Tarea 3

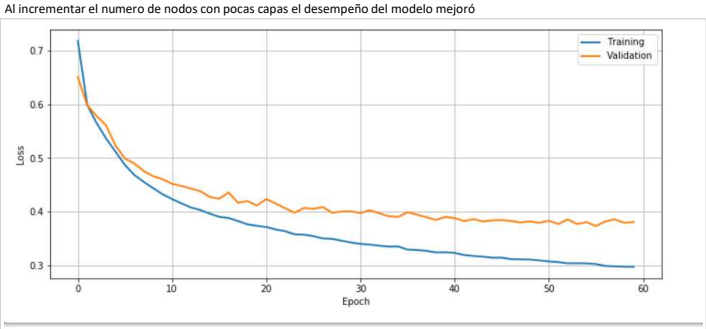
Monday, January 20, 2020 11:30 AM

Javier Valencia Goujon 123227
Alfie Sergio González Salcedo 181574

1) ¿Hay algún efecto si normalizas o dejas de normalizar tus datos?
Si hay un efecto, al no normalizar los datos la red NEURONAL tiene un peor desempeño ya que los gradientes pueden oscilar de un lado a otro y tardar mucho tiempo antes de que finalmente pueda llegar al mínimo global / local. El tamaño de paso muy fácilmente puede pasar de ser muy pequeño a ser muy grande y nunca converger por la diferencia en los órdenes de magnitud de las variables.

2) ¿Cómo se comporta la distancia entre el error de entrenamiento y el de validación al incrementar o disminuir el número de capas?
Con la configuración inicial estaba **haciendo overfitting** desde la época 30 aproximadamente, es decir que la distancia entre Train y validation test era alta.
La intuición nos dice que debemos de reducir la complejidad del modelo y reducir el número de capas o de neuronas.

3) ¿Y al incrementar o disminuir el número de nodos en cada capas?
Al incrementar el numero de nodos con pocas capas el desempeño del modelo mejoró




```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_55 (Dense)             (None, 64)                1664
_________________________________________________________________
dense_56 (Dense)             (None, 128)               8320
_________________________________________________________________
OUTPUT (Dense)               (None, 2)                 258
=================================================================
Total params: 10,242
Trainable params: 10,242
Non-trainable params: 0
_________________________________________________________________
```

```
# Define training parameters
DNN.compile(optimizer='adam', loss='mse')
```

4) ¿Hay algún impacto en el desempeño si cambias el tamaño de los lotes?
Si cambio el tamaño de lote cambia la velocidad a la que corre el modelo. Si se disminuye el tamaño de lote tardan más las iteraciones.

5) ¿Cuál es la intuición para usar la métrica de generalización que está en la celda debajo de la gráfica con las curvas de desempeño por época?
La métrica sirve para medir cuánto se está sobre parametrizando el modelo.

6) Incluye una tabla comparativa para tus 5 mejores métodos. Compáralos en términos de los hiperparámetros seleccionados y las pérdidas de entrenamiento y validación?
NOTA: Puse mis 5 peores métodos para mostrar cómo no hacer una red neuronal

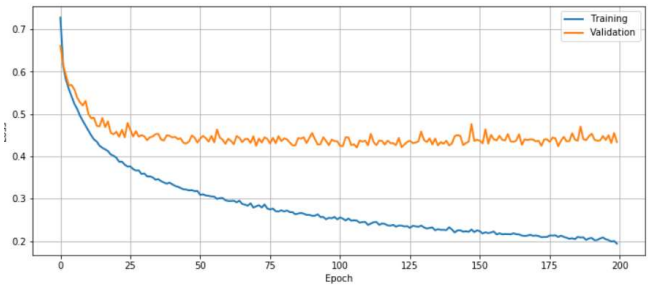


Figura 1: configuración inicial

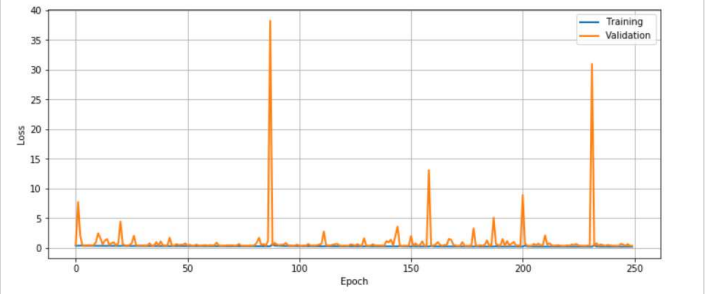


Figura 2: optimizador **sgd,** 3 capas , 250 epocas

```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_48 (Dense)             (None, 64)                1664
_________________________________________________________________
dense_49 (Dense)             (None, 128)               8320
_________________________________________________________________
dense_50 (Dense)             (None, 64)                8256
_________________________________________________________________
OUTPUT (Dense)               (None, 2)                 130
=================================================================
Total params: 18,370
Trainable params: 18,370
Non-trainable params: 0
_________________________________________________________________
```

```
# Define training parameters
DNN.compile(optimizer='sgd', loss='mse')
```

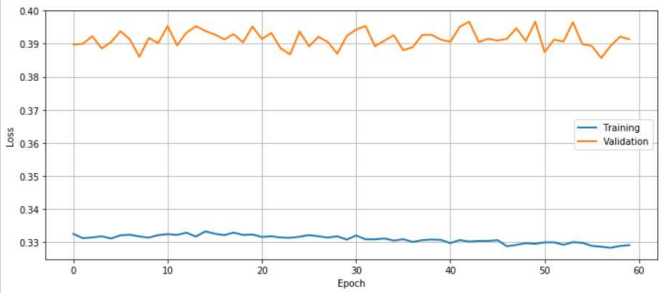


Figura3: modelo Underfitting, con sesgo

```
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_53 (Dense)             (None, 32)                832
_________________________________________________________________
dense_54 (Dense)             (None, 32)                1056
_________________________________________________________________
OUTPUT (Dense)               (None, 2)                 66
=================================================================
Total params: 1,954
Trainable params: 1,954
Non-trainable params: 0
_________________________________________________________________
```

```
[51] # Define training parameters
     DNN.compile(optimizer='adam', loss='mse')
```

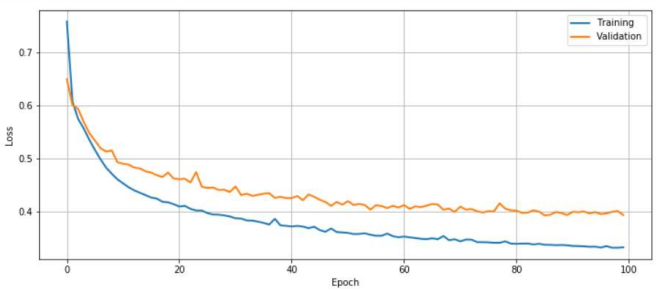
7) ¿La pérdida de test de tu mejor modelo es comparable con la de validación y entrenamiento? ¿Qué sugiere eso?

0.392244101263756	Sí, es similar al error de validación. Esto sugiere que el modelo no está tan sobre parametrizado.

8) Reporta el tiempo promedio que tardaba en correr cada uno de tus experimentos.
2 minutos

9) Incluye gráficas de desempeño de tus mejores modelos.

El mejor modelo fue una red con 2 capas , con la siguiente arquitectura:



time: 207 ms

El error de validación alrededor de .40

Generalization loss 0.1552%

```
[ ] # Generalization metric
    min_loss = np.min([DNN.history.history['loss'][-1], DNN.history.history['val_loss'][-1]])
    max_loss = np.max([DNN.history.history['loss'][-1], DNN.history.history['val_loss'][-1]])
    general_loss = (max_loss - min_loss) / max_loss
    print(f"Generalization loss: {general_loss:6.4f}%")
```

```
Generalization loss: 0.1552%
time: 3.46 ms
```

```
[ ] # Compute test loss
    loss_test = DNN.evaluate(x=x_test, y=y_test, verbose=False)
    print("Test loss:", loss_test)
```

```
Test loss: 0.392244101263756
time: 94.8 ms
```

```
[ ] # Predict on test set
    y_test_hat = DNN.predict(x=x_test)
    print(y_test_hat.shape)
```

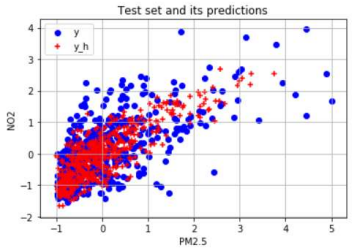
```
(1591, 2)
time: 76.8 ms
```

```
Model: "sequential_18"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_76 (Dense)             (None, 45)                1170
_________________________________________________________________
dense_77 (Dense)             (None, 32)                1472
_________________________________________________________________
OUTPUT (Dense)               (None, 2)                 66
=================================================================
Total params: 2,708
Trainable params: 2,708
Non-trainable params: 0
_________________________________________________________________
time: 66.4 ms
```

```
# Define training parameters
DNN.compile(optimizer='adam', loss='mse')
```

Tamaño de batch **64**

```
# Train model
history = DNN.fit(x=x_train, y=y_train, batch_size=64, epochs=100, verbose=1, validation_split=0.1, shuffle=True)
```



time: 232 ms