EXAMEN PARCIAL 2: REDES NEURONALES ARTIFICIALES

Alberto Javier Pelayo Brambila – A01636406

24 DE OCTUBRE DE 2022

PROFESOR: EDGAR MACIAS GARCÍA

Proyecto de Control Avanzado 1

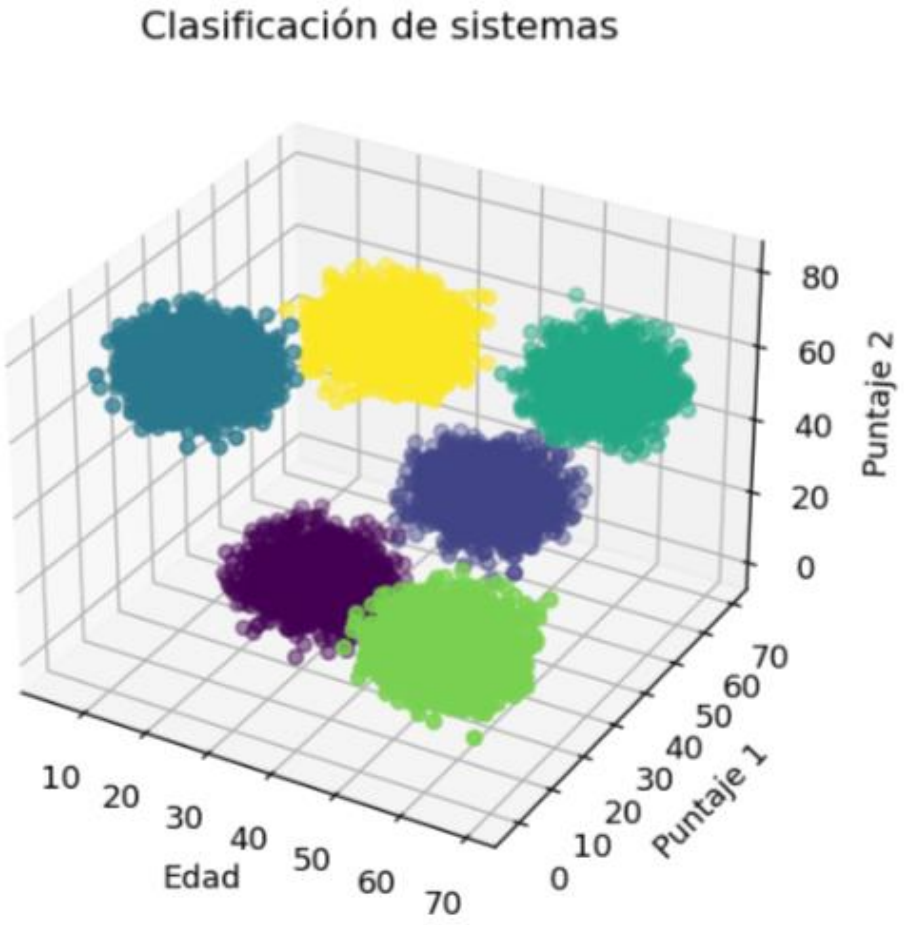
Problema 1: Aplicaciones en sistemas operativos

Análisis de datos

Esteban Trabajos y Bernardo Puentes son dos prestigiosos informáticos que desean competir en el mercado de sistemas operativos contra los gigantes Apple (de Steve Jobs) y Microsoft (de Bill Gates), para lo cual desarrollaron el sistema operativo multiplataforma ”Jihuatsi”(lobo, en la lengua purhepecha).

Uno de los principales atractivos de su sistema operativo es que dispondrá de versiones de software especializadas en función de las necesidades del cliente, el cual podrá elegir (o alternar) entre alguna de las configuraciones de manera gratuita una vez adquirido el sistema.

Para ello, recopilaron información de más de 6000 personas que eligieron entre 6 diferentes tipos de colecciones de software. Se pretende crear un modelo que permita recopilar datos del cliente durante el primer inicio de sesión, a fin de recomendarle la configuración más apropiada para su tipo de experiencia de usuario.



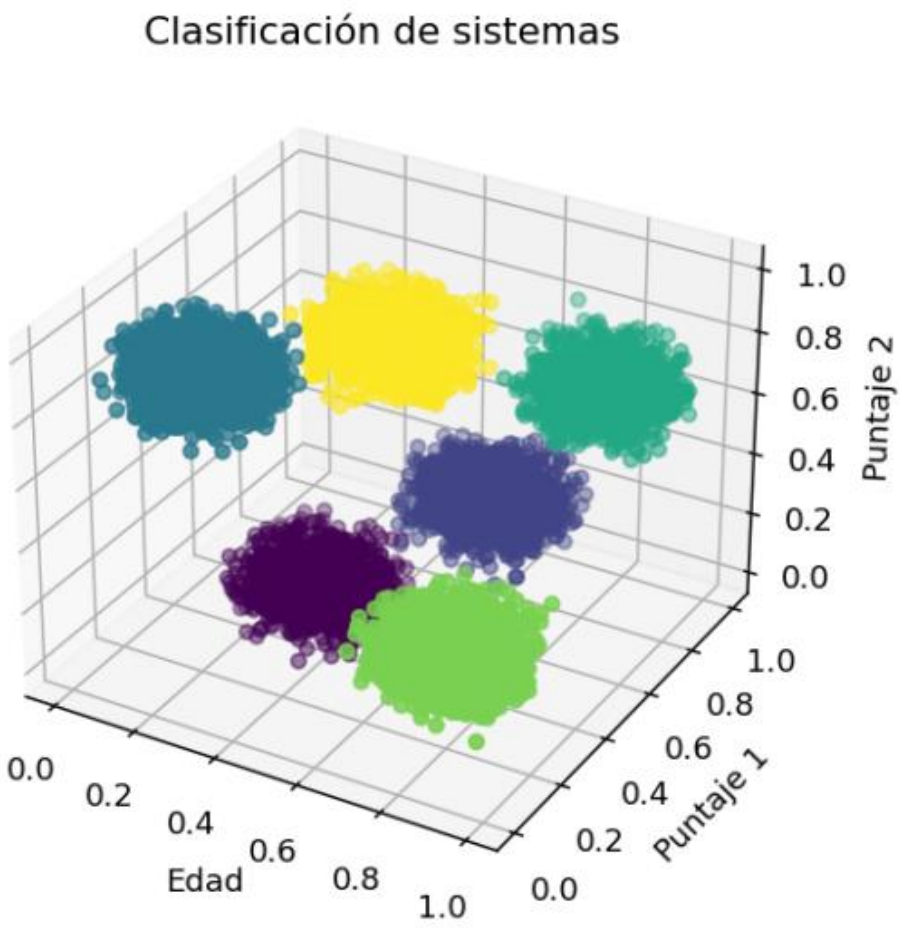
El primer paso fue observar y analizar los datos de entrada y salida, observando sus dimensiones se obtiene (7500,4). Por lo tanto, se extraen los valores de los cuales 3 columnas para la entrada “x”, la última columna sería la de la salida “y”.

Posteriormente procedimos a normalizar cada una de las entradas con el fin de tener valores limitados entre 0 y 1 en ambos ejes, para esto se aplican la siguiente fórmula de normalización para cada una de las salidas:

$$z = \frac{z - z_{min}}{z_{max}}$$

El siguiente paso es ordenar los datos de forma aleatoria para que la red neuronal no se vicie en el aprendizaje. Para dicho ejercicio nos apoyamos de las funciones de random.

Posterior al aplicar todo esto obtenemos normalizado nuestros datos de la siguiente forma:



Planteamiento del modelo

Para este caso se decidió implementar un modelo de clasificación tomando en cuenta los datos de entrada y salida con el fin de llegar a una estimación más cercana al de los valores de la base de datos descargados.

Procedemos a definir los hiperparámetros de la red neuronal:

- Neuronas de entrada = ne = 5.
- Neuronas de la capa oculta = no = 5.
- Neuronas de salida = ns = 6 =número de clases o salida del problema.

- Dendritas = d = 3 = número de columnas de los datos de entrada.

Procedemos a definir funciones auxiliares

Función de activación sigmoide:

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

Función de salida Softmax:

$$y_m = \frac{e^{h_3}}{\sum_{i=0}^{h_3} e^{h_3}}$$

Función de error de entropía cruzada

$$e = -\log(y_m)$$

El siguiente paso es plantear las ecuaciones que se usaran en cada etapa:

Capa de entrada:

$$h_1 = w_1 x + b_1$$

$$y_1 = sigmoide(h_1)$$

Dimensiones: h_1 & $y_1 = (n_e, 1)$

Capa oculta:

$$h_2 = w_2 y_1 + b_2$$

$$y_2 = sigmoide(h_2)$$

Dimensiones: h_2 & $y_2 = (n_o, 1)$

Capa de salida:

$$h_3 = w_3 y_2 + b_3$$

$$y_m = softmax(h_3)$$

Dimensiones: h_3 & $y_m = (n_s, 1)$

Entrenamiento del modelo

Para poder optimizar nuestro problema a resolver se pretende minimizar a lo más posible nuestro error (e) mediante ajustar los hiperparámetros ne y no de nuestro modelo propuesto y el número de épocas o el Lr para ajustar el entrenamiento. Para lograr lo anterior usamos el algoritmo del gradiente descendiente:

$$W_{k+1} = W_k - L_r \frac{\partial e}{\partial W}$$

Por la naturaleza de nuestros parámetros se debe de aplicar la regla de la cadena para obtener los gradientes correspondientes

Para llegar a nuestra solución primero debemos de proponer los gradientes

Capa de salida

$$\frac{\partial e}{\partial w_3} = \frac{\partial e}{\partial y_m} \cdot \frac{\partial y_m}{\partial h_3} \cdot \frac{\partial h_3}{\partial w_3}$$

$$\frac{\partial e}{\partial b_3} = \frac{\partial e}{\partial y_m} \cdot \frac{\partial y_m}{\partial h_3} \cdot \frac{\partial h_3}{\partial b_3}$$

$$\frac{\partial e}{\partial y_2} = \frac{\partial e}{\partial y_m} \cdot \frac{\partial y_m}{\partial h_3} \cdot \frac{\partial h_3}{\partial y_2}$$

Capa oculta

$$\frac{\partial e}{\partial w_2} = \frac{\partial e}{\partial y_2} \cdot \frac{\partial y_2}{\partial h_2} \cdot \frac{\partial h_2}{\partial w_2}$$

$$\frac{\partial e}{\partial b_2} = \frac{\partial e}{\partial y_2} \cdot \frac{\partial y_2}{\partial h_2} \cdot \frac{\partial h_2}{\partial b_2}$$

$$\frac{\partial e}{\partial y_1} = \frac{\partial e}{\partial y_2} \cdot \frac{\partial y_2}{\partial h_2} \cdot \frac{\partial h_2}{\partial y_1}$$

Capa de entrada

$$\frac{\partial e}{\partial w_1} = \frac{\partial e}{\partial y_1} \cdot \frac{\partial y_1}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_1}$$

$$\frac{\partial e}{\partial b_1} = \frac{\partial e}{\partial y_1} \cdot \frac{\partial y_1}{\partial h_1} \cdot \frac{\partial h_1}{\partial b_1}$$

Después encontramos los gradientes por separado

Función de error:

$$\frac{\partial e}{\partial y_m} = \frac{1}{\partial y_m}$$

Función Softmax:

Función clase correcta:

$$\frac{\partial y_m}{\partial h_3} = y_{m[i]}(1 - y_{m[i]})$$

Función clase incorrecta:

$$\frac{\partial y_m}{\partial h_3} = -y_{m[i]}(y_{m[j]})$$

Capa de salida

$$\frac{\partial h_3}{\partial w_3} = y_2$$

$$\frac{\partial h_3}{\partial b_3} = 1.0$$

$$\frac{\partial h_3}{\partial y_2} = w_3$$

Capa oculta

$$\frac{\partial y_2}{\partial h_2} = y_2(1 - y_2)$$

$$\frac{\partial h_2}{\partial w_2} = y_1$$

$$\frac{\partial h_2}{\partial b_3} = 1.0$$

$$\frac{\partial h_2}{\partial y_1} = w_2$$

Capa de entrada

$$\frac{\partial y_1}{\partial h_1} = y_1(1 - y_1)$$

$$\frac{\partial h_1}{\partial w_1} = x$$

$$\frac{\partial h_1}{\partial b_1} = 1.0$$

Por último, actualizamos los parámetros con la ayuda del gradiente descendiente

$$w_1 = w_1 - L_r\left(\frac{\partial e}{\partial w_1}\right)$$

$$b_1 = b_1 - L_r\left(\frac{\partial e}{\partial b_1}\right)$$

$$w_2 = w_2 - L_r\left(\frac{\partial e}{\partial w_2}\right)$$

$$b_2 = b_2 - L_r\left(\frac{\partial e}{\partial b_2}\right)$$

$$w_3 = w_3 - L_r\left(\frac{\partial e}{\partial w_3}\right)$$

$$b_3 = b_3 - L_r\left(\frac{\partial e}{\partial b_3}\right)$$

Resultados experimentales

Se creó una instancia para nuestro modelo con unos parámetros iniciales:

$$n_e = 5, \qquad n_o = 5, \qquad n_s = 6, \qquad d = 3$$

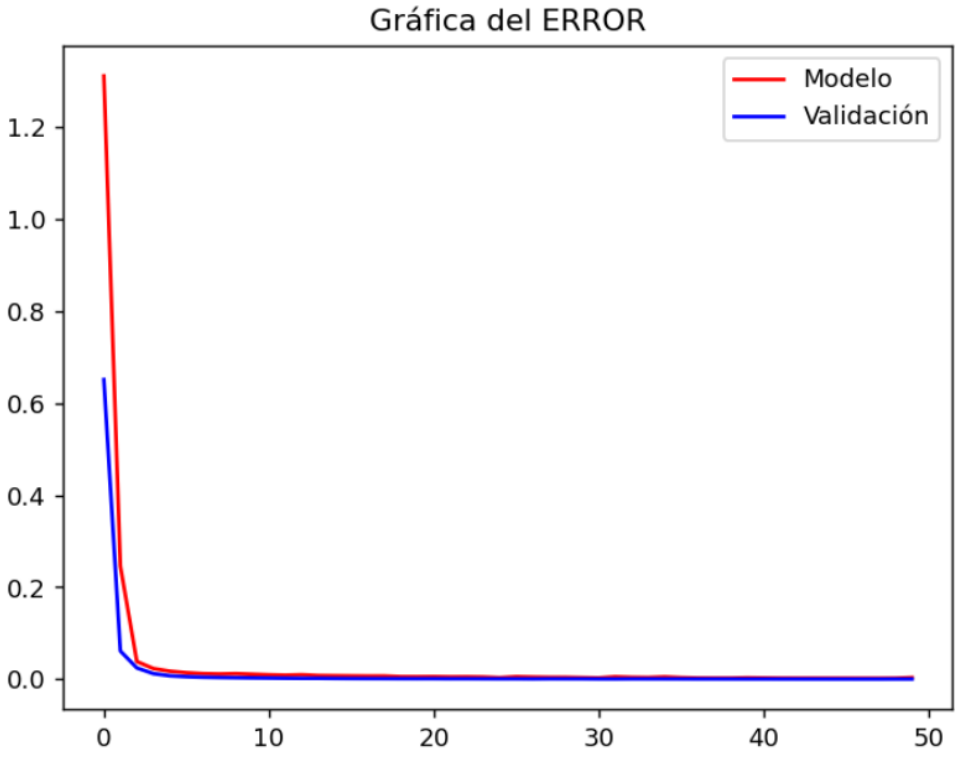
```
#crear instancia del modelo
SisOp = mlp(5,5,6,3)
```

A este modelo se le sometió a un entrenamiento con una taza de aprendizaje $L_r = 0.1$ y durante 50 épocas.

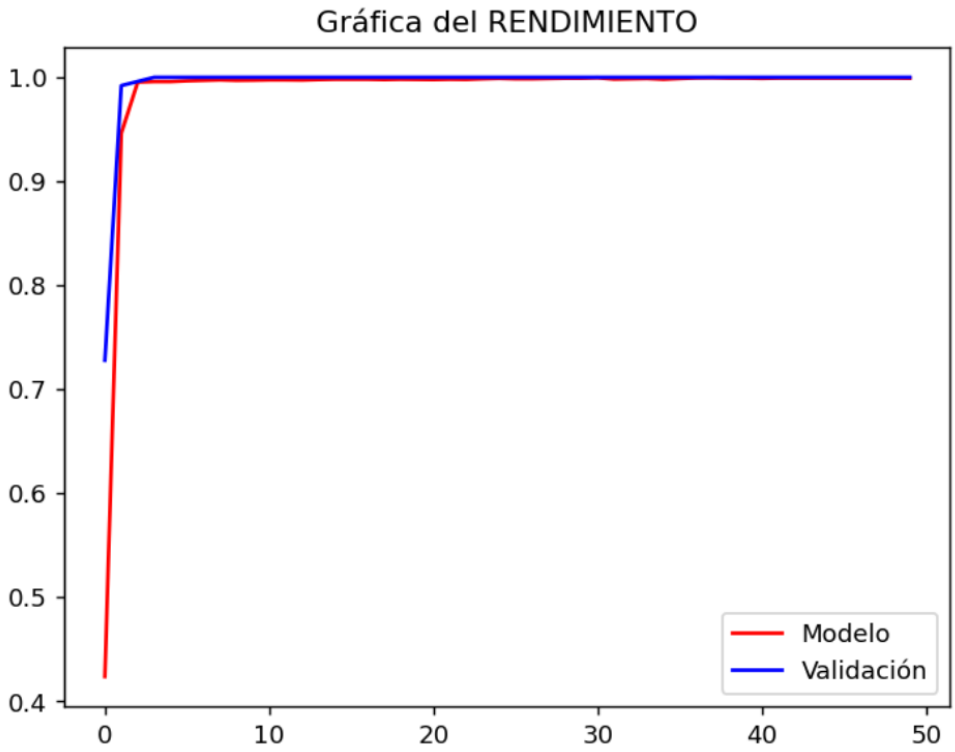
```
#entrenar modelo
SisOp.train(x_train,y_train,x_test,y_test,0.1,50)
```

Para este modelo el 90% de los datos son utilizados para el modelo, mientras que un 10% de los datos se utilizó para la validación del modelo.

Con dicho entrenamiento obtenemos cómo se comporta el error en el modelo y en la validación:



También podemos observar después del entrenamiento cómo se comporta el rendimiento del modelo y de la validación:

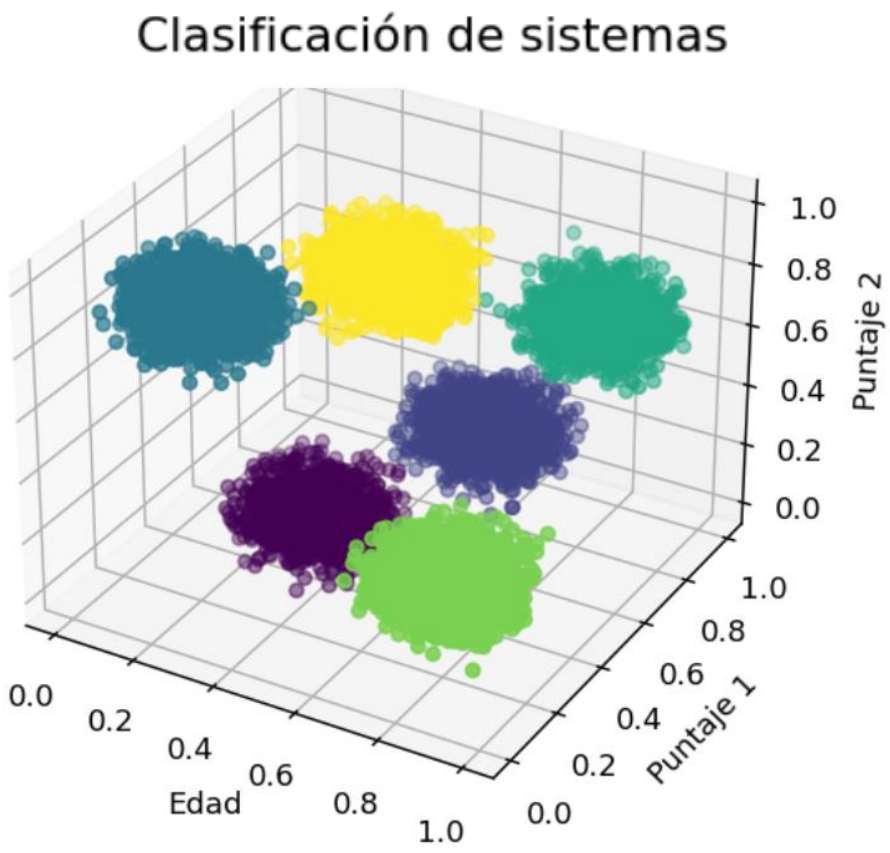


Con nuestra validación podemos buscar 2 objetivos, alcanzar un rendimiento máximo y lograr un error mínimo, para ello con los parámetros obtenidos en la validación (rv=rendimiento de validación y ev=error de validación) obtenemos los siguientes resultados:

```
Rendimiento MÁXIMO
Época en la que se alcanza el rendimiento máximo:3
Rendimiento de validación: 100.0%
Error de validación: 1.2396508061135272%
Rendimiento del modelo: 99.58518518518518%
Error del modelo: 2.351123272647249%

Error MÍNIMO
Época en la que se alcanza el error mínimo:47
Rendimiento de validación: 100.0%
Error de validación: 0.057297587356447945%
Rendimiento del modelo: 99.92592592592592%
Error del modelo: 0.2290256616943305 8%
```

Como podemos observar con los datos anteriores al obtener el error mínimo en este caso a la vez estamos alcanzando el rendimiento máximo, por lo que al final logramos cumplir con el objetivo teniendo un error muy pequeño tanto en la validación como en el modelo y a su vez tener el máximo rendimiento tanto en la validación como en nuestro modelo

Posteriormente podemos visualizar la evaluación de nuestro modelo



Conclusiones

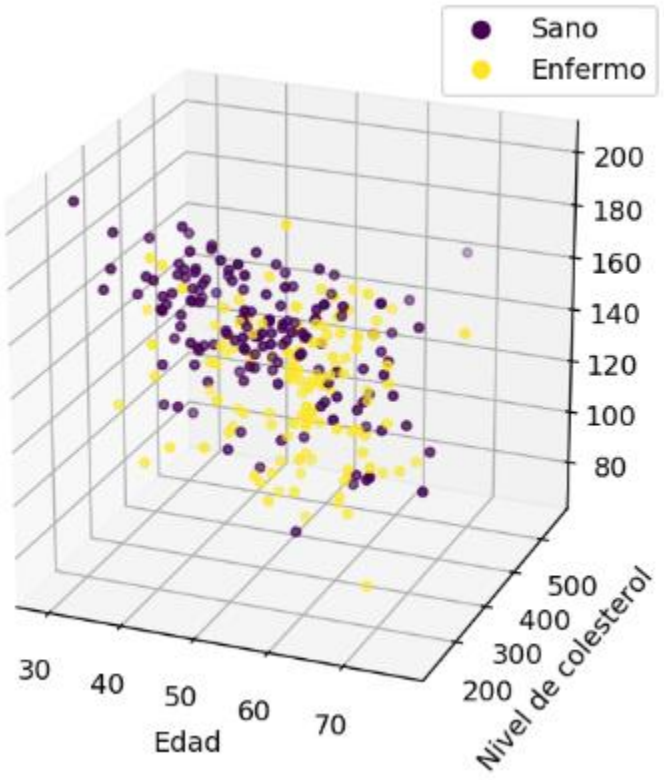
Con los resultados obtenidos se puede observar, que si se pudo clasificar bien los softwares recomendados para cada tipo de personas con la ayuda de los datos recopilados. Vemos que nuestro modelo cuenta con un rendimiento de 99.926% y un rendimiento del 100% en la validación, lo cual lo vuelve un modelo muy confiable para clasificación. Esto se puede observar al evaluar nuestro modelo y ver que logra clasificar de manera muy precisa los datos.

Problema 2: Enfermades cardiovasculares

Análisis de datos

Un prestigioso médico de Zapopan desea crear un tratamiento preventivo para personas con riesgo de padecer enfermedades cardiovasculares. Como una primera etapa, desea determinar si por medio de la edad, el nivel de colesterol y la frecuencia cardiaca promedio es posible determinar si una persona es propensa a desarrollar alguna enfermedad. Para lo cual, elaboró una base de datos con la información de 270 de sus pacientes (que desarrollaron y no desarrollaron problemas).

Se pretende desarrollar un modelo que permita estimar la probabilidad de que un peciente sea propenso a desarrollar algún tipo de padecimiento carciaco.



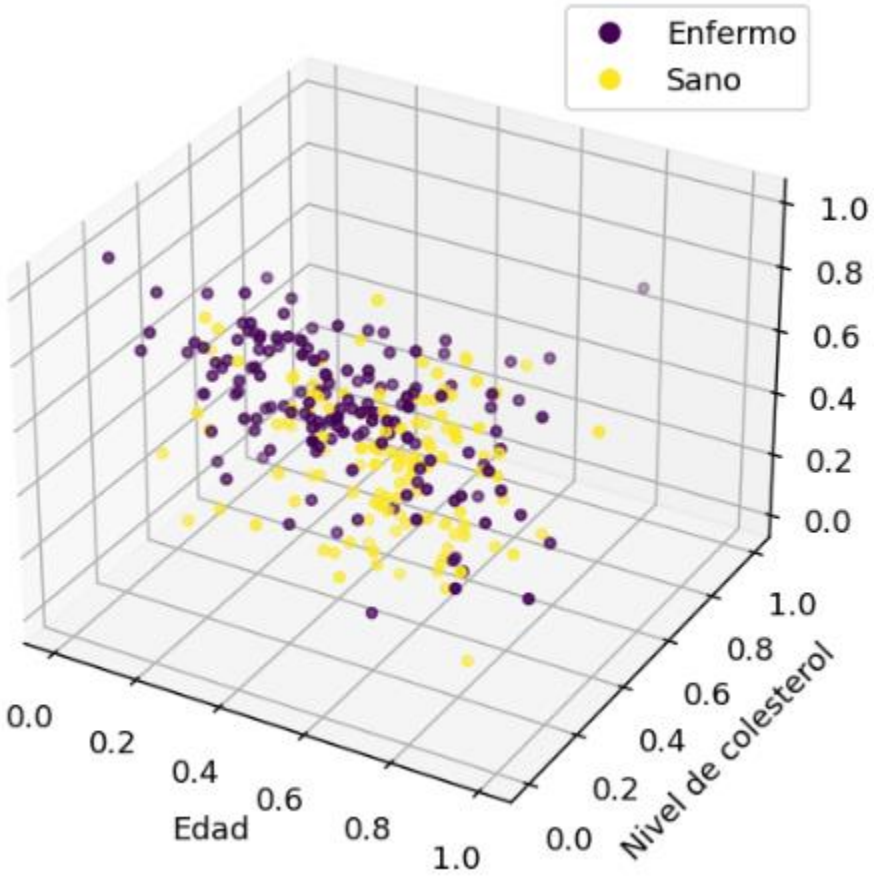
El primer paso fue observar y analizar los datos de entrada y salida, observando sus dimensiones se obtiene (7500,4). Por lo tanto, se extraen los valores de los cuales 3 columnas para la entrada “x”, la última columna sería la de la salida “y”.

Posteriormente procedimos a normalizar cada una de las entradas con el fin de tener valores limitados entre 0 y 1 en ambos ejes, para esto se aplican la siguiente fórmula de normalización para cada una de las salidas:

$$z = \frac{z - z_{min}}{z_{max}}$$

El siguiente paso es ordenar los datos de forma aleatoria para que la red neuronal no se vicie en el aprendizaje. Para dicho ejercicio nos apoyamos de las funciones de random.

Posterior al aplicar todo esto obtenemos normalizado nuestros datos de la siguiente forma:



Planteamiento del modelo

Para este caso se decidió implementar un modelo de clasificación tomando en cuenta los datos de entrada y salida con el fin de llegar a una estimación más cercana al de los valores de la base de datos descargados.

Procedemos a definir los hiperparámetros de la red neuronal:

- Neuronas de entrada = ne = 6.
- Neuronas de la capa oculta = no = 15.
- Neuronas de salida = ns = 2 =número de clases o salida del problema.
- Dendritas = d = 3 = número de columnas de los datos de entrada.

Procedemos a definir funciones auxiliares

Función de activación sigmoide:

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

Función de salida Softmax:

$$y_m = \frac{e^{h_3}}{\sum_{i=0}^{h_3} e^{h_3}}$$

Función de error de entropía cruzada

$$e = -\log(y_m)$$

El siguiente paso es plantear las ecuaciones que se usaran en cada etapa:

Capa de entrada:

$$h_1 = w_1 x + b_1$$

$$y_1 = sigmoide(h_1)$$

Dimensiones: h_1 & $y_1 = (n_e, 1)$

Capa oculta:

$$h_2 = w_2 y_1 + b_2$$

$$y_2 = sigmoide(h_2)$$

Dimensiones: h_2 & $y_2 = (n_o, 1)$

Capa de salida:

$$h_3 = w_3 y_2 + b_3$$

$$y_m = softmax(h_3)$$

Dimensiones: h_3 & $y_m = (n_s, 1)$

Entrenamiento del modelo

Para poder optimizar nuestro problema a resolver se pretende minimizar a lo más posible nuestro error (e) mediante ajustar los hiperparámetros ne y no de nuestro modelo propuesto y el número de épocas o el Lr para ajustar el entrenamiento. Para lograr lo anterior usamos el algoritmo del gradiente descendiente:

$$W_{k+1} = W_k - L_r \frac{\partial e}{\partial W}$$

Por la naturaleza de nuestros parámetros se debe de aplicar la regla de la cadena para obtener los gradientes correspondientes

Para llegar a nuestra solución primero debemos de proponer los gradientes

Capa de salida

$$\frac{\partial e}{\partial w_3} = \frac{\partial e}{\partial y_m} \cdot \frac{\partial y_m}{\partial h_3} \cdot \frac{\partial h_3}{\partial w_3}$$

$$\frac{\partial e}{\partial b_3} = \frac{\partial e}{\partial y_m} \cdot \frac{\partial y_m}{\partial h_3} \cdot \frac{\partial h_3}{\partial b_3}$$

$$\frac{\partial e}{\partial y_2} = \frac{\partial e}{\partial y_m} \cdot \frac{\partial y_m}{\partial h_3} \cdot \frac{\partial h_3}{\partial y_2}$$

Capa oculta

$$\frac{\partial e}{\partial w_2} = \frac{\partial e}{\partial y_2} \cdot \frac{\partial y_2}{\partial h_2} \cdot \frac{\partial h_2}{\partial w_2}$$

$$\frac{\partial e}{\partial b_2} = \frac{\partial e}{\partial y_2} \cdot \frac{\partial y_2}{\partial h_2} \cdot \frac{\partial h_2}{\partial b_2}$$

$$\frac{\partial e}{\partial y_1} = \frac{\partial e}{\partial y_2} \cdot \frac{\partial y_2}{\partial h_2} \cdot \frac{\partial h_2}{\partial y_1}$$

Capa de entrada

$$\frac{\partial e}{\partial w_1} = \frac{\partial e}{\partial y_1} \cdot \frac{\partial y_1}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_1}$$

$$\frac{\partial e}{\partial b_1} = \frac{\partial e}{\partial y_1} \cdot \frac{\partial y_1}{\partial h_1} \cdot \frac{\partial h_1}{\partial b_1}$$

Después encontramos los gradientes por separado

Función de error:

$$\frac{\partial e}{\partial y_m} = \frac{1}{\partial y_m}$$

Función Softmax:

Función clase correcta:

$$\frac{\partial y_m}{\partial h_3} = y_{m[i]}(1 - y_{m[i]})$$

Función clase incorrecta:

$$\frac{\partial y_m}{\partial h_3} = -y_{m[i]}(y_{m[j]})$$

Capa de salida

$$\frac{\partial h_3}{\partial w_3} = y_2$$

$$\frac{\partial h_3}{\partial b_3} = 1.0$$

$$\frac{\partial h_3}{\partial y_2} = w_3$$

Capa oculta

$$\frac{\partial y_2}{\partial h_2} = y_2(1 - y_2)$$

$$\frac{\partial h_2}{\partial w_2} = y_1$$

$$\frac{\partial h_2}{\partial b_3} = 1.0$$

$$\frac{\partial h_2}{\partial y_1} = w_2$$

Capa de entrada

$$\frac{\partial y_1}{\partial h_1} = y_1(1 - y_1)$$

$$\frac{\partial h_1}{\partial w_1} = x$$

$$\frac{\partial h_1}{\partial b_1} = 1.0$$

Por último, actualizamos los parámetros con la ayuda del gradiente descendiente

$$w_1 = w_1 - L_r\left(\frac{\partial e}{\partial w_1}\right)$$

$$b_1 = b_1 - L_r\left(\frac{\partial e}{\partial b_1}\right)$$

$$w_2 = w_2 - L_r\left(\frac{\partial e}{\partial w_2}\right)$$

$$b_2 = b_2 - L_r\left(\frac{\partial e}{\partial b_2}\right)$$

$$w_3 = w_3 - L_r\left(\frac{\partial e}{\partial w_3}\right)$$

$$b_3 = b_3 - L_r\left(\frac{\partial e}{\partial b_3}\right)$$

Resultados experimentales

Se creó una instancia para nuestro modelo con unos parámetros iniciales:

$$n_e = 6, \qquad n_o = 15, \qquad n_s = 2, \qquad d = 3$$

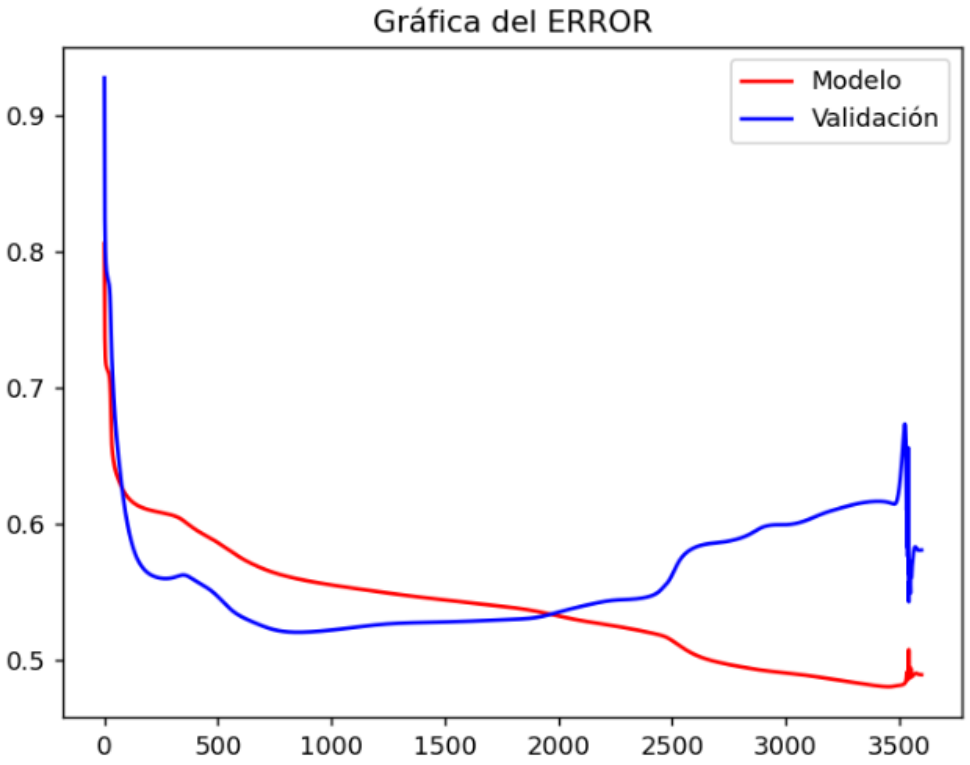
```
#crear instancia del modelo
enfermedad = mlp(6,15,2,3)
```

A este modelo se le sometió a un entrenamiento con una taza de aprendizaje $L_r = 0.175$ y durante 3600 épocas.

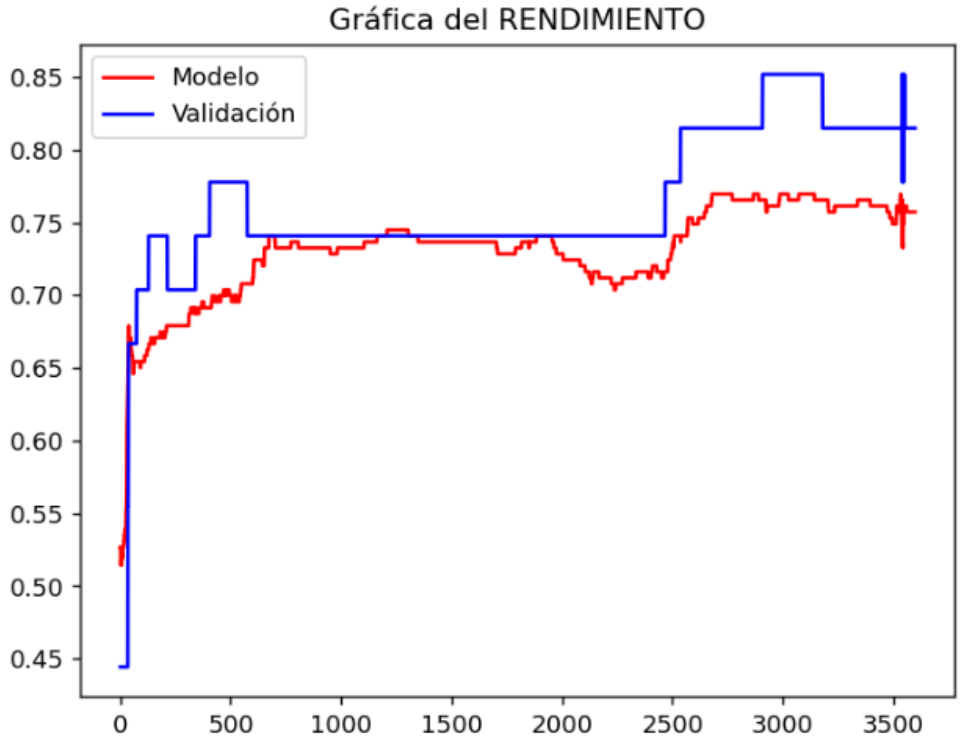
```
#entrenar modelo
enfermedad.train(x_train,y_train,x_test,y_test,0.175,3600)
```

Para este modelo el 90% de los datos son utilizados para el modelo, mientras que un 10% de los datos se utilizó para la validación del modelo.

Con dicho entrenamiento obtenemos cómo se comporta el error en el modelo y en la validación:



También podemos observar después del entrenamiento cómo se comporta el rendimiento del modelo y de la validación:

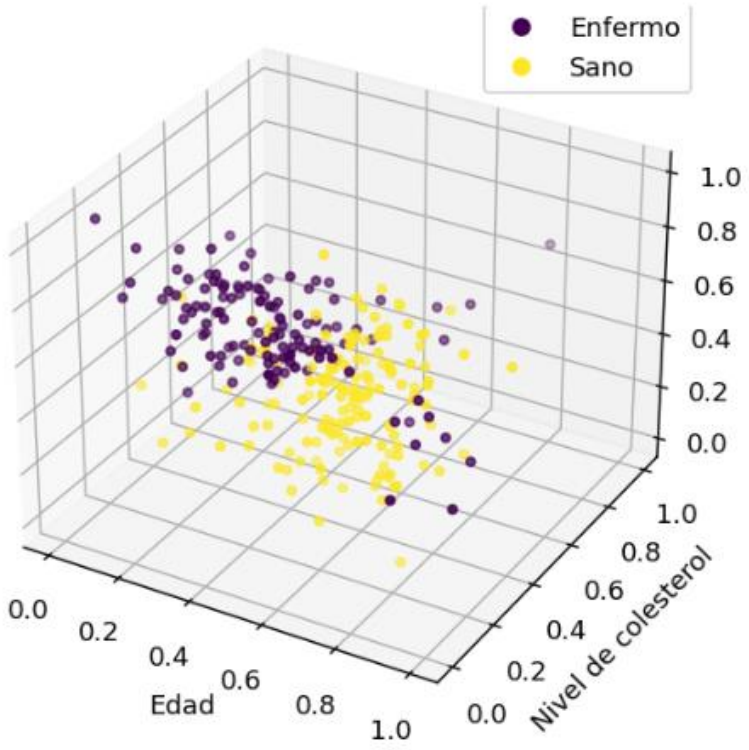


Con nuestra validación podemos buscar 2 objetivos, alcanzar un rendimiento máximo y lograr un error mínimo, para ello con los parámetros obtenidos en la validación (rv=rendimiento de validación y ev=error de validación) obtenemos los siguientes resultados:

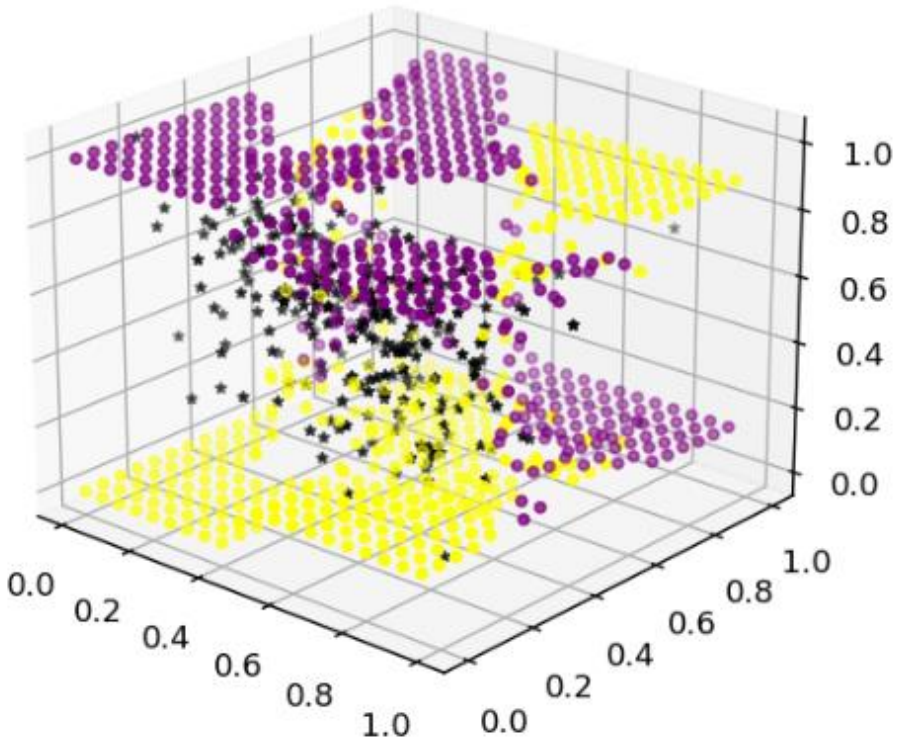
```
Rendimiento MÁXIMO
Época en la que se alcanza el rendimiento máximo:2909
Rendimiento de validación: 85.18518518518519%
Error de validación: 59.817595139754744%
Rendimiento del modelo: 76.5432098765432%
Error del modelo: 49.2343509873017%

Error MÍNIMO
Época en la que se alcanza el error mínimo:853
Rendimiento de validación: 74.07407407407408%
Error de validación: 52.05060976291633%
Rendimiento del modelo: 73.25102880658436%
Error del modelo: 55.96155551323621%
```

Como podemos observar con los datos anteriores al obtener el rendimiento máximo de la validación obtenemos un rendimiento mayor del modelo que al momento de obtener el rendimiento mínimo. En este caso nuestro modelo fue muy difícil de ajustar para tratar de satisfacer los requisitos de obtener el 90% de rendimiento.

Posteriormente podemos visualizar la evaluación de nuestro modelo



Al mismo tiempo podemos ver los hiperplanos con los que el modelo hace la clasificación



Conclusiones

Los datos de este modelo y su distribución no poseen algún patrón o características en común y están altamente mezclados entre sí, haciendo bastante sino es que hasta imposible de lograr el encontrar un modelo perfecto para esta situación, se hizo el esfuerzo mayor por poder cumplir con el 90% del rendimiento.

El modelo no fue tan satisfactorio ya que a duras penas se logro conseguir un 85.2% de rendimiento de validación, además de contar con un porcentaje de error tanto del modelo como de la validación exageradamente alto, haciendo que este modelo no sea confiable al 100% ya que tiene un error mínimo de validación del 52% y cuando el rendimiento de validación es el máximo el modelo tiene un error de 49.23%

Problema 3: Mejoramiento de la potencia de vehículos eléctricos

Análisis de datos

Elías Muñoz es un ingeniero mexicano fundador de la empresa Camarena. La compañía desea desarrollar un nuevo vehículo eléctrico inteligente que sea económico, eficiente y con la tecnología suficiente para competir directamente con la compañía Tesla de Elon Musk.

Actualmente, se encuentra desarrollando un sistema de manejo asistido que permita a los conductores mantener su velocidad mientras atraviesan pendientes inclinadas, lo que es un problema de potencia común para la mayoría de los automóviles. Su idea es desarrollar un modelo que permita estimar la potencia requerida del automóvil ante diferentes grados de inclinación de las pendientes, a fin de compensar (en medida de lo necesario) la potencia actual del vehículo. Para ello, recopiló información experimental en campo que relaciona ambas variables.

El primer paso fue observar y analizar los datos de entrada y salida, observando sus dimensiones se obtiene (500,1). Por lo tanto, se extraen los valores para la entrada “x”, y para la salida “y”.

Posteriormente procedimos a normalizar “x” y “y” con el fin de tener valores limitados entre 0 y 1 en ambos ejes, para esto se aplican la siguiente fórmula de normalización para cada una de las salidas:

$$x = \frac{x - x_{min}}{x_{max}}$$

$$y = \frac{y - y_{min}}{y_{max}}$$

Posterior al aplicar todo esto obtenemos normalizado nuestros datos de la siguiente forma:

Planteamiento del modelo

Para este caso se decidió implementar un modelo de clasificación tomando en cuenta los datos de entrada y salida con el fin de llegar a una estimación más cercana al de los valores de la base de datos descargados.

Procedemos a definir los hiperparámetros de la red neuronal:

- Neuronas de entrada = ne = 4.
- Neuronas de la capa oculta = no = 3.
- Neuronas de salida = ns = 1 =número de clases o salida del problema.
- Dendritas = d = 1 = número de columnas de los datos de entrada.

Procedemos a definir funciones auxiliares

Función de activación sigmoide:

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

Función de error de entropía cruzada

$$e = \frac{1}{2}(y_m - y_d)^2$$

El siguiente paso es plantear las ecuaciones que se usaran en cada etapa:

Capa de entrada:

$$h_1 = w_1 x + b_1$$

$$y_1 = sigmoide(h_1)$$

Dimensiones: h_1 & $y_1 = (n_e, 1)$

Capa oculta:

$$h_2 = w_2 y_1 + b_2$$

$$y_2 = sigmoide(h_2)$$

Dimensiones: h_2 & $y_2 = (n_o, 1)$

Capa de salida:

$$h_3 = w_3 y_2 + b_3$$

$$y_m = sigmoide(h_3)$$

Dimensiones: h_3 & $y_m = (n_s, 1)$

Entrenamiento del modelo

Para poder optimizar nuestro problema a resolver se pretende minimizar a lo más posible nuestro error (e) mediante ajustar los hiperparámetros ne y no de nuestro modelo propuesto y el número de épocas o el Lr para ajustar el entrenamiento. Para lograr lo anterior usamos el algoritmo del gradiente descendiente:

$$W_{k+1} = W_k - L_r \frac{\partial e}{\partial W}$$

Por la naturaleza de nuestros parámetros se debe de aplicar la regla de la cadena para obtener los gradientes correspondientes

Para llegar a nuestra solución primero debemos de proponer los gradientes

Capa de salida

$$\frac{\partial e}{\partial w_3} = \frac{\partial e}{\partial y_m} \cdot \frac{\partial y_m}{\partial h_3} \cdot \frac{\partial h_3}{\partial w_3}$$

$$\frac{\partial e}{\partial b_3} = \frac{\partial e}{\partial y_m} \cdot \frac{\partial y_m}{\partial h_3} \cdot \frac{\partial h_3}{\partial b_3}$$

$$\frac{\partial e}{\partial y_2} = \frac{\partial e}{\partial y_m} \cdot \frac{\partial y_m}{\partial h_3} \cdot \frac{\partial h_3}{\partial y_2}$$

Capa oculta

$$\frac{\partial e}{\partial w_2} = \frac{\partial e}{\partial y_2} \cdot \frac{\partial y_2}{\partial h_2} \cdot \frac{\partial h_2}{\partial w_2}$$

$$\frac{\partial e}{\partial b_2} = \frac{\partial e}{\partial y_2} \cdot \frac{\partial y_2}{\partial h_2} \cdot \frac{\partial h_2}{\partial b_2}$$

$$\frac{\partial e}{\partial y_1} = \frac{\partial e}{\partial y_2} \cdot \frac{\partial y_2}{\partial h_2} \cdot \frac{\partial h_2}{\partial y_1}$$

Capa de entrada

$$\frac{\partial e}{\partial w_1} = \frac{\partial e}{\partial y_1} \cdot \frac{\partial y_1}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_1}$$

$$\frac{\partial e}{\partial b_1} = \frac{\partial e}{\partial y_1} \cdot \frac{\partial y_1}{\partial h_1} \cdot \frac{\partial h_1}{\partial b_1}$$

Después encontramos los gradientes por separado

Función de error:

$$\frac{\partial e}{\partial y_m} = y_m - y$$

Capa de salida

$$\frac{\partial y_m}{\partial h_3} = y_m(1 - y_m)$$

$$\frac{\partial h_3}{\partial w_3} = y_2$$

$$\frac{\partial h_3}{\partial b_3} = 1.0$$

$$\frac{\partial h_3}{\partial y_2} = w_3$$

Capa oculta

$$\frac{\partial y_2}{\partial h_2} = y_2(1 - y_2)$$

$$\frac{\partial h_2}{\partial w_2} = y_1$$

$$\frac{\partial h_2}{\partial b_3} = 1.0$$

$$\frac{\partial h_2}{\partial y_1} = w_2$$

Capa de entrada

$$\frac{\partial y_1}{\partial h_1} = y_1(1 - y_1)$$

$$\frac{\partial h_1}{\partial w_1} = x$$

$$\frac{\partial h_1}{\partial b_1} = 1.0$$

Por último, actualizamos los parámetros con la ayuda del gradiente descendiente

$$w_1 = w_1 - L_r \left(\frac{\partial e}{\partial w_1}\right)$$

$$b_1 = b_1 - L_r\left(\frac{\partial e}{\partial b_1}\right)$$

$$w_2 = w_2 - L_r\left(\frac{\partial e}{\partial w_2}\right)$$

$$b_2 = b_2 - L_r\left(\frac{\partial e}{\partial b_2}\right)$$

$$w_3 = w_3 - L_r\left(\frac{\partial e}{\partial w_3}\right)$$

$$b_3 = b_3 - L_r\left(\frac{\partial e}{\partial b_3}\right)$$

Resultados experimentales

Se creó una instancia para nuestro modelo con unos parámetros iniciales:

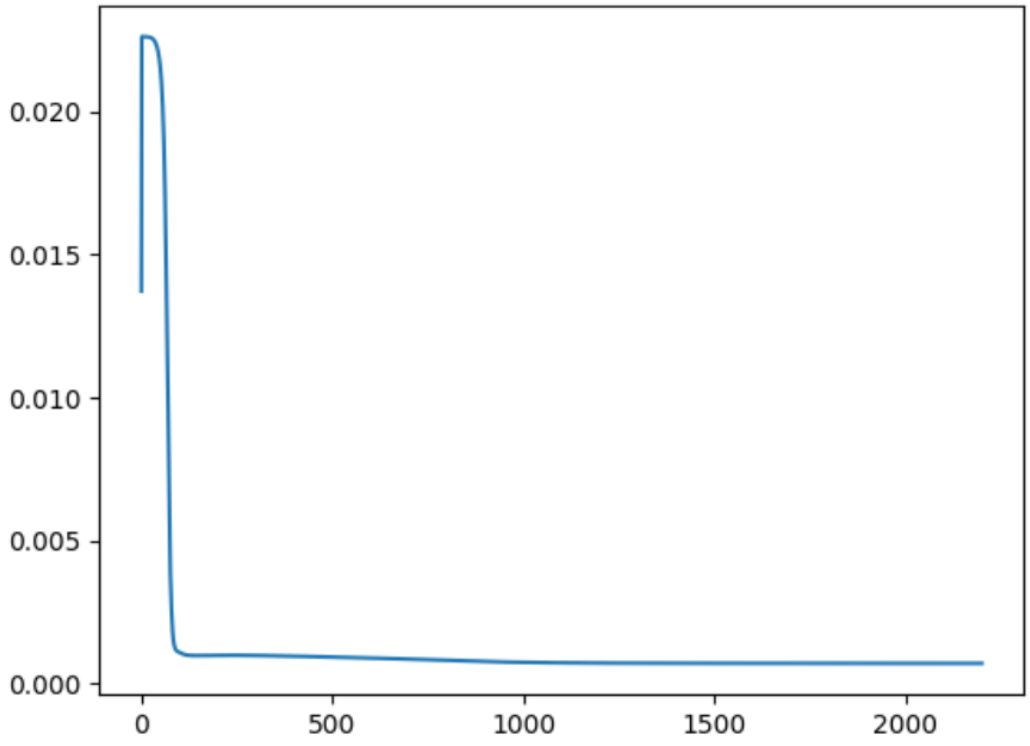
$$n_e = 4, \quad n_o = 3, \quad n_s = 1, \quad d = 1$$

```
#crear instancia de red neuronal
vehiculos=mlp(4,3,1,1)
```

A este modelo se le sometió a un entrenamiento con una taza de aprendizaje $L_r = 0.1$ y durante 2200 épocas.

```
#entrenar red neuronal
vehiculos.train(x,y,0.1,2200)
```

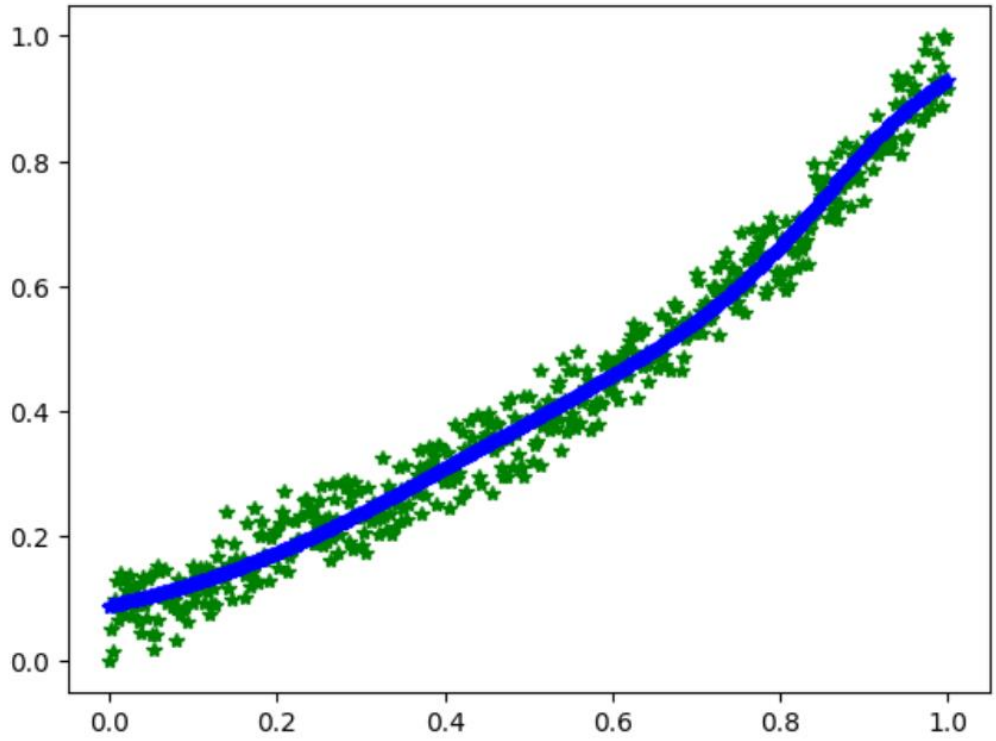
Con dicho entrenamiento obtenemos cómo se comporta el error (error medio cuadrático):



Con nuestro modelo parametrizado obtenemos un error igual a:

$$Error\ (e) = 0.07063229\%$$

Posteriormente podemos visualizar el comportamiento de nuestro modelo parametrizado (color azul) comparado con nuestra gráfica inicial obtenida de los datos (color verde)



Conclusiones

Observando los resultados de la regresión podemos determinar y estimar de una manera más precisa y certera la potencia requerida del automóvil ante diferentes grados de inclinación de las pendientes. Podemos observar que el error al ser bastante pequeño nos indica que el modelo de red neuronal es bastante acertado y se puede estimar a un futuro los valores con ayuda de este modelo.