

# LABORATORIO 1

# Ejercicios

En el tutorial aprendimos sobre los componentes básicos de las redes neuronales: unidades lineales. Vimos que un modelo de una sola unidad lineal ajustará una función lineal a un conjunto de datos (equivalente a la regresión lineal). En este ejercicio, creará un modelo lineal y obtendrá algo de práctica para trabajar con modelos en Keras.

Antes de comenzar, ejecute la celda de código a continuación para configurar todo.

```
[ ]: # Setup plotting
import matplotlib.pyplot as plt

plt.style.use('seaborn-whitegrid')
# Set Matplotlib defaults
plt.rc('figure', autolayout=True)
plt.rc('axes', labelweight='bold', labelsize='large',
       titleweight='bold', titlesize=18, titlepad=10)

# Setup feedback system
from learntools.core import binder
binder.bind(globals())
from learntools.deep_learning_intro.ex1 import *
```

# Ejercicios

El conjunto de datos de la calidad del vino tinto consta de mediciones fisicoquímicas de aproximadamente 1600 vinos tintos portugueses. También se incluye una calificación de calidad para cada vino a partir de pruebas de cata a ciegas.

Primero, ejecute la siguiente celda para mostrar las primeras filas de este conjunto de datos.

[ ]:

```
import pandas as pd

red_wine = pd.read_csv('../input/dl-course-data/red-wine.csv')
red_wine.head()
```

[2]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

# Ejercicios

## *Paso1: Input shape*

¿Qué tan bien podemos predecir la calidad percibida de un vino a partir de las mediciones fisicoquímicas? El objetivo es la "calidad" y las columnas restantes son las características. ¿Cómo establecería el parámetro **input\_shape** para un modelo de Keras en esta tarea?



```
# YOUR CODE HERE  
input_shape = [11]  
# Check your answer  
q_1.check()
```

Correct

# Ejercicios

## ***Paso2: Define a linear model***

Ahora defina un modelo lineal apropiado para esta tarea. Preste atención a cuántas entradas y salidas debe tener el modelo.

```
[7]: from tensorflow import keras
      from tensorflow.keras import layers

      # YOUR CODE HERE
      model = keras.Sequential([
          layers.Dense(units=1, input_shape=[11])
      ])
      # Check your answer
      q_2.check()
```

Correct

# Ejercicios

## *Paso3: Look at the weights*

Internamente, Keras representa los pesos de una red neuronal con **tensors**. Los tensores son básicamente la versión de TensorFlow de una matriz Numpy con algunas diferencias que los hacen más adecuados para el aprendizaje profundo. Uno de los más importantes es que los tensores son compatibles con los aceleradores **GPU** y **TPU**). Las TPU, de hecho, están diseñadas específicamente para cálculos de tensores.

Los pesos de un modelo se mantienen en su atributo de pesos como una lista de tensores. Obtenga los pesos del modelo que definió anteriormente. (Si lo desea, puede mostrar los pesos con algo como: **print ("Weights \n {} \n \n Bias \n {}". Format (w, b)))**).

[14]:

```
# YOUR CODE HERE
w, b = model.weights

# Check your answer
q_3.check()
```

**Correct:** Do you see how there's one weight for each input (and a bias)? Notice though that there doesn't seem to be any pattern to the values the weights have. Before the model is trained, the weights are set to random numbers (and the bias to 0.0). A neural network learns by finding better values for its weights.

(Por cierto, Keras representa pesos como tensores, pero también usa tensores para representar datos. Cuando configuras el argumento `input_shape`, le estás diciendo a Keras las dimensiones de la matriz que debe esperar para cada ejemplo en los datos de entrenamiento. Configurando `input_shape = [ 3]` crearía una red aceptando vectores de longitud 3, como `[0.2, 0.4, 0.6]`.)

# Ejercicios

## ***Optional: Plot the output of an untrained linear model***

Los tipos de problemas en los que trabajaremos a lo largo de la Lección 5 serán problemas de regresión, donde el objetivo es predecir algún objetivo numérico. Los problemas de regresión son como problemas de "ajuste de curvas": estamos tratando de encontrar una curva que se ajuste mejor a los datos. Echemos un vistazo a la "curva" producida por un modelo lineal. (¡Probablemente hayas adivinado que es una línea!)

Mencionamos que antes de entrenar, los pesos de un modelo se establecen al azar. Ejecute la celda de abajo varias veces para ver las diferentes líneas producidas con una inicialización aleatoria. (No hay codificación para este ejercicio, es solo una demostración).



```
import tensorflow as tf
import matplotlib.pyplot as plt

model = keras.Sequential([
    layers.Dense(1, input_shape=[1]),
])

x = tf.linspace(-1.0, 1.0, 100)
y = model.predict(x)

plt.figure(dpi=100)
plt.plot(x, y, 'k')
plt.xlim(-1, 1)
plt.ylim(-1, 1)
plt.xlabel("Input: x")
plt.ylabel("Target y")
w, b = model.weights # you could also use model.get_weights() here
plt.title("Weight: {:.2f}\nBias: {:.2f}".format(w[0][0], b[0]))
plt.show()
```