Algoritmos Genéticos, Una aplicación en optimización y una aplicación de búsqueda, clasificación mediante GA

González Blandón Edison David, Velásquez Franco Juan Esteban
edavid.gonzalez@udea.edu.co, juan.velasquez12@udea.edu.co
Departamento de Ingeniería Electrónica
Universidad de Antioquia
Medellín, Colombia

***Resumen*—En este trabajo se realiza la solución de la tarea número tres del curso de fundamentos de inteligencia computacional. Se compone este trabajo de dos actividades, con la solución de las dos situaciones problema basados en técnicas y/o métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización como lo son los Algoritmos Genéticos (AG abreviatura en español - ING: Genetic Algorithm (GA)). Se muestra el desarrollo seguido para las soluciones y los resultados obtenidos. El trabajo realizado es realizado sobre la plataforma Jupyter, lenguaje Python 3.**

***Índice de Términos*—Función Fitness o Función Objetivo. Optimización. Espacio de búsqueda. Población. Gen. Cromosoma. Cruce. Selección de padres. Cruce. Mutación. Supervivencia.**

I. INTRODUCCIÓN

El algoritmo genético es una técnica de búsqueda basada en la teoría de la evolución de Darwin, Los principios básicos de los Algoritmos Genéticos fueron establecidos por Holland (1975), y se encuentran bien descritos en varios textos – Goldberg (1989), Davis (1991), Michalewicz (1992), Reeves (1993). Se presentarán aquí los conceptos básicos que se requieren para abordar dos problemas de machine learning en los cuales se puede usar esta técnica. Esta representación del conocimiento pese a ser relativamente nuevo, sus aplicaciones son cada vez más diversas, en especial en problemas de optimización y de búsqueda de soluciones viables (búsqueda adecuada de puntos de inflexión en un conjunto de datos), problemas difíciles que, de lo contrario, llevaría toda la vida resolver; desde el análisis de los organismos vivos de nuestra naturaleza, se puede observar generación tras generación, la evolución acorde con los principios de la selección natural y la supervivencia de los más fuertes, postulados por Darwin (1859). Por imitación de este proceso, los Algoritmos Genéticos son capaces de ir creando soluciones para problemas del mundo real. La evolución de dichas soluciones hacia valores óptimos del problema depende en buena medida de una adecuada codificación de las mismas.

II. DESARROLLO

A. Definiciones y Contextualización

Los algoritmos genéticos (ESP: AG ó ENG: GA) son algoritmos basados en los conceptos de selección natural y genética, partiendo de un grupo o población compuesto por cromosomas quienes brindan soluciones posibles al problema dado, un cromosoma es una solución posible, estas soluciones luego experimentan recombinación (cruzados - crossover) y mutación (Según la teoría de la evolución de Darwin), produciendo hijos, este proceso se repite durante generaciones, y según el criterio de supervivencia más natural, las personas más aptas (higher fitness) tienen más posibilidades de aparearse y tener más hijos (Cromosomas, individuos) “más aptos”.

En el momento de trasladar el comportamiento natural en la evolución de los organismos vivos a un

algoritmo debemos aclarar y delimitar los siguientes aspectos:

- Cromosomas: Se compone de genes, cada cromosoma es una de las soluciones brindadas para el problema:

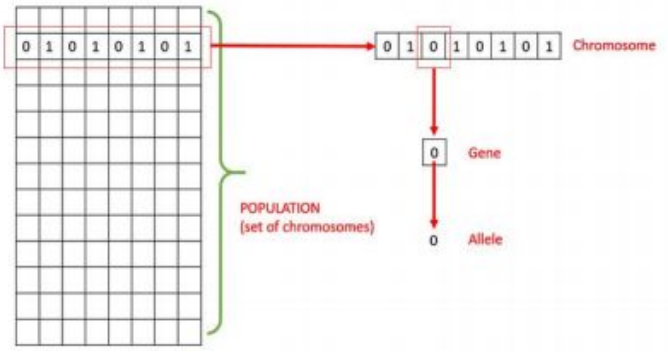

Fig 1. Composición de la población inicial, explicación cada una de sus partes (Tomado de referencia [5])

  Varios cromosomas componen una población, esta población se encuentra representada en dos espacios, el del cálculo (Como lo ve nuestro algoritmo para poder ser procesado) conocido como genotipo, y el espacio de soluciones del mundo real, conocido como fenotipo, el fenotipo se representa normalmente idéntico a un organismo o la representación de un sistema natural quien decide y elige características claras de supervivencia y de mutación que permite continuar con su existencia, es decir, evolución necesaria, adaptativa.
- Operadores genéticos: Quienes alteran la composición genética de la descendencia, entre ellos se encuentra: 1. Estrategia de cruce: análogo a la reproducción y al cruce biológico. En este, se selecciona más de un padre y se producen uno o más descendientes utilizando el material genético de los padres, reconocidos cruce de un punto, cruce multipunto, cruce uniforme, recombinación aritmética, crude de orden de davis. 2. Estrategia de mutación: se puede definir como un pequeño ajuste aleatorio en el cromosoma, para obtener una nueva solución, entre las mutaciones mencionadas durante el curso de inteligencia computacional se encuentran: bit flip, random resetting, swap mutation, scramble mutation, inversion mutation. 3. Estrategia de selección: hace referencia a la selección de los padres que aparean y se recombinan para crear los descendientes para la próxima generación, en el curso es usado mediante dos grandes grupos, el primero acorde al fitness en el cual se encuentra el método de la ruleta, muestreo estocástico universal y rank, el segundo es por torneo.
- Política de supervivencia: Después de pasar de generaciones, quienes sobreviven, en el curso fueron vistos la supervivencia basado en edad y basado en la aptitud (fitness).
- Función de Fitness: Es una función que trabaja entorno a una característica que deseamos maximizar, pues es lo requerido por el entorno.
- Restricciones que de pueden tener: tanto los valores de entrada, los de salida y dentro del proceso realizado por el algoritmo genético, una res
- Criterio de parada: normalmente mediante generaciones o mediante el análisis de convergencia de la función.

Teniendo en cuenta lo anterior, nuestro algoritmo se comporta de la siguiente manera:

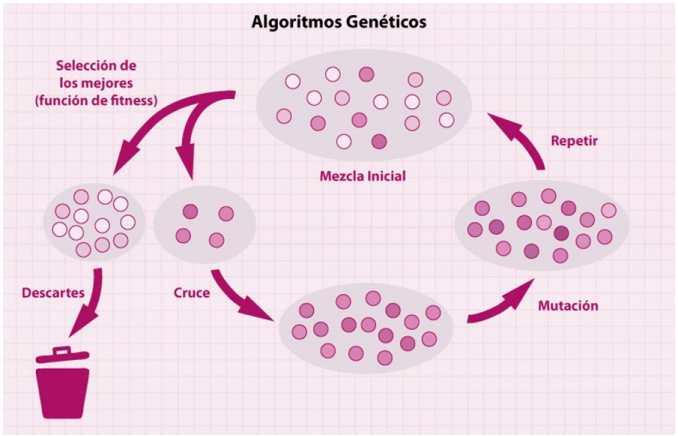

Fig 2. Esquema de funcionamiento de un GA (Tomado de referencia [10])

Comenzamos con una población inicial (que puede ser generada al azar o sembrada por otras heurísticas). Se seleccionan los padres de esta población para el apareamiento. Se aplican operadores de cruce y mutación a los padres para generar nuevos descendientes. Y, finalmente, estos descendientes reemplazan a los individuos existentes en la población y el proceso se repite.

B. Primer Problema: "Problema de la Mochila" - Optimización

El problema consiste en que dado un conjunto finito de ítems, cada uno de los cuales tiene asociado un peso y una ganancia, seleccionar el subconjunto de ítems a incluir en una mochila, capaz de soportar un peso máximo finito - cuya inclusión proporcione una ganancia máxima. Sean n ítems y una mochila capaz de soportar un peso máximo C. Denotamos por bj el beneficio obtenido al introducir el ítem j en la mochila mientras que wj denota el peso asociado a dicho ítem j.

La implementación brindada fue mediante:

- Cromosomas: Se decidió que los cromosomas se generan aleatoriamente con una población de 30 individuos, donde cada gen representa cada uno de los elementos que llevará y el alelo toma valor de 1 o 0, donde 1 es que el elemento se llevará y 0 es cuando no se llevará en la mochila.
- Estrategia de cruce: Se implementó la estrategia de cruce punto a punto, la cual consiste en seleccionar un punto de cruce aleatorio y los padres se intercambian para obtener nuevos descendientes.
- Estrategia de mutación: Bit flip, la cual asigna un valor aleatorio del conjunto de valores permisibles a un gen elegido al azar
- Estrategia de selección: Usada la ruleta, se elige un punto fijo en la circunferencia de la rueda como se muestra (Fixed point) y se gira la rueda. La región de la rueda que se encuentra delante del punto fijo se elige como principal, igualmente lo que se hace es que se coge cada cromosoma y su valor de fitness es dividido por la suma total de todos los fitness y se le asocia una probabilidad.
- Estrategia de supervivencia: Basado en aptitud, los hijos tienden a reemplazar a los individuos menos aptos de la población.
- Función de Fitness: Donde la ganancia sea máxima con el mínimo peso.
- Restricciones que de pueden tener: La población no incrementan sino que se incluyen los hijos y posteriormente se recorta nuevamente la población.
- Criterio de parada: Se define bajo número de iteraciones

Con lo anterior se obtuvieron los siguientes resultados

Fig 3. los puntos naranja representan el peso que, en los vectores es el inferior, por otro lado, los puntos azules representan la ganancia, que en los vectores es el superior.

C. Segundo Problema: "Realizando clasificación" - Búsqueda en el espacio

Agrupar N números en K grupos disjuntos minimizando la suma de las diferencias entre los grupos.

Para llevar a cabo la solución de este problema se creó un conjunto de datos, datos (x,y), aprovechando que son datos en pares o tuplas, tal

que se puede ver gráficamente los clusters en un plano cartesiano, estos datos son un conjunto de tuplas (Problema bidimensional) en los cuales el usuario puede ingresar su cantidad N, así como la cantidad de grupos K en los cuales se desean dividir los datos, se hizo uso de gaussianas multivariadas separadas para la generación de datos

La implementación fue realizada mediante Jupyter, el entorno de notebook computacional interactivo basado en la web, usando python 3. Teniendo en cuenta que:

- Cromosomas: Se permitió que el usuario (Quien ejecuta el Script de Python, con el algoritmo genético) decida cuál es su valor.

- Estrategia de cruce: Implementados tres cruces: 1. Cruce de un punto: se selecciona un punto de cruce aleatorio y las colas de sus dos padres se intercambian para obtener nuevos descendientes. 2. Cruce multipunto: es una generalización del cruce de un punto en el que se intercambian segmentos alternados para obtener nuevos descendientes. 3. Cruce uniforme: Tratamos cada gen por separado, lanzamos una moneda para cada cromosoma para decidir si se incluirá o no en la descendencia

Fig 4. Funcionamiento gráfico de los cruces implementados (Tomado de referencia [5])

  No fueron implementados, recombinación aritmética, puesto que nuestras etiquetas son valores enteros y tampoco Davis, porque su procesamiento es innecesario.

- Estrategia de mutación: Usada, Random Resetting: es una extensión del Bit flip para la representación entera. En esta, se asigna un valor aleatorio del conjunto de valores permisibles a un gen elegido al azar. bit flip: mutación invertida, seleccionamos uno o más genes aleatoriamente y se cambia su valor de etiqueta. Esto se usa para GA codificados binarios, aunque fue implementado para nuestro problema con etiquetas enteras, (random resetting), pues deseamos brindar variedad a nuestra población, incluyendo entropía y variabilidad sin tener que realizar muchas mutaciones, pues una variabilidad muy alta, muchas veces considera una tasa de convergencia muy alta, de igual manera, en la forma en que se crearon nuestros datos, no es viable después de tener un buen fitness, realizar una búsqueda "extraña" de datos.

Fig 5. Funcionamiento gráfico de la mutación implementada (Tomado de referencia [5])

- Estrategia de selección: Usada la ruleta, se elige un punto fijo en la circunferencia de la rueda como se muestra (Fixed point) y se gira la rueda. La región de la rueda que se encuentra delante del punto fijo se elige como principal, igualmente lo que se hace es que se coge cada cromosoma y su valor de fitness es dividido por la suma total de todos los fitness y se le asocia una probabilidad.

Fig 6. Estrategia de selección, ruleta(Tomado de referencia [9])

  En la anterior figura se observa como un cromosoma con mayor fitness tiene un pedazo mayor de la rueda, es decir, aumenta

la probabilidad de que sea el padre, es por eso, que se eligió esta opción, puesto que queremos hijos más fuertes pero no deseamos eliminar la viabilidad probabilística que sea el padre alguien que no es tan fuerte, quien incluye la búsqueda en el espacio de diferentes soluciones.

- Estrategia de supervivencia: Elegida la supervivencia basada en aptitud: los hijos tienden a reemplazar a los individuos menos aptos de la población, sobreviven los más fuertes, los mejores.

- Función de Fitness:

Fig 7.Función Fitness.

Se calcula la distancia euclidiana entre dos puntos, estos dos puntos deben de pertenecer al mismo cluster (se hace por grupos), se guarda en un acumulador, después se suman las distancias que dieron entre los grupos y así se obtiene el valor de fitness del cromosoma. Esto se realiza para todos los cromosomas C, en cuanto tan grande Q sea la población.

- Restricciones que se pueden tener: La población no incrementan sino que se incluyen los hijos y posteriormente se recorta nuevamente la población. Los valores que pueden tomar los genes son valores desde 1 hasta la cantidad de clusters, son números enteros.

- Criterio de parada: Elegido el criterio de parada por la cantidad de generaciones, en este criterio comparado con el de convergencia se puede limitar el uso del recurso computacional.

Basado en lo anterior, los resultados obtenidos, por nuestro AG fue:

Fig 8. Datos generados con dos gaussianas multivariadas, 50 datos.

Fig 9. Resultado de implementar algoritmos genéticos, con 50 cromosomas, 50 genes, 2 cluster, 500 generaciones, cruce multipunto, mutación random resetting.

Fig 10. Datos generados con dos gaussianas multivariadas, 50 datos.

cromosomas, 16 genes, 2 cluster, 20 generaciones, cruce de un punto, mutación random resetting.

Fig 11. Resultado de implementar algoritmos genéticos, con 50 cromosomas, 50 genes, 4 cluster, 500 generaciones, cruce multipunto, mutación random resetting.

Hasta el momento se puede observar cómo el algoritmo funciona adecuadamente según se solicitan los clusters, pero esto debido a que costo computacional?, igual es implementado otros métodos de cruce:

Fig 12. Datos generados con dos gaussianas multivariadas, 16 datos

Fig 13. Resultado de implementar algoritmos genéticos, con 30

Fig 14. Datos generados con dos gaussianas multivariadas, 20 datos

Fig 14. Gráfico para la ejecución entre la figura 1 y 2, el resultado de fitness vs generación

Fig 15. Resultado de implementar algoritmos genéticos, con 40 cromosomas, 20 genes, 3 cluster, 20 generaciones, cruce uniforme, mutación random resetting.

Importante a destacar es que los algoritmos de un punto, multipunto y uniforme tienen un buen rendimiento, la función fitness se encuentra en

crecimiento, lo importante a destacar es que carga computacional está sufriendo, también se podrían variar los valores de las tuplas, para que no sean grupos tan marcados y visualizar cómo se comporta el algoritmo en la búsqueda de grupos.

III. CONCLUSIONES

- La aplicación más común de los algoritmos genéticos se encuentra en la optimización, la literatura ha enmarcado que estos son eficientes y confiables en este aplicativo, sin embargo, no todos los problemas son apropiados para resolverse mediante esta herramienta, en general, su uso debe de tener en cuenta el espacio de búsqueda (Sus posibles soluciones), estas deben encontrarse en un rango limitado, además debe de ser viable definir una función de aptitud (Fitness, objetivo) que nos indique que tan bien o mal es una respuesta.
- El poder de los algoritmos genéticos se encuentra en que es una técnica robusta, capaz de tratar con éxito gran variedad de problemas de diferentes áreas, incluyendo aquellos en los que otros métodos encuentran dificultades, su problemática se presenta en el costo computacional, en algunos casos elevada, en otros manifiesta ser un tiempo competitivo con el resto de algoritmos de optimización combinatoria.
- Se destaca de la mutación con respecto al cruce, que el primero operador genético es esencial para la convergencia del AG, es quien incluye diversidad en la población genética mientras que el segundo no lo es.
- La selección de los padres es decisiva, puesto que, buenos padres contraen normalmente buenos hijos, es decir, soluciones mejores y más adecuadas.
- Se destaca que los algoritmos genéticos, no requieren de información adicional, de hecho no son recomendables de usarse, si se tiene información adicional, quién resuelve el problema debe de tener una idea del espacio de sus variables de entrada y salida para manejar la elección de cromosomas, generaciones, y así balancear la carga computacional.
- El GA continuamente recalcula el valor fitness, es un proceso adaptativo y estocástico, por lo tanto no es seguro que se optimice la salida y que la solución sea la mejor, en algunos casos puede no converger, es un algoritmo que debe de ser supervisado.

IV. REFERENCIAS

[1]http://educagratis.cl/moodle/course/view.php?id=255

[2]Jyh-Sing. ¨*Neuro-Fuzzy and Soft Computing*¨,1997

[3]http://educagratis.cl/moodle/course/view.php?id=255

[4]http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/t2geneticos

[5]https://github.com/williamegomez/Machine-Learning-Teaching/tree/master/lessons/2018-1/Genetic%20Algorithms

[6]http://algoritmosdelainteligenciaartificia.blogspot.com.co/2015/04/algoritmos-de-la-inteligencia-artificial.html

[7]http://nando1-utb.blogspot.com.co/p/algoritmos-geneticos.html

[8]https://mega.nz/#!s6Z3mS4D!RIYFDw_B30nKMFX2f-0fRJXhS7ETrPS8zMPqLysMees

[9]https://www.google.com.co/search?biw=1366&bih=662&tbm=isch&sa=1&ei=HGb4WsSpNsjr5gKpvoSQCQ&q=seleccion+de+padres%2C+ruleta&oq=seleccion+de+padres%2C+ruleta&gs_l=img.3...21487.23020.0.23251.8.8.0.0.0.0.185.965.4j4.8.0....0...1c.1.64.img..0.2.318...35i39k1.0.drvJ0MV5VRs#imgdii=iQaapEdwIPl1KM:&imgrc=pFCRvOdL7t89eM:

[10]https://www.google.com.co/search?biw=1366&bih=662&tbm=isch&sa=1&ei=NWb4Wp6mDcm05gLt97k4&q=algoritmos+geneticos&oq=algor&gs_l=img.3.0.35i39k1l2j0j0i67k1l3j0l4.309980.310971.0.312046.5.5.0.0.0.0.180.608.0j5.5.0....0...1c.1.64.img..0.5.608....0.eDp1zrFN1_g#imgrc=JRoxjNhNhthLAM: