Escuela Superior de Cómputo

Práctica 2. Redes neuronales usando *Python* y *Tensorflow*

Redes neuronales y aprendizaje profundo

Profesora: Dra. Camacho Vázquez Vanessa Alejandra.

Objetivo general: el alumno será capaz de implementar redes neuronales mediante el uso de Python y Tensorflow.

Objetivos específicos:

- ✓ El alumno analizará e introducirá el conjunto de datos de entrada.
- ✓ El alumno graficará y procesará los datos para su entrenamiento.
- ✓ El alumno construirá y entrenará el modelo.
- ✓ El alumno realizará predicciones con el conocimiento adquirido y graficará los resultados.

Marco teórico

Observar el video ya que te servirá como base para desarrollar esta práctica: https://www.youtube.com/watch?v=iX_on3VxZzk

1. Elaborar un resumen a mano*** (mínimo 3 cuartillas) que cubra los temas siguientes:

   - ✓ Aprendizaje automático vs programación regular.
   - ✓ Escenario que resolveremos.
   - ✓ Conceptos que debes conocer.
   - ✓ Proceso que sigue la red neuronal.
   - ✓ ¿Cómo va a aprender?

***Puedes añadir diagramas y notas que consideres importantes.

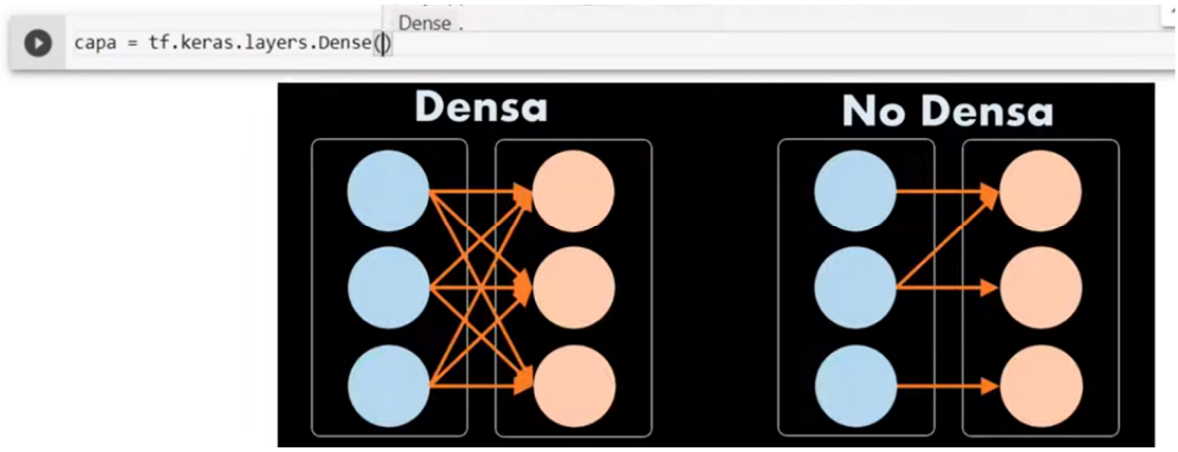
Implementación

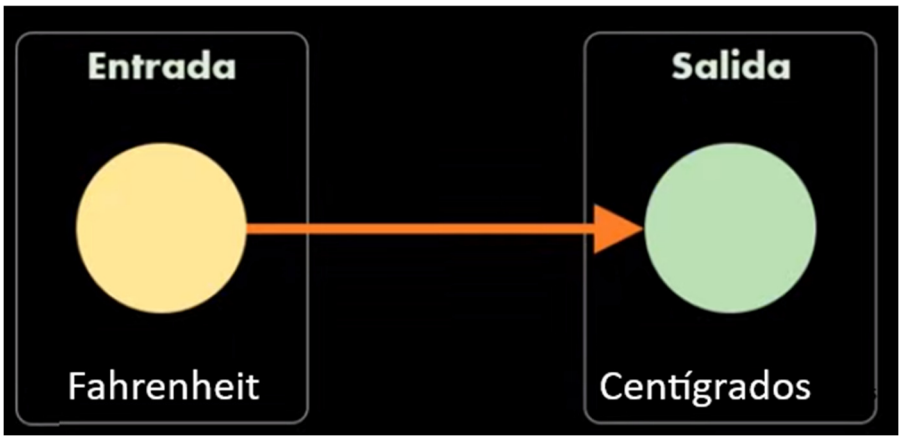
Esta red neuronal será capaz de convertir grados **Fahrenheit a centígrados**. El conjunto de datos de entrada está compuesto por solamente 10 instancias y se muestra en la tabla a continuación:

Fahrenheit	Centígrados
32	0
20	-6.66667
15	-9.44444
50	10
65	18.3333
72	22.2222
5	-15
85	29.4444
100	37.7778
0	-17.7778

Nota: *Google Colab* te permite programar tus redes neuronales. De lo contrario, deberás instalar Python y Tensorflow en tu computadora.

Tu primera red neuronal utilizará una capa densa *Dense* → este tipo de capa tiene conexiones desde cada neurona a todas las neuronas de la siguiente capa. Aunque dicha red neuronal sencilla solamente tendrá 2 neuronas así que no habrá mucho por conectar.





El siguiente código te permite implementar una red neuronal muy parecida que convierte de grados Celsius a Farenheit.

2. Entonces **deberás adaptar dicho código para que tu red neuronal convierta de grados Farenheit a Centígrados** usando los datos de entrada que se encuentran en la tabla anterior.

3. Además, tendrás que **añadir comentarios a cada parte de tu código para explicar detalladamente lo que hace.**

Código de la red neuronal de ejemplo

Construcción del modelo

```
import tensorflow as tf
import numpy as np
```

```
celsius = np.array([-40, -10, 0, 8, 15, 22, 38], dtype=float)
fahrenheit = np.array([-40, 14, 32, 46, 59, 72, 100], dtype=float)
```

```
capa = tf.keras.layers.Dense(units=1, input_shape=[1])
modelo = tf.keras.Sequential([capa])
```

Propiedades para el aprendizaje

```
[ ]  modelo.compile(
         optimizer=tf.keras.optimizers.Adam(0.1),
         loss='mean_squared_error'
     )
```

Entrenamiento

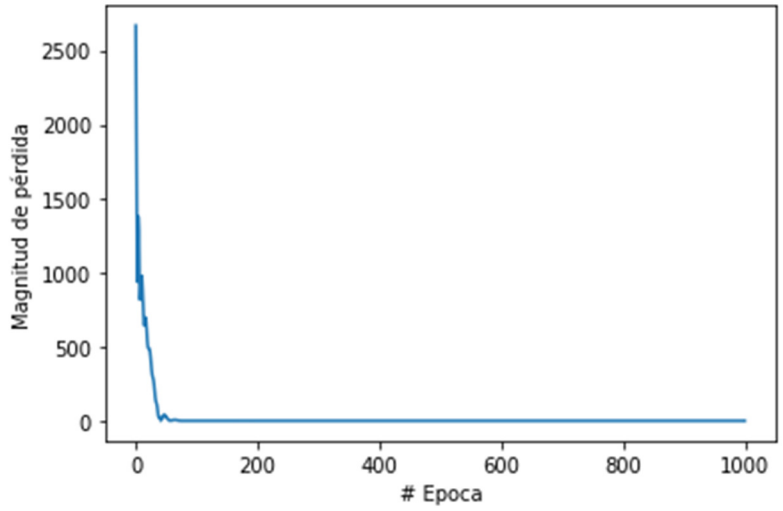
```
print("Comenzando entrenamiento...")
historial = modelo.fit(celsius, fahrenheit, epochs=1000, verbose=False)
print("Modelo entrenado!")
```

```
Comenzando entrenamiento...
Modelo entrenado!
```

Gráfica de los resultados de la función de pérdida

```
import matplotlib.pyplot as plt
plt.xlabel("# Epoca")
plt.ylabel("Magnitud de pérdida")
plt.plot(historial.history["loss"])
```

```
[<matplotlib.lines.Line2D at 0x7fd4af0d3c90>]
```



Predicciones y resultados

```
print("Hagamos una predicción!")
resultado = modelo.predict([100.0])
print("El resultado es " + str(resultado) + " fahrenheit!")
```

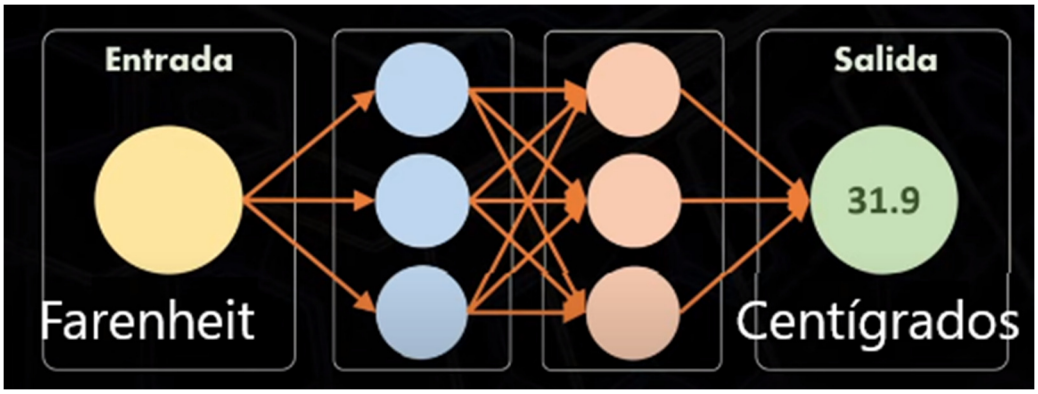
```
Hagamos una predicción!
El resultado es [[211.74745]] fahrenheit!
```

```
print("Variables internas del modelo")
print(capa.get_weights())
```

4. Adjunta capturas de pantalla y tu código con comentarios. No olvides ejecutar cada etapa: construcción del modelo – propiedades para el aprendizaje – entrenamiento – Gráfica de los resultados de la función de pérdida y resultados.

Implementación de la segunda red neuronal

Tu segunda red neuronal contendrá más capas → esta red también llevará a cabo la conversión de **Farenheit a Centígrados** usando los datos de entrada que se encuentran en la tabla anterior.



Contiene 2 capas ocultas.

A continuación, se encuentra el código de una red neuronal parecida que convierte de grados Celsius a Farenheit y deberás adaptar el código para implementar la segunda red neuronal que se te pide.

```
import tensorflow as tf
import numpy as np
```

```
celsius = np.array([-40, -10, 0, 8, 15, 22, 38], dtype=float)
fahrenheit = np.array([-40, 14, 32, 46, 59, 72, 100], dtype=float)
```

```
oculta1 = tf.keras.layers.Dense(units=3, input_shape=[1])
oculta2 = tf.keras.layers.Dense(units=3)
salida = tf.keras.layers.Dense(units=1)
modelo = tf.keras.Sequential([oculta1, oculta2, salida])
```

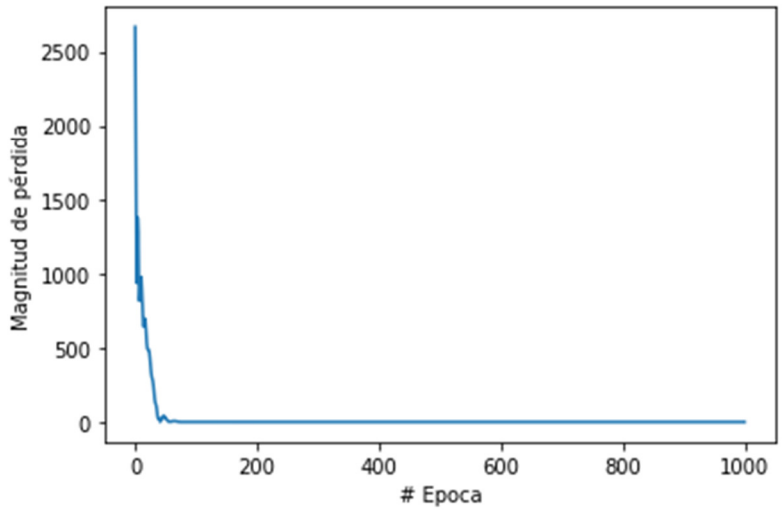
```
modelo.compile(
    optimizer=tf.keras.optimizers.Adam(0.1),
    loss='mean_squared_error'
)
```

```
print("Comenzando entrenamiento...")
historial = modelo.fit(celsius, fahrenheit, epochs=1000, verbose=False)
print("Modelo entrenado!")
```

```
Comenzando entrenamiento...
Modelo entrenado!
```

```
import matplotlib.pyplot as plt
plt.xlabel("# Epoca")
plt.ylabel("Magnitud de pérdida")
plt.plot(historial.history["loss"])
```

```
[<matplotlib.lines.Line2D at 0x7fd4af0d3c90>]
```




```
print("Hagamos una predicción!")
resultado = modelo.predict([100.0])
print("El resultado es " + str(resultado) + " fahrenheit!")
```

```
Hagamos una predicción!
El resultado es [[211.74745]] fahrenheit!
```

```
print("Variables internas del modelo")
print(oculta1.get_weights())
print(oculta2.get_weights())
print(salida.get_weights())
```

```
Variables internas del modelo
[array([[-0.26688448,  0.81386125,  0.586152  ]], dtype=float32), array([2.6633756, 4.0691733, 4.0954447], dtype=float32)]
[array([[ 1.0695152 , -0.8056518 ,  0.29660237],
       [ 0.6458193 , -0.397615  ,  0.78575206],
       [ 1.0877354 , -0.27557883,  0.81521314]], dtype=float32), array([ 4.1576405, -4.012629 ,  4.1956296], dtype=float32)]
[array([[ 0.9644579],
       [-0.6735023],
       [ 0.7411787]], dtype=float32), array([3.8273103], dtype=float32)]
```

5. **Aquí deberás añadir tu código que implementa esta segunda red neuronal que convierte de grados Farenheit a Centígrados** usando los datos de entrada que se encuentran en la tabla anterior. Además, incluye capturas de pantalla que muestren cada etapa ejecutada.

6. Incluso deberás añadir **comentarios a las partes de código modificadas explicando a detalle lo que realizan.**

7. Explica con tus propias palabras los cambios que notaste al implementar esta segunda red neuronal **en comparación con tu primera red neuronal sencilla** (usa al menos 2 párrafos).

Conclusiones - anota tus conclusiones acerca del desarrollo de esta práctica.

Referencias bibliográficas