Ejercicio Tensorflow 1
https://playground.tensorflow.org/

1. **Capas y patrones:** Prueba a entrenar la red neuronal por defecto haciendo clic en el botón de ejecución (arriba izquierda). Observa cómo encuentra rápidamente una buena solución para la tarea de clasificación. Observa que las neuronas de la primera capa oculta han aprendido patrones simples, mientras que las neuronas de la segunda capa oculta han aprendido a combinar los patrones simples de la primera capa oculta en patrones más complejos. En general, cuantas más capas, más complejos pueden ser los patrones.

2. **Función de activación:** Sustituye la función de activación Tanh por la función de activación ReLU y vuelve a entrenar la red. Observa que encuentra una solución aún más rápido, pero ésta vez los límites son lineales. Esto se debe a la forma de la función ReLU.

3. **Mínimos locales:** Modifica la arquitectura de la red para tener una sola capa oculta con tres neuronas. Entrénela varias veces (para restablecer los pesos de la red, haz clic en el botón de reset situado junto al botón de reproducción). Observa que el tiempo de entrenamiento varía mucho, y a veces incluso se queda atascado en un mínimo local.

4. **Demasiado pequeño:** Ahora elimina una neurona para quedarte con sólo 2. Observa que la red neuronal es ahora incapaz de encontrar una buena solución, aunque lo intente varias veces. El modelo tiene muy pocos parámetros y se adapta sistemáticamente al conjunto de entrenamiento.

5. **Suficientemente grande:** A continuación, ajusta el número de neuronas a 8 y entrena la red varias veces. Fíjate en que ahora es consistentemente rápida y nunca se atasca. Esto pone de manifiesto una importante cuestión en la teoría de las redes neuronales: las redes neuronales grandes casi nunca se quedan atascadas en los mínimos locales, e incluso cuando lo hacen, estos óptimos locales son casi tan buenos como el óptimo global. Sin embargo, pueden quedarse atascadas durante mucho tiempo para salir de “una meseta”.

6. **Red profunda y “gradientes de fuga”:** Ahora cambia el conjunto de datos para que sea la espiral (abajo a la derecha en "DATA"). Cambia la arquitectura de la red para que tenga 4 capas ocultas con 8 neuronas cada una. Observa que el entrenamiento es mucho más largo, y a menudo se queda atascado en las mesetas durante largos periodos de tiempo. Observa también que las neuronas de las capas más altas (es decir, las de la derecha) tienden a evolucionar más rápido que las neuronas de las capas más bajas (es decir, las de la izquierda). Este problema, llamado , denominado "gradientes de fuga", puede aliviarse mediante una mejor inicialización de los pesos y otras técnicas, mejores optimizadores (como AdaGrad o Adam), o utilizando Batch Normalización.

NOTA: Gradiente de fuga (desvanecimiento del gradiente): A medida que aumentan las capas de red, el valor derivado de la retroprogagación disminuye, o sea, que el gradiente se va desvaneciendo a valores muy bajos, por lo que los pesos no varian o lo hacen de manera muy pequeña.