Programación genética

Introducción

La **Programación Genética** consiste en la evolución automática de programas usando ideas basadas en la selección natural (Darwin).

No sólo se ha utilizado para generar programas, también para otros tipos de soluciones cuya estructura sea similar a la de un programa. Por ejemplo, fórmulas matemáticas o circuitos electrónicos.

Introducción

La evolución se produce en la naturaleza gracias a que:

- Existe reproducción entre individuos de una población.
- Las características de los individuos afectan a su probabilidad de supervivencia.
- Existe herencia.
- Existen recursos finitos, lo que ocasiona competencia.

En programación genética se busca que las poblaciones de programas evolucionen, transmitiendo su herencia de manera que se adapten mejor al medio. Los mejores individuos tienen mayores probabilidades de reproducirse. La medida de calidad del individuo depender´a del tipo de problema.

Introducción

Un algoritmo de programación genética sigue el esquema:

1. Genera una población inicial.

2. Mientras no se cumpla el criterio de parada:

   a) Seleccionar individuos (para reproducción y eliminación), considerando su calidad.

   b) Combinar y/o variar individuos nuevos.

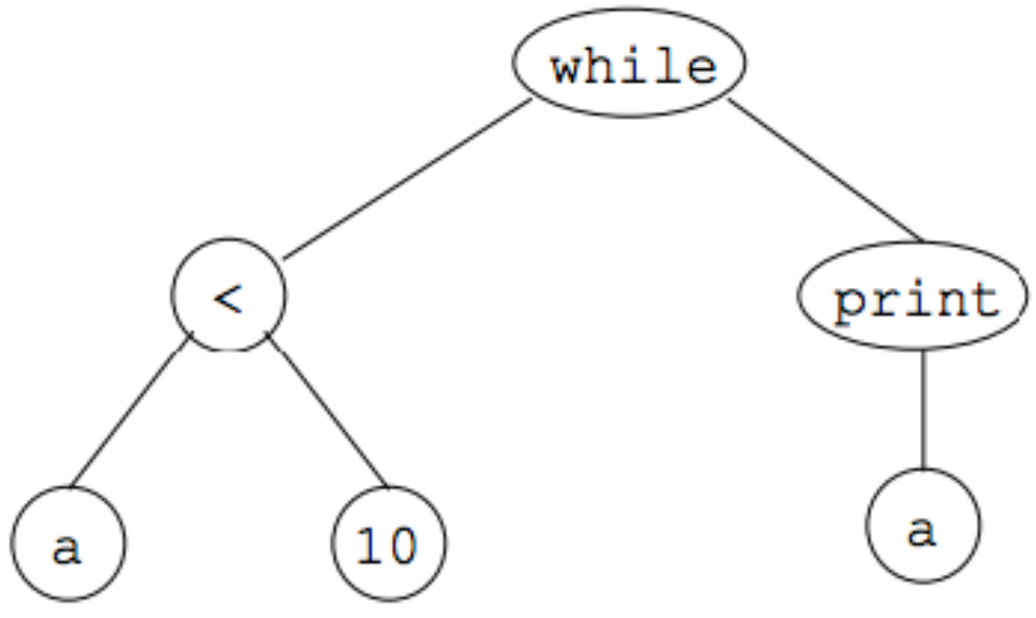
   c) Agregar y eliminar individuos.

Representación

En programación genética, los programas (o individuos) se representan como árboles. Es así como el segmento de código:

```
while (a<10) {
 print(a);
 a++
}
```

puede representarse por el árbol:



Terminales y funciones

El conjunto de terminales está compuesto por las entradas posibles al individuo, constantes y funciones de aridad 0.

El conjunto de funciones está compuesto por los operadores y funciones que pueden componer a un individuo. Por ejemplo:

- Funciones booleanas: AND, OR, NOT, XOR.
- Funciones aritméticas: PLUS, MINUS, MULT, DIV.
- Sentencias condicionales: IF, THEN, ELSE, CASE, SWITCH
- Sentencias para iteraciones: WHILE, FOR, REPEAT..UNTIL

El conjunto de terminales y funciones elegidos para resolver un problema particular debe ser, obviamente, suficiente para representar una solución al problema. Por otro lado, no es conveniente usar un número grande de funciones, debido a que esto aumenta el tamañoo del espacio de búsqueda (principio de parsimonia). Además es deseable que las funciones puedan manejar todos los argumentos que eventualmente podrían llegar a tener.

Inicialización

El primer paso en programación genética consiste en formar la población inicial de individuos.

Uno de los parámetros principales para un algoritmo genético es el tamaño máximo de un programa. Este límite puede estar impuesto sobre el **número de nodos** o sobre la **profundidad del árbol**.

Usualmente, se utilizan dos métodos para generar esta población: el método de **grow** y el **full**.

Método *grow*

- Sea ***T*** el conjunto de terminales y ***F*** el conjunto de funciones.
- Se elige aleatoriamente un elemento de ***F*** para que conforme la raíz del árbol.
- El contenido de los nodos hijos de la raíz se elige desde ***F*** ∪ ***T***. Si el valor elegido es una función, se repite este procedimiento con los hijos. (si el valor elegido es una constante, se termina esa rama del árbol.)

Método *full*

- El método *full* hace crecer el árbol en forma similar al método *grow*, pero siempre se eligen elementos del conjunto de funciones ***F***, a menos que el nodo haya alcanzado la profundidad máxima permitida, en cuyo caso sólo se eligen elementos de ***T***.

- El resultado de este método son siempre árboles balanceados de profundidad máxima.

- Si se usa el número de nodos como límite de tamaño, el crecimiento se termina cuando el tamaño árbol ha alcanzado el límite

Operadores genéticos

Los operadores básicos son:

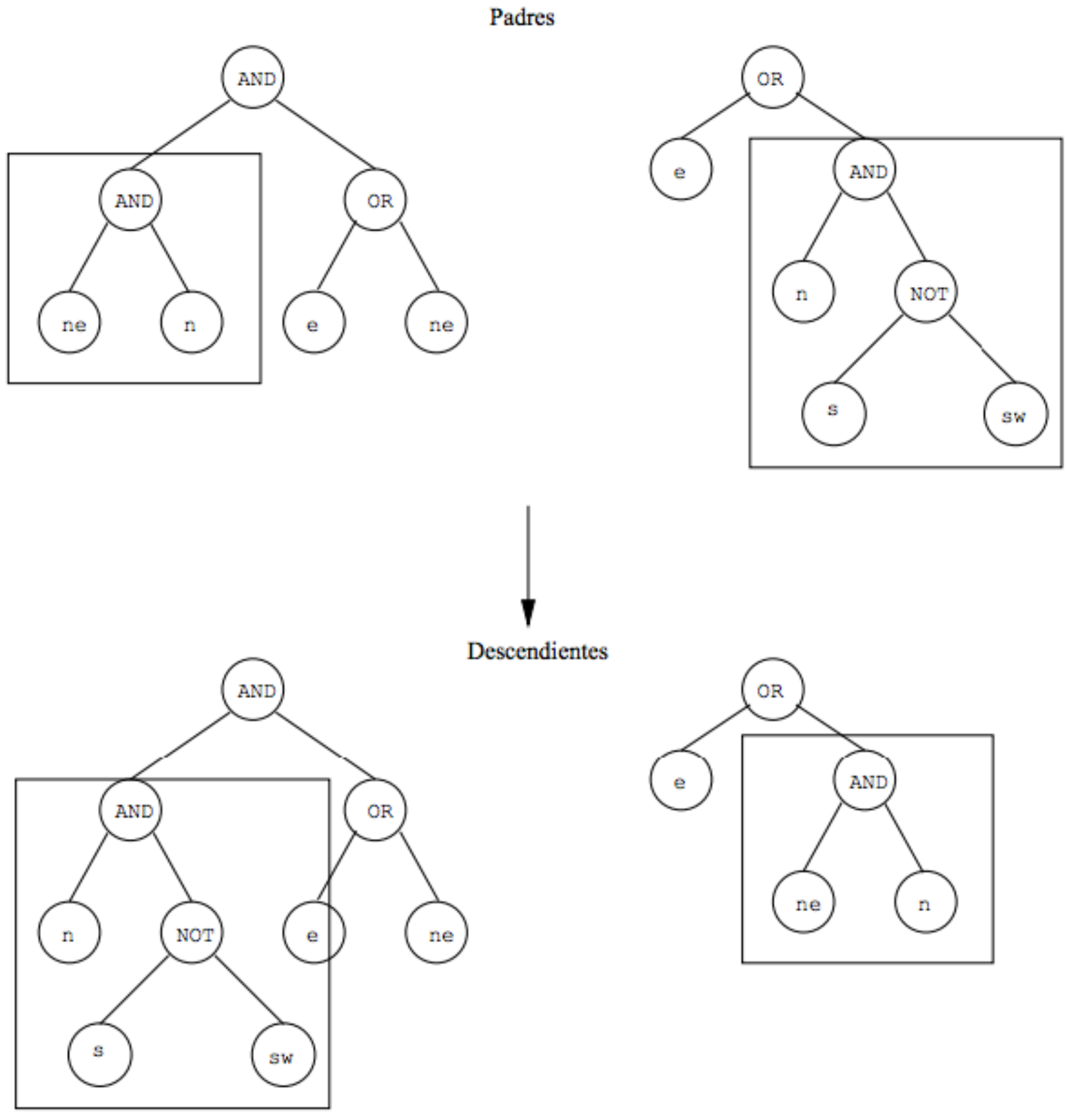
- Crossover
- Mutación
- Reproducción

Crossover

- El operador de cruce (crossover) combina el material genético de individuos intercambiando pedazos de los dos progenitores para producir dos descendientes.

- Si los individuos se representan como árboles, se elige al azar un nodo de cada árbol y luego se intercambian los subárboles bajo estos nodos.

Crossover


Padres
AND
AND
OR
ne
n
e
ne
OR
e
AND
n
NOT
s
sw
Descendientes
AND
AND
OR
n
NOT
e
ne
s
sw
OR
e
AND
ne
n

Mutación

- Actúa sobre un solo individuo, generalmente, uno resultante de un crossover.

- La mutación actúa con una probabilidad, en general, muy baja y es un parámetro del algoritmo de programación genética.

- Un tipo de mutación consiste en escoger en forma aleatoria un nodo del árbol y generar bajo éste otro subárbol (por ejemplo, usando el método grow).

Nombre del operador	**Descripción del efecto**
Mutación puntual	Un solo nodo es intercambiado por otro de la misma clase
Permutación	Los argumentos de un nodo son permutados
Levantamiento	Nuevo individuo es generado a partir de un subárbol
Expansión	Un terminal es cambiado por un árbol generado al azar
Colapso	Subárbol es intercambiado por un terminal
Mutación de Subárbol	Subárbol es reemplazado por otro generado al azar.
Duplicación de Gen	Subárbol es reemplazado por un terminal al azar.

Reproducción

- Es el operador más simple de todos. Se selecciona un individuo y se duplica, quedando dos copias dentro de la población.

Función de calidad

- La **calidad** es la medida usada por el algoritmo de programación genética durante la evolución simulada para medir qué tan buena es una solución.

- Una función de calidad se dice **estandarizada positiva** si siempre asigna el valor 0 al individuo con mejor solución para el problema.

- Una función de calidad se dice **normalizada** si siempre asigna valores entre 0 y 1.

- Si el objetivo es aprender una función matemática $g(x)$, entonces una medida de calidad puede ser el **error cuadrático medio**.

Función de calidad

- Así, si ***I*** es un conjunto de entradas y $g(i) = o_i$ es la salida para una entrada i, entonces podríamos medir la **calidad** de un programa p mediante:

$$f_p = \sum_{i \in \mathcal{I}} (p_i - o_i)^2$$

Selección

La **selección** es el proceso por el cual se transmiten individuos de una generación a otra. Hay distintos tipos de selección.

Método basado en **algoritmos genéticos**:

1. Elegir a dos individuos progenitores, privilegiando los que posean mejor calidad.

$$Pr(i) = \frac{f(i)}{\sum_i f(i)}$$

2. Aplicar crossover (si corresponde, probabilísticamente).
3. Aplicar mutación (si corresponde, probabilísticamente).
4. Reproducir.
5. Repetir hasta completar una nueva generación de N individuos.

Selección

Otra técnica consiste en efectuar **torneos**:

1. Elegir dos grupos de *n* individuos aleatoriamente desde la población.

2. Seleccionar al mejor elemento del primer grupo (padre), y al mejor elemento del segundo (madre).

3. Aplicar *crossover* (si corresponde, probabilísticamente).

4. Aplicar mutación (si corresponde, probabilísticamente).

5. Los dos nuevos individuos reemplazan a los peores de cada uno de los grupos.

Esta última técnica es generalmente preferida por razones de eficiencia.

Condición de terminación

Hemos visto que para hacer evolucionar una población es necesario:

- Generar una población inicial.
- Seleccionar individuos para producir nuevas generaciones.

¿Cuando nos detenemos? Depende de lo que queramos, pero en general cuando el mejor individuo de la población tiene una **calidad aceptable**.