# Algoritmos Genéticos:

Resumen: Un algoritmo genético es un algoritmo aproximado para resolver problemas de optimización, típicamente sobre conjuntos discretos. Aunque no dan garantía de producir el valor óptimo global producen una "población" de puntos factibles con buenos valores de la función objetivo. Son fáciles de programar y permiten atacar una enorme cantidad de Problemas muy distintos de manera eficiente y robusta.

Más formalmente, tenemos:

$X$ — Conjunto finito de cadenas admisibles.

(Típicamente todas las cadenas de una longitud $L$ de $\{0,1\}$ $00\ldots0$, $0110\ldots1$ ← enorme $2^L$ elementos; $L$-letras)

$f$ — Función objetivo $f: X \longrightarrow \mathbb{R}$

que queremos maximizar

(típicamente esta es una "caja negra" no pedimos ninguna propiedad particular en $f$)

$I_0$ — Población inicial $I_0 \subseteq X$

(típicamente es un conjunto pequeño dentro de $X$ de tamaño $n << |X|$).

Queremos $\max_{x \in X} f(x)$ y $\operatorname{argmax}_{x \in X} f(x)$

($\max$: valor; $\operatorname{argmax}$: cadenas de valor máximo.)

El algoritmo genético procede mediante 3 etapas que se repiten en orden:

(i) Reproducción

(ii) Cross-Over (recombinación)

(iii) Mutación

produciendo una población que evoluciona

$I_0 \to I_1 \to I_2 \to \dots \to I_k$ (k = #de iteraciones que queramos.)

Describiremos $I_t \to I_{t+1}$ ① (i), (ii) y (iii)

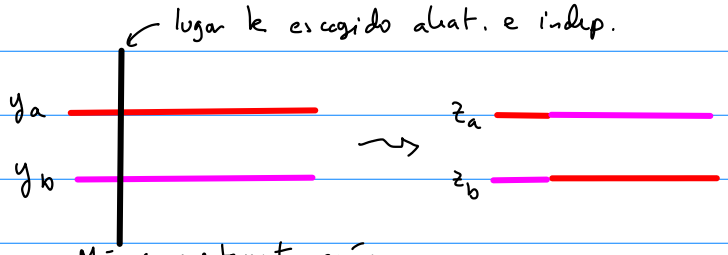
(i) Reproducción

(i.1) Para cada $x_i \in I_t$ calculamos $f(x_i) = f_i$ y definimos su probabilidad de reproducción como $\tilde{f}_i := \frac{f_i}{\sum_{i=1}^{n} f_i}$

(i.2) Sorteamos n candidatos independientemente haciendo que el candidato $y_j$ sea una copia de $x_i$ con probabilidad $\tilde{f}_i$

(Los $y_i$ no son todos distintos, los $x_i$ más exitosos se reproducen más).

(ii) CrossOver: Partimos el conjunto de los $y_i$ en n parejas. Cada pareja $\{y_a, y_b\}$ se recombina escogiendo un lugar aleatoriamente así:




Más concretamente, así:

00 110 → 00 001

10 001 → 10 110

Obtenemos $z_1, \dots, z_n$

(iii) Mutación para cada posición de la cadena $z_i$ sorteamos una moneda. Con muy baja probabilidad (q ~ 0 escogida por nosotros) el bit correspondiente se cambia "mutando" $z_i$ por $\tilde{z}_i$.

DEFINIMOS $I_{t+1} := \{\tilde{z}_1, \dots, \tilde{z}_n\}$ ← nueva población

**Ejemplo:** Dígitos en binario

$11111 = 1+2+2^2+2^3+2^4 = 2^5-1 = 31$

$00010 = 2$

$00001 = 1$

$00000 = 0$

Tome $f(x) = x^2$ entonces $f(11111) = 31^2$

Población inicial (completamente aleatoria) $I_0 =$

0 1 1 0 1
1 1 0 0 0
0 1 0 0 0
1 0 0 1 0

sólo 4 strings

¿Cómo evoluciona la población? En todos los instantes tenemos una población de tamaño 4 y lo importante es entender cómo varía:

(1) El mejor fitness de la población y
(2) El fitness promedio de la población.
(3) El peor fitness de la población.

¿Qué pasa si quitamos la recombinación?
¿Qué pasa si variamos la tasa de mutación?