Universidad del Valle de Guatemala
Departamento de Matemática
Licenciatura en Matemática Aplicada

Estudiante: Rudik Roberto Rompich
Correo: rom19857@uvg.edu.gt
Carné: 19857

CC3066 - Data Science I - Catedrático: Luis Furlan
2 de septiembre de 2021

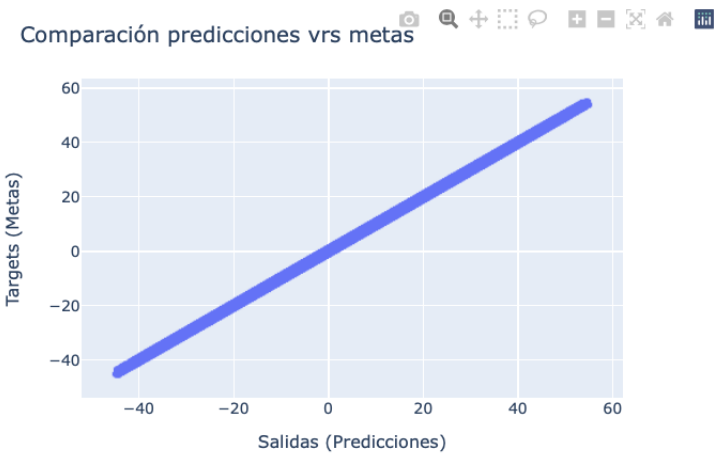
Laboratorio 5

Instrucciones: en clase vimos un modelo simple de una red neuronal utilizando *TensorFlow* 2. Utilizando el código desarrollado (o si lo desea uno propio pero que funcione correctamente), responda a las siguientes preguntas:

Problema 1. *Cambie el número de observaciones a 100,000. ¿Qué ocurre?*

Solución. Los resultados a simple vista parecen ser lo mismos; tales que

1. La cantidad de observaciones hace que el modelo sea un poco más lento y mucho más tardado.
2. El error parece estabilizarse bastante más rápido y converge aproximadamente a un mismo número.
3. Los pesos y sesgos no variaron en nada, se mantienen cercanos a las cifras esperadas.
4. La gráfica del problema sigue la misma tendencia (solo que con más datos).
5. Cabe recalcar que estamos en un caso de *overfitting* ya que se usaron los mismos datos de entrenamiento.
6. La gráfica:



□

Problema 2. *«Juegue» un poco con la tasa de aprendizaje. Los valores como 0.0001, 0.001, 0.1, 1 son interesantes para observar ¿Qué diferencias se observan? ¿Se comporta bien el algoritmo?*

Solución. Tenemos los siguientes casos:

0.0001 (1) El algoritmo arroja pesos y sesgos bastante precisos. (2) El algoritmo tomó un poco más de tiempo en ejecutarse, comparado a la tasa de aprendizaje anterior. (3) La pérdida se estabilizó completamente en 0.3323.

0.001 (1) El algoritmo arroja pesos y sesgos bastante precisos (aunque por décimas impreciso a lo esperado). (2) El algoritmo fue un poco más eficiente en ejecutarse, comparado a la tasa de aprendizaje del inciso anterior. (3) La pérdida se estabilizó completamente en 0.3333.

0.1 El modelo explotó; la tasa de aprendizaje es muy grande.

1 El modelo explotó; la tasa de aprendizaje es muy grande.

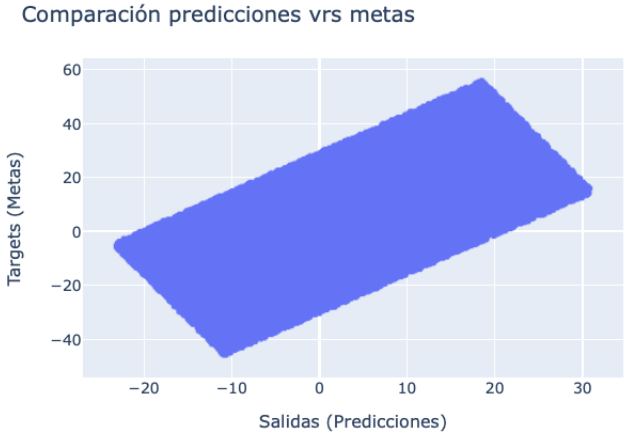
□

Problema 3. *Cambie la función de pérdida. Una función altenativa es la "Huber Loss".*

La función de pérdida Huber es más adecuada que la L2.norm cuando tenemos valores atípicos, ya que es menos sensitiva a los mismos (en nuestro ejemplo no tenemos valores atípicos, pero seguramente se topará con ellos en el futuro). La L2-norm eleva todas las diferencias al cuadrado, por lo que los valores atípicos tienen mucha influencia sobre los resultados. La sintáxis correcta de la función de pérdida Huber es "huber_loss".

¿Cómo se comparan los resultados al cambiar la función de pérdida?

Solución. Por los datos de entrenamiento «idealistas» que se usaron para hacer las predicciones, ya que fueron los mismos que para la construcción del modelo; se pensaría que no habría un efecto directo en los resultados. Sin embargo, con una tasa de aprendizaje de 0.02 se produce la siguiente gráfica, con los puntos más homogéneamente distribuidos:



□

Referencia: https://www.tensorflow.org/versions/r1.15/api_docs/python/tf/keras