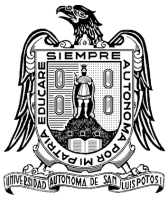


Parcial 3, Cómputo Bio-inspirado Universidad Autónoma de San Luis Potosí

Leonardo Alejandro González López
308096

4 de diciembre de 2022

Índice

1. Planteamiento del problema

Investigar sobre el conjunto de datos (dataset) IRIS. Deberá decir quién lo creó y la fecha. Describir las características relevantes de este dataset y qué es lo que contiene (incluyendo numero y descripción de las clases y número y descripción de las variables). Bajar el dataset e indicar de dónde lo bajó.

Resolver el problema de clasificación del dataset utilizando el enfoque de clasificación binaria. Utilizar el algoritmo de programación genética para encontrar una regla de inferencia que sea capaz de clasificar en clase X y no clase X cada una de las clases del dataset.

1.1. IRIS

El conjunto de datos de Iris se usó en el artículo de Ronald Fisher de 1936, El uso de medidas múltiples en problemas taxonómicos, y también se puede encontrar en el repositorio de aprendizaje automático de UCI. Incluye tres especies de iris con 50 muestras cada una, así como algunas propiedades de cada flor. Una especie de flor es linealmente separable de las otras dos, pero las otras dos no son linealmente separables entre sí. La información que se presenta es la siguiente:

- SepalLengthCm
- SepalWidthCm
- PetalLengthCm
- PetalWidthCm
- Species

Los datos pueden describirse de dos maneras, la primera es un archivo .txt o .data donde esta acomodado en forma de columnas sin encabezado o id[1], la otra es en un archivo .csv que contiene un id y encabezados [2]. El dataset incluye 150 clases, 50 por cada flor, con cada una de las 4 variables descritas en centímetros y expresados como números Racionales.

2. Descripción de la solución

Para aproximarme a la solución primero analice los datos y como estos se relacionan, para comprender la forma en que tenía que realizar la función de aptitud. Para esto, creé un script de python para poder ver su comportamiento gráficamente, posteriormente, necesitaba de donde basarme para crear el programa de iris y la función de aptitud, por lo que utilicé los ejemplos previamente vistos en clase, la función de aptitud está basada en 'regfitness.m' de la biblioteca GPLAB[3] y el resto del programa esta basado en el archivo 'demo.m' de la misma biblioteca.

En primer instancia, el codigo de python utilizado para visualizar los datos puede ser encontrado en los anexos del examen.

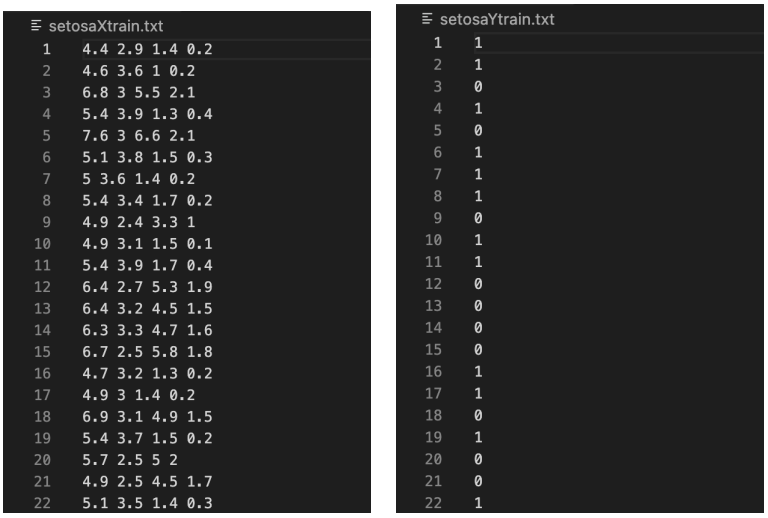
Gracias a esta visualización puedo hacerme a la idea de que separar linealmente estos datos no es posible debido a la naturaleza de estos datos, por lo que hay que aproximar el resultado de una forma diferente, a lo que 'regfitness.m' puede darnos una mejor solución, la lógica es separar los resultados del valor numérico del árbol del árbol 'res', en palabras más simples, separar los positivos, negativos, falsos positivos y falsos negativos, lo que nos permitirá hacer uso de las funciones para evaluar las reglas. Las que se intentaron usar fueron:

- Max {F = #Correctos – n * #Incorrectos}
- PA = tp / (tp + fp)
- S=tp/(tp+fn)
- Max { F = p1 PA + p2 CP + p3 S }

Sin embargo, nos hace falta el parámetro de comprehensibilidad para utilizar la última fórmula listada. El punto de la función de evaluación es ver que tan acertada es, entonces, al ver las fórmulas de la capacidad predictiva y la sensibilidad a la regla, me doy cuenta que están sacando un tipo de factor o normalización(probablemente esta ultima definición este errada), estos factores, son complementarios es decir, que al sumarlos deberían dar la unidad, entonces, si queremos saber que tan acertada es esta función, debemos restarle a la unidad la suma

de estos dos factores, cada uno multiplicado por 0.5. Otra forma sería utilizar la primera fórmula en la que solo necesitamos el numero de correctos e incorrectos, además del número de evaluaciones hechas.

```
%Max { F = p1 PA + p2 CP + p3 S }
%buscamos que se acerque mas a 0
sumdif=1-((0.5*pa)+(0.5*s));
%Max {F = #Correctos { n * #Incorrectos}
%sumdif=(positivo+negativo)-10*(falsoneg+falsopos);
```



(a) Datos en X (b) Datos en Y

Figura 1: Datos de trabajo

El siguiente paso es acomodar la forma en que trabajaremos los datos, afortunadamente, la mayoría es númerico a excepción de los nombres, y siguiendo la lógica del problema Knapsack1-0, es mas facil trabajarlos si los combertimos a un símbolo; siendo 1 cuando si se trata del dato y 0 cuando no, la siguiente parte es separar los datos de entrenamiento y prueba en sus ejes X y Y lo que fue un proceso tedioso 'Dividir este pequeño dataset en un conjunto de entrenamiento y uno de prueba (80 %-20 %)' fue lo solicitado por el problema, entonces los datos quedarían resumidos como se observa en la figura 1.

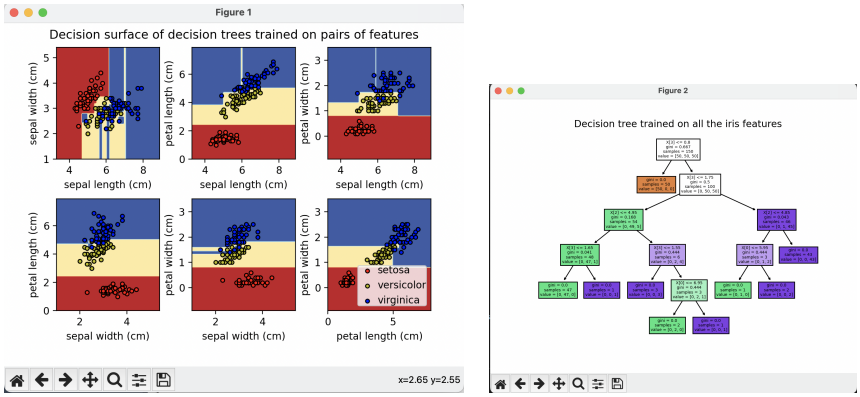
2.1. Main

Como se menciono anteriormente, el archivo que se utilizo como base para correr GPLAB, fue 'demo.m' y lo que se modificó y agregó fue lo siguiente:

```
%functions, no log10 pq el resultado se vuelve muy inexacto
p.addfunctions={p, 'mydivide', 2, 'mysqrt','mylog2',1};
p.calcfitness='regfitnessperso';
%modif
p.datafilex='setosaXtrain.txt';
p.datafiley='setosaYtrain.txt';

p.usetestdata=1;
p.testdatafilex='setosaXprueba.txt';
p.testdatafiley='setosaYprueba.txt';
%modif
```

3. Descripcion de los resultados



(a) Conjunto de datos (b) Árbol de desición

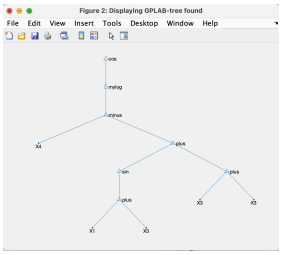
Figura 2: Visualización de datos

Los operadores elegidos para el desarrollo del codigo fueron 'crossover'2,2,'mutation',1,1.
La visualizacion de los datos en el script de python quedaron de la siguiente manera, también se hizo un árbol de desición para comparar

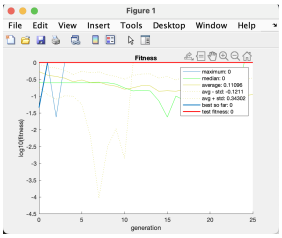
los resultados obtenidos en GPLAB.

Gracias a esto pude observar que posiblemente iba a tener problemas para clasificar versicolor de virginica porque es donde mas convergen sus datos

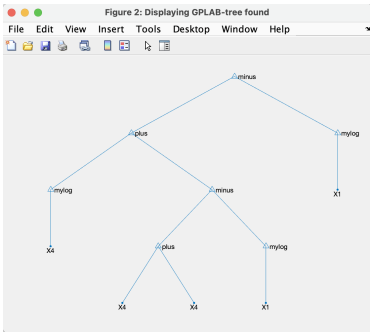
3.1. Resultados



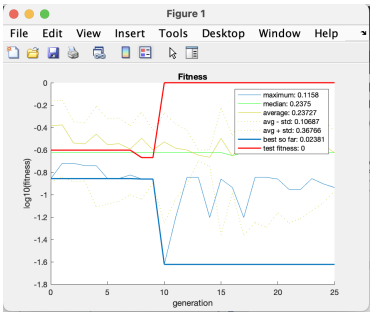
(a) Arbol setosa



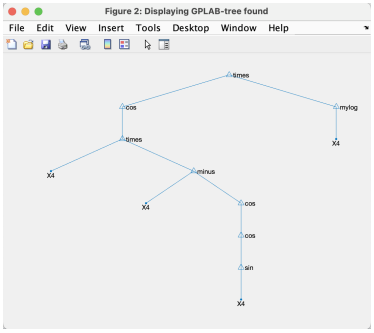
(b) Datos setosa



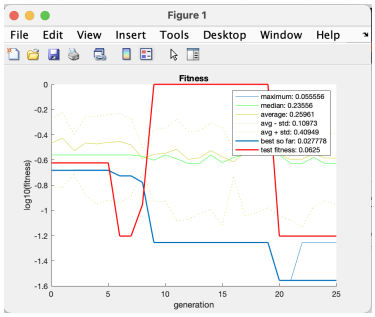
(c) Arbol virginica



(d) Datos virginica



(e) Arbol versicolor



(f) Datos versicolor

Figura 3: Arboles y Resultados

4. Discusión

Primeramente pido al lector que note la similitud de los arboles obtenidos con GPLAB al propuesto en hecho con python, lo que me hace creer que la confiabilidad de estos es muy buena, en segundo lugar, quiero mencionar que la predicción sobre los problemas que se tendrían en la clase versicolor al analizar sus datos fueron acertadas, a continuación expongo las matrices de confusión de las 3 clases, en la mayoria de las corridas hechas Setosa y Virginica obtubieron un resultado perfecto, pero en cuanto a Versicolor, 2 de 3 corridas tuvieron algun error:

SETOSA	Positivos	Negtivos
Positivos	5	5
Negativos	0	0

VIRGINICA	Positivos	Negtivos
Positivos	4	6
Negativos	0	0

VERSICOLOR	Positivos	Negtivos
Positivos	7	2
Negativos	1	0

Las otras dos corridas de versicolor, dieron como resultado: '8 correctos, 1 negativo y 1 falso positivo' y '7 correctos, 3 negativos y 0 falsos' este resultado pudiese mejorar con mas datos de entrenamiento que le permitan al programa tener un mejor sesgo de los datos.

5. Conclusion

Al realizar este examen por fin pude entender la diferencia de GA a GP, además de aplicar conocimientos pasados para poder comprender como era necesario plantear el problema, finalmente, lo importante de un algoritmo genético es tener los datos de entrenamiento necesarios para poder tener resultados óptimos, en eso recae que en la actualidad existan dataset enormes.

Referencias

[1] R. A. Fisher. Iris data set. UCI. [Online]. Available: https://archive.ics.uci.edu/ml/machine-learning-databases/iris/

[2] MATHNERD. Iris data set csv. Kaggle. [Online]. Available: https://www.kaggle.com/datasets/arshid/iris-flower-dataset

[3] S. Silva. A genetic programming toolbox for matlab. GPLAB. [Online]. Available: https://gplab.sourceforge.net