ALGORITMOS GENÉTICOS

Autor: Miguel Ángel Muñoz Pérez.
Primera versión: Noviembre, 1997.
Última modificación: Abril, 2005.

ADVERTENCIA

Si ya sabes lo que es un algoritmo genético y esperas hallar algo novedoso en esta página, me temo que aquí no lo vas a encontrar. Este documento no es de investigación, sino de divulgación y es "free". En este caso acepto gustoso todas tus críticas y sugerencias.

Si no te interesan las matemáticas, no creo que sea de interés lo que aquí se cuenta.

Entonces, ¿a quién le puede interesar? Si no has oído hablar de un algoritmo genético, pero te gustan los números o si ya sabes que existen pero no conoces cómo funcionan, te invito a que sigas leyendo.

No es preciso que seas un mago con los números. Basta con que sepas el concepto de función y poco más. Si conoces la representación de números en forma binaria, pues mejor, y si no, yo te la explico.

FUNCIONAMIENTO DE UN ALGORITMO GENETICO

Vamos a partir de una función f(x) muy sencilla:

$f(x) = x^2$

(es decir, x al cuadrado).

Imagina que deseas encontrar el valor de x que hace que la función f(x) alcance su valor máximo, pero restringiendo a la variable x a tomar valores comprendidos entre 0 y 31. Aún más, a x sólo le vamos a permitir tomar valores enteros, es decir: 0,1, 2, 3,..., 30, 31. Obviamente el máximo se tiene para x = 31, donde f vale 961. No necesitamos saber algoritmos genéticos para resolver este problema, pero su sencillez hace que el algoritmo sea más fácil de comprender.

Lo primero que debemos hacer es encontrar una manera de codificar las posibles soluciones (posible valores de x). Una manera de hacerlo es con la codificación binaria. Con esta codificación un posible valor de x es

(0, 1, 0, 1, 1).

¿Cómo se interpreta esto? Muy sencillo: multiplica la última componente (un 1) por 1, la penúltima (un 1) por 2, la anterior (un 0) por 4, la segunda (un 1) por 8 y la primera (un 0) por 16 y a continuación haz la suma: 11. Observa que (0, 0, 0, 0, 0) equivale a x = 0 y que (1, 1, 1, 1, 1) equivale a x = 31.

A cada posible valor de la variable x en representación binaria le vamos a llamar individuo. Una colección de individuos constituye lo que se denomina población y el número de individuos que la componen es el tamaño de la población.

Una vez que tenemos codificada la solución, debemos escoger un tamaño de población. Para este ejemplo ilustrativo vamos a escoger 6 individuos.

Debemos partir de una población inicial. Una manera de generarla es aleatoriamente: coge una moneda y lánzala al aire; si sale cara, la primera componente del primer individuo es un 0 y en caso contrario un 1. Repite el lanzamiento de la moneda y tendremos la segunda componente del primer individuo (un 0 si sale cara y un 1 si sale cruz). Así hasta 5 veces y obtendrás el primer individuo. Repite ahora la secuencia anterior para generar los individuos de la población restantes. En total tienes que lanzar 5 * 6 = 30 veces la moneda.

Nuestro siguiente paso es hacer competir a los individuos entre sí. Este proceso se conoce como selección. La tabla 1 resume el proceso.

Tabla 1.- SELECCION

(1)	(2)	(3)	(4)	(5)
1	(0,1,1,0,0)	12	144	6
2	(1,0,0,1,0)	18	324	3
3	(0,1,1,1,1)	15	225	2
4	(1,1,0,0,0)	24	576	5
5	(1,1,0,1,0)	26	676	4
6	(0,0,0,0,1)	1	1	1

Cada fila en la tabla 1 está asociada a un individuo de la población inicial. El significado de cada columna de la tabla es el siguiente:

(1) = Número que le asignamos al individuo.

(2) = Individuo en codificación binaria.

(3) = Valor de x.

(4) = Valor de f(x).

Observa que el mejor individuo es el 5 con f = 676. Calcula la media de f y obtendrás fmed =324.3.

En cuanto a la columna (5) ahora te lo explico. Una manera de realizar el proceso de selección es mediante un torneo entre dos. A cada individuo de la población se le asigna una pareja y entre ellos se establece un torneo: el mejor genera dos copias y el peor se desecha. La columna (5) indica la pareja asignada a cada individuo, lo cual se ha realizado aleatoriamente. Existen muchas variantes de este proceso de selección, aunque este método nos vale para ilustrar el ejemplo.

Después de realizar el proceso de selección, la población que tenemos es la mostrada en la columna (2) de la tabla 2. Observa, por ejemplo, que en el torneo entre el individuo 1 y el 6 de la población inicial, el primero de ellos ha recibido dos copias, mientras que el segundo cae en el olvido.

Tabla 2.- CRUCE

(1)	(2)	(3)	(4)
1	(0,1,1,0,0)	5	1
2	(0,1,1,0,0)	3	3
3	(1,0,0,1,0)	2	3
4	(1,0,0,1,0)	6	1
5	(1,1,0,1,0)	1	1
6	(1,1,0,1,0)	4	1

Tras realizar la selección, se realiza el cruce. Una manera de hacerlo es mediante el cruce 1X: se forman parejas entre los individuos aleatoriamente de forma similar a la selección. Dados dos individuos pareja se establece un punto de cruce aleatorio, que no es más que un número aleatorio entre 1 y 4 (la longitud del individuo menos 1). Por ejemplo, en la pareja 2-3 el punto de cruce es 3, lo que significa que un hijo de la pareja conserva los tres primeros bits del padre y hereda los dos últimos de la madre, mientras que el otro hijo de la pareja conserva los tres primeros bits de la madre y hereda los dos últimos del padre. La población resultante se muestra en la columna (2) de la tabla 3.

Tabla 3.- POBLACION TRAS EL CRUCE			
(1)	**(2)**	**(3)**	**(4)**
1	(0,1,0,1,0)	10	100
2	(1,1,1,0,0)	28	784
3	(0,1,1,1,0)	14	196
4	(1,0,0,0,0)	16	256
5	(1,1,0,1,0)	26	676
6	(1,0,0,1,0)	18	324

En la columna (3) tienes el valor de x; en la siguiente tienes el valor de f correspondiente.

Fíjate en que ahora el valor máximo de f es 784 (para el individuo 2), mientras que antes de la selección y el cruce era de 676. Además fmed ha subido de 324.3 a 389.3. ¿Qué quiere decir esto? Simplemente que los individuos después de la selección y el cruce son mejores que antes de estas transformaciones.

El siguiente paso es volver a realizar la selección y el cruce tomando como población inicial la de la tabla 3. Esta manera de proceder se repite tantas veces como número de iteraciones tú fijes. Y ¿cuál es el óptimo? En realidad un algoritmo genético no te garantiza la obtención del óptimo pero, si está bien construido, te proporcionará una solución razonablemente buena. Puede que obtengas el óptimo, pero el algoritmo no te confirma que lo sea. Así que quédate con la mejor solución de la última iteración. También es buena idea ir guardando la mejor solución de todas las iteraciones anteriores y al final quedarte con la mejor solución de las exploradas.

CONSIDERACIONES ADICIONALES

En problemas reales en los que se aplican los algoritmos genéticos, existe la tendencia a la homogeinización de la población, es decir a que todos los individuos de la misma sean idénticos. Esto impide que el algoritmo siga explorando nuevas soluciones, con lo que podemos quedar estancados en un mínimo local no muy bueno.

Existen técnicas para contrarrestar esta "deriva genética". El mecanismo más elemental, aunque no siempre suficientemente eficaz, es introducir una mutación tras la selección y el cruce. Una vez que has realizado la selección y el cruce escoges un número determinado de bits de la población y los alteras aleatoriamente. En nuestro ejemplo consiste simplemente en cambiar algunos(s) bit(s) de 1 a 0 ó de 0 a 1.

BIBLIOGRAFÍA

Nota. No se trata de una relación exhaustiva y completa de toda la bibliografía existente sobre algoritmos genéticos. El lector puede consultar en la bibliografía aquí citada, referencias adicionales sobre esta técnica.

DAVIS, L. (1991): Handbook of Genetic Algorithms. Van Nostrand Reinhold.

DIAZ, A. y GLOVER, F. (1996): Optimización Heurística y Redes Neuronales en Dirección de Operaciones e Ingeniería. Paraninfo.

GOLDBERG, D.E. (1989): Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, MA.

HOLLAND, J. (1975): Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor.

MICHALEWICZ, Z. (1992): Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag.

HIPERENLACES

GRUPO DE INGENIERÍA DE ORGANIZACIÓN. UNIVERSIDAD DE SEVILLA. http://io.us.es/enlaces.htm [Última consulta: Abril de 2005].

GALIB: MATTHEW'S GENETIC ALGORITHMS LIBRARY. http://lancet.mit.edu/ga/ [Última consulta: Abril de 2005].

GENEWOOD. http://www.genewood.host.sk [Última consulta: Abril de 2005].

INFORMÁTICA EVOLUTIVA: ALGORITMOS GENÉTICOS. http://geneura.ugr.es/~jmerelo/ie/ags.htm [Última consulta: Abril de 2005].

REDCIENTÍFICA - INTRODUCCIÓN A LOS ALGORITMOS GENÉTICOS. http://www.redcientifica.com/imprimir/doc199904260011.html [Última consulta: Abril de 2005].