Revisión y exploración herramienta playground.tensorflow

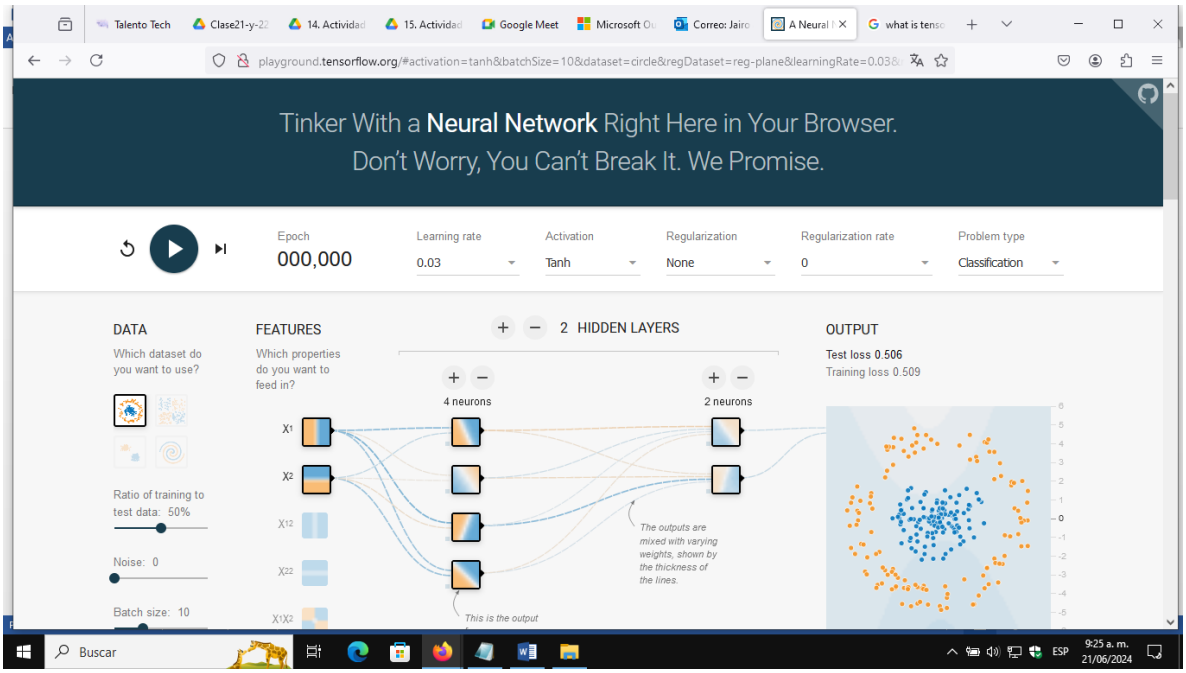
Que es TensorFlow

TensorFlow es una biblioteca de software de código abierto para la programación de redes neuronales y el aprendizaje automático. Fue desarrollada por Google Brain Team y se lanzó al público en 2015.

TensorFlow permite a los desarrolladores construir modelos de aprendizaje profundo de forma eficiente, utilizando una arquitectura flexible basada en grafos que puede ejecutarse en una variedad de plataformas, como CPU, GPU y TPU. La biblioteca proporciona una amplia gama de herramientas y técnicas para el preprocesamiento de datos, la creación y entrenamiento de modelos, la evaluación del rendimiento y la implementación en producción.

TensorFlow se utiliza en una amplia variedad de aplicaciones de aprendizaje automático, como el procesamiento del lenguaje natural, el reconocimiento de voz y de imágenes, la clasificación y segmentación de datos, entre otros. Además, TensorFlow tiene una comunidad de usuarios y desarrolladores activos que contribuyen al desarrollo y mejora de la biblioteca, lo que la convierte en una herramienta potente y en constante evolución para la construcción de modelos de aprendizaje automático.

https://gamco.es/glosario/tensorflow/



Elementos de la herramienta:

Data: conjunto de datos de la herramienta

Ratio of training: Proporción de datos de entrenamiento

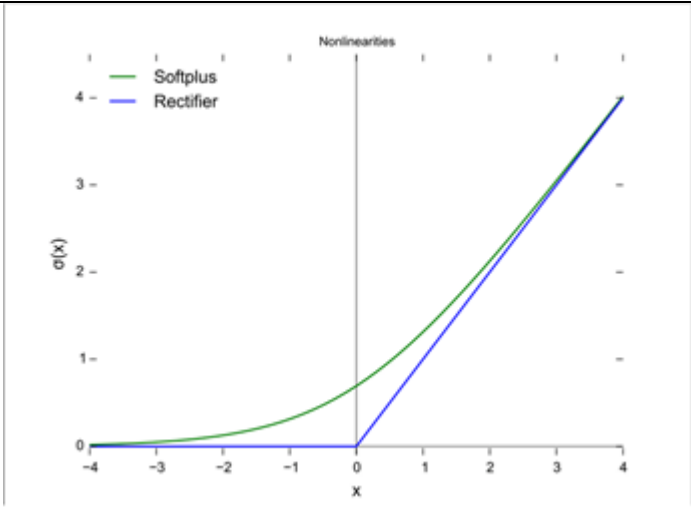
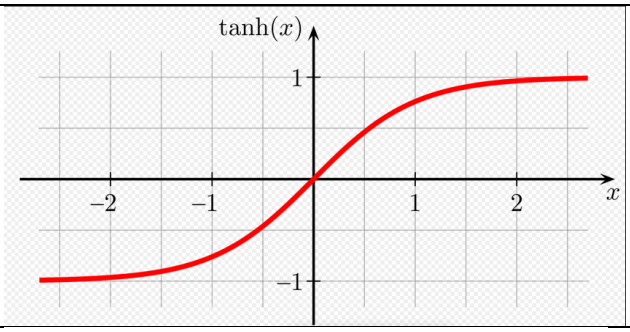
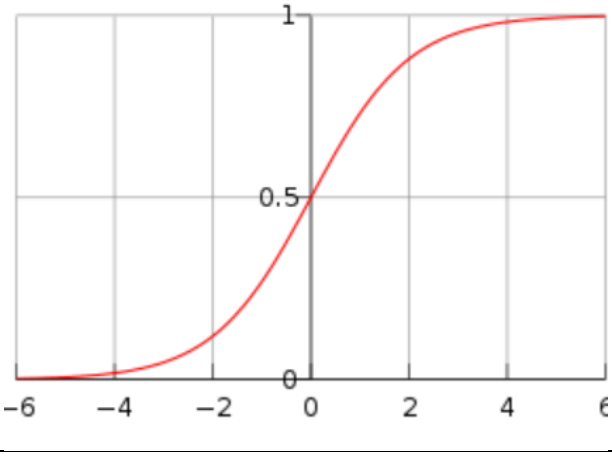
Noise: Ruido en el proceso

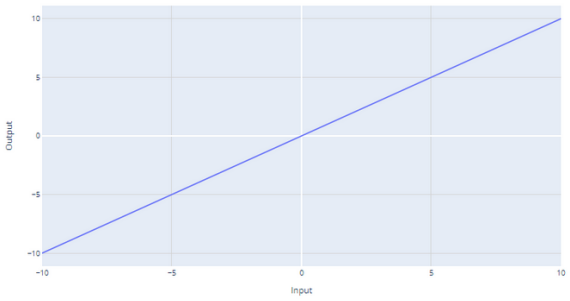
Features: Características de entrada para análisis

Neuronas por nivel: número de neuronas por nivel

Hidden layers: número de capas ocultas

Activation: Funciones de activación la **función de activación** de un nodo define la salida de un nodo dada una entrada o un conjunto de entradas. La función de activación es usualmente una abstracción representando una tasa de potencial de activación gatillándose en la celda. En su forma simplificada, esta función es binaria, esto es, se activa la neurona o no.

RELU Rectified Lineal Unit Se utiliza en redes neuronales convolucionales, para obtener resultados de menor error que los generados con función logística (la cual es inspirada por la teoría de la probabilidad) y existe evidencia que es más práctica que su contraparte, la función hiperbólica. El rectificador es una función de activación común en redes neuronales profundas. Se utiliza en redes neuronales convolucionales, para obtener resultados de menor error que los generados con función logística (la cual es inspirada por la teoría de la probabilidad) y existe evidencia que es más práctica que su contraparte, la función hiperbólica. El rectificador es una función de activación común en redes neuronales profundas.	
Tanh: Tangente hiperbólica Las funciones hiperbólicas son unas funciones cuyas definiciones se basan en la función exponencial, están ligadas entre sí mediante operaciones racionales y son análogas a las funciones trigonométricas.	
Sigmoid: Sigmoide Se utiliza como una función de activación para las capas ocultas. Esta función toma un valor de entrada y lo comprime en un rango comprendido entre 0 y 1, lo que la convierte en una función útil para modelar probabilidades y realizar clasificaciones binarias. La función sigmoide tiene una forma característica de «S» que le da su nombre.	

Linear: Lineal Devuelve la entrada x como salida. Gráficamente, es una línea recta con una pendiente de 1. El principal caso de uso es en la capa de salida de una red neuronal utilizada para la regresión., utilizar una función de activación lineal en la capa de salida garantiza que la red neuronal produzca un valor numérico. La función de activación lineal no reduce ni transforma la salida, por lo que se devuelve el valor real previsto. Sin embargo, la función de activación lineal rara vez se utiliza en las capas ocultas de las redes neuronales. Esto se debe a que no proporciona ninguna no linealidad. El objetivo de las capas ocultas es aprender combinaciones no lineales de las características de entrada. Utilizar una activación lineal todo el tiempo restringiría el modelo a aprender solo transformaciones lineales de la entrada.	

https://www.datacamp.com/es/tutorial/introduction-to-activation-functions-in-neural-networks

Otras funciones de activación

https://es.linkedin.com/pulse/la-importancia-de-las-funciones-activaci%C3%B3n-en-una-red-calvo-martin

https://www.datacamp.com/es/tutorial/introduction-to-activation-functions-in-neural-networks

https://jacar.es/la-funcion-sigmoide-una-herramienta-clave-en-redes-neuronales/

Regularization

Previene las altas variaciones u el sobreajuste: Castigando los pesos por capa con el valor absoluto o la norma en cada peso

REGULARIZACIÓN L1 Y L2

Regularización L1 (Lasso Regression):

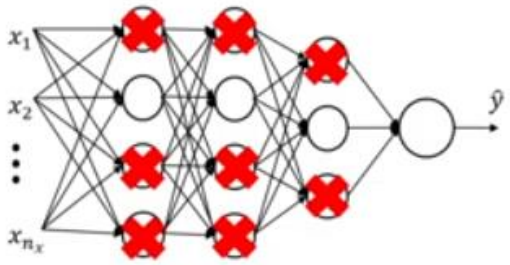
$$\mathcal{J}(\omega, b) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\sum_{l=1}^{L}|W^{[l]}| \qquad Donde: \quad |W^{(l)}| = \sum_{i=1}^{n^{(l)}}\sum_{j=1}^{n^{(l-1)}}|W_{ij}^{[l]}|$$

Regularización L2 (Ridge Regression):

$$\mathcal{J}(\omega, b) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\sum_{l=1}^{L}||W^{[l]}||^2$$

En la regularización al castigar los pesos, estos tienden acero y a desactivar neuronas y el modelo no aprende tantas características de los datos de entrenamientos obteniendo en algunos casos una mejor exactitud con la realidad. La regulación puede activar el problema de alta variación.

FUNCIONAMIENTO



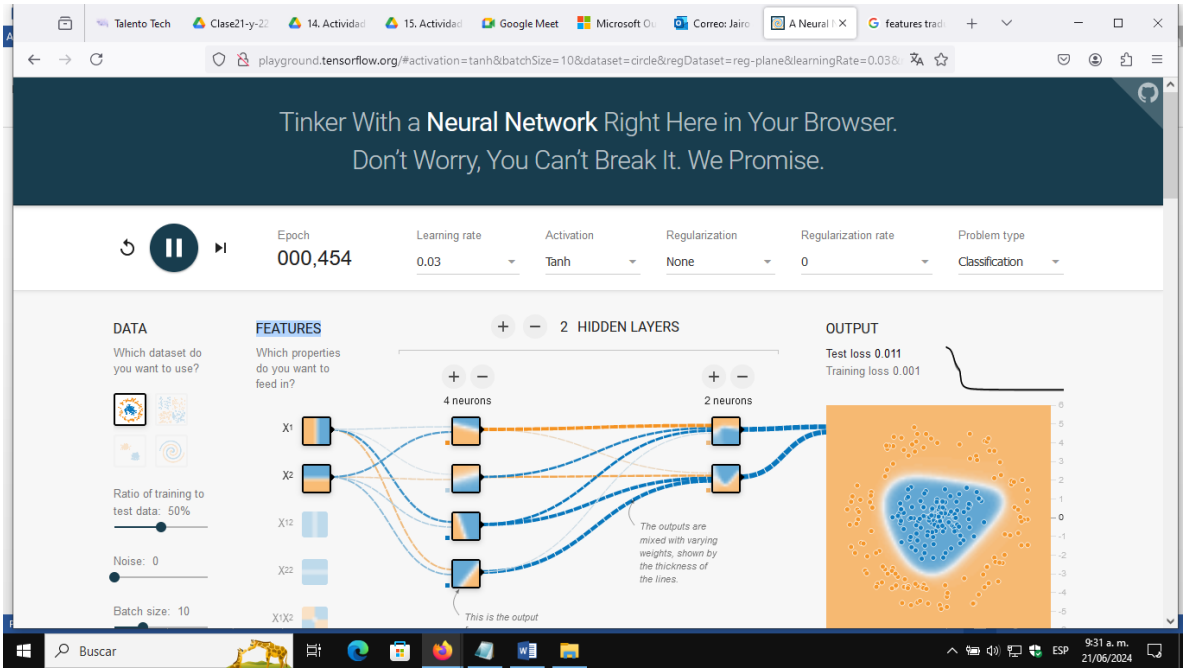
$$\mathcal{J}(\omega, b) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\sum_{l=1}^{L}||W^{[l]}||^2$$

$W \approx 0$ →

[Regularization in Neural Networks | Pinecone]

https://www.youtube.com/watch?v=KR8WW7DxY4A

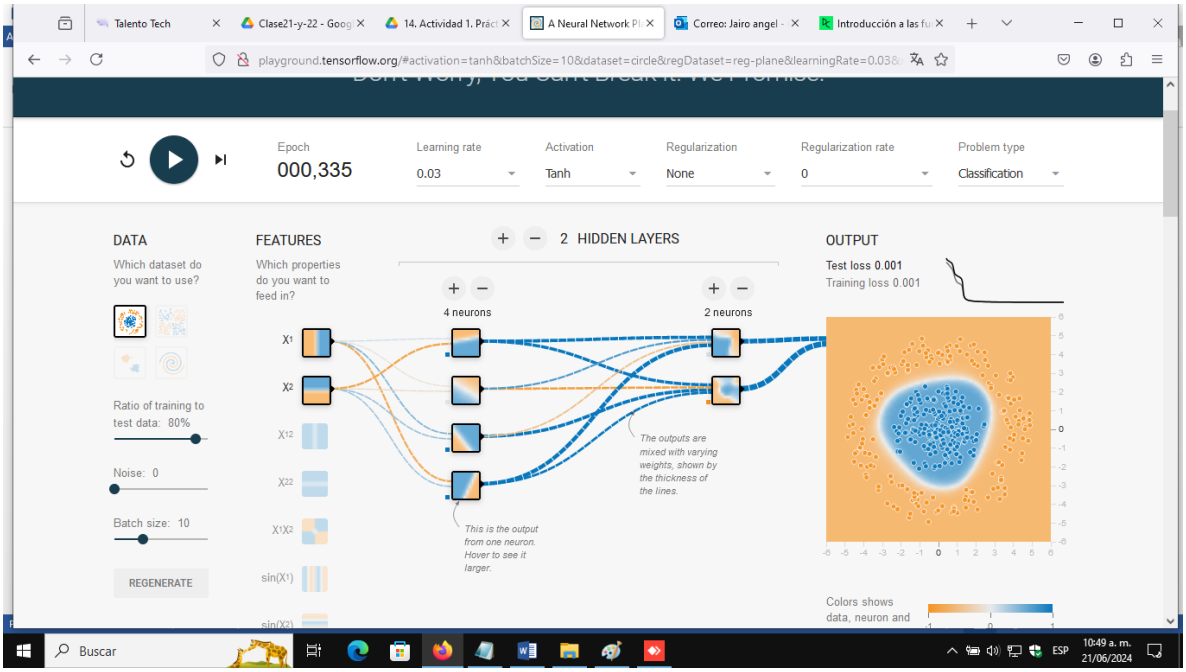
Ejecuciones



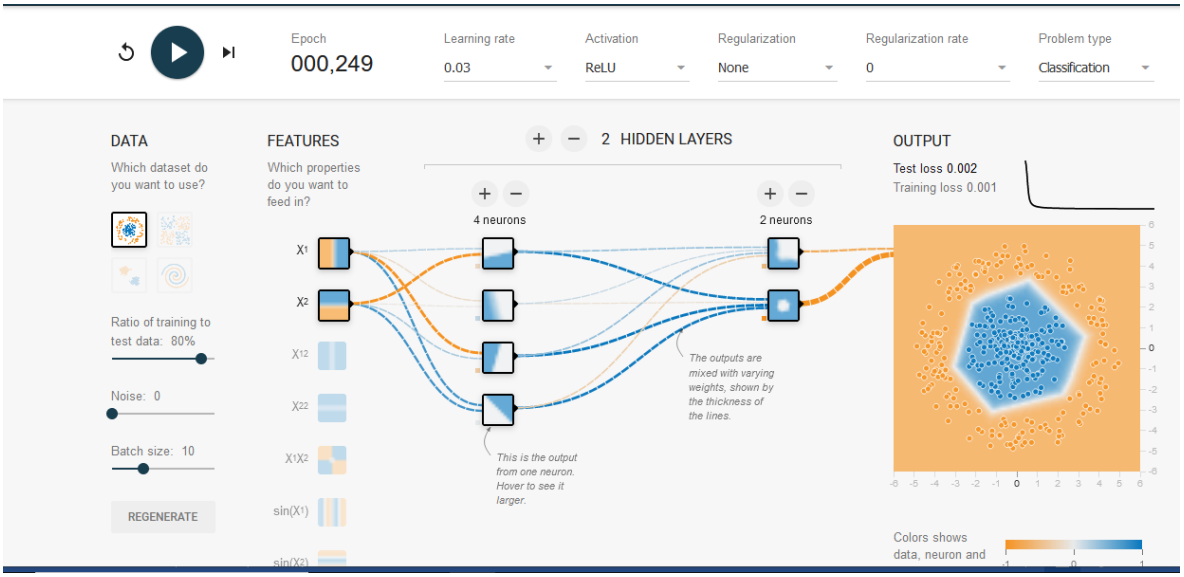
Taza de entrenamiento: 80% , sin regularización

Tipo de problema clasificación

Tanh

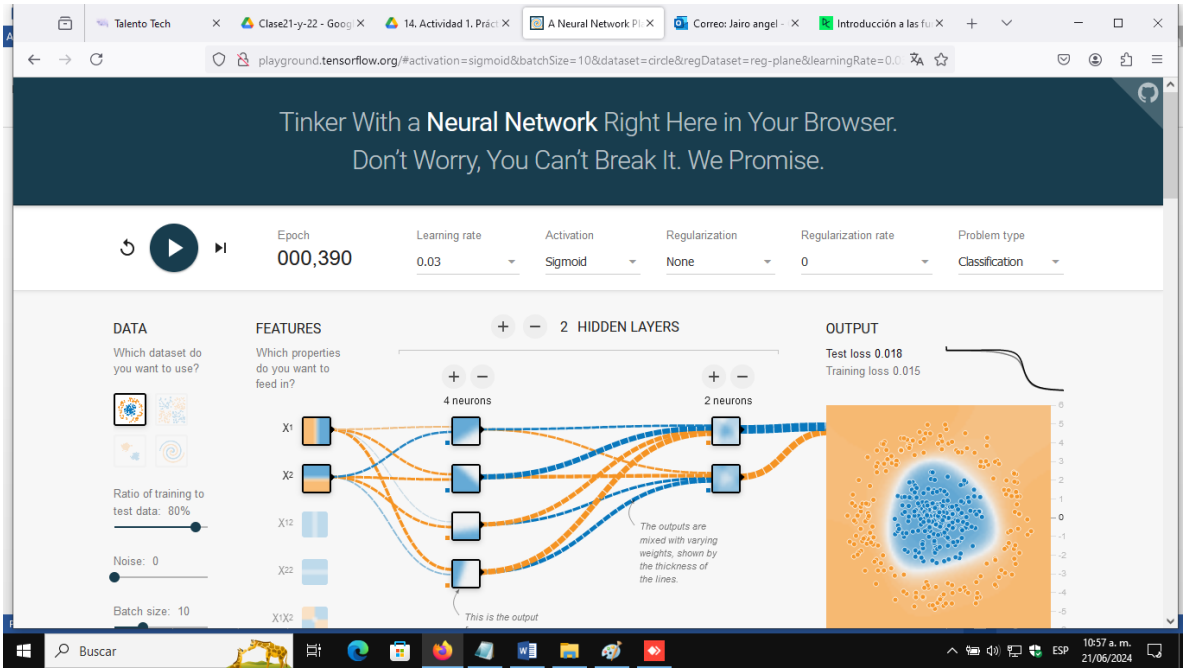


ReLU



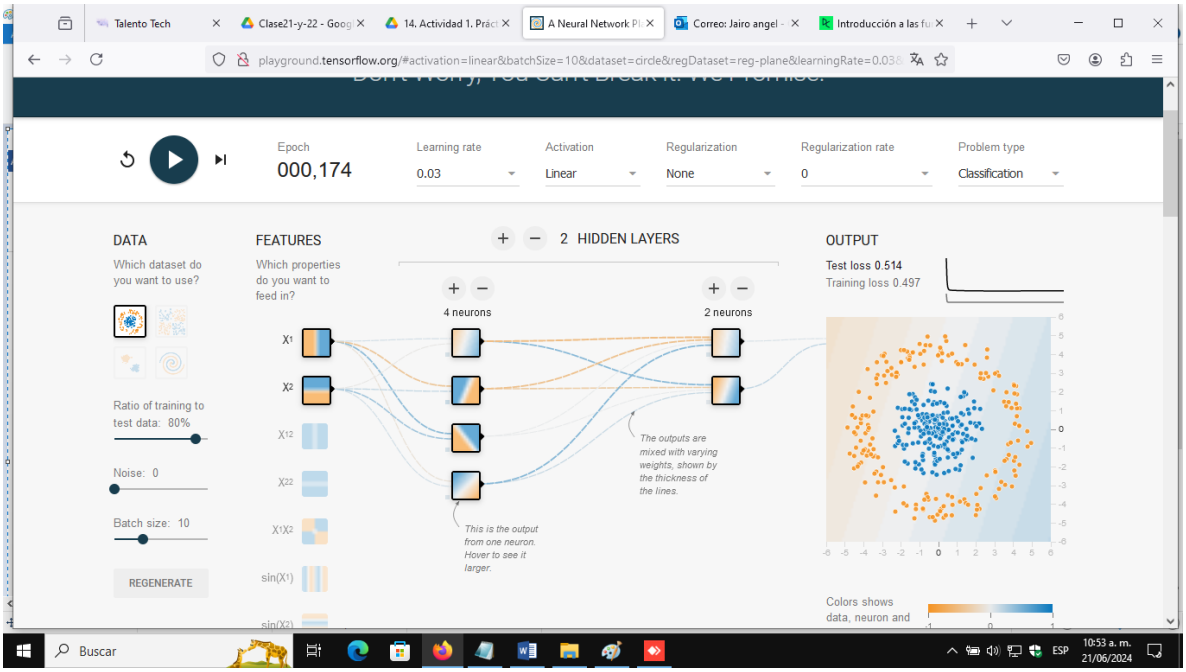
Convergencia rápida con pocas iteraciones a un error bajo

Sigmoid



Un proceso diferente de convergencia y la gráfica del error

Linear



Proceso de convergencia rapido

Conclusión: parecería que para procesos de clasificación la ReLU y la lineal

Tiene un mejor comportamiento en convergencia y error final para dos entradas y una capa oculta.

Jairo Angel