Algoritmos Genéticos

como motor de clasificaciones

IEEE ARGENTINA

UP | Universidad de Palermo
Facultad de Ingeniería

Próxima charla

Análisis de calidad Java con Java (con NN) (D. López De Luise, M. Agüero) 21/11/05 (18:30hs)

informes:

upgrade@palermo.edu

sec.argentina@ieee.org

UP

Universidad de Palermo (Depto. Ing. Informática)

IEEE

Institute of Electric and Electronic Engineers (cap. local CIS)

Internetworking the world

Objetivo

- Presentar el uso de AG para clasificaciones
- Presentar resultados con un prototipo en Java

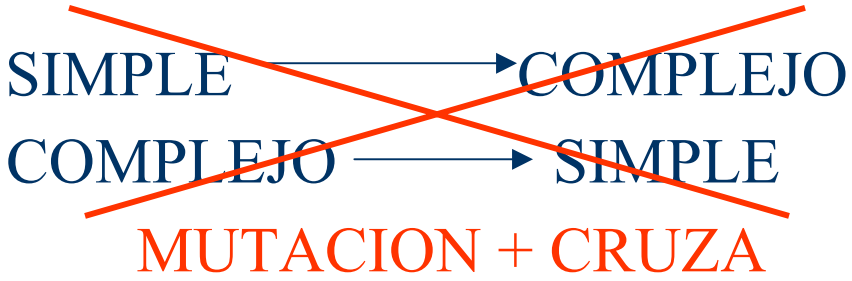
Temario

- Origen de AG
- AG como clasificador
- Proyecto FIC

Origen de AG

Motivación

analogía biológica.



- La evolución es un mecanismo robusto
- Modela interacciones complejas
- Paralelizables

Origen de AG

Historia

fines ‘50 → primeros intentos → basado en mutación

ppios ‘60 → Hans J. Bremermann → cruza como suma

mediados ‘60 → John H Holland → cruza + mutación

‘70 → John H Holland → código clasificador

perros :: [011 111 000001]

color (alto-medio-bajo) | país origen

Actualidad → extensión a varios campos

→ alternativas de modelización

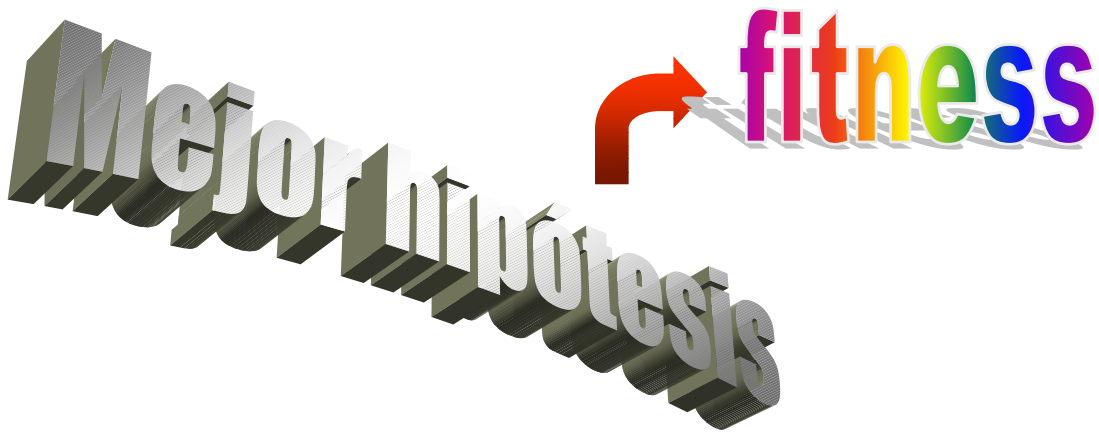
AG como clasificador

Aplicabilidad

- AG permite paralelizar implícitamente la búsqueda=> más regiones
- AG permite resolver problemas lineales y no lineales
- Balance exploración-explotación
- Se probó exitosamente en varios contextos:
  - control de tuberías de gas
  - diseño de redes de comunicaciones
  - diseño de turbinas de jets
  - detección de metales
  - simulación de agentes económicos
- Aún queda mucha potencialidad (alrededor de 8000 reglas)
- Rick L. Riolo observó que los AG como "latent learning"

AG como clasificador

El problema



AG como clasificador

La operatoria

Estructura

AG como clasificador

El algoritmo

AG(Fitness, Fitness_threshold, p, r, m) {

P:=conjunto de p hipótesis generadas al azar

calcular Fitness(h), para todo h en P

mientras (max{Fitness(h)} < Fitness_threshold:

crear PS: a) seleccionar (1-r)p miembros de P para PS con $P(hi)=\frac{Fitness\ (H_i)}{\Sigma_i Fitness(h_i)}$

agregarlos a PS

b) seleccionar r.p/2 pares de hipótesis y cruzarlos.

agregar los hijos a PS

c) mutar el m% de los miembros de PS

P:=PS

Evaluar Fitness(h)

retornar hipótesis dentro de P con mayor fitness }


fin