

Minería de datos

Hoja de trabajo 7

1. Optimización de Hiperparámetros:

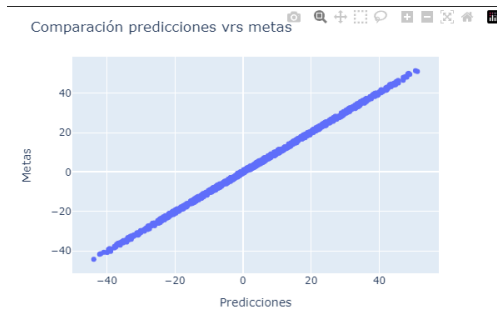
- Cambie el número de observaciones a 100,000. Este no es un hiperparámetro pero puede afectar el rendimiento del modelo. ¿Qué ocurre?

Podemos ver que al tener 100,000 observaciones se llenan más espacios vacíos en las gráficas lo que permite que el modelo y predicciones sean más exactos. También podemos observar que las producciones con menos datos se acercan más a las metas. En Cuanto a los pesos y el sesgo podemos ver que los pesos se reducen entre 3 decimos y 1 centésimo Y el sesgo se reduce aproximadamente un centésimo.

- Experimente con diferentes valores para la tasa de aprendizaje. Los valores como 0.0001, 0.001, 0.1, 1 son interesantes para observar. Prueba también cambiando el número de épocas (50, 100, 200). ¿Qué diferencias se observan? ¿Se comporta bien el algoritmo?

1. Learning rate 0.02:

Los pesos fueron : [1.9771714], [-2.9714637]. Estos pesos determinan cuánto contribuye cada entrada a la salida de la neurona. Un peso positivo aumenta la salida cuando la entrada es positiva, mientras que un peso negativo disminuye la salida. El sesgo obtenido fue el siguiente: [4.9921584] Este valor se suma a la combinación lineal de las entradas ponderadas por los pesos antes de aplicar la función de activación.

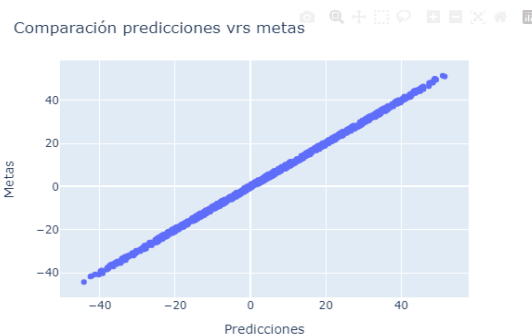


Se puede observar que el resultado es el mismo que el del ejercicio anterior, cumpliendo con las predicciones del modelo.

2. Learning rate 0.0001:

Los pesos fueron de $[1.9923216, -2.9978154]$. El primer peso ha aumentado ligeramente, lo que sugiere una mayor influencia de la primera entrada. El segundo peso ha disminuido ligeramente, aumentando la influencia negativa de la segunda entrada. El sesgo fue de $[4.9926157]$. Este pequeño cambio en el sesgo muestra que el modelo está haciendo ajustes finos en su salida para mejorar la precisión.

Un aprendizaje de 0.0001 es mucho más pequeño que el anterior de 0.02. Este valor más pequeño hace que los pasos de ajuste de los pesos y sesgos sean mucho más pequeños en cada iteración del entrenamiento.



En esta gráfica se pueden ver los resultados de precisión, los cuales son un poco más dispersos que el anterior, sin embargo sigue siendo un modelo efectivo.

3. Learning rate 1:

Al tener un learning rate tan grande los pesos y sesgos son increíblemente grandes, por lo que no se puede tomar el valor exacto de los mismos, en la gráfica resultante se mira que esta no contiene nada

por lo dispersos que están los datos , por lo que hace que el modelo sea inútil.

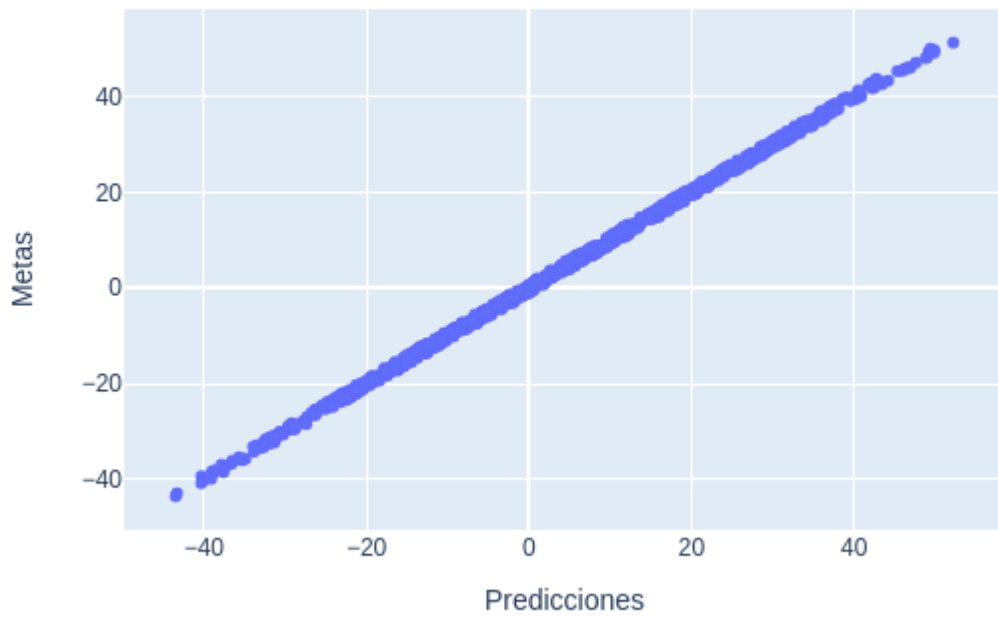
- Cambie la función de pérdida. ¿Cómo se comparan los resultados al cambiar la función de pérdida? Una función alternativa es la “Huber Loss”.

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”.

Podemos ver que al cambiar la función por huber loss se obtienen pesos que son menores que los pesos de mean square error al igual que los sesgos. En cuanto a las predicciones podemos ver que en general son menores al igual que los targets que se obtienen con huber loss, pero el promedio del delta de cada valor es menor en el square mean error lo que indica que es más preciso el square men error para este caso.

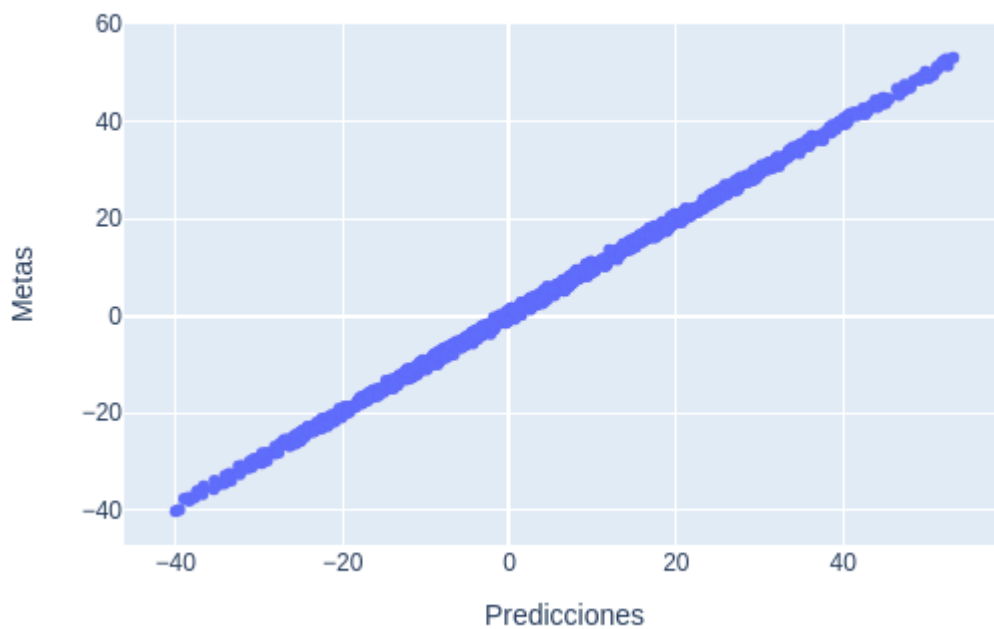
Gráfica Huber loss

Comparación predicciones vrs metas



Grafica Square Means error

Comparación predicciones vrs metas



Al comparar las gráficas podemos ver que la gráfica de Huber loss está más dispersa que la gráfica de square means error lo que confirma que es más precisa la función de Square means error para este caso.

- Evalúe cómo estos cambios afectan la precisión y la pérdida del modelo.

Huber Loss

predicciones

10.4, -19.8, 11.8, 2.2, 14.9

target

9.7, -19.2, 12.8, 2.6, 15.4

$$\frac{(0.7+0.6+1.0+0.4+0.5)}{5} = 0.64$$

Square Mean Error

predicciones

19.6, -29.9, -33.5, -24.6, -6.9

target

19.1, -29.2, -32.5, -24.7, -6.5

$$\frac{(0.5+0.7+1+0.1+0.4)}{5} = 0.54$$

En general podemos observar que square means es más preciso a la hora de evaluar la pérdida generada por cada función.

2. Validación Cruzada:

- Implemente una técnica de validación cruzada para evaluar la estabilidad y la generalización del modelo.

Implementación de validación cruzada tras compilar el modelo:

```
kf = KFold(n_splits=5, shuffle=True, random_state=42)

result_mse = []

for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    modelo.fit(X_train, y_train, epochs = 100, verbose = 0)
    perdida_mse = modelo.evaluate(X_test, y_test, verbose = 0)
    result_mse.append(perdida_mse)
    # pérdida de cada iteración
    print('MSE: ', perdida_mse)


promedio_mse = np.mean(result_mse)

print('El promedio de MSE es: ', promedio_mse)

modelo.fit(X, y, epochs = 100, verbose = 0)
pesos = modelo.layers[0].get_weights()[0]
print('Pesos: ', pesos)
sesgo = modelo.layers[0].get_weights()[1]
print('Sesgo: ', sesgo)
predicciones = modelo.predict(X[:5])
print('Predicciones: ', predicciones)
targets = y.flatten()[:5]
print('Targets: ', targets)
sumaPredict = np.sum(predicciones)
print("Perdida: ", sumaPredict/len(predicciones))
```

- Reporte los resultados de precisión y pérdida para cada iteración de la validación cruzada.

Resultados:

```
MSE: 0.3822559714317322
MSE: 0.4001358449459076
MSE: 0.32633164525032043
MSE: 0.40411660075187683
MSE: 0.4116416275501251
El promedio de MSE es: 0.38489633798599243
Pesos: [[ 2.0498946]
 [-3.0635114]]
Sesgo: [5.0240784]
1/1  0s 27ms/step
Predicciones: [[ 21.825235 ]
 [ 29.72574 ]
 [-13.363466 ]
 [-2.8613586 ]
 [ 24.476849 ]]
Targets: [ 22.41812498 29.16969245 -13.87051406 -2.67330012 24.5753794 ]
Perdida: 11.960599517822265
```

3. Interpretación de Resultados:

- Analice los resultados obtenidos, enfocándose en cómo los diferentes hiperparámetros y la validación cruzada afectan el rendimiento del modelo.

La variabilidad en los valores de MSE (Mean Squared Error) obtenidos a lo largo de las iteraciones de la validación cruzada puede proporcionar información valiosa sobre la estabilidad y consistencia del modelo. En este caso, los valores de MSE obtenidos en las distintas iteraciones son:

1. MSE: 0.3822559714317322
2. MSE: 0.4001358449459076
3. MSE: 0.32633164525032043
4. MSE: 0.40411660075187683
5. MSE: 0.4116416275501251

Aunque estos valores de MSE no difieren de manera extremadamente significativa, existe una diferencia notable entre la iteración con el MSE más bajo (0.3263) y la iteración con el MSE más alto (0.4116). Esta variabilidad sugiere que el modelo tiene un rendimiento relativamente consistente, pero no completamente estable, lo que indica consistencia relativa, lo cual se refiere a que el modelo no está teniendo resultados drásticamente diferentes en distintas particiones del conjunto de datos.

Por otro lado, las predicciones del modelo están razonablemente cerca de los valores reales (targets), lo cual es una buena señal de que el modelo está capturando las tendencias generales de los datos.

- Sugiera posibles mejoras basadas en sus observaciones.

Para mejorar el rendimiento del modelo, se recomienda implementar early stopping para evitar el sobreajuste, introducir regularización para penalizar pesos grandes, y utilizar técnicas como Grid Search o Random Search para optimizar hiperparámetros como la tasa de aprendizaje y el tamaño del batch. Además, es útil experimentar con diferentes números de épocas para encontrar el equilibrio adecuado entre subajuste y sobreajuste, aumentar las particiones en la validación cruzada a 10 o más para obtener una estimación más robusta, y probar arquitecturas de modelo más complejas con más capas ocultas o unidades y diversas funciones de activación para capturar patrones complejos en los datos.