Área Académica Ingeniería en Mecatrónica

Curso: Inteligencia Artificial

Tarea 2: Estudio Profundo de Sistemas Conexionistas

Profesor:

Juan Luis Crespo Mariño

Estudiantes (Grupo 12):

David Quesada Solís, 2015100428

Braulio Rojas Montenegro 2015073567

II Semestre

2020

Índice

Introducción 1

Descripción del problema 1

Metodología 2

Desarrollo 3

Descripción básica de aplicación 3

Consideraciones para el cálculo y construcción de las expresiones matemáticas 4

Desarrollo de aplicación 4

Selecciones del Usuario en base a los requerimientos del proyecto 4

Importación datos 5

Set de Datos para entrenamiento del modelo 6

Creación de la Neurona 6

Tasas de Aprendizaje 7

Gráficas de Resultados 8

Exportación de datos 9

Experimentos sobre el rendimiento de redes 9

Objetivos de las pruebas 10

Factores de influencia y variables de respuesta 10

Pruebas y rangos de prueba 11

Número de neuronas ocultas 12

Número de iteraciones de entrenamiento 12

Función de activación 12

Valor de tazas de aprendizaje 12

Porcentaje de datos destinados a entrenamiento y pruebas de validación 12

Definición de pruebas y matrices de diseño 13

Metodología de las pruebas 18

Conclusiones 19

Referencias bibliográficas **20**

Introducción

Descripción del problema

Se requiere realizar una aplicación en Python en la que se pueda entrenar una red neuronal artificial (RNA) tipo perceptrón multicapa de una sóla capa oculta (PM), con una forma de optimización de descenso de gradiente y con una función de pérdida L2 [1]. El resto de argumentos se pueden configurar por medio de la consola de Python, tal como el número de entradas y salidas, función de activación, entre otros. Todo el desarrollo de la aplicación debe realizarse sin utilizar bibliotecas de Python relacionadas a RNAs.

Metodología

Para mantener un cierto orden durante la ejecución de la tarea, se plantean los siguientes pasos:

1. Determinar los pasos teóricos del algoritmo de cada elemento a programar, por medio de una investigación previa. Estos elementos se describen con detalle en la sección de desarrollo.
2. Encontrar consideraciones importantes que se deban tomar en cuenta al implementar cada algoritmo y estructura de la red neuronal.
3. Establecer la relación entre perceptrones con otros perceptrones, para facilitar su programación posterior.
4. Escribir el código necesario para construir una red neuronal “desde cero”, junto con formas de entrenarlo y evaluarlo, así como importación y exportación de datos.
5. Redactar los objetivos de las pruebas a realizar.
6. Definir las variables de respuesta (objetivo) que se desea estudiar así como la serie de valores a poner a prueba para cada variable, con base a la información recolectada.
7. Definir los factores de influencia que inciden en las variables de respuesta, así como el conjunto de valores a poner a prueba para cada factor, con base a la información recolectada.
8. Redactar una metodología a seguir durante las pruebas, para evitar error humano y para obtener resultados útiles.
9. Programar redes neuronales conexionistas basadas en las características investigadas.
10. Poner en marcha el experimento, teniendo en cuenta los pasos, y apuntando los resultados de forma ordenada.
11. Realizar un análisis de los resultados.

Desarrollo

Descripción básica de aplicación

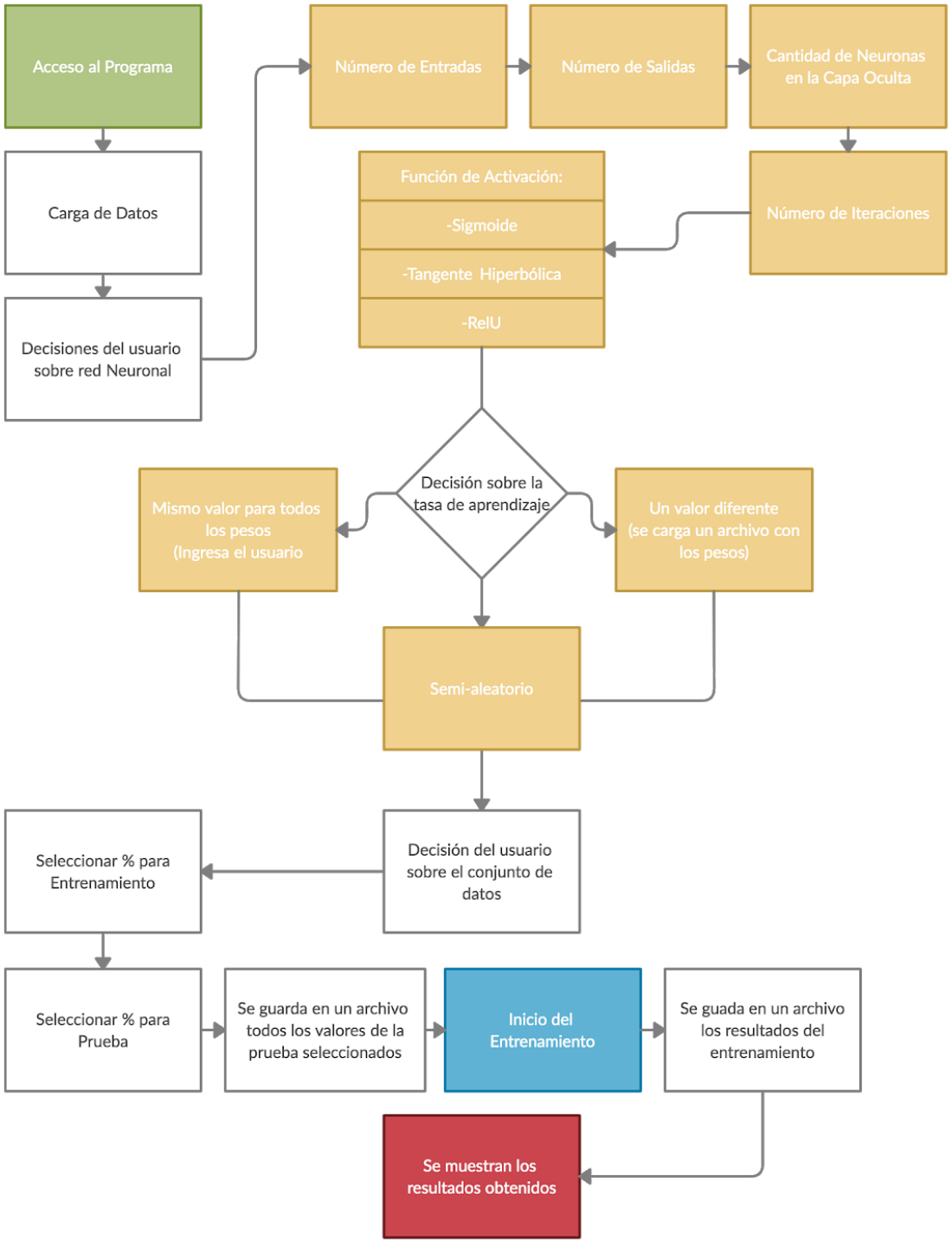

Diagrama 1. Bloques progresivos en la aplicación del Modelo.

Consideraciones para el cálculo y construcción de las expresiones matemáticas

En base a los requerimientos por el proyecto en desarrollo, se establece no usar ninguna otra función que no sea absolutamente básica: sólo accesos a memoria, operaciones aritmético lógicas, bucles, estructuras de programación, etc.

Por lo tanto la creación de arrays y el manejo de funciones con arrays para el manejo de los valores y actualización de pesos, es clave, en esto se usará la librería “numpy” la cual es fundamental para la computación científica en Python.

Para el manejo de los datos, lectura y escritura se usará la librería “pandas” la cual posee un amplio número de funciones esenciales para el análisis de datos, con extensiones de tipo: (.csv),(.txt),(.xlsx), entre otros.

El uso de la librería “matplotlib” la cual permite la visualización y creación de gráficos en base a los diferentes análisis que se requiera, en nuestro caso, podremos demostrar de manera gráfica las curvas de aprendizaje en base a las iteraciones de los entrenamientos en cuestión.

La función de activación estará determinada por el usuario, lo cual implica la necesidad de creación de una clase neuronal, que posea como entrada, la función de activación requerida de acuerdo a la solicitud así como la suma de productos de las entradas y el valor de sesgo necesario en la evaluación de la función de activación a nivel de cada neurona.

Desarrollo de aplicación

Librerías utilizadas en el desarrollo de la aplicación

Librerías

```
[ ]  import os
     import matplotlib.pyplot as pyplot
     import numpy as np
     import pandas as pd
```

Figura 1. Librerías Utilizadas

Selecciones del Usuario en base a los requerimientos del proyecto

Se solicita el ingreso de los datos referentes a la red y el entrenamiento para llevar a cabo un desarrollo óptimo de la aplicación, estos van a ser ingresados por medio del teclado en la interfaz del programa, tales como el número de neuronas en la capa oculta, el número de iteraciones y la función de activación.

```
Selecciones del usuario
-Seleccionar el número de neuronas en la capa oculta.
Valor :8
-Seleccionar el número de iteraciones.
Valor :200
-Seleccionar la función de activación
1. Relu
2. Tangente Hiperbólica
3. Sigmoide
Selección :3
...Los datos se han guardado correctamente...
```

Figura 2. Entradas del usuario

Importación datos

La importación de datos en el desarrollo del proyecto es clave, ya que estos van a representar los porcentajes de información referentes a entrenamientos y pruebas, el desarrollo del mismo permite verificar las entradas y las salidas del archivo csv, así como la separación de datos para llevar a cabo la tarea.

Obtención de Matrices CSV to arrays, Carga del Set de datos y selección de los valores porcentuales de Train y Test

```
#Lectura del dataset cargado por el usuario

print("Seleccione el set de Datos:")
print("1. Airfoil")
print("2. Space Shuttle")
seleccion = float(input("Selección :"))

if seleccion==1:

    print("Se trabajara con el Set AirFoil:")
    data = pd.read_csv('/content/drive/My Drive/T_2_IA_22020/airfoil_self_noise.csv')
    X, y = data.values[:, :-1], data.values[:, -1]

    #El usuario selecciona los valores porcentuales de cantidad de datos para Entrenamiento y Pruebas
    #Ejemplo, sí el usuario selecciona 25% para Entramiento, se sabe que el 75% restante es para Pruebas
    #El usario ingresa el valor de la variable "valor"
    print("Ingrese el valor porcentual del set de Datos para entrenamiento:")
    valor = float(input("Porcentaje :"))
    a,=y.shape
    b,c=X.shape
    d=1
    tamaño=float(a)
    ventrenamiento=int(tamaño*valor*0.01)
    X_train=data.values[:ventrenamiento,:-1]
    X_test=data.values[ventrenamiento:,:-1]
    Y_train=data.values[:ventrenamiento,-1]
    Y_test=data.values[ventrenamiento:,-1]
```

Figura 3. Importación de datos para el manejo de vectores

Set de Datos para entrenamiento del modelo

Se poseen dos Set de Datos, para realizar el modelo y el entrenamiento de la red, el usuario tiene la capacidad de decidir sobre cual set de datos trabajar así como el valor porcentual para train y test.

```
Seleccione el set de Datos:
1. Airfoil
2. Space Shuttle
Selección :1
Se trabajara con el Set AirFoil:
Ingrese el valor porcentual del set de Datos para entrenamiento:
Porcentaje :40
Se van a usar : 600 Datos para train
Se van a usar : 901 Datos para test
Cantidad de entradas: 5
Cantidad de salidas: 1
Vectores de trabajo: 1
(600, 6) (901, 6) (600,) (901,)
```

Figura 4. Set de Datos Airfoil.

```
Seleccione el set de Datos:
1. Airfoil
2. Space Shuttle
Selección :2
Se trabajara con el Set SpaceShuttle:
Ingrese el valor porcentual del set de Datos para entrenamiento:
Porcentaje :40
Se van a usar : 8 Datos para train
Se van a usar : 14 Datos para test
Cantidad de entradas: 3
Cantidad de salidas: 1
Vectores de trabajo:
(8, 4) (14, 4) (8,) (14,)
```

Figura 5. Set de Datos Space Shuttle

Creación de la Neurona

El desarrollo de la neurona viene dado por la capacidad de la misma de evaluar la función de activación en base la suma del producto de las entradas, los pesos establecidos y tomando en cuenta el valor de sesgo en los casos que se ameriten.

Se tienen 3 opciones de Neurona en base a la función de activación requerida por el usuario, la selección viene dada por:

- Relu
- Tangente Hiperbólica
- Sigmoide

Creación de la Neurona

```
[ ]  #La Neurona posee una salida que es la evaluación de la función de activación
     #Para este caso se tienen 3 Funciones de activación, por lo tanto pueden haber 3 tipos de Neuronas
     #net=sumadeproductos


     def Neurona(f_activación, net, sesgo):

     #tipo 1=Relu
             if (f_activación==1):
                 if(net-sesgo>=0):
                   return (net-sesgo)

                 else:
                   return 0
     #tipo 2=Tangente Hiperbólica
             if (f_activación==2):
               y= (2/(1+(math.e)^(-2(net-sesgo))))-1
               return y

     #tipo 3=Sigmoide
             if (f_activación==3):
               y= (1/(1-(math.e)^(-(net-sesgo))))
               return y
```

Figura 6. Función de Neurona para el manejo de Redes

Tasas de Aprendizaje

La tasa de aprendizaje también tendrá tres opciones:

- El mismo valor para todos los pesos (el usuario lo introduce desde teclado).

- Un valor diferente para cada peso (se carga un archivo con las tasas)

- Semialeatorio (el usuario introduce un valor desde teclado y se establece para cada peso un valor aleatorio que consiste en la modificación del valor de teclado con un ruido de tipo gaussiano máxima desviación 25 % del valor).

```
# Las tasas de aprendizaje , se deciden de 3 formas

print("Tasas de Aprendizaje")
print("-Seleccionar la opción :")
print("Tipo 1: Mismo valor para todos los pesos")
print("Tipo 2: Semi-aleatorio")
print("Tipo 3: Ingresado por el usuario")
tasadeaprendizaje=input("Selección :")
#tipo 1=mismo valor
if (tasadeaprendizaje==1):
  print("Mismo valor para todos los pesos")
  valorpesostodos=input("Valor :")
  print("...Los datos se han guardado correctamente...")


#tipo 2=semi-aleatorio
if (tasadeaprendizaje==2):
  print("Semialeatorio...con un ruido de tipo gaussiano -máxima desviación 25 % del valor")
  valorsemialeatorio=input("Valor :")
  print("...Los datos se han guardado correctamente...")


#tipo 3=Ingresado por el usuario
if (tasadeaprendizaje==3):
  print("Ingrese el archivo de los pesos en la carpeta con el nombre: pesos.csv")
  data2 = pd.read_csv('/content/drive/My Drive/Tarea2_IA_22020/pesos.csv')
  print("...Los datos se han guardado correctamente...")
```

Figura 7. Desarrollo de las entradas por el usuario en Tasas de Aprendizaje

```
Tasas de Aprendizaje
-Seleccionar la opción :
Tipo 1: Mismo valor para todos los pesos
Tipo 2: Semi-aleatorio
Tipo 3: Ingresado por el usuario
Selección :
```

Figura 8. Entradas del usuario para valores de Tasas de Aprendizaje

Gráficas de Resultados

Se determina necesaria la capacidad de evaluar de forma visual el rendimiento del entrenamiento del modelo, por lo tanto haciendo uso de la librería matplotlib se pueden representar las curvas de error y entrenamiento.

Curvas y Gráficas de Error y Entrenamiento

```
#Curvas de error
pyplot.title('Curvas de entrenamiento')
pyplot.xlabel('Epocas')
pyplot.ylabel('Error')
```

Figura 9. Plot de las curvas de error del modelo

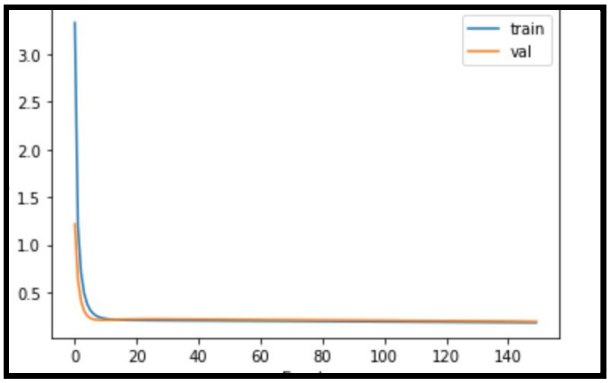

Figura 10. Representación visual de las Tasas

Exportación de datos

Exportación de Archivos de Salida

```
#Guardar dataframe como un nuevo csv
result_data.to_csv('ResultadodePrueba_1.csv')
```

Figura 11. Exportación de datos de salida en un archivo .csv

Experimentos sobre el rendimiento de redes

Para poder determinar el rendimiento de la red que se desea construir [1], se pueden realizar ciertas pruebas que permitan obtener resultados relevantes. Para ello, se utiliza la teoría encontrada en [2] sobre conceptos y planeación de diseño de experimentos. En primer lugar, se deben definir los objetivos. Seguido a esto, se definen así, las variables respuesta a medir y los factores de influencia. Igualmente, se deben asignar valores a las variables de respuesta y factores de influencia para las pruebas. Por último, se establece un plan de acción para llevar a cabo las pruebas.

Las pruebas se deben realizar sobre 2 conjuntos de datos distintos. De esta forma, se realizarán 2 pruebas para cada conjunto de datos, por cada etapa del proceso de experimentación.

Objetivos de las pruebas

- Determinar la configuración topológica de la red neuronal más adecuada para cada uno de los problemas mencionados previamente.
- Entender el proceso de aprendizaje de una neurona seleccionada aleatoriamente, observando principalmente el peso de una de sus conexiones.
- Estudiar la influencia individual de la asignación de tazas de aprendizaje tanto fijas como semi-aleatorias, según lo especificado en la descripción del problema, sobre el proceso y los resultados del entrenamiento de una red neuronal.

Factores de influencia y variables de respuesta

Dada la descripción del problema [1], se tiene que los únicos parámetros que el usuario puede cambiar son:

- Número de entradas
- Número de salidas
- Número de neuronas ocultas
- Número de iteraciones de entrenamiento

- Función de activación
- Valor de tazas de aprendizaje
- Modo de asignación de tazas de aprendizaje
- Porcentaje de datos destinados a entrenamiento y pruebas de validación.

De forma que se deben elegir factores que correspondan a un número de parámetros del listado anterior para cada prueba a realizar. Ahora bien, respecto a las variables respuesta se tienen que éstas pueden ser, dependiendo de la prueba:

- Valor final de curva de aprendizaje de entrenamiento (VFE)
- Valor final de curva de aprendizaje de validación (VFV)
- Tiempo de entrenamiento total
- Número de iteraciones hasta llegar a VFE

Con respecto a la penúltima variable, se refiere al tiempo que transcurre durante el número de iteraciones de entrenamiento seleccionado. Además, se define la última variable de respuesta de la lista como el número de iteraciones hasta llegar a un rango ±1% del VFE, donde los datos posteriores también se encuentran en este rango. Este rango es aceptable para esta aplicación, ya que según la naturaleza del problema, este no requiere tener mayor exactitud.

Antes de dar lugar a los rangos o valores que tomará cada uno de los factores y las variables, se plantean las pruebas a realizar.

Pruebas y rangos de prueba

Para cumplir con el primer objetivo, se desean encontrar relaciones entre topología de red y variables de respuesta, pero se deben considerar los otros factores, ya que son influyentes de forma equivalente a los parámetros de topología, y pueden variar incluso la influencia de los parámetros de topología sobre las variables. Para no alargar el proceso de experimentación, y permitir una gran cantidad de pruebas, sólo se contempla el modo gaussiano de asignación de taza de aprendizaje, ya que de cierta forma tiene características de los tres modos.

Como se tiene conjuntos de datos fijos, el número de entradas y salidas será único para cada conjunto, se tienen 6 factores cuyos valores pueden cambiar.

Número de neuronas ocultas

El número de neuronas ocultas se elige teniendo en cuenta reglas básicas de selección del número de neuronas ocultas, como los que se mencionan en [3]. De este modo, se eligen 3 valores diferentes para número de neuronas ocultas con base en las siguientes reglas, sin seguirlos estrictamente:

- Un número entre el número de entradas y el número de salidas.
- ⅔ del número de entradas más el número de salidas.
- Un número menor al doble del número de entradas.

Número de iteraciones de entrenamiento

Este número dependerá exclusivamente del número de datos del set que se use, así como el porcentaje de datos destinados a entrenamiento.

Función de activación

Según [1], las funciones de activación disponibles serán:

- Sigmoide
- Tangente hiperbólica
- ReLU

Valor de tazas de aprendizaje

Un valor que comúnmente se utiliza al inicio del entrenamiento de una red neuronal es de 0.001, el cual también se utiliza de forma frecuente como valor base dentro de [4]. Se utilizarán entonces 2 valores de taza de aprendizaje: 0.001 y 0.008 los cuales están cerca del valor dicho, y tienen suficiente diferencia para que la red se comporte de manera distinta, reduciendo la posibilidad de inestabilidad durante el entrenamiento [4].

Porcentaje de datos destinados a entrenamiento y pruebas de validación

Respecto a porcentaje de datos de entrenamiento, [5] indica que un buen punto de partida es usando un 70% de los datos. Por lo tanto, se usará este valor, así como un 60%, para comprobar si para cada uno de los tratamientos, resulta mejor una etapa de validación con mayor cantidad de datos.

Definición de pruebas y matrices de diseño

Durante las pruebas se medirán tanto los factores de cada matriz de diseño, como las 4 variables de respuesta, con las que se mide de cierta forma u otra, el rendimiento de la red neuronal. Por supuesto, también se analizarán las curvas de entrenamiento y validación, en caso de que se dé sobreentrenamiento o subentrenamiento, entre otras consideraciones. En la Tabla 1, se muestran los valores de número de neuronas ocultas a utilizar para el experimento, los cuales se trataron de distribuir uniformemente. En dicha tabla también, se consideró útil observar el comportamiento de la red con un número de neuronas ocultas igual al doble de las entradas no menor como sugiere [3]. Después, en las Tablas 2 y 3 se muestran las matrices de diseño de todas las variables de respuesta para cada conjunto de datos. Se realizará 1 corrida, es decir, un solo entrenamiento por tratamiento, lo que resulta para la primera etapa de pruebas de 72 experimentos individuales.

Tabla 1. Valores para factores de influencia.

Factor de influencia	Valores para pruebas	
	Conjunto de datos 1	Conjunto de datos 2
No. entradas	5	3
No. salidas	1	1
No. neuronas ocultas	3, 6, 10	1, 2, 4

Tabla 2. Matriz de diseño para las primeras pruebas con el conjunto de datos 1.

No. Tr.	A: No. Neuronas Ocultas	B: Func. Activación	C: Taza de aprendizaje base	D: Porcentaje entrenamiento-validación
1	-1: 3	-1: Sigmoide	C-: 0.001	D-: 70%-30%
2	-1: 3	-1: Sigmoide	C-: 0.001	D+: 60%-40%
3	-1: 3	-1: Sigmoide	C+: 0.008	D-: 70%-30%
4	-1: 3	-1: Sigmoide	C+: 0.008	D+: 60%-40%
5	-1: 3	0: Tanh	C-: 0.001	D-: 70%-30%
6	-1: 3	0: Tanh	C-: 0.001	D+: 60%-40%
7	-1: 3	0: Tanh	C+: 0.008	D-: 70%-30%
8	-1: 3	0: Tanh	C+: 0.008	D+: 60%-40%
9	-1: 3	+1: ReLU	C-: 0.001	D-: 70%-30%
10	-1: 3	+1: ReLU	C-: 0.001	D+: 60%-40%
11	-1: 3	+1: ReLU	C+: 0.008	D-: 70%-30%
12	-1: 3	+1: ReLU	C+: 0.008	D+: 60%-40%
13	0: 6	-1: Sigmoide	C-: 0.001	D-: 70%-30%
14	0: 6	-1: Sigmoide	C-: 0.001	D+: 60%-40%
15	0: 6	-1: Sigmoide	C+: 0.008	D-: 70%-30%
16	0: 6	-1: Sigmoide	C+: 0.008	D+: 60%-40%
17	0: 6	0: Tanh	C-: 0.001	D-: 70%-30%
18	0: 6	0: Tanh	C-: 0.001	D+: 60%-40%
19	0: 6	0: Tanh	C+: 0.008	D-: 70%-30%
20	0: 6	0: Tanh	C+: 0.008	D+: 60%-40%
21	0: 6	+1: ReLU	C-: 0.001	D-: 70%-30%
22	0: 6	+1: ReLU	C-: 0.001	D+: 60%-40%
23	0: 6	+1: ReLU	C+: 0.008	D-: 70%-30%
24	0: 6	+1: ReLU	C+: 0.008	D+: 60%-40%

25	+1: 10	-1: Sigmoide	C-: 0.001	D-: 70%-30%
26	+1: 10	-1: Sigmoide	C-: 0.001	D+: 60%-40%
27	+1: 10	-1: Sigmoide	C+: 0.008	D-: 70%-30%
28	+1: 10	-1: Sigmoide	C+: 0.008	D+: 60%-40%
29	+1: 10	0: Tanh	C-: 0.001	D-: 70%-30%
30	+1: 10	0: Tanh	C-: 0.001	D+: 60%-40%
31	+1: 10	0: Tanh	C+: 0.008	D-: 70%-30%
32	+1: 10	0: Tanh	C+: 0.008	D+: 60%-40%
33	+1: 10	+1: ReLU	C-: 0.001	D-: 70%-30%
34	+1: 10	+1: ReLU	C-: 0.001	D+: 60%-40%
35	+1: 10	+1: ReLU	C+: 0.008	D-: 70%-30%
36	+1: 10	+1: ReLU	C+: 0.008	D+: 60%-40%

Tabla 3. Matriz de diseño para las primeras pruebas con el conjunto de datos 2.

No. Tr.	A: No. Neuronas Ocultas	B: Func. Activación	C: Taza de aprendizaje base	D: Porcentaje entrenamiento-validación
1	-1: 1	-1: Sigmoide	C-: 0.001	D-: 70%-30%
2	-1: 1	-1: Sigmoide	C-: 0.001	D+: 60%-40%
3	-1: 1	-1: Sigmoide	C+: 0.008	D-: 70%-30%
4	-1: 1	-1: Sigmoide	C+: 0.008	D+: 60%-40%
5	-1: 1	0: Tanh	C-: 0.001	D-: 70%-30%
6	-1: 1	0: Tanh	C-: 0.001	D+: 60%-40%
7	-1: 1	0: Tanh	C+: 0.008	D-: 70%-30%
8	-1: 1	0: Tanh	C+: 0.008	D+: 60%-40%
9	-1: 1	+1: ReLU	C-: 0.001	D-: 70%-30%
10	-1: 1	+1: ReLU	C-: 0.001	D+: 60%-40%
11	-1: 1	+1: ReLU	C+: 0.008	D-: 70%-30%

12	-1: 1	+1: ReLU	C+: 0.008	D+: 60%-40%
13	0: 2	-1: Sigmoide	C-: 0.001	D-: 70%-30%
14	0: 2	-1: Sigmoide	C-: 0.001	D+: 60%-40%
15	0: 2	-1: Sigmoide	C+: 0.008	D-: 70%-30%
16	0: 2	-1: Sigmoide	C+: 0.008	D+: 60%-40%
17	0: 2	0: Tanh	C-: 0.001	D-: 70%-30%
18	0: 2	0: Tanh	C-: 0.001	D+: 60%-40%
19	0: 2	0: Tanh	C+: 0.008	D-: 70%-30%
20	0: 2	0: Tanh	C+: 0.008	D+: 60%-40%
21	0: 2	+1: ReLU	C-: 0.001	D-: 70%-30%
22	0: 2	+1: ReLU	C-: 0.001	D+: 60%-40%
23	0: 2	+1: ReLU	C+: 0.008	D-: 70%-30%
24	0: 2	+1: ReLU	C+: 0.008	D+: 60%-40%
25	+1: 4	-1: Sigmoide	C-: 0.001	D-: 70%-30%
26	+1: 4	-1: Sigmoide	C-: 0.001	D+: 60%-40%
27	+1: 4	-1: Sigmoide	C+: 0.008	D-: 70%-30%
28	+1: 4	-1: Sigmoide	C+: 0.008	D+: 60%-40%
29	+1: 4	0: Tanh	C-: 0.001	D-: 70%-30%
30	+1: 4	0: Tanh	C-: 0.001	D+: 60%-40%
31	+1: 4	0: Tanh	C+: 0.008	D-: 70%-30%
32	+1: 4	0: Tanh	C+: 0.008	D+: 60%-40%
33	+1: 4	+1: ReLU	C-: 0.001	D-: 70%-30%
34	+1: 4	+1: ReLU	C-: 0.001	D+: 60%-40%
35	+1: 4	+1: ReLU	C+: 0.008	D-: 70%-30%
36	+1: 4	+1: ReLU	C+: 0.008	D+: 60%-40%

En el segundo conjunto de pruebas, se realizará un monitoreo de la actualización de pesos en una neurona en las primeras pruebas, para cumplir con el segundo objetivo. Este análisis se realizará de manera reducida para poder entender con mayor facilidad los cálculos que está realizando la red y en específico, la neurona a estudiar, sin tener que analizar un amplio rango de parámetros. De esta forma, se elige el conjunto de datos 2, en alguno de los casos de 2 neuronas ocultas, que resulte en la combinación que tenga el menor valor de VFE, por 10 iteraciones. Las condiciones específicas de este análisis se redactará una vez obtenido los resultados de las primeras pruebas.

Para terminar con la sección de pruebas, se realizarán experimentos para cumplir con el tercer objetivo. En este sentido, se desea observar únicamente la influencia del modo de asignación de tazas durante el entrenamiento, así como en los resultados. Esto requiere que se analicen las 4 variables de respuesta al mismo tiempo. De manera similar al conjunto de pruebas anterior, se eligen las condiciones de la mejor combinación del primer conjunto de pruebas para el conjunto de datos correspondiente.

En la Tabla 4 se muestra la matriz de diseño de las 4 variables de respuesta para cada conjunto de datos, i.e. el VFE, VFV, el tiempo de entrenamiento y el número de iteraciones hasta llegar al VFV. Se realizarán 3 corridas por tratamiento, es decir, 3 entrenamientos diferentes, para observar patrones que surjan de las 3 corridas (12 experimentos). Cabe notar que se mantendrán también aquellas corridas que no posean un comportamiento adecuado, ya que forman parte del experimento, y pueden tener información de las tazas usadas respecto a la sensibilidad de la red durante el entrenamiento ante éstas.

Tabla 4. Matriz de diseño para las pruebas relacionadas al objetivo 3.

No. Tratamiento	A: Modo tazas de aprendizaje
1	A-: Fijo
2	A+: Semi-aleatorio

Por último, se escribirá un plan de acción antes del desarrollo del experimento, de forma que se mantenga un cierto orden al realizar las pruebas, reduciendo el error humano en la medida de lo posible.

Metodología de las pruebas

Como el proceso se realizará completamente dentro de una computadora se pueden guardar los datos completos de cada experimento en varias hojas de cálculo, por lo que no existe posibilidades de error humano.

Si es importante mencionar que las hojas de cálculo exportadas como resultado de los experimentos tendrán para cada tratamiento, el valor de cada variable de respuesta medida por el mismo programa, y el análisis posterior se puede realizar por medios gráficos utilizando el conjunto de datos generado.

Conclusiones

Se tiene que la implementación de una red neuronal como la que se pide en este proyecto se ve beneficiada de usar el paradigma de programación orientada a objetos, como es el caso de la programación de una clase de python de neuronas, así como una de red neuronal que una todas las neuronas en la forma de perceptrón multicapa. Es una forma de agrupar cada elemento para separar el código en bloques de cada elemento de una red neuronal, y se facilita la lectura del código.

Al lidiar con parámetros de entrada se deben establecer de igual forma rangos de prioridad ya que tienen un mayor impacto en la salida, para poder evaluar el rendimiento del modelo y entender de una manera más conclusa el comportamiento de la red, se debe dar una prioridad dependiendo de los valores de entrada en el sistema, la capacidad de solución del modelo va estar directamente relacionada con la función de activación así como la función de optimización y de pérdida que posea el modelo en cuestión.

Referencias bibliográficas

[1] J. Crespo, *"Tarea 2: Estudio profundo de sistemas conexionistas"*, Costa Rica, 2020.

[2] H. Gutiérrez Pulido and R. de la Vara Salazar, *Análisis y diseño de experimentos*, 3ra ed. México: McGraw-Hill, 2012, p. 2-11.

[3] J. Heaton, *Introduction to neural networks with Java*, 2nd ed. Chesterfield (MO, USA): Heaton Research, Inc., 2008, pp. 158, 159.

[4] L. Smith, "*A DISCIPLINED APPROACH TO NEURAL NETWORK HYPER-PARAMETERS: PART 1 – LEARNING RATE, BATCH SIZE, MOMENTUM, AND WEIGHT DECAY*", JUS Naval Research Laboratory Technical Report, 2018.

[5] J. Crespo, *"Algunas notas básicas sobre operación de redes"*, Costa Rica, 2020.