INSTITUTO TECNOLOGICO DE TLAXCALA

FACULTAD DE INGENIERÍA

INTELIGENCIA ARTIFICIAL Y REDES NEURONALES ENFOCADO A LA CLASIFICACIÓN DE DATOS

P R O Y E C T O

PRESENTA:

ROSENDO FLORES ZAPATA

ASESOR:

TLAXCALA, TLAX, JULIO 2019.

Introducción

El cerebro es el órgano más enigmático del cuerpo humano, el cual establece la forma en la que percibimos las imágenes, sonidos, sabores etc. Nos permite almacenar información, crear ideas, recuerdos de nuestra vida, e incluso experimentar emociones, sensaciones y al mismo tiempo poder comunicarnos.

Si los seres humanos no contáramos con este órgano, seriamos organismos primitivos que solo podríamos responder por reflejos e instintos, el cerebro es el que nos hace ser distinto de todas las especies vivientes, es el que nos hace ser inteligentes.

Durante años la humanidad ha soñado con fabricar maquinaria con inteligencia como la nuestra capaz de resolver distintos problemas a la vez.

Erigir estas maquinarias artificialmente inteligentes obliga a la humanidad a resolver el problema más enigmático de la programación. La forma de atacar y resolver estos problemas se le conoce como inteligencia artificial.

Contenido de investigación

Tabla de ilustraciones

Objetivo

Conocer la estructura, aplicaciones, funciones como el comportamiento, predicción, clasificación de inteligencia artificial y redes neuronales en la actualidad. Realizando experimentos para comprender más sobre estos temas.

Justificación

Este proyecto es una sinopsis de inteligencia artificial y redes neuronales, la información es dirigida al área de la computación.

La inteligencia artificial en la actualidad tiene un gran número de aplicaciones. El ser humano se ha adaptado a vivir con esta herramienta debido a que facilita algunas actividades cotidianas.

La función de esta investigación es aportar conocimiento no solo a mi si no a quien consulte este documento.

Hipótesis

La inteligencia artificial está íntimamente relacionada con la inteligencia humana. Esto se demostrará principalmente con la investigación y la implementación de una red neuronal.

Capítulo I
Inteligencia artificial

Inteligencia artificial.

La inteligencia artificial o mejor conocida como “**IA**”, es una disciplina científica-informática que se concentra en la creación de algoritmos y mecanismos que muestran un nivel de inteligencia mayor a la de un ser humano.

Un sistema con inteligencia artificial es capaz de analizar datos en grandes cantidades y por lo tanto formula predicciones de forma automática con rapidez y precisión.

Hoy en día el ser humano sueña con desarrollar máquinas capaces de realizar su trabajo, siendo más eficiente que nosotros.

Casi todas las industrias recurren a algún tipo de sistema de IA para efectuar labores que mejoren su eficiencia. Al mismo tiempo, la IA se abre paso poco a poco en las escuelas, el trabajo, el hogar con las aplicaciones de Smart-phones, tabletas y otros dispositivos.

Cuando la IA entre al funcionamiento cambiara nuestra forma de vida. Aunque sus defensores esperan mejorar las tareas y los procesos laborales, su adaptación generalizada puede alterar la fuerza laboral del planeta.

El trabajo humano es necesario para la mayoría de las tareas, pero es probable que la IA, incluyendo los sistemas y robots gobernados por ella, llegue a remplazarlo.

Pese que la IA puede cambiar a la sociedad y es considerada amenazadora por alguno, son muchos los que siguen entusiasmados con sus posibles aplicaciones.

El Internet es otra tecnología que precisó varios años de desarrollo antes de convertirse en algo esencial en nuestro trabajo y nuestra vida. Aunque la IA lleve décadas investigándose, aún está por despegar. Sin embargo, la tecnología en este campo crece de forma exponencial, y cada vez más compañías encuentran modos de crear productos que la utilizan. En solo cuestión de años, adquirirá mayor relevancia.

Historia de la inteligencia artificial

El origen de la inteligencia artificial (IA) ha conducido a la aparición de aplicaciones que en la actualidad tiene un gran impacto a nuestras vidas. Esta tecnología tiene aproximadamente décadas de existencia.

La idea de que un nivel de inteligencia y conciencia comparable a la humana pueda existir fuera del ser humano lleva siglos considerándose.

Una de sus primeras representaciones conocidas pertenece a la mitología griega. Talos, cuya historia data del 500 a. C., creó un gigantesco autómata (máquina de funcionamiento autónomo) de bronce que protegía la isla de Creta. Según ciertas versiones, recorría la isla acarreando placas de bronce con leyes grabadas para ejercer de juez y resolver las disputas de sus habitantes.

Se puede considerar a los autómatas como los primeros ejemplos de robots y de los intentos de crear maquinas que imitan el comportamiento del humano. Todos utilizaban gran variedad de mecanismos de ingeniería, como alimentación de vapor y muelles de acero que les permitían realizar acciones concretas. Antiguamente, ciertos espectadores llegaban a pensar que tales creaciones albergaban algún tipo de voluntad propia.

El término **“Inteligencia artificial”** se acuño por primera vez en 1956 por **Marvin Minsky, John McCarthy y Claude Shannon** para referirse a la ciencia e ingenio de hacer maquinas inteligentes especialmente crearon programas de cálculo, los tres científicos creían que para la década de los 70s la humanidad estaría rodeada con la IA.

Tras el fiasco de esta teoría las investigaciones sobre la inteligencia artificial se detuvieron. Fue hasta los años 90s cuando las empresas tecnológicas decidieron realizar inversiones gigantescas en este terreno con el fin de mejorar la capacidad de procesamiento y análisis de datos que se generan con el crecimiento digital.

En 1997 IBM demostró que un sistema informático era capaz de vencer al mejor jugador de ajedrez del mundo **Gari Kaspárov**. Esta aplicación se llamó “Deep Blue” y sirvió de base para que la industria tecnológica y la sociedad en general cobrar conciencia de la relevancia de la inteligencia artificial

A partir de ese momento aparecieron los primeros ordenadores digitales en los laboratorios universitarios.

Figura 1 Gari Kaspárov jugando con Deep Blue

Enfoques

Principales sistemas con inteligencia artificial.

1. **Sistemas que se comportan como humanos:** Se encargan de crear maquinas capaces de realizar funciones, se requeriría un humano inteligente la maquina debe poseer estas capacidades.

    - **Procesamiento de lenguaje natural:** Le permitirá comunicarse satisfactoriamente.

    - **Representación del conocimiento.** Para almacenar lo que se conoce o siente.

    - **Razonamiento automático:** Utiliza la información almacenada para responder a preguntas y extraer conclusiones concisas.

    - **Aprendizaje automático:** Se adapta a nuevas circunstancias y detecta patrones.

    - **Visión computacional:** Percibe objetos.

Aprendizaje automático.

EL aprendizaje automático es un tipo de inteligenciar artificial (AI) que proporciona a las computadoras la capacidad de aprender, sin ser programadas explícitamente.

Arthur Samuel fue un pionero estadounidense recordado por la creación de juegos informáticos y la inteligencia artificial, Nombro el término "Machine Learning" en 1959. Evolucionado a partir del estudio de reconocimiento de patrones y la teoría del aprendizaje computacional en la inteligencia artificial, el machine learning explora el estudio y construcción de algoritmos que pueden aprender y hacer predicciones sobre datos.

El proceso de aprendizaje automático es similar a la minería de datos. Ambos sistemas buscan entre los datos para encontrar patrones. Sin embargo, en lugar de extraer los datos para la compresión humana como es el caso de las aplicaciones de minería de datos el **aprendizaje automático** utiliza esos datos para detectar patrones en los datos y ajustar las acciones de programa en consecuencia.

Un ejemplo de aprendizaje automático en la actualidad es el *feed* de noticias de Facebook que personaliza cada perfil. Si un miembro detiene frecuentemente su desplazamiento para leer una noticia o poner like a una foto o publicación de algún otro perfil el *feed* de noticias empezara a mostrar más actividades de ese amigo.

Clasificación de aprendizajes automáticos

Aprendizaje supervisado: Es una técnica para deducir una función a partir de datos de entrenamiento para detectar patrones. Los datos de entrenamiento consisten de pares de objetos (vectores). El objetivo del aprendizaje supervisado es el crear una función capaz de predecir el valor correspondiente a cualquier objeto de entrada valida después de haber visto una serie de ejemplos, los datos de entrenamiento. Para ello, tienen que generalizar a partir de4 los datos presentados a las situaciones no vistas previamente.

Aprendizaje no supervisado: Es un método de aprendizaje automático donde un modelo es ajustado a las observaciones. Se distingue del aprendizaje supervisado por el hecho de que no hay un conocimiento a priori. El aprendizaje no supervisado típicamente trata a los objetos de entrada como un conjunto de variables aleatorias, siendo construido un modelo de densidad para el conjunto de datos.

Capítulo II
Redes neuronales artificiales

Red neuronal artificial.

Una de las ramas más destacadas del campo científico de la inteligencia artificial es la que corresponde a las redes neuronales artificiales (RNAs) entendiendo como tales aquellas redes en las que existen elementos procesados de información de cuyas interacciones locales depende el comportamiento del conjunto del sistema.

Las RNA tratan de emular el comportamiento del cerebro humano, caracterizado por el aprendizaje a través de la experiencia y la extracción del conocimiento genérico a partir de un conjunto de datos. Estos sistemas imitan esquemáticamente la estructura neuronal del cerebro, mediante un programa de ordenador.

Las neuronas artificiales se clasifican por capas, estas realizan diferentes tipos de transformaciones a sus entradas estas viajan desde la primera hasta la última.

Neurobiología.

Una neurona típica posee el aspecto y las partes que se muestran en la figura 4. Sin embargo, debemos observar que la imagen no está a escala, el axón alcanza un largo típico de centímetros y en ocasiones de varios metros, las dendritas y las terminales sinápticas, son más largas, numerosas y tupidas

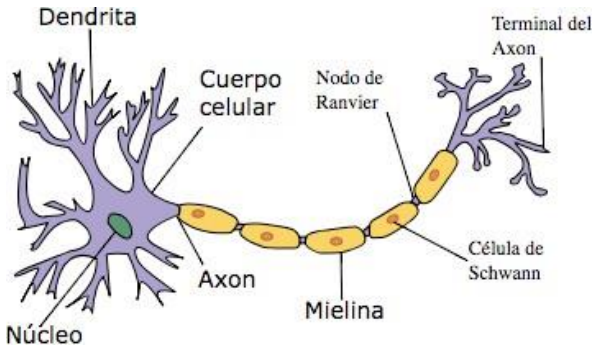

Figura 2 Neurona y sus partes.

Típicamente, las neuronas son 6 o 5 órdenes de magnitud más lentas que una compuerta lógica de silicio, los eventos en un chip de silicio toman alrededor de nanosegundos (10i ^9 s), mientras que en una neurona este tiempo es del orden de los milisegundos (10i^3). Sin embargo, el cerebro compensa en forma excepcional la lentitud relativa en el funcionamiento neuronal con un numero inmenso de neuronas con interconexiones masivas entre ellas.

La mayoría de las neuronas codifican sus salidas como una serie de breves pulsos periódicos, llamados *potenciales de acción,* que se originan cercanos al soma de la célula y se propaga a través del axón. Luego, este pulso llega a la sinapsis y de ahí a las dendritas de la neurona siguiente.

Una *sinapsis* es una interconexión entre dos neuronas, en la imagen de abajo se representa. En ella el botón sináptico corresponde al termino del axón de una neurona pre-sináptica.

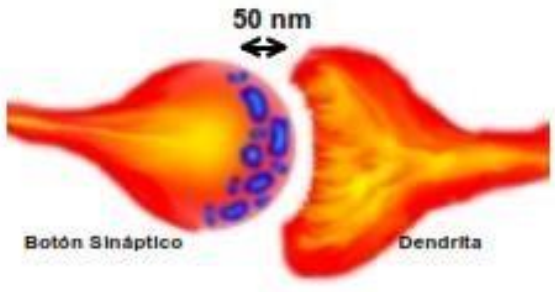

Figura 3 Salto sináptico.

Elementos de una red neuronal artificial.

Entradas: Estas capas reciben la información desde el exterior (Entradas de la neurona).

Pesos: Normalmente una neurona recibe muchas y múltiples entradas simultáneas. Cada entrada tiene su propio peso relativo el cual proporciona la importancia de la entrada dentro de la función de agregación de la neurona. Es pesos realizan la misma función que realizan las fuerzas sinápticas de las neuronas biológicas. En ambos casos, algunas entradas son más importantes que otras de manera que tiene mayor efecto sobre el procesamiento de la neurona al combinarse para producir la respuesta neuronal. Los pesos son coeficientes que pueden adaptarse dentro de la red que determinan la intensidad de la señal de entrada registrada por la neurona artificial.

Salidas: Cada elemento de procesamiento tiene permitido una única salida que puede estar asociada con un número elevado de otras neuronas. Normalmente, la salida es directamente equivalente al valor resultante de la función de activación.

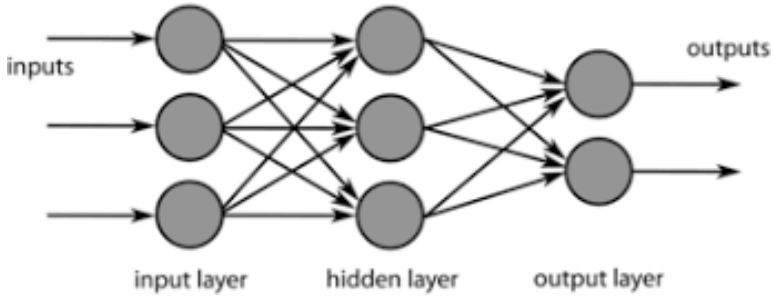

Figura 4 Diagrama de una red neuronal artificial.

Perceptrón.

El modelo biológico más simple de un perceptrón es una neurona y viceversa. Es decir, el modelo matemático más simple de una neurona es un perceptrón. La neurona es una célula especializada y caracterizada por poseer una cantidad indefinida de canales de entrada llamados dendritas y un canal de salida llamado axón. Las dendritas operan como sensores que recogen información de la región donde se hallan y la derivan hacia el cuerpo de la neurona que reacciona mediante una sinapsis que envía una respuesta hacia el cerebro, esto en el caso de los seres vivos.

Una neurona sola y aislada carece de razón de ser. Su labor especializada se torna valiosa en la medida en que se asocia a otras neuronas, formando una red. Normalmente, el axón de una neurona entrega su información como "señal de entrada" a una dendrita de otra neurona y así sucesivamente. El perceptrón que capta la señal en adelante se extiende formando una red de neuronas, sean éstas biológicas o de sustrato semiconductor (compuertas lógicas).

En el perceptrón, existen dos tipos de aprendizaje, el primero utiliza una tasa de aprendizaje mientras que el segundo no la utiliza. Esta tasa de aprendizaje amortigua el cambio de los valores de los pesos.

El algoritmo de aprendizaje es el mismo para todas las neuronas, todo lo que sigue se aplica a una sola neurona en el aislamiento. Se definen algunas variables primero:

- El ***x(j)*** denota el elemento en la posición j en el vector de la entrada.
- El ***w(j)*** el elemento de la posición j en el vector de peso.
- El ***y*** denota la salida de la neurona.
- El ***s*** denota la salida esperada.
- El ***α*** es una constante tal que $0 < \alpha < 1$

El algoritmo de aprendizaje es el mismo para todas las neuronas, todo lo que sigue se aplica a una sola neurona en el aislamiento. Se definen algunas variables primero: Los dos tipos de aprendizaje difieren en este paso. Para el primer tipo de aprendizaje, utilizaremos la siguiente regla de actualización de los pesos:

$$w(j)' = w(j) + \alpha\,(s - y)\,x(j)$$

Se definen algunas variables primero: Para el segundo tipo de aprendizaje, sin utilizar tasa de aprendizaje, la regla de actualización de los pesos será la siguiente.

$$w(j)' = w(j) + (s - y)\,x(j)$$

Perceptrón Multicapa.

El perceptrón multicapa puede ser totalmente o localmente conectado. En el primer caso cada salida de una neurona de la capa “i” es la entrada de todas las neuronas de la capa “i+1”, mientras que en el segundo cada neurona de la capa “i” es entrada de una serie de neuronas (región) de la capa “i+1”

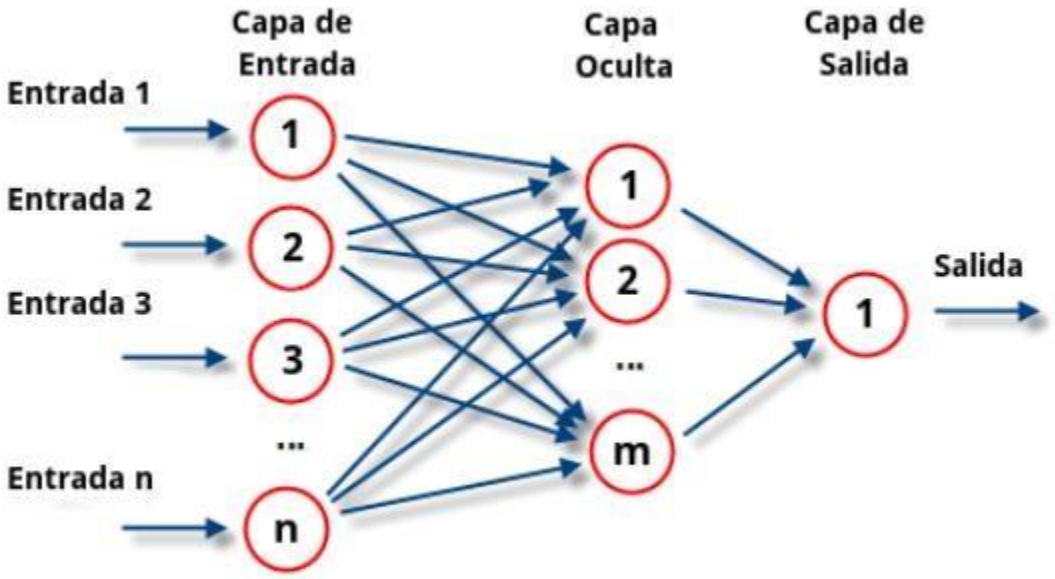

Figura 5 Perceptrón multicapa.

Capas de perceptrón multicapa.

- Capa de entrada: constituida por aquellas neuronas que introducen los patrones de entrada en la red. En estas neuronas no se producen procesamiento.

- Capas ocultas: Formada por aquellas neuronas cuyas entradas provienen de capas anteriores y cuyas salidas pasan a neuronas de capas posteriores.

- Capa de salida: Neuronas cuyos valores de salida corresponden con las salidas de toda la red.

Limitaciones.

- El perceptrón multicapa no explora bien, es decir, si la red se entrena mal o de manera insuficiente, las salidas pueden ser imprecisas.

- La existencia de mínimos locales en la función de error dificulta considerablemente el entrenamiento, pues una vez alcanzado un mínimo el entrenamiento se detiene, aunque no se haya alcanzado la tasa de convergencia fijada.

Experimento de perceptrón.

Objetivo del experimento: Entrenar un perceptrón que pueda clasificar un conjunto de números sin clasificar.

Lo que se hizo, fue crear una clase llamada Perceptrón, de esta manera se podrá crear todos los perceptrones que se necesitan y emplearlos para diferentes objetivos.

Tiene dos entradas y una salida (Categoría p1 o Categoría p2).

Se declara la clase con sus atributos.

```
# Importamos la libreria random
import random

#creamos clase
class Perceptron:
    def __init__(self, sample, exit, learn_rate=0.01, epoch_number=1000, bias=-1)

#atributos de class
        self.sample = sample    #entrenamiento
        self.exit = exit # Salida esperada para cada dato
        self.learn_rate = learn_rate # Que tanto aprendera la red
        self.epoch_number = epoch_number
        self.bias = bias # Bias de la red
        self.number_sample = len(sample)  # Numero de ejemplos
        self.col_sample = len(sample[0])   # Columnas de los datos
        self.weight = []  # Lista de pesos
```

Figura 6 Clase del perceptrón.

Epoch: Cada presentación completa al perceptrón multicapa del set de entrenamiento se denomina epoch. Así, el proceso de aprendizaje se repite epoch tras epoch hasta que los pesos sinápticos se estabilizan y la red converge a un valor aceptable.

```
def sort(self, sample):


    #Se inserta el bias
   # sera una neurona que siempre estara activada.



    sample.insert(0, self.bias)
    u = 0
    for i in range(self.col_sample + 1):

         # Función de activación
        u = u + self.weight[i] * sample[i]

    y = self.sign(u)  # Comprobamos el valor de la función de activación

    if  y == -1:   # Si y es igual a -1, la clasificación corresponde a P1
        print(('Ejemplo: ', sample))
        print('Categoria: P1')
    else:
        print(('Ejemplo: ', sample))
        print('Categoria: P2')
```

Figura 7 Epoch.

El método sort utilizara la función de activación. Luego se comprobará la salida y se le asignará una clasificación a cada uno de los datos. Si y= -1 se asignara a la clasificación P1 de lo contrario, se le asignara la clasificación P2.

```
# Datos de entrenamiento

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

exit = [-1, 1, -1, 1, -1, -1, 1, 1, 1]  # Clasificación de los datos de entrenamiento (salidas que esperamos para cada conjunto de dato)
```

```
# Entrenamos a la neurona
network.trannig()
```

Figura 8 Entrenamiento.

Este es el resultado del perceptrón

```
Inserte un numero >:c :3 : 4
Inserte un numero >:c :3 : 2
('Ejemplo: ', [-1, 4.0, 2.0])
Categoria: P2
Inserte un numero >:c :3 :
```

Figura 9 Funcionamiento del perceptrón.

Capítulo III

Implementación de la red neuronal.

Red neuronal artificial.

El objetivo fue desarrollar una red neuronal de clasificación donde se separe un grupo del otro.

Se desarrolló en Google Colaboratoy el cual es un entorno de notebook Jupyter que no requiere configuración para usar, el código se ejecuta en línea.

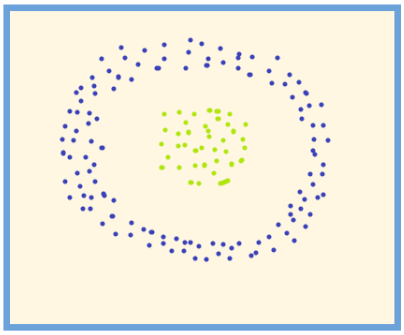

Figura 10 Red neuronal.

Lo primero fue importar librerías como numpy para el procesamiento matemático, scipy que expande las funcionabilidades de numpy, matplotlib.pyplot para gráficos y make_circles para hacer círculos.

```
[1]  import numpy as np
     import scipy as sc
     import matplotlib.pyplot as plt

     from sklearn.datasets import make_circles
```

Figura 11 librerías para la RA.

Una vez importadas las librerías se creó el DATASET se definieron dos variables. Donde “n” se refiere el número de registros que tenemos en nuestros datos “500” y “p” refiere a las características sobre cada uno de los registros “2”.

```
##############DATASET###############

n = 200
p = 2

X, Y = make_circles(n_samples=n, factor=0.5, noise=0.05)

Y = Y[:, np.newaxis]

plt.scatter(X[Y[:, 0] == 0, 0], X[Y[:, 0] == 0, 1], c="skyblue")
plt.scatter(X[Y[:, 0] == 1, 0], X[Y[:, 0] == 1, 1], c="salmon")
plt.axis("equal")
plt.show()
```

Figura 12 Código DATASET.

Ejecutamos el DATASET y este es el resultado.

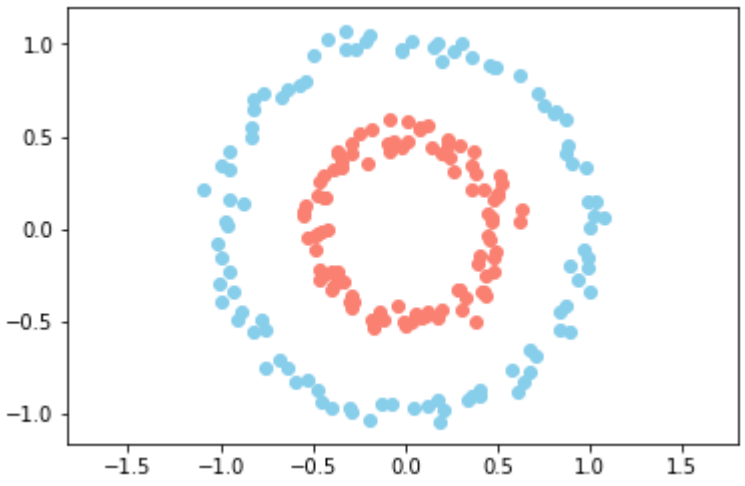

Figura 13 Resultado del DATASET.

Creamos la clase neurona.

```
############ CLASE DE CAPA  ###################

class neural_layer():

  def __init__(self, n_conn, n_neur, act_f):

    self.act_f = act_f

    self.b = np.random.rand(1, n_neur)      * 2 - 1
    self.W = np.random.rand(n_conn, n_neur) * 2 - 1
```

Figura 14 Resultado del DATASET.

El siguiente paso fue crear una función de activación la cual se encarga en devolver una salida a partir de un valor de entrada, normalmente el conjunto de valores de salida en un rango determinado como (0,1) o (-1,1).

En este caso se optó por adaptar la función sigmoide que transforma los valores introducidos a una escala (0,1), donde los valores altos tienen de manera asíntota a 1 y los valores muy bajos tienden de manera asintótica a 0.

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

Figura 15 Función de activación Sigmoide.

```
#############ACTIVACION#######################

sigm = (lambda x: 1 / (1 + np.e ** (-x)),
        lambda x: x * (1 - x))

relu = lambda x: np.maximum(0, x)

_x = np.linspace(-5, 5, 100)
plt.plot(_x, relu(_x))
```

```
[<matplotlib.lines.Line2D at 0x7fcededb0128>]
```

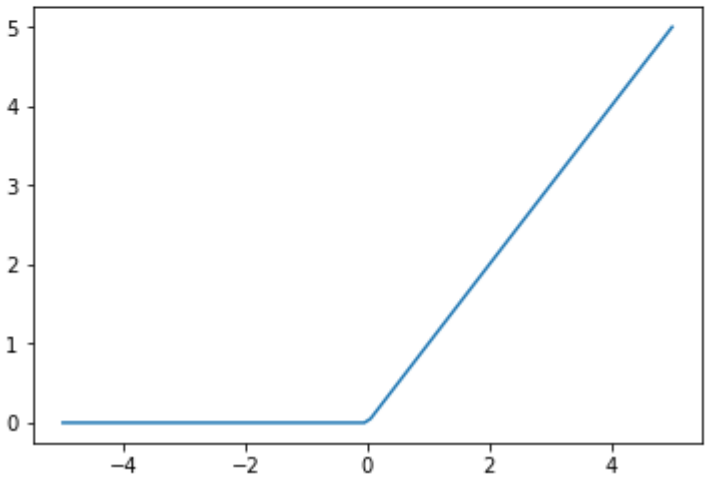

Figura 16 Implementación de la función Sigmoide.

Creamos la red neuronal definiendo el número de neuronas

```
#### RN

l0 = neural_layer(p, 4, sigm)
l1 = neural_layer(4, 8, sigm)


def create_nn(topology, act_f):

  nn = []

  for l, layer in enumerate(topology[:-1]):

    nn.append(neural_layer(topology[l], topology[l+1], act_f))

  return nn
```

Figura 17 Función de activación Sigmoide.

Con ayuda del error cuadrático medio entrenamos a la red. En este caso pide dos valores donde “Yp” (Y predicha) y “Yr” (Y real) calcula la diferencia de cuanto difiere un valor de otro posteriormente elevarlas al cuadrado. Como no solo hay un único elemento en este vector es necesario calcular la media (np.mean)

```
#ENTRENAMIENTO

topology = [p, 4, 8, 1]

neural_net = create_nn(topology, sigm)

l2_cost = (lambda Yp, Yr: np.mean((Yp - Yr) ** 2),
           lambda Yp, Yr: (Yp - Yr))
```

Figura 18 Entrenamiento.

La función de este algoritmo fue tomar el vector de entrada y pasarlo capa por capa ejecutando las operaciones que se realizan en las neuronas. Se utilizó el método de la suma ponderada donde se multiplica el valor de entrada “X” por “W”, después se le suma el parámetro de vallas y por ultimo pasa a la función de activación.

```
# Sum Ponderada
deltas = []

for l in reversed(range(0, len(neural_net))):

  z = out[l+1][0]
  a = out[l+1][1]

  if l == len(neural_net) - 1:
    deltas.insert(0, l2_cost[1](a, Y) * neural_net[l].act_f[1](a))
  else:
    deltas.insert(0, deltas[0] @ _W.T * neural_net[l].act_f[1](a))

  _W = neural_net[l].W

  # Gradient descent
  neural_net[l].b = neural_net[l].b - np.mean(deltas[0], axis=0, keepdims=True) * lr
  neural_net[l].W = neural_net[l].W - out[l][1].T @ deltas[0] * lr
```

Figura 19 Implementación de suma ponderada.

Con este algoritmo hacemos funcionar la red neuronal, llegando al objetivo deseado.

```
import time
from IPython.display import clear_output

neural_n = create_nn(topology, sigm)

loss = []

for i in range(2500):

  #
  pY = train(neural_n, X, Y, l2_cost, lr=0.05)

  if i % 25 == 0:

    print(pY)

    loss.append(l2_cost[0](pY, Y))

    res = 50

    _x0 = np.linspace(-1.5, 1.5, res)
    _x1 = np.linspace(-1.5, 1.5, res)

    _Y = np.zeros((res, res))

    for i0, x0 in enumerate(_x0):
      for i1, x1 in enumerate(_x1):
        _Y[i0, i1] = train(neural_n, np.array([[x0, x1]]), Y, l2_cost, train=False)[0][0]

    plt.pcolormesh(_x0, _x1, _Y, cmap="PuOr")
    plt.axis("equal")

    plt.scatter(X[Y[:,0] == 0, 0], X[Y[:,0] == 0, 1], c="skyblue")
    plt.scatter(X[Y[:,0] == 1, 0], X[Y[:,0] == 1, 1], c="salmon")
```

Figura 20 Prueba de red neuronal.

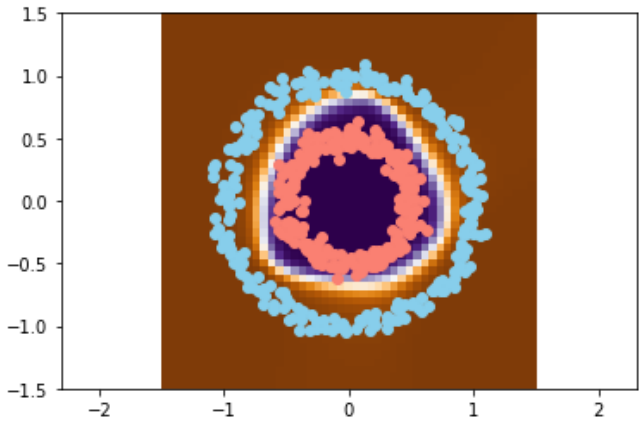

Figura 21 Resultado de la red.

Conclusión

En la sociedad, dentro de las ciencias informáticas la Inteligencia Artificial es una de las áreas que causa más altas expectativas. Un sistema o programa que mejore su comportamiento en base a su experiencia y que, además, tenga una noción de lo que es un error y que a la vez lo evite resulta muy impresionante.

No obstante, la realización de esta investigación me ha servido para darme cuenta que la IA no es una tecnología nueva, lleva décadas de estudio y está en constante evolución.

Por otro lado, me ha parecido apasionante todo lo relacionado con las redes neuronales pues sus aplicaciones son prácticamente infinitas.

Referencias

- Ivan Martínez Ortiz. (2014). Introducción a la Redes Neuronales. Universidad Complutense de Madrid: Facultad de Informática.

- Yann LeCun. (2015). Deep learning. Octubre 01, 2017, de Nature International Journal of Science Sitio web: https://www.nature.com/articles/nature14539

- Desconocido. (2018). Fundamentos de las redes neuronales. 2018, de Desconocido Sitio web: https://thales.cica.es/rd/Recursos/rd98/TecInfo/07/capitulo2.html

- Miquel Barceló García. (2015). Inteligencia Artificial. España: UOC La Universidad Virtual (eBook).

- Anónimo. (2017). GPGPU. Noviembre 12, 2017, de Wikipedia Sitio web: https://es.wikipedia.org/wiki/GPGPU

- Yann LeCun. (2015). Deep learning. Octubre 01, 2017, de Nature International Journal of Science Sitio web: https://www.nature.com/articles/nature14539

- Henry J. Kelley. (1960). Gradient Theory of Optimal Flight Paths. Octubre 20, 2017, de Aerospace Research Central Sitio web: https://arc.aiaa.org/doi/abs/10.2514/8.5282?journalCode=arsj