Inteligencia artificial avanzada para la ciencia de datos I

Momento de Retroalimentación: Módulo 2 Uso de framework o biblioteca de aprendizaje máquina para la implementación de una solución. (Portafolio Implementación)

Sobre el código

Modelo realizado

Para esta entrega se realizó el código de un perceptron utilizando el método de descenso de gradiente estocástico que puede recibir una cantidad n de características de entradas y una de salida. Se utiliza un formato csv para ingresar los datos, donde la última columna del csv es la característica de salida y las primeras n-1 columnas son las características de entrada..

Se realizó utilizando la librería pandas para la lectura y el tratamiento de los datos y usando la librería numpy para algunas operaciones como el producto punto.

Hiperparámetros ajustables

Al usuario se permite ajustar los siguientes hiperparámetros:

- Número de épocas: el usuario puede elegir un número de épocas que quiere que sucedan en el algoritmo. Este hiperparámetro es opcional y si no se indica, se atravesarán épocas hasta que los pesos lleguen a la convergencia (lo cual no significa un buen resutado)..
- Learning rate: El usuario puede elegir el nivel de learning rate que él desee. Este hiperparámetro no es opcional.
- Porcentaje de datos para entrenamiento: Este hiperparámetro indicará qué porcentaje de los datos se desean para usarse para entrenar al modelo y, por consiguiente, qué porcentaje de datos se usarán para testing del dataset. Este hiperparámetro no es opcional.

Es importante mencionar que aunque los hiperparámetros sean iguales en distintas corridas, es probable que no se obtengan los mismos resultados en cada corrida ya que el perceptron está diseñado para tomar muestras aleatorias de los datos cada vez que se corra el algoritmo.

Pruebas

Datasets de pruebas

En el repositorio de github se tienen 3 datasets distintos para pruebas:

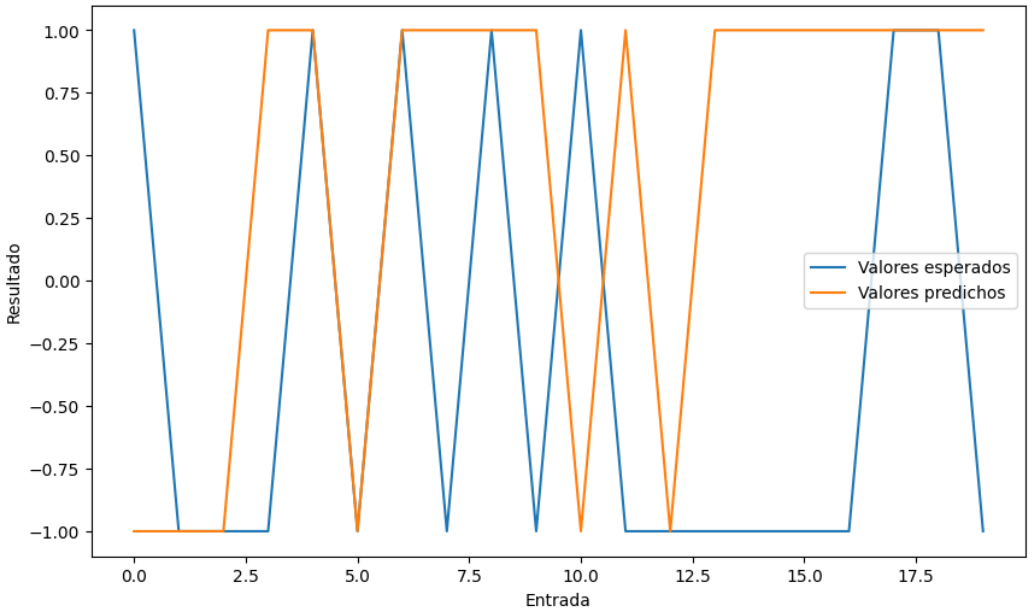
- gatoperro: se tienen varios datos sobre perros y gatos donde el modelo tratará de predecir si con las características dadas se está hablando de un perro o de un gato.
- test: donde se tienen características de personas y el algoritmo intentará determinar si una persona me cae bien o no.

- xor: donde se tiene una tabla de verdad y el algoritmo intentará predecir el restante de la tabla de verdad habiendo visto solo una parte de ella.

Pruebas con los mismos hiperparámetros

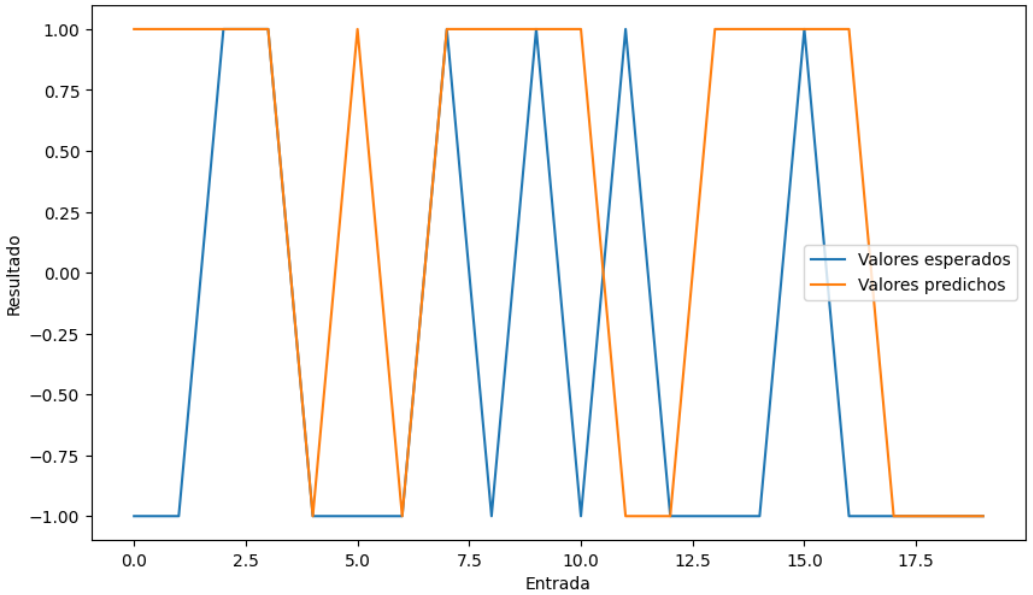
Se van a realizar dos pruebas con los mismos hiperparámetros, para demostrar la importancia de la toma de muestras aleatorias.

Se realizará la prueba con el archivo *test* y un learning rate de 0.1, 70% de datos de entrenamiento y 3 épocas.



Esto nos da como resultado un 45% de precisión en tu nuestro sample de pruebas y un mean square error de 2.2.

Si hacemos una segunda corrida con los mismos hiperparámetros, obtenemos los siguientes resultados



Vemos que la precisión de nuestro modelo aumentó hasta 55.0%, con respecto a nuestro sample de pruebas. Y nuestro mean square error es de 1.8. Todo esto sin cambiar los hiperparámetros y es debido a que tomamos muestras aleatorias en cada corrida. Así que siempre se entrena de forma diferente el modelo.

Análisis de accuracy y error

Información otorgada por el programa al finalizar

Al finalizar una corrida, se nos da información sobre la precisión del modelo contra el sample de pruebas y se calcula el mean square error.

Conclusiones importantes sobre el modelo

Como vimos anteriormente, los resultados pueden variar incluso sin cambiar los hiperparámetros ya que los registros se mezclan de forma aleatoria en cada corrida. Esto de cierta manera puede producir algún tipo de sesgo al momento de entrenar al modelo o quitar todos los sesgos. Es por ello que podemos obtener mejores o peores resultados en cada corrida.