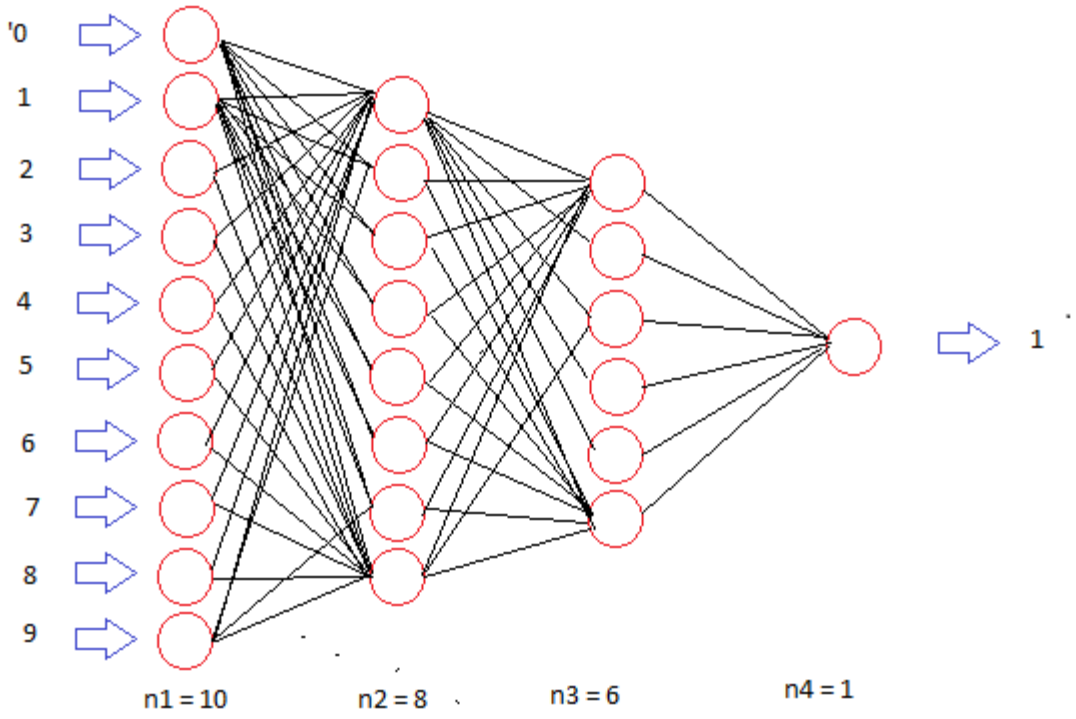
Las variante son en las columnas 0, 3, y 8 que hacen referencia a desayunar, almorzar y cenar respectivamente, la última columna siempre es 1, ya que es el resultado esperado para todo este conjunto de rutina, siempre y cuando el resultado sea una después de una rutina la alarma no

tiene por qué encenderse, de lo contrario si se tiene cualquier otro resultado, la alarma se encenderá ya que en teoría el estudiante se salió de su rutina habitual y no se sabe que está pasando.

[0,1,2,0,4,5,6,7,0,9,1]
[1,1,2,1,4,5,6,7,1,9,1]
[2,1,2,2,4,5,6,7,2,9,1]
[3,1,2,3,4,5,6,7,3,9,1]
[4,1,2,4,4,5,6,7,4,9,1]
[5,1,2,5,4,5,6,7,5,9,1]
[6,1,2,6,4,5,6,7,6,9,1]
[7,1,2,7,4,5,6,7,7,9,1]
[8,1,2,8,4,5,6,7,8,9,1]
[9,1,2,9,4,5,6,7,9,9,1]

Los conceptos que se usaron para resolver este problema son los siguientes:

Un **perceptrón multicapa** que es una red neuronal artificial (RNA) formada por múltiples capas, en este caso específico se usó la siguiente red neuronal:



Donde la entradas son una rutina, y de acuerdo a esta rutina esperamos que la salida sea 1.

Tenemos 2 capas ocultas, las capas 2, y 3, cada una con 8 y 6 neuronas respectivamente, la capa de entrada tiene 10 neuronas, y la capa de salida 1 neurona.

Se usa la función sigmoidea que está dada de la siguiente manera:

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

Se hace uso de artificios matemáticos en base a derivadas parciales que dan como resultado el algoritmo de Back-Propagation.

Calculo del error:

$$e = \frac{1}{2}(s_1 - y_1)^2$$

$$\frac{\partial e}{\partial \blacksquare} = \frac{\partial e}{\partial y_i} \cdot \frac{\partial y_1}{\partial \blacksquare}$$

$$\frac{\partial e}{\partial y_1} = \frac{1}{2} \cdot 2(s_1 - y_1)^{2-1} \cdot (s_1 - y_1)^1 = (s_1 - y_1)(-1)$$

$$\boxed{\frac{\partial e}{\partial y_1} = -(s_1 - y_1)}$$

$$\frac{\partial e}{\partial \blacksquare} = \sum_{i=1}^{n_4=1} -(s_i - i) \cdot \frac{\partial y_i}{\partial \blacksquare} \qquad \blacksquare \begin{matrix} \nearrow U_k \\ \searrow W_{ij} \end{matrix}$$

Modificación de pesos y umbrales:

$$W = W - \propto \frac{\partial e}{\partial W} \;\; ; \propto: 0{,}1 \;\; ; \;\; 0{,}2 \ldots . \Rightarrow Razon\ de\ aprendizaje$$

$$U = U - \propto \frac{\partial e}{\partial U} \;\; ; \propto: 0{,}1 \;\; ; \;\; 0{,}2 \ldots . \Rightarrow Razon\ de\ aprendizaje$$

Derivadas parciales de la salida respecto de los pesos y umbrales:

Pesos: (W)

$$\frac{\partial y_i}{\partial w_{jk}^{(1)}} = a_j^{(1)} \cdot a_k^{(2)}(1-a_k)^2 \cdot \left[\sum_{p=1}^{n_3} W_{kp}^{(2)} a_p^{(3)}\left(1-a_p\right)^3 \cdot W_{p_i}^{(3)}\right] \cdot y_i(1-y_i)$$

$$\frac{\partial y_i}{\partial w_{jk}^{(2)}} = a_j^{(2)} \cdot a_k^{(3)}\left(1-a_k^{(3)}\right) \cdot W_{k_i}^{(3)} y_i(1-y_i)$$

$$\frac{\partial y_i}{\partial w_{ji}^{(3)}} = a_j^{(3)} \cdot y_i(1-y_i)$$

Umbrales: (U)

$$\frac{\partial y_i}{\partial U_j^{(1)}} = a_j^{(2)} \cdot \left(1-a_j^{(2)}\right) \cdot \left[\sum_{p=1}^{n_3} W_{jp}^{(2)} a_p^{(3)} \cdot \left(1-a_p^{(3)}\right) W_{p_i}^{(3)}\right] \cdot y_i(1-y_i)$$

$$\frac{\partial y_i}{\partial U_j^{(2)}} = a_j^{(3)} \cdot \left(1-a_j^{(3)}\right) \cdot W_{ji}^{(3)} y_i(1-y_i)$$

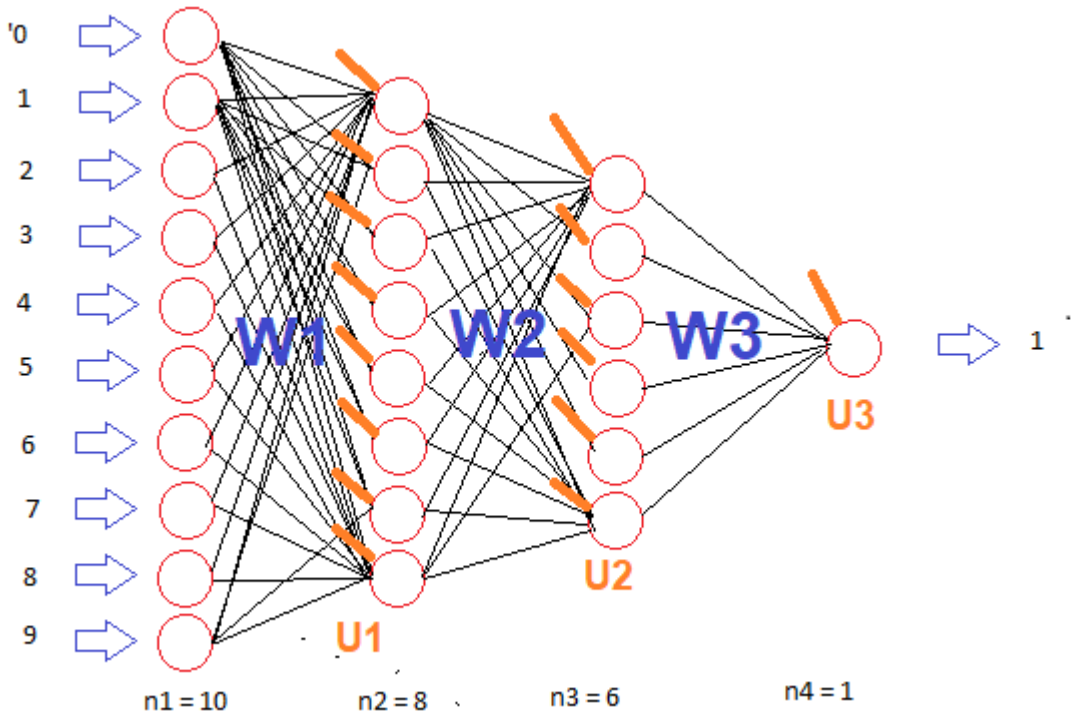
$$\frac{\partial y_i}{\partial U_j^{(3)}} = y_i(1-y_i)$$

5. INGENIERÍA

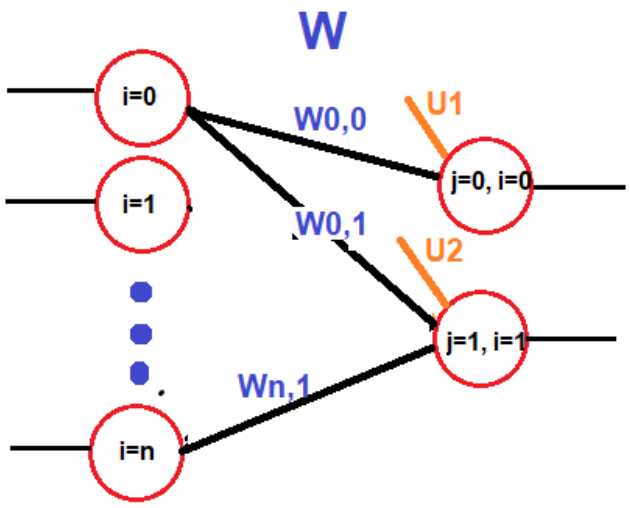
Las entradas para la red neuronal es una matriz, donde cada fila representa una columna distinta, y la última columna de cada fila representa la salida esperada para que la alarma no se active, en este caso tendrá el valor de 1.

```
entrada = []
entrada.append([0,1,2,0,4,5,6,7,0,9,1])
entrada.append([1,1,2,1,4,5,6,7,1,9,1])
entrada.append([2,1,2,2,4,5,6,7,2,9,1])
entrada.append([3,1,2,3,4,5,6,7,3,9,1])
entrada.append([4,1,2,4,4,5,6,7,4,9,1])
entrada.append([5,1,2,5,4,5,6,7,5,9,1])
entrada.append([6,1,2,6,4,5,6,7,6,9,1])
entrada.append([7,1,2,7,4,5,6,7,7,9,1])
entrada.append([8,1,2,8,4,5,6,7,8,9,1])
entrada.append([9,1,2,9,4,5,6,7,9,9,1])
```

Las representaciones los pesos y umbrales se dan de la siguiente manera:



Donde W1, W2, W3, podemos representarlos con matrices 2D de la misma manera los umbrales U1, U2 y U3 podemos representarlo en una matriz 2D, en la siguiente imagen veremos cómo representarlo:



Como vemos en cada neurona tenemos un índice, esto ayudara con la matriz.

Peso W1: (Índices de la matriz)

00	10	20	30	40	50	60	70
10	11	12	13	14	15	16	17
20	21	22	23	24	25	26	27
30	31	32	33	34	35	36	37
40	41	42	43	44	45	46	47
50	51	52	53	54	55	56	57
60	61	62	63	64	65	66	67
70	71	72	73	74	75	76	77
80	81	82	83	84	85	86	87
90	91	92	93	94	95	96	97

Donde:

La fila 0 representa las los pesos de la neurona 0 de la capa N1 hacia las neuronas de la capa N2.

- 0,0 = peso de la neurona i=0 de la capa N1 hacia la neurona j=0 de la capa N2.
- 0,1 = peso de la neurona i=0 de la capa N1 hacia la neurona j=1 de la capa N2
- 0,2 = peso de la neurona i=0 de la capa N1 hacia la neurona j=2 de la capa N2
- ……

La fila 1 representa las los pesos de la neurona 1 de la capa N1 hacia las neuronas de la capa N2.

La fila 2 representa las los pesos de la neurona 2 de la capa N1 hacia las neuronas de la capa N2

Y así sucesivamente, como se puede apreciar tenemos una matriz de 10 x 8 para W1, debido a que en la capa N1 del perceptrón que implementamos tiene 10 neuronas, y la capa N2 tiene 8 neuronas

Peso W2: (Índices de la matriz)

00	10	20	30	40	50
10	11	12	13	14	15
20	21	22	23	24	25
30	31	32	33	34	35
40	41	42	43	44	45
50	51	52	53	54	55
60	61	62	63	64	65
70	71	72	73	74	75

Como se puede apreciar tenemos una matriz de 8 x 6 para W2, debido a que en la capa N2 del perceptrón que implementamos tiene 8 neuronas, y la capa N3 tiene 6 neuronas, se aplica la misma lógica de los pesos de W1

Peso W3: (Índices de la matriz)

00
10
20
30
40
50

Como se puede apreciar tenemos una matriz de 6 x 1 para W3, debido a que en la capa N3 del perceptrón que implementamos tiene 6 neuronas, y la capa N4 tiene 1 neurona, se aplica la misma lógica de los pesos de W1

Umbrales U1, U2, U3. (Índices de los umbrales)

00	01	02	03	04	05	06	07
10	11	12	13	14	15		
20							

Donde cada fila 0, 1 y 2 representa los umbrales de la capa N2, N3, y N4 respectivamente.

Los umbrales de la fila 0 tiene 8 columnas debido a que la capa N2 tiene 8 neuronas, la fila 1 tiene 6 columnas debido a que la capa N3 tiene 6 neuronas, y la fila 2 tiene una columna debido a que la capa N4 tiene 1 neurona.

La explicación anterior permitirá entender el proceso de aprendizaje, tomaremos la primera rutina como ejemplo y veremos el proceso de aprendizaje paso a paso, también se imprimirá los pesos W1, W2, W3 y los umbrales U1, U2 y U3 antes del aprendizaje y después del aprendizaje para ver los cambios.

Antes de continuar explicaremos lo que se denomina escalamiento, debido a que es vital para entender el la solución al problema.

Trabajamos con valores de [0,1], debido a que usamos la función sigmoidea, pero ingresamos valores en el rango [0,9], entonces lo que se hace tanto para entrada como para la salida es escalar cualquier valor dentro del rango [0,9] a un valor que este dentro del rango [0,1] de la siguiente manera:

0 = 0

1= 0.xx

5 = 0.5

6= 0.xx

9 = 1

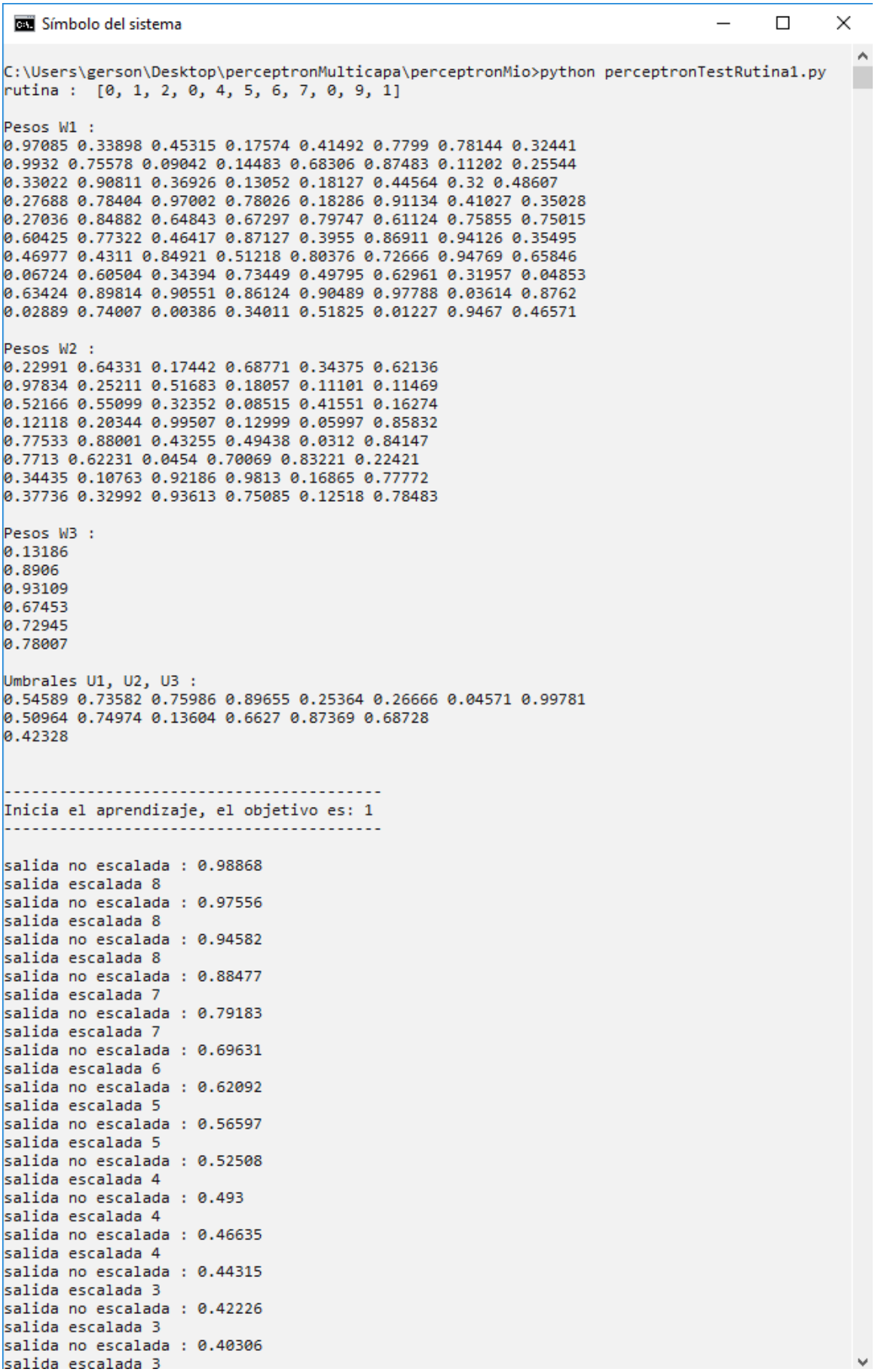
Con la siguiente formula:

Entrada: (Entrada - valor mínimo) / (valor máximo – valor mínimo)

Salida: (Salida * (valor máximo – valor mínimo) + valor mínimo)

Dicho esto pasemos a ver la salida del código:

La razón de aprendizaje para este ejemplo es de: alpha = 0.1



Símbolo del sistema

```
C:\Users\gerson\Desktop\perceptronMulticapa\perceptronMio>python perceptronTestRutina1.py
rutina :  [0, 1, 2, 0, 4, 5, 6, 7, 0, 9, 1]

Pesos W1 :
0.97085 0.33898 0.45315 0.17574 0.41492 0.7799 0.78144 0.32441
0.9932 0.75578 0.09042 0.14483 0.68306 0.87483 0.11202 0.25544
0.33022 0.90811 0.36926 0.13052 0.18127 0.44564 0.32 0.48607
0.27688 0.78404 0.97002 0.78026 0.18286 0.91134 0.41027 0.35028
0.27036 0.84882 0.64843 0.67297 0.79747 0.61124 0.75855 0.75015
0.60425 0.77322 0.46417 0.87127 0.3955 0.86911 0.94126 0.35495
0.46977 0.4311 0.84921 0.51218 0.80376 0.72666 0.94769 0.65846
0.06724 0.60504 0.34394 0.73449 0.49795 0.62961 0.31957 0.04853
0.63424 0.89814 0.90551 0.86124 0.90489 0.97788 0.03614 0.8762
0.02889 0.74007 0.00386 0.34011 0.51825 0.01227 0.9467 0.46571

Pesos W2 :
0.22991 0.64331 0.17442 0.68771 0.34375 0.62136
0.97834 0.25211 0.51683 0.18057 0.11101 0.11469
0.52166 0.55099 0.32352 0.08515 0.41551 0.16274
0.12118 0.20344 0.99507 0.12999 0.05997 0.85832
0.77533 0.88001 0.43255 0.49438 0.0312 0.84147
0.7713 0.62231 0.0454 0.70069 0.83221 0.22421
0.34435 0.10763 0.92186 0.9813 0.16865 0.77772
0.37736 0.32992 0.93613 0.75085 0.12518 0.78483

Pesos W3 :
0.13186
0.8906
0.93109
0.67453
0.72945
0.78007

Umbrales U1, U2, U3 :
0.54589 0.73582 0.75986 0.89655 0.25364 0.26666 0.04571 0.99781
0.50964 0.74974 0.13604 0.6627 0.87369 0.68728
0.42328


----------------------------------------
Inicia el aprendizaje, el objetivo es: 1
----------------------------------------

salida no escalada : 0.98868
salida escalada 8
salida no escalada : 0.97556
salida escalada 8
salida no escalada : 0.94582
salida escalada 8
salida no escalada : 0.88477
salida escalada 7
salida no escalada : 0.79183
salida escalada 7
salida no escalada : 0.69631
salida escalada 6
salida no escalada : 0.62092
salida escalada 5
salida no escalada : 0.56597
salida escalada 5
salida no escalada : 0.52508
salida escalada 4
salida no escalada : 0.493
salida escalada 4
salida no escalada : 0.46635
salida escalada 4
salida no escalada : 0.44315
salida escalada 3
salida no escalada : 0.42226
salida escalada 3
salida no escalada : 0.40306
salida escalada 3
```

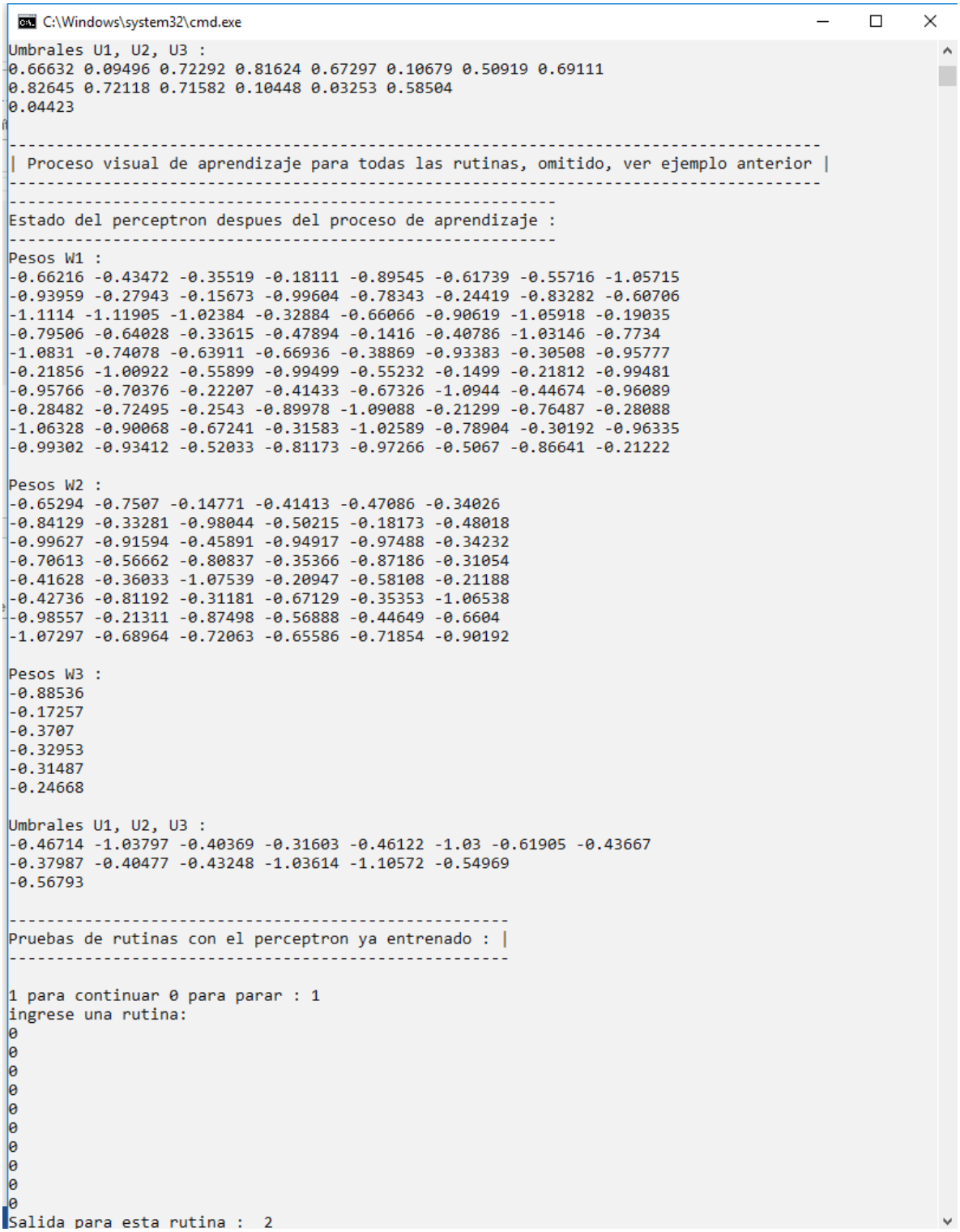
Símbolo del sistema

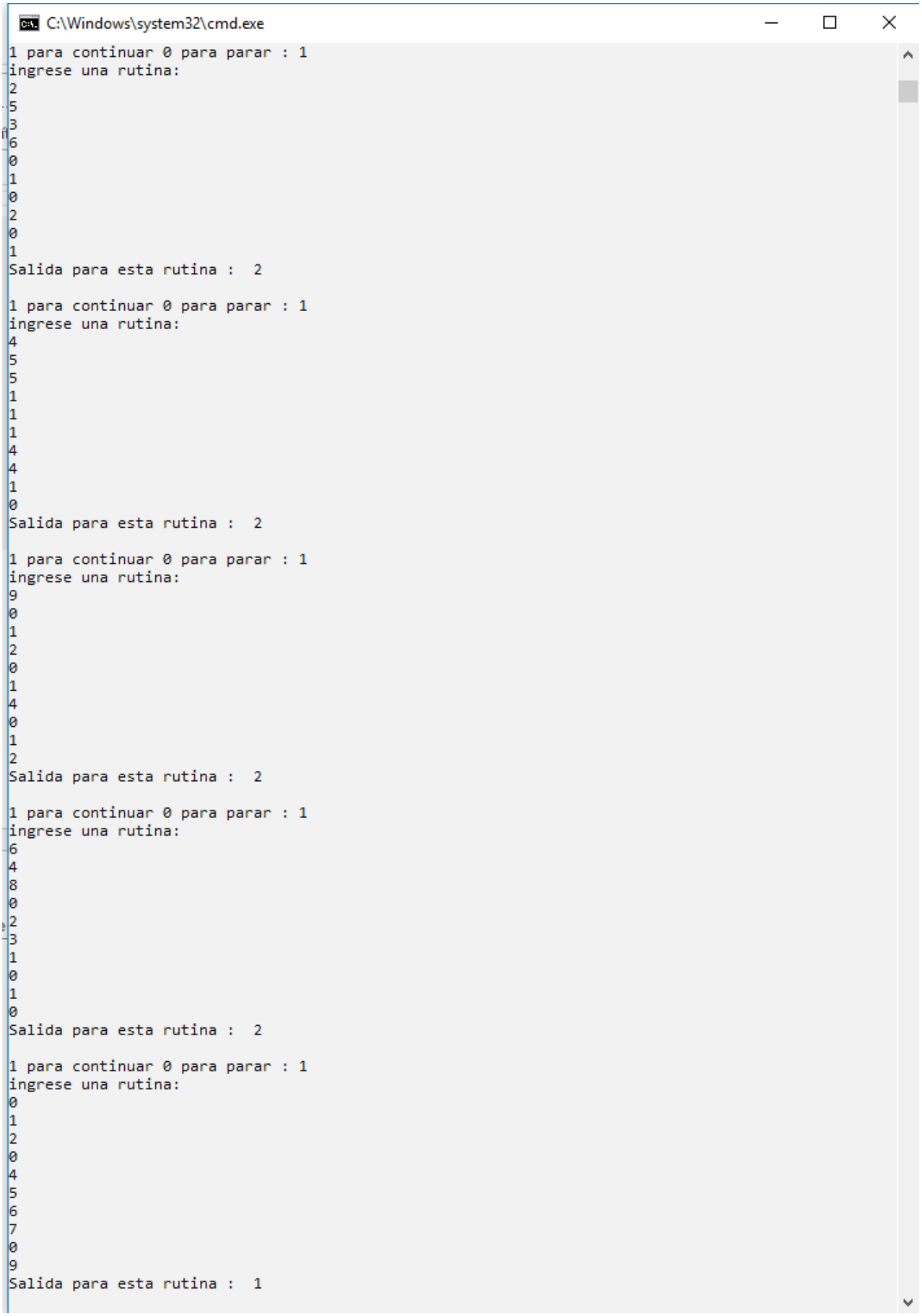
```
salida no escalada : 0.38521
salida escalada 3
salida no escalada : 0.36858
salida escalada 3
salida no escalada : 0.35306
salida escalada 3
salida no escalada : 0.33862
salida escalada 3
salida no escalada : 0.32522
salida escalada 2
salida no escalada : 0.31281
salida escalada 2
salida no escalada : 0.30135
salida escalada 2
salida no escalada : 0.29078
salida escalada 2
salida no escalada : 0.28104
salida escalada 2
salida no escalada : 0.27207
salida escalada 2
salida no escalada : 0.26382
salida escalada 2
salida no escalada : 0.25621
salida escalada 2
salida no escalada : 0.24921
salida escalada 2
salida no escalada : 0.24276
salida escalada 2
salida no escalada : 0.23681
salida escalada 2
salida no escalada : 0.23131
salida escalada 2
salida no escalada : 0.22623
salida escalada 2
salida no escalada : 0.22153
salida escalada 1

-----------------------------------------------
Finalizo el aprendizaje, objetivo alcanzado
-----------------------------------------------


Nuevo estado de los Pesos y Umbrales:
-------------------------------------

Pesos W1 :
-0.44494 -1.07682 -0.96265 -1.24005 -1.00088 -0.6359 -0.63436 -1.09138
-0.4223 -0.65984 -1.32518 -1.26976 -0.73241 -0.54095 -1.30299 -1.15962
-1.08497 -0.50741 -1.04614 -1.28289 -1.23391 -0.97015 -1.09423 -0.92825
-1.13891 -0.63176 -0.44578 -0.63554 -1.23293 -0.50445 -1.00552 -1.06551
-1.14421 -0.56633 -0.76664 -0.73798 -0.61711 -0.80453 -0.65419 -0.66273
-0.81001 -0.64184 -0.9507 -0.53849 -1.01879 -0.54666 -0.47071 -1.0572
-0.94421 -0.98375 -0.56549 -0.89634 -0.61022 -0.6891 -0.4635 -0.75293
-1.34646 -0.80966 -1.07056 -0.67284 -0.91576 -0.78615 -1.09085 -1.36216
-0.78157 -0.51767 -0.51029 -0.55455 -0.51091 -0.43792 -1.37966 -0.5396
-1.38418 -0.67434 -1.41025 -1.06479 -0.89485 -1.40349 -0.46222 -0.94353

Pesos W2 :
-1.19132 -0.76794 -1.23656 -0.72655 -1.06928 -0.7918
-0.456 -1.15338 -0.8875 -1.23349 -1.30044 -1.29579
-0.90316 -0.85821 -1.08512 -1.32873 -0.99668 -1.24932
-1.30977 -1.20315 -0.41061 -1.28386 -1.35159 -0.55254
-0.65055 -0.5286 -0.9755 -0.91943 -1.38082 -0.5703
-0.6533 -0.78698 -1.36337 -0.71319 -0.58001 -1.18791
-1.08509 -1.2995 -0.48446 -0.43247 -1.24298 -0.63339
-1.05027 -1.07794 -0.47102 -0.66292 -1.28663 -0.62658
```

```
Símbolo del sistema

Pesos W3 :
-1.07739
-0.31865
-0.27816
-0.53472
-0.4798
-0.42918

Umbrales U1, U2, U3 :
-0.86715 -0.67684 -0.65352 -0.50711 -1.15942 -1.14903 -1.36271 -0.4111
-0.99523 -0.66208 -1.26934 -0.77692 -0.55938 -0.73954
-0.39376


C:\Users\gerson\Desktop\perceptronMulticapa\perceptronMio>
```

Ahora se hará correr algunas rutinas que no formaron parte del entrenamiento del perceptrón, estas rutinas correrán sobre el perceptrón ya entrenado, para probar como está funcionando.

```
C:\Windows\system32\cmd.exe

C:\Users\gerson\Desktop\perceptronMulticapa\perceptronMio>python PerceptronRutinaFinal.py
Entrenamiento con las siguientes rutinas:
-----------------------------------------
Rutina  0  : 0 1 2 0 4 5 6 7 0 9 1
Rutina  1  : 1 1 2 1 4 5 6 7 1 9 1
Rutina  2  : 2 1 2 2 4 5 6 7 2 9 1
Rutina  3  : 3 1 2 3 4 5 6 7 3 9 1
Rutina  4  : 4 1 2 4 4 5 6 7 4 9 1
Rutina  5  : 5 1 2 5 4 5 6 7 5 9 1
Rutina  6  : 6 1 2 6 4 5 6 7 6 9 1
Rutina  7  : 7 1 2 7 4 5 6 7 7 9 1
Rutina  8  : 8 1 2 8 4 5 6 7 8 9 1
Rutina  9  : 9 1 2 9 4 5 6 7 9 9 1

-----------------------------------------------------.----
Estado del perceptron antes de iniciar el aprendizaje :
-----------------------------------------------------
Pesos W1 :
0.47525 0.70268 0.78221 0.95629 0.24196 0.52002 0.58025 0.08025
0.19733 0.85744 0.97956 0.14075 0.35358 0.89312 0.30358 0.52936
0.02506 0.01729 0.11133 0.80744 0.47599 0.2311 0.07618 0.945
0.34235 0.49713 0.80125 0.65847 0.9958 0.72954 0.10594 0.36401
0.05242 0.39458 0.49379 0.46582 0.7472 0.20332 0.82829 0.17561
0.91653 0.12556 0.57277 0.13961 0.58317 0.98719 0.91426 0.13753
0.17697 0.43052 0.90854 0.71971 0.46181 0.04262 0.68463 0.17048
0.84933 0.40882 0.87524 0.23367 0.04389 0.924 0.36551 0.84949
0.07412 0.23673 0.465 0.82157 0.11151 0.34837 0.83549 0.17406
0.14018 0.19863 0.60694 0.32063 0.16125 0.63012 0.26193 0.91616

Pesos W2 :
0.49684 0.38003 0.98798 0.7201 0.66268 0.79275
0.30248 0.80068 0.15566 0.63321 0.95311 0.65458
0.1656 0.21112 0.67744 0.18453 0.15763 0.7887
0.44479 0.56369 0.32729 0.78047 0.26144 0.82218
0.73214 0.77096 0.06033 0.925 0.55272 0.92147
0.72348 0.31841 0.82382 0.46284 0.77978 0.06733
0.16691 0.91666 0.26072 0.56514 0.68666 0.472
0.08324 0.4389 0.41521 0.47789 0.41427 0.22987

Pesos W3 :
0.04018
0.75297
0.55484
0.59601
0.61067
0.67886
```


C:\Windows\system32\cmd.exe
Umbrales U1, U2, U3 :
0.66632 0.09496 0.72292 0.81624 0.67297 0.10679 0.50919 0.69111
0.82645 0.72118 0.71582 0.10448 0.03253 0.58504
0.04423
| Proceso visual de aprendizaje para todas las rutinas, omitido, ver ejemplo anterior |
Estado del perceptron despues del proceso de aprendizaje :
Pesos W1 :
-0.66216 -0.43472 -0.35519 -0.18111 -0.89545 -0.61739 -0.55716 -1.05715
-0.93959 -0.27943 -0.15673 -0.99604 -0.78343 -0.24419 -0.83282 -0.60706
-1.1114 -1.11905 -1.02384 -0.32884 -0.66066 -0.90619 -1.05918 -0.19035
-0.79506 -0.64028 -0.33615 -0.47894 -0.1416 -0.40786 -1.03146 -0.7734
-1.0831 -0.74078 -0.63911 -0.66936 -0.38869 -0.93383 -0.30508 -0.95777
-0.21856 -1.00922 -0.55899 -0.99499 -0.55232 -0.1499 -0.21812 -0.99481
-0.95766 -0.70376 -0.22207 -0.41433 -0.67326 -1.0944 -0.44674 -0.96089
-0.28482 -0.72495 -0.2543 -0.89978 -1.09088 -0.21299 -0.76487 -0.28088
-1.06328 -0.90068 -0.67241 -0.31583 -1.02589 -0.78904 -0.30192 -0.96335
-0.99302 -0.93412 -0.52033 -0.81173 -0.97266 -0.5067 -0.86641 -0.21222
Pesos W2 :
-0.65294 -0.7507 -0.14771 -0.41413 -0.47086 -0.34026
-0.84129 -0.33281 -0.98044 -0.50215 -0.18173 -0.48018
-0.99627 -0.91594 -0.45891 -0.94917 -0.97488 -0.34232
-0.70613 -0.56662 -0.80837 -0.35366 -0.87186 -0.31054
-0.41628 -0.36033 -1.07539 -0.20947 -0.58108 -0.21188
-0.42736 -0.81192 -0.31181 -0.67129 -0.35353 -1.06538
-0.98557 -0.21311 -0.87498 -0.56888 -0.44649 -0.6604
-1.07297 -0.68964 -0.72063 -0.65586 -0.71854 -0.90192
Pesos W3 :
-0.88536
-0.17257
-0.3707
-0.32953
-0.31487
-0.24668
Umbrales U1, U2, U3 :
-0.46714 -1.03797 -0.40369 -0.31603 -0.46122 -1.03 -0.61905 -0.43667
-0.37987 -0.40477 -0.43248 -1.03614 -1.10572 -0.54969
-0.56793
Pruebas de rutinas con el perceptron ya entrenado :
1 para continuar 0 para parar : 1
ingrese una rutina:
0
0
0
0
0
0
0
0
0
Salida para esta rutina : 2


C:\Windows\system32\cmd.exe
1 para continuar 0 para parar : 1
ingrese una rutina:
2
5
3
6
0
1
0
2
0
1
Salida para esta rutina : 2
1 para continuar 0 para parar : 1
ingrese una rutina:
4
5
5
1
1
1
4
4
1
0
Salida para esta rutina : 2
1 para continuar 0 para parar : 1
ingrese una rutina:
9
0
1
2
0
1
4
0
1
2
Salida para esta rutina : 2
1 para continuar 0 para parar : 1
ingrese una rutina:
6
4
8
0
2
3
1
0
1
0
Salida para esta rutina : 2
1 para continuar 0 para parar : 1
ingrese una rutina:
0
1
2
0
4
5
6
7
0
9
Salida para esta rutina : 1

6. CONCLUSIONES

Con la construcción de una red neuronal podemos asociar semánticamente distintos tipos de entrada a un conjunto de valores y hacer que nuestra RNA aprenda lo que convenga para un conjunto de datos con características similares, lo interesante de una RNA es que en teoría "aprende" pero en palabras sencillas se podría decir que hace es ajustar tanto los pesos y umbrales en base a artificios netamente matemáticos, definiciones matemáticas estudiadas y probadas que permite emular el aprendizaje de una red neuronal humana.

7. REFERENCIAS

[1] Redes neuronales: conceptos básicos y aplicaciones. Universidad Tecnológica Nacional – Facultad Regional Rosario, *Damián Jorge Matich, Marzo de 2001.*

[2] https://medium.com/@juanzambrano/aprendizaje-supervisado-o-no-supervisado-9ccf1fd6e7b

[3] https://es.wikipedia.org/wiki/Aprendizaje_autom%C3%A1tico

[4] https://es.wikipedia.org/wiki/Perceptr%C3%B3n_multicapa

[5] http://avellano.fis.usal.es/~lalonso/RNA/index.htm

[6]https://www.youtube.com/watch?v=jaEIv_E29sk&list=PLAnA8FVrBl8AWkZmbswwWiF8a_52dQ3JQ