IMPLEMENTACION DE UN MODELO DE MACHINE LEARNING USANDO FRAMEWORK

Kevin Joan Delgado Pérez – A01706328

Resumen

Se pretende analizar y documentar el proceso de desarrollo de una red neuronal usando el framework Tensorflow con el motivo de realizar una regresión lineal para predecir los valores en el conteo de bicicletas rentadas en la ciudad de Seúl, Corea del Sur.

Introducción

Un modelo de regresión lineal pretende predecir valores que forman parte de un mismo set de datos, para generar el modelo es necesario indicar las variables dependientes e independientes del modelo, con estas variables, el algoritmo será entrenado para predecir los valores del set.

El dataset contiene distintos parámetros, en este caso, el modelo considerará sólo los siguientes:

1. Variable dependiente:
    - Conteo de bicicletas rentadas
2. Variables independientes
    - Hora del día
    - Temperatura en grados Celsius
    - Velocidad del viento en metros por segundo
    - Temporada (Esta variable es categórica, por lo cual se usaron dummy variables para poder analizarse, las opciones son, Spring, Summer, Autumn y Winter)

Descripción del método

Para implementar el método primero es necesario extraer el set de datos hacia un dataframe mediante la librería Pandas, después, verificar la información que contiene mediante la página donde obtuvimos la base de datos, en este caso, fue obtenido del UCI Machine Learning Repository. (UCI, 2020)

Un poco de la información de contexto para contiene la página de donde fue extraída la base de datos, por lo cual se conoce el significado de cada variable.

Dentro de nuestras variables se tiene una variable categórica, debido a esto, se utiliza la función de pandas llamada *get_dummies* para generar columnas igual al numero de categorías, de esta manera se podrán leer esos datos que nos pueden ayudar a generar un mejor modelo, teniendo el dataframe limpio y listo para ser trabajado, se realiza una separación de datos mediante *pd.sample* donde utilizamos el 75% de los datos para ser entrenados y el retos de la información será utilizada para las pruebas. Antes de comenzar a normalizar los datos para ser entrenados, fueron separadas las columnas de las variables independientes de la variable dependiente y se imprime las estadísticas descriptivas del set de entrenamiento. Posteriormente, se normalizan los datos para mejorar su calidad y agilizar el proceso de entrenamiento, también se imprimen parte de ellos para verificar que no se tengan incongruencias. Para iniciar el entrenamiento primero se diseña la red neuronal, donde sus características fueron:

- 1 input con el tamaño del número de variables independientes.
- 3 niveles con 64 neuronas.
- 1 nivel con 8 neuronas.
- Los niveles de neuronas utilizarán la función de activación Relu.
- 1 output con 1 neurona para definir el modelo.

- Modelo de optimización Adam con un learning rate de 0.03.
- Funcion de loss de mean squared error.
- Métodos de evaluación de mean squared error y mean absolute error.

Con estas características es que se entrenará el modelo, mediante la función *summary()* podemos ver ésta información, éste último realizará n número de épocas (en este caso y con el mejoramiento del modelo, será de 300 épocas). Después del entrenamiento, se guarda el histograma con los datos entrenados para poder graficarlos, también se imprimen la información de las últimas 10 épocas de entrenamiento para notar el avance en la minimización de errores.

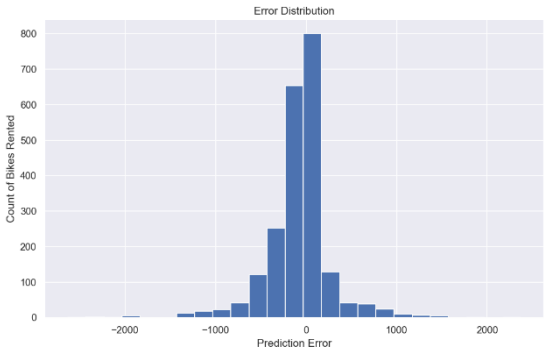
Por último, se generan 3 gráficos donde diferente información es posible ser obtenida. La primera con la información de los errores de entrenamiento y la de prueba a través de las épocas, la segunda con una grafica de dispersión para ver el comportamiento de las predicciones y la última con un histograma de los valores de predicción en el conteo de bicicletas rentadas, esto para verificar que tenga un comportamiento Gaussiano, es decir, que el modelo sea congruente realizando predicciones.

Análisis de gráficos

En el primer gráfico tenemos la información acerca de los errores de entrenamiento en contra de los errores de prueba a través de las épocas realizadas. Esto permite conocer información del modelo como el sesgo y la varianza en el modelo, y con ello, conocer si el modelo puede ser desajustado, ajustado o sobre ajustado, es una de las mayores pruebas que se deben realizar para conocer el desempeño del modelo y su efectividad.

El segundo gráfico consiste en una gráfica de dispersión para conocer la efectividad en las

predicciones, puesto que el comportamiento de una regresión lineal es precisamente ser lineal, según la gráfica podemos saber que tiene cierto patrón lineal y que mayormente tiene predicciones que tienden a acercarse al resultado, por lo que parece ser un modelo aceptable de predicción.



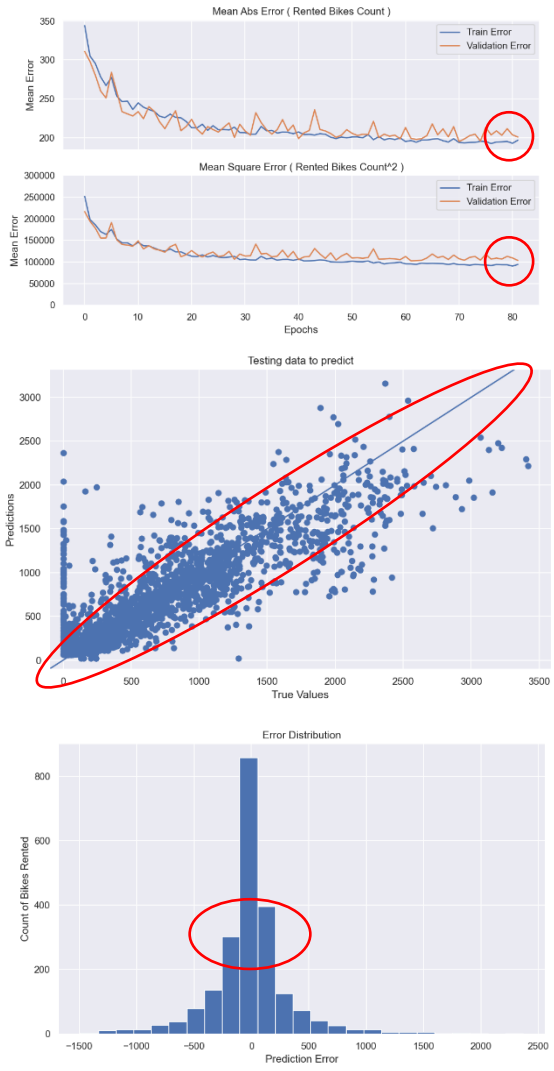
El último gráfico consiste en un histograma con los valores de predicción y el error que contiene cada una de ellas. Con este gráfico podemos observar la consistencia de las predicciones de acuerdo con el comportamiento gaussiano, presente en la mayoría de los buenos modelos estadísticos. Dado que se presenta tal comportamiento en la gráfica, se dice que el modelo tendrá congruencia con sus predicciones.

Mejoramiento del modelo

Tras tener el primer modelo realizado, fue necesario plantearse una mejora para evitar la tendencia al sobre ajuste del modelo, por lo cual se implementó un segundo programa con la implementación de una función de *Keras* llamada *EarlyStopping* donde, a partir de un crecimiento en la diferencia de errores entre el proceso de entrenamiento y el de prueba, la función contiene con un valor de "paciencia" en el cual definirá el valor de desajuste en la diferencia de ambos cálculos de errores, la paciencia de la función será de un valor de 20, el entrenamiento terminará el proceso para evitar el sobre ajuste que puede existir al determinar un número de épocas muy alto, entonces, obviando el proceso del nuevo modelo, el proceso tendrá una alta probabilidad de evitar el sobre ajuste y en dados ciertos casos, tendrá mayor ajuste que el primer programa, además, otro beneficio será el de disminuir la cantidad de procesamiento al entrenar la red neuronal.

Resultados y Conclusiones

Finalmente llegando a los resultados, con el nuevo programa implementado, y el valor de paciencia definido, el entrenamiento del modelo fue realizado con menos épocas de las definidas, algo que ya era esperado, a pesar de ello, con los datos obtenidos de las gráficas, fue definido un mejor modelo de predicción, puesto que **el primer modelo contiene mayor varianza** en la diferencia de los errores en el proceso de entrenamiento y el de prueba, cosa que fue corregida con la función de Keras y **ya no es posible notar esa diferencia en el segundo modelo**. Por otro lado, al ser una regresión línea, es decir, un modelo no tan complejo para predecir o procesar datos, **el sesgo no cambió en cierta medida,** cosa que es buena, teniendo en cuenta que, debido a su complejidad, **una regresión lineal tiene un alto sesgo, pero una varianza baja, si se requiere un bajo sesgo también, se debería de cambiar el modelo para obtener mejores resultados.**



De las gráficas anteriores podemos concluir que, a diferencia del anterior modelo:

- La varianza entre el valor de los errores de entrenamiento y prueba disminuyó.
- El proceso de entrenamiento tuvo menos épocas de procesamiento.
- La gráfica de dispersión mostró predicciones más lineales.
- El histograma de los errores presenta mayor consistencia para ser similar a la campana de Gauss, comportamiento que es positivo para las predicciones.

Referencias

1. V E, S., Park, J., Cho, Y., (2020). *Using data mining techniques for bike sharing demand prediction in metropolitan city*. Article from Science Direct. Obtained from: https://www.sciencedirect.com/science/article/pii/S0140366419318997#sec4

2. UCI. (2020). *Machine Learning Repository UCI: Seoul Data Set.* Obtained from: https://archive.ics.uci.edu/ml/datasets/Seoul+Bike+Sharing+Demand#