Programación genética

Alejandro Mendoza Puig[1]

[1] *Posgrado en Ciencia e Ingeniería de la Computación, Universidad Nacional Autónoma de México, Ciudad Universitaria Unidad de Posgrado, Edificio "C" 1er. nivel, Delegación Coyoacán, C.P. 04510, CDMX.*

Fecha de publicación: 21/04/2024

INTRODUCCIÓN

La programación genética es una rama de la computación evolutiva que funciona similar a los algoritmos genéticos, pero en lugar de realizar las operaciones de **cruza, mutación, selección y reemplazo** sobre los individuos de una población los cuales codifican a genotipos que representan una posible solución al problema, con la programación genética se realizan los mismos operadores pero a una función la cual esta buscando optimizar su comportamiento. La población pueden ser vistos como programas de computadora (1)

Los operadores se utilizan para realizar cambios en la estructura de la función, cambiando las operaciones y los valores de las constantes que la componen, con el fin de encontrar una función que se ajuste a los datos de entrenamiento y que sea capaz de generalizar a los datos de prueba.

En esta tarea trabajamos sobre dos problemas, el primero es el de paridad, en el cual dados 3 bits se tiene que generar una respuesta binaria, 1 cuando el número de unos en los 3 bits son pares y 0 de lo contrario. El segundo problema es una regresión simbólica en la cual contamos con 21 valores de X junto con el resultado de f(x).

Para el primer problema estaremos utilizando los siguientes parametros:

- F1: and, or, not
- F2: and, or, not, xor
- T: A, B, C

Para el segundo problema estaremos utilizando los siguientes parametros:

- F: +, -, *, div, cos, sin, log, exp, abs, sqrt, $x\hat{y}$
- T: x, R, k's

Objetivos

- Implementar un algoritmo de programación genética para resolver el problema de paridad.
- Implementar un algoritmo de programación genética para resolver el problema de regresión simbólica.
- Realizar un análisis de los resultados obtenidos.

DESARROLLO

La implementación se realizó en python, utilizando una clase llamada Nodo la cual representa cada nodo en el árbol de la función, se implementaron funciones recursivas para la navegación del árbol, esto trajo problemas a la aplicación los cuales veremos a mayor detalle más adelante.

Clase Nodo

Para contener la estructura de los árboles de las funciones se implementó una clase llamada Nodo, la cual contiene los siguientes atributos:

- **value:** Valor del nodo, puede ser una operación o una constante.
- **hijos:** Lista de los hijos a los que apunta un nodo en especifico.
- **agregar_hijo():** Función que agrega un nuevo nodo al árbol. Implementado para la función de cruza.

Crear Población

Se hizo una funcion llamada crear_arbol() la cual va generando los nodos de manera recursiva. Si el valor para el nodo es una operación que requiere de dos parametros entonces crean los dos hijos de manera recursiva. Cuando se llega a una profuncidad máxima o se cumple con una probabilidad de terminar el arbol se agregan los nodos terminales (A,B,C).

Mutación

PAra la mutación se realizó una función que recorre el árbol con una probabilidad de mutación de 1/num_nodos. Cuando en un nodo se cumple la probabilidad de mutación entonces se genera un árbol nuevo y se coloca en el nodo actual.

Cruza

La cruza es la parte de la implementación donde mayor problema se presentó. Para la cruza se tienen que elegir dos nodos aleatorios en cada uno de los padres y se intercambia los nodos incluyendo toda la subestructura de los nodos.

Para la cruza se realizaron tres funciones:

- **encontrar_subarbol**: Función que recorre el árbol y dada una probabilidad de 1/num_nodos encuentra un nodo aleatorio. Regresa ese nodo junto con todos sus descendientes.

- **insertar_subarbol**: Función que recibe un nodo y un subarbol y lo inserta en el nodo.
- **cruzamiento**: Función que ejecuta las funciones anteriores para cada uno de los padres. Recibe dos padres y una probabilidad de cruza y retorna dos hijos ya con mutación aplicada.

El problema que se presentó fue al momento de encontrar los subarboles e intercambiarlos, al ser una función recursiva se me ha dificultado el poder debuggear a detalle cada una de las cruzas que se realizan. En la función de encontrar_subarbol tuve complicaciones pues en algunos casos en especifico me estaba regresando la funcion un arbol vacio (None) el cual impide que se realice una cruza exitosa pues ninguno de los operadores pueden funcionar con una entrada como (None).

Evaluación

Para la evaluación se realizaron dos funciones, una para evaluar al individuo con un solo renglon de la tabla de verdad y otra que evalua con toda la tabla de verdad y da una metrica de fitness o accuracy.

$$fitness = RespuestasCorrectas/TotalRespuestas \quad (1)$$

Selección

Para selección se utilizó el método de la ruleta sin elitismo, eventualmente lo implementaré con elitismo pero debido al problema encontrado en la cruza (de mayor prioridad) no se pudo implementar.

RESULTADOS

Por el momento no he podido obtener resultados debido a mi problema con la cruza, al ejecutar el programa se paridad el mejor resultado que he obtenido es 0.875, lo cual quiere decir que el algoritmo obtuvo 7/8. Para poder mejorar esto solucionaré el problema con la cruza y me asegurare también que la profundidad de cada individuo no crezca de más al momento de hacer una mutación o una cruza ya que esto tambien puede terminar ofuscando las soluciones.

CONCLUSIONES

Durante el desarrollo de la tarea aprendí muy bien como es que se aplican los operadores de algoritmos genéticos en la programación genética. Las mutaciónes y cruzas son operadores que afectan demasiado a los individuos ya que se intercambian secciones completas del árbol, esto puede ayudar mucho para salir de optimos locales pero también puede afectar mucho a la solución. En la implementación se permitió que la mutación o la cruza generara individuos con profundidad mayor a la permitida y luego con otra función se recortan al máximo permitido, esto puede ser como los genes que no codifican hacia ningún fenotipo y que en realidad no afectan mucho sobre la selección natural.

REFERENCIAS

[1] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, Massachusetts and London, England: MIT Press, 1992, page iii.