Clasificador lineal de flores Iris según la longitud y anchura de su sépalo

Proyecto final – ACECOM

16/11/2019

1 Objetivo

Este proyecto consiste en la programación de un algoritmo de clasificación lineal (Perceptrón) desde cero y su aplicación en la clasificación de flores Iris según las características de su sépalo.

2 Dataset

El dataset utilizado es un archivo .csv de 5 columnas. Las dos primeras contienen la longitud y anchura del sépalo de las flores; mientras que las dos siguientes contienen información sobre el pétalo, la cual no fue utilizada. La última columna contiene el tipo de flor Iris que presenta las características anteriores y abarca tres tipos de flores. Para el presente proyecto, solo se usaron los tipos Iris Setosa e Iris virginica.

Datos de entrenamiento: 25
Datos de validación: 100

2.1 Descarga de dataset:

Este dataset puede ser descargado de manera pública en el repositorio UCI:
• Iris Data Set: https://archive.ics.uci.edu/ml/datasets/Iris

3 Metodología

Dado que en este proyecto, el código consiste principalmente en una clase denominada Iris, se procederá a analizar sus métodos para la explicación del código.

3.1 Método "init":

Es el constructor, el cual se encarga de asignarle a cada objeto de la clase dos atributos:

• self.df: Es un dataframe, obtenido mediante la librería Pandas, que contiene al dataset del proyecto.
• self.X: Es una lista con los valores y etiquetas de la data de entrenamiento, la cual se obtiene con el empleo del segundo método.

Asimismo, este método requiere el número de datos de entrenamiento a emplear (definido como 25).

3.2 Método "generatePoints":

Este método se encarga de la extracción de los datos de entrenamiento de manera aleatoria del dataset. Para esto, se verifica que cada dato extraído corresponda a uno de los dos tipos de flores por clasificar. Luego, se le asigna una etiqueta a cada fila extraída correspondiente a los siguientes valores:

• Iris Setosa: -1
• Iris Virginica: 1

Por último, cada fila se anexa a la lista final en forma de una lista de 3 elementos correspondientes a la longitud del sépalo, la anchura del sépalo y la etiqueta respectiva.

3.3 Método "plot":

Sirve para gráficar todos los datapoints en un mismo gráfico, de acuerdo a sus elementos:

• Eje X: Longitud del sépalo
• Eje Y: Anchura del sépalo
• Color:
- Azul: Iris Setosa
- Rojo: Iris Virginica

Asimismo, el método permite la entrada de tres variables opcionales, cuyo empleo se de la siguiente forma:

• mispts: Se emplea para graficar los datos de validación en la misma gráfica que los datos de entrenamiento. Para su mejor distinción, los puntos de entrenamiento se representan con puntos más grandes que los de validación.
• vec: Representa una lista de 3 elementos, los cuales representan los coeficientes de una recta, y se emplea para graficar dicha recta de color verde. En el proyecto, en esta variable se ingresa el peso actualizado tras cada iteración.
• save: Representa un valor booleano, el cual de ser verdadero, habilita el almacenamiento de las gráficas de los datapoints y de los pesos hallados en cada

iteración. Asimismo, se le asigna un título correspondiente tanto para datos de entrenamiento, como para la representación de final de los datos de validación.

3.4 Método "classificationError":

Método en el que se ingresa cada peso hallado y se usa para determinar la cantidad de datapoints de entrenamiento que están clasificados incorrectamente según la etiqueta correcta. Presenta solo un valor opcional, el cual se usa para alternar entre los datos de entrenamiento y los datos de validación.

3.5 Método "chooseMisclPoint":

Funciona de manera similar al método anterior; sin embargo, se emplea para generar una lista que contenga todos los puntos mal ubicados respecto al peso ingresado. Retorna un datapoint, escogido al azar, de dicha lista.

3.6 Método "pla":

Este método representa al algoritmo de clasificación (Perceptrón). En primer lugar, inicializa una variable peso con ceros. Luego, dicho peso es ingresado al método "chooseMisclPoint" y se obtiene un dato mal clasificado con el fin de usarlo para la actualización del peso. Este proceso, se itera hasta que todos los puntos hayan sido clasificados correctamente. Además, en cada iteración, se grafica el peso con el método "plot" y se almacena dicha gráfica. Por último, se genera una lista que almacena todos los errores generados en cada iteración.

3.7 Método "checkError":

El método "checkError" se emplea para determinar el error generado por el peso final cuando se emplean todos los datos de validación. Requiere el número de datos de validación (definido como 100 y correspondiente al total de puntos del dataset).

3.8 Método "plotError":

Por último, este método se emplea para la gráfica del porcentaje de error generado tras cada iteración

4 Resultados

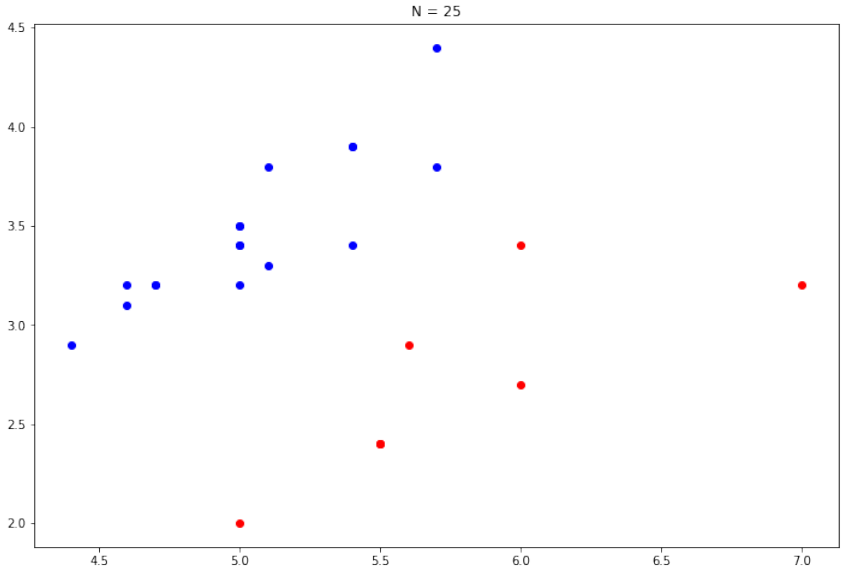

Figure 1: Datos de entrenamiento

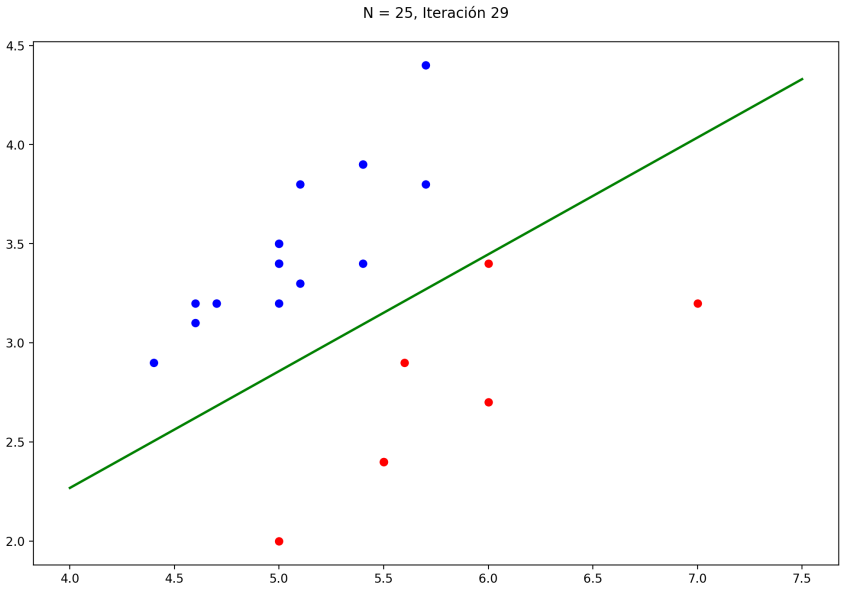

Figure 2: Peso Final

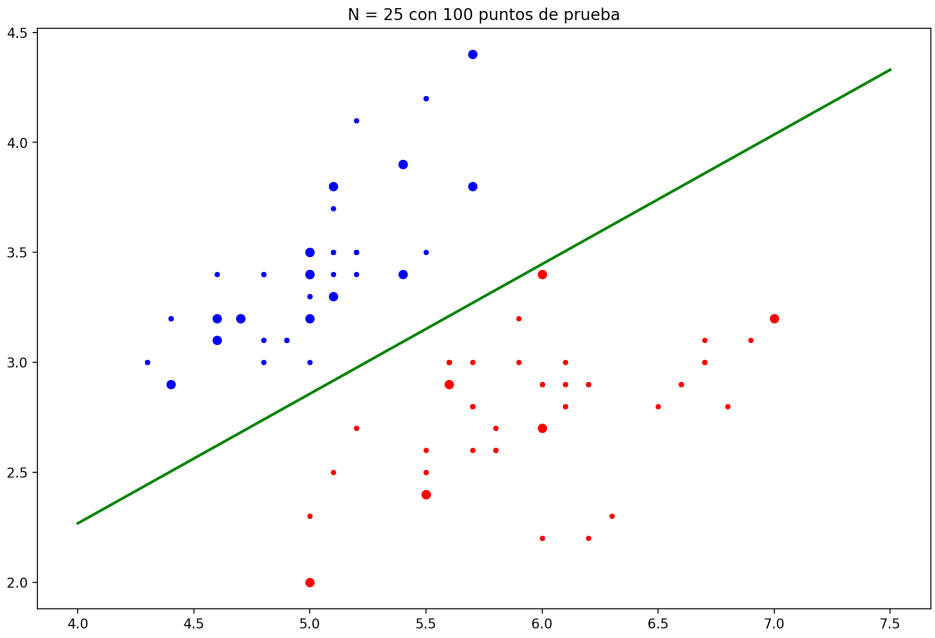

Figure 3: Datos de validación

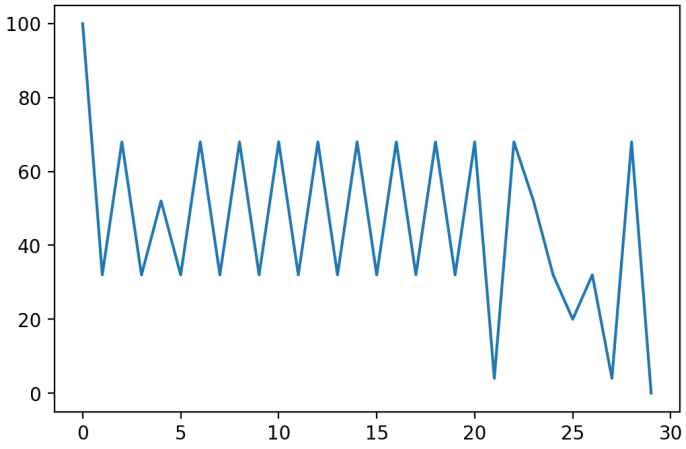

Figure 4: Error porcentual vs Iteraciones