Miniproyecto 2:
Multilayer Perceptron

Por: David Damián Arbeu

Enero de 2023

PREGUNTA

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

SOLUCIÓN

Como se especificó 'None' para la función de activación, significa que se usa una función de activación lineal particular, a decir la función identidad. Más precisamente, que los *inputs* son transformados vía la siguiente función:

$$s(\mathbf{x}_{sepal}) = \mathbf{x}_{sepal}$$

PREGUNTA

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

SOLUCIÓN

Una matriz, cuyas filas son el vector de pesos asociado a cada capa de la red neuronal.

PREGUNTA

3 Revisa la documentación de la función .fit() para saber cómo puedes darle un subset de validación que no cambie entre diferentes corridas de entrenamiento.

SOLUCIÓN

Si se especifica el argumento de validation_data (cuyo formato son np.arrays o tensores) en los argumentos de Model.fit, entonces dichos datos se usarán en cada época para evaluar pérdida o cualquier otra métrica al final de cada época.

PREGUNTA

4 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, y error obtenido en entrenamiento y validación.

SOLUCIÓN

Después de experimentar con diferentes arquitecturas para la red, la que mejor resultados me dió fue una red neuronal con los siguientes hiperparámetros.

- Número de capas: 3. La capa de entrada, una intermedia y la capa de salida.
- Número de perceptrones: 2 en la capa de entrada y 64 en la segunda capa(capa intermedia)
- número total de pesos: 64*4= 256
- Tamaño de lote: 4
- Número de épocas: 60
- Error promedio (train, val)= (0.24, 0.89)

PREGUNTA

5 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, y error obtenido en entrenamiento y validación.

SOLUCIÓN

El modelo parece no ser estable pues tiene un comportamiento *nervioso* . Podemos ver que en cada corrida hay mucha variabilidad del error en el conjunto de entrenamiento tanto como en el de validación.

Al final ésto lo podemos comprobar, de modo que el error promedio en entrenamiento es $\overline{E}_{train} = 5.71$ y varianza $\sigma^2_{train-err} = 3.36^2$ y en validación error promedio $\overline{E}_{test} = 8.19$ y varianza $\sigma^2_{train-err} = 5.23^2$