DL_02_MLP

Brandon Francisco Hernández Troncoso

== Q1 ==

Revisa la documentación de tensorflow: ¿qué tipo de activación tiene la capa de salida de la red que acabamos de crear?

R= Activación lineal

== Q2 ==

Según la documentación de tensorflow (y/o el código de la siguiente celda), ¿Qué extructura de datos usa tensorflow para organizar las capas de una red?

R= Tensores

== Q3 ==

En el ejercicio anterior le indicamos a tensorflow que separara 20% de los datos de entrenamiento para hacer validación. Dicho 20% es seleccionado de manera aleatoria. Revisa la documentación de la función .fit() para saber cómo puedes darle un subset de validación que no cambie entre diferentes corridas de entrenamiento. Reporta tu respuesta.

R= Con el argumento **validation_data** y eliminando el argumento **validation_split**.

== Q4 ==

Nota que esta segunda red, aunque más grande, parece generar peores resultados en comparación con la primera red. Encuentra la mejor combinación de hiperparámetros para minimizar el error tanto como sea posible. Reporta: número de capas, número de perceptrones en cada capa, número total de pesos, tamaño del lote, y número de épocas.

Se crearon 3 combinaciones de red. Se reporta a continuación la información solicitada para las arquitecturas creadas y para los mejores modelos:

Modelo 1 [2, 32, 16, 8, 4, 2]:

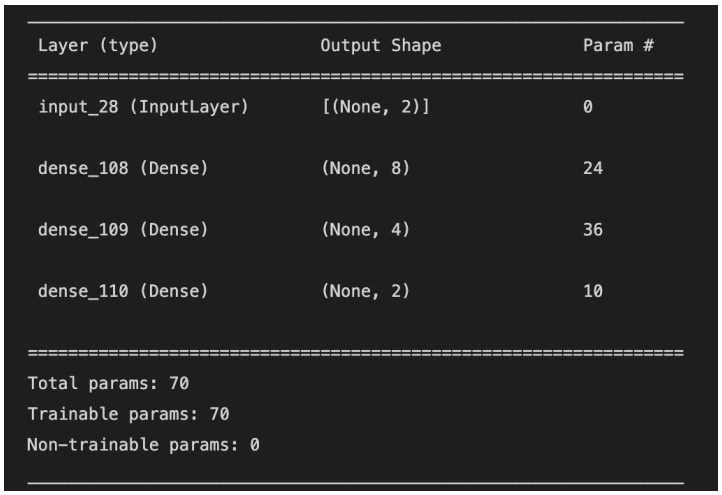
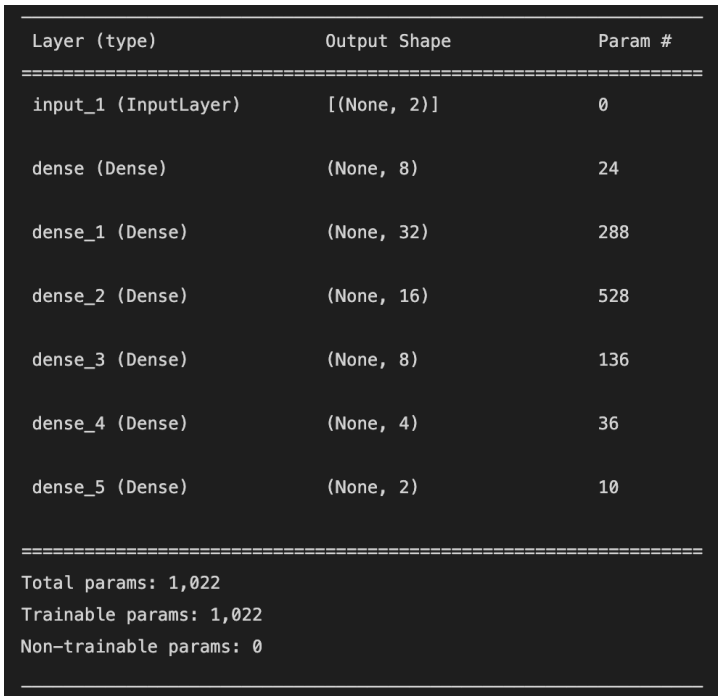


Figure 1: A nice image.

Mejores resultados:

1) [2, 32, 16, 8, 4, 2]; batch_size=4, epochs=200, validation_split=0.2 -> Average – train loss: 4.8648 +- 3.4672; test loss: 5.2218 +- 4.2117
2) [2, 32, 16, 8, 4, 2]; batch_size=4, epochs=200, validation_split=0.1 -> Average – train loss: 4.6441 +- 3.5963; test loss: 4.9694 +- 4.3508
3) [2, 32, 16, 8, 4, 2]; batch_size=2, epochs=200, validation_split=0.1 -> Average – train loss: 3.6965 +- 4.1280; test loss: 4.1307 +- 4.7731

Modelo 2: [2, 8, 32, 16, 8, 4, 2]



Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 2)]	0
dense (Dense)	(None, 8)	24
dense_1 (Dense)	(None, 32)	288
dense_2 (Dense)	(None, 16)	528
dense_3 (Dense)	(None, 8)	136
dense_4 (Dense)	(None, 4)	36
dense_5 (Dense)	(None, 2)	10

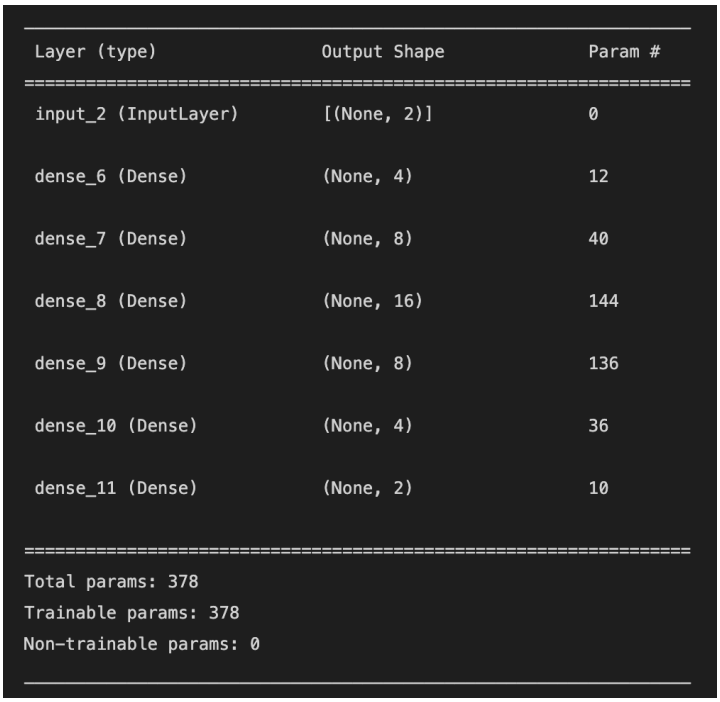
Total params: 1,022
Trainable params: 1,022
Non-trainable params: 0

Figure 2: A nice image.

Mejores resultados:

- [2, 8, 32, 16, 8, 4, 2]; batch_size=4, epochs=200, validation_split=0.2 -> Average – train loss: 4.6602 +- 3.3445; test loss: 4.9514 +- 4.0859

Modelo 3: [2, 4, 8, 16, 8, 4, 2]



Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 2)]	0
dense_6 (Dense)	(None, 4)	12
dense_7 (Dense)	(None, 8)	40
dense_8 (Dense)	(None, 16)	144
dense_9 (Dense)	(None, 8)	136
dense_10 (Dense)	(None, 4)	36
dense_11 (Dense)	(None, 2)	10

Total params: 378
Trainable params: 378
Non-trainable params: 0

Figure 3: Resumen modelo 3.

Mejores resultados:

- [2, 4, 8, 16, 8, 4, 2]; batch_size=4, epochs=200, validation_split=0.2 -> Average – train loss: 5.3751 +- 3.3875; test loss: 5.8204 +- 4.1399

== Q5 ==

De acuerdo con los resultados obtenidos después de correr el modelo 100 veces, ¿pareciera que el modelo es confiables/estable? ¿por qué si o por qué no?

R= No pareciera ser confiable porque los errores son muy pequeños o muy grandes según el conjunto de datos usado para el entrenamiento y la validación.