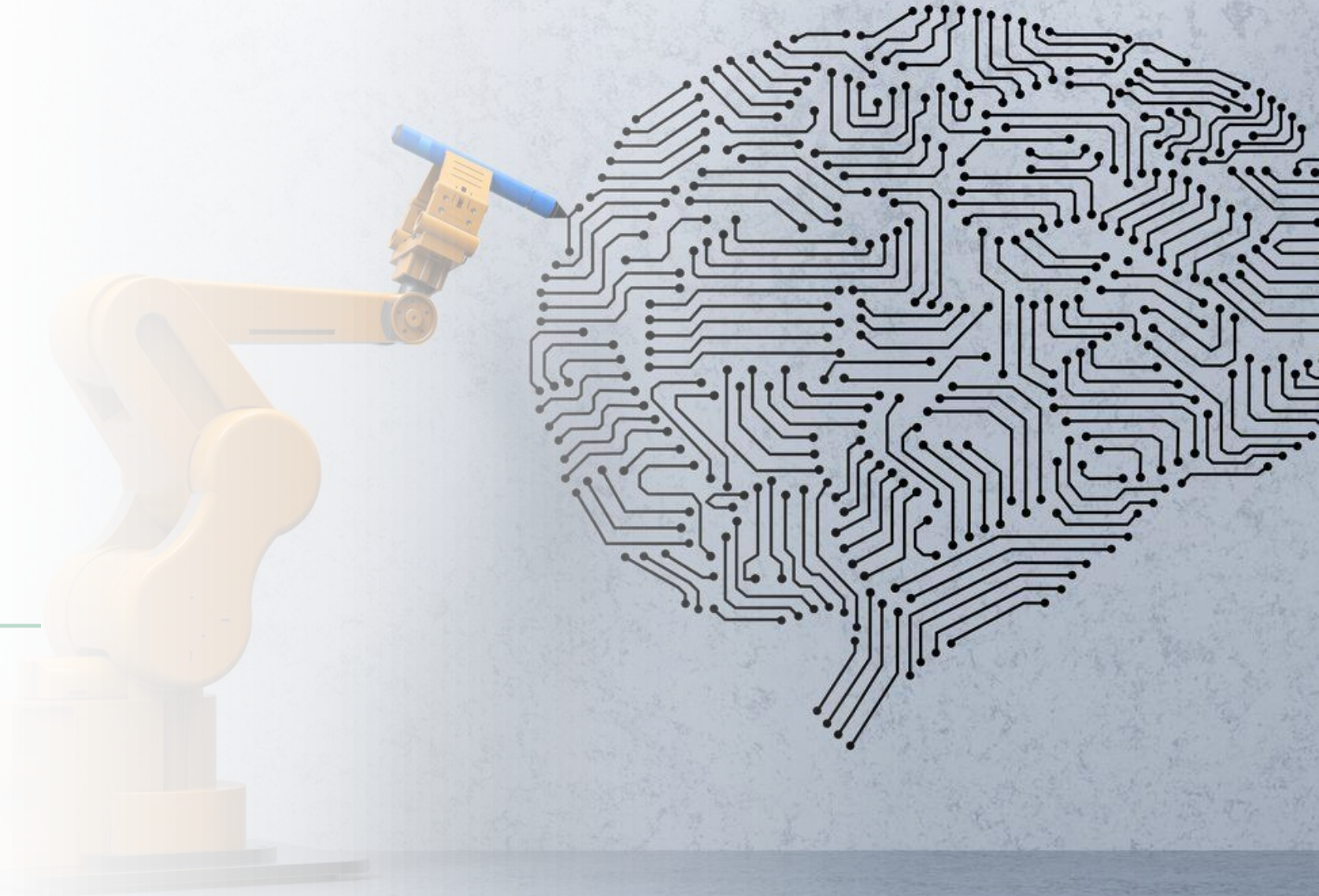


Aprendizaje por refuerzo

Clase 13: ES y neuroevolución



iimas



Antes de empezar I...

- En máximo dos páginas, entregar lo siguiente:
- Categoría (tesis, aplicación, reproducción, investigación)
- Integrantes
- El problema
 - ¿Qué tarea o problema se estudiará?
 - ¿Dónde se obtendrán los datos, simulador o sistema del mundo real?
 - ¿Cuál es la principal hipótesis que se investigará?
 - ¿Cómo se relaciona con RL?
- Objetivos
- Metodología
 - Describir la clase de métodos que se utilizarán (de acuerdo al curso)
 - ¿Qué literatura se utilizará para evaluar los resultados? cualitativa y cuantitativa?

Antes de empezar II... tarea 3

Implementación de

- (20 puntos) DDQN
- (20 puntos extra) Estrategias evolutivas
- (20 puntos extra) Agregar un método de exploración diferente a (épsilon) voraz

Problemas

- (10 puntos) CartPole
- (10) 2 de los 5 juegos de MinAtar (<https://github.com/kenjyoung/MinAtar>)
- (20 puntos) Pong

Entregables (en pdf)

- (-20 puntos) Parámetros: número de episodios, α , política utilizada, otros parámetros relevantes
- (-20 puntos) Entrenamiento: gráfica de convergencia episodios vs recompensa obtenida para cada algoritmo
- (-20 puntos) Resultado: gráfica de media de recompensa acumulada de cada método en cada problema

Notas

- Referencia: Deep reinforcement learning Hands-on (Capítulos 8 y 20)
- Consideren usar Colab para acceso a GPU
- Entrega: 2 de abril

Para el día de hoy...

- Estrategias evolutivas
- Optimización de política
- Neuroevolución
- Un poco de implementación





E| escandalo...

Evolution Strategies as a Scalable Alternative to Reinforcement Learning

Tim Salimans Jonathan Ho Xi Chen
OpenAI Szymon Sidor Ilya Sutskever

Natural Evolution Strategies

Daan Wierstra DAAN@DEEPMIND.COM
Tom Schaul TOM@DEEPMIND.COM
DeepMind Technologies Ltd.
Fountain House, 130 Fenchurch Street
London, United Kingdom

Tobias Glasmachers TOBIAS.GLASMACHERS@INI.RUB.DE
Institute for Neural Computation
Universitätsstrasse 150
Ruhr-University Bochum, Germany

Yi Sun YI@IDSIA.CH
Google Inc.
1600 Amphitheatre Pkwy
Mountain View, United States

Jan Peters MAIL@JAN-PETERS.NET
Intelligent Autonomous Systems Institute
Hochschulstrasse 10
Technische Universität Darmstadt, Germany

Jürgen Schmidhuber JUERGEN@IDSIA.CH
Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA)
University of Lugano (USI)/SUPSI
Galleria 2
Manno-Lugano, Switzerland

Editor: Una-May O'Reilly

Simple random search provides a competitive approach
to reinforcement learning

Horia Mania Aurelia Guy Benjamin Recht

Department of Electrical Engineering and Computer Science
University of California, Berkeley

March 20, 2018

Deep Neuroevolution: Genetic Algorithms are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning

Felipe Petroski Such Vashisht Madhavan Edoardo Conti Joel Lehman Kenneth O. Stanley Jeff Clune

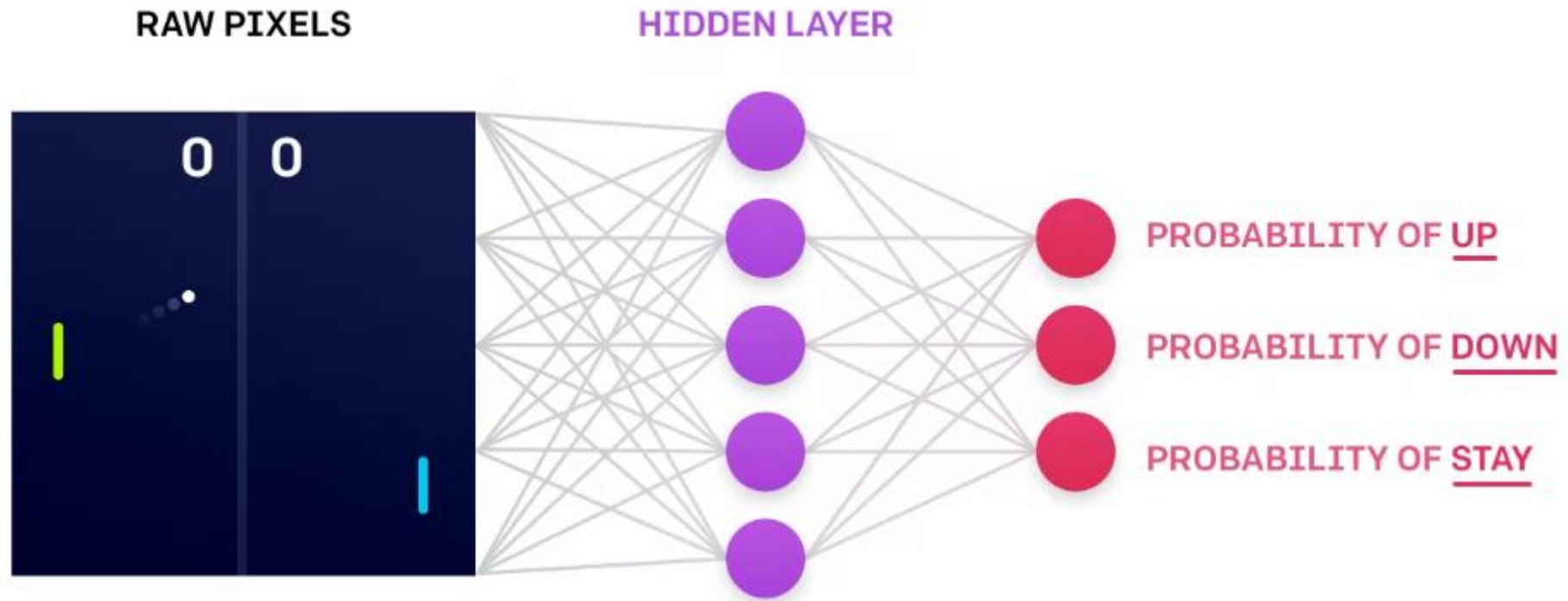
Uber AI Labs
{felipe.such, jeffclune}@uber.com

¿Escandalo?

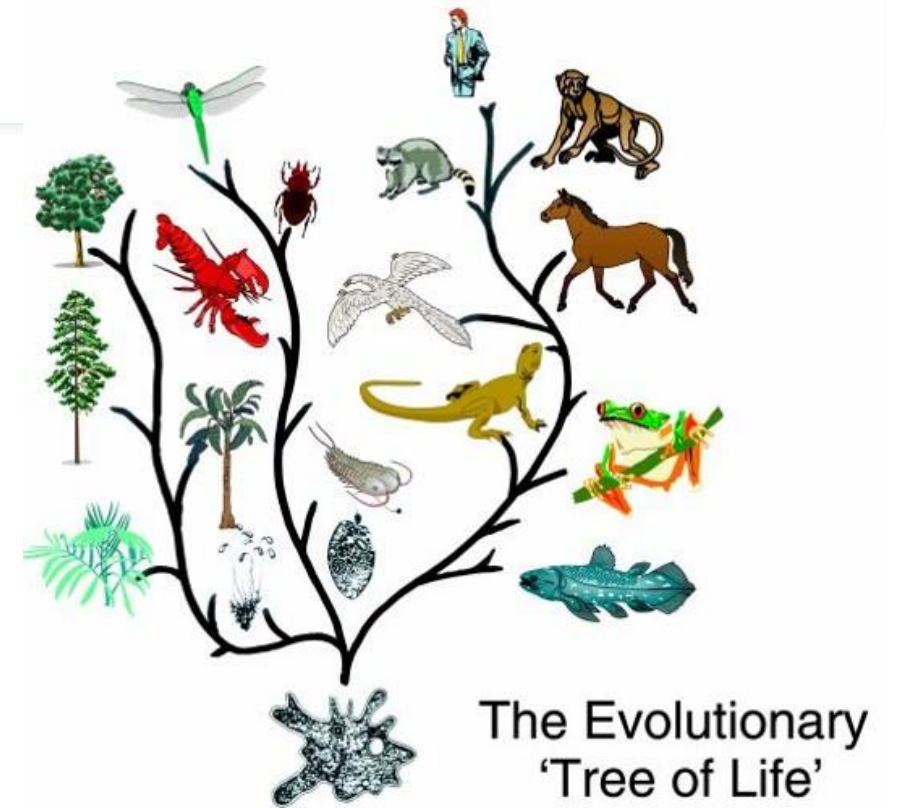
In practice, genetic algorithms have had a widespread impact on optimization problems, such as circuit layout and job-shop scheduling. At present, it is not clear whether the appeal of genetic algorithms arises from their performance or from their aesthetically pleasing origins in the theory of evolution. Much work remains to be done to identify the conditions under which genetic algorithms perform well.


In practice, **genetic** algorithms have their place within the broad landscape of optimization methods (Marler and Arora, 2004), particularly for complex structured problems such as circuit layout or job-shop scheduling, and more recently for evolving the architecture of deep neural networks (Miikkulainen *et al.*, 2019). It is not clear how much of the appeal of **genetic** algorithms arises from their superiority on specific tasks, and how much from the appealing metaphor of evolution.

El problema



- La evolución nos ha traído hasta aquí
- Se trata de un proceso natural bajo el cual aquellos individuos más aptos tienen mayor oportunidad de pasar sus genes a otras generaciones
- Supervivencia del más apto
- ¿Funcionará para resolver nuestros problemas?
- La idea: diseñar agentes que simulen el proceso evolutivo para resolver problemas





Elementos para simular el proceso evolutivo

Codificar las estructuras que se replicarán

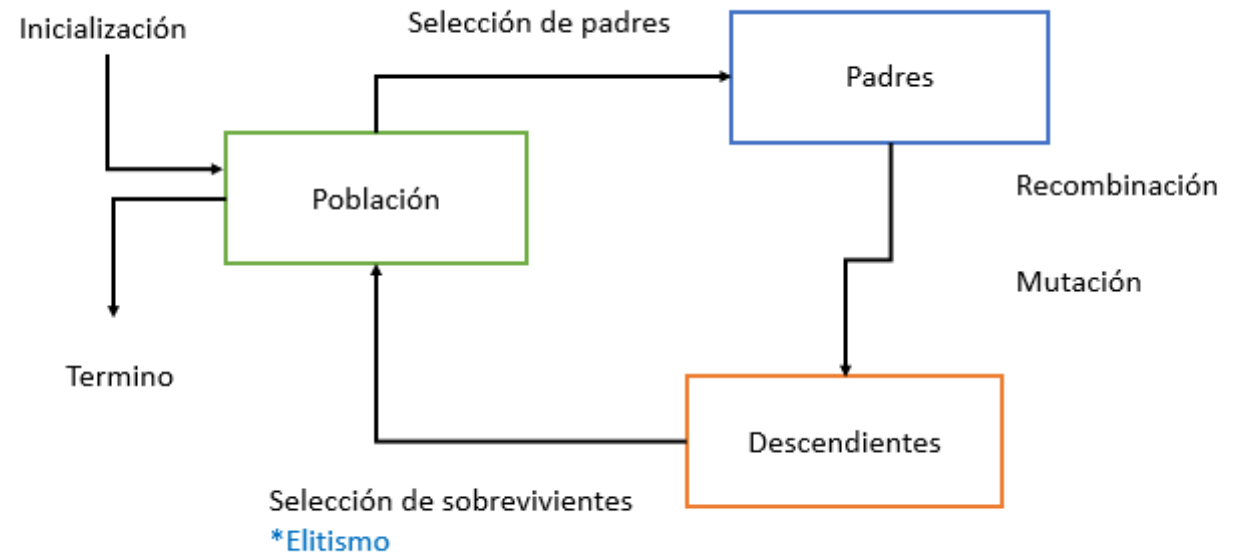
Operaciones que afecten a los individuos

Una función de aptitud

Un mecanismo de selección

El algoritmo básico

- Generar (aleatoriamente) una población inicial.
- Calcular aptitud de cada individuo.
- Seleccionar (probabilísticamente) con base en aptitud.
- Aplicar operadores genéticos (cruza y mutación) para generar la siguiente población.
- Ciclar hasta que cierta condición se satisfaga



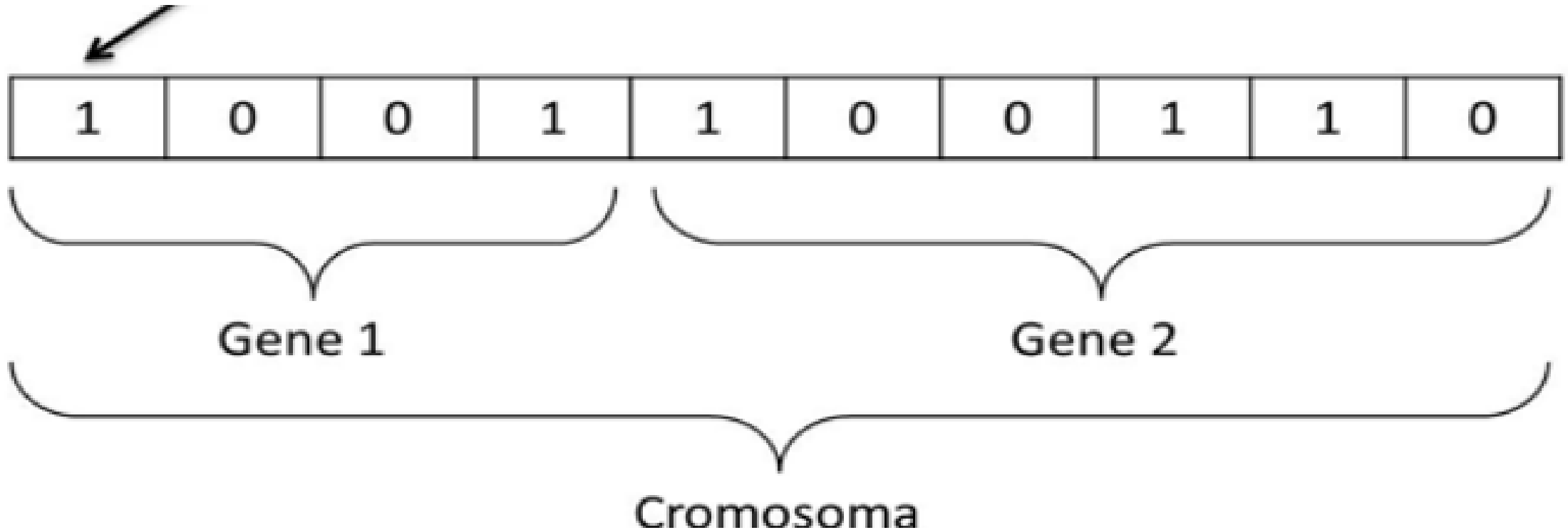
Selección por torneo

- La idea es seleccionar a los padres con base en comparaciones directas
 - Determinista: el mejor gana siempre
 - Probabilística: el mejor gana con probabilidad $0.5 \leq p < 1$
- Algoritmo: hasta seleccionar el número de padres deseado
 - Shuffle de la población
 - Seleccionar P individuos que participaran en el torneo
 - Compararlos con base en aptitud
 - El ganador es el individuo más apto
- Complejidad $O(n)$

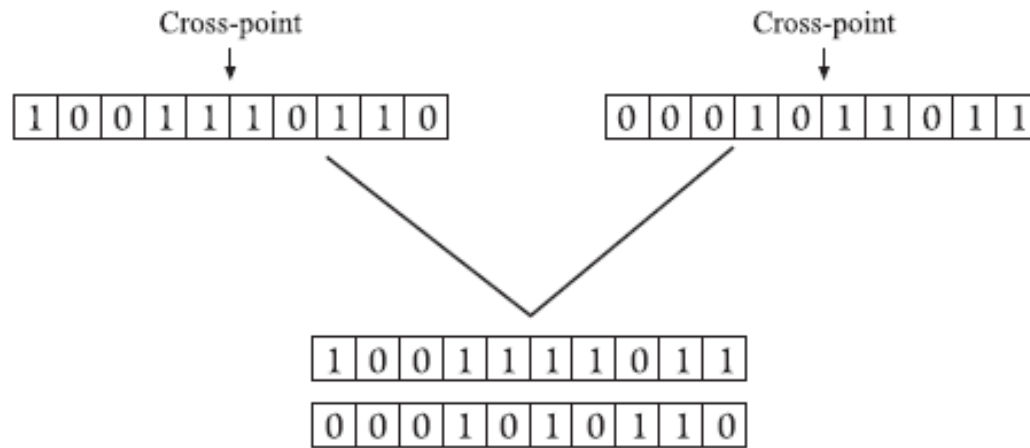
Ind	Apt	Torneo		
0	254	1		
1	47	2		
2	457	3		
3	194	1		
4	85	2		
5	1037	3		
	2074			

Representación

- La representación tradicional es la binaria
- A la cadena binaria se le llama "cromosoma"
- A cada cadena binaria que codifica una variable se le denomina "gene"
- Al valor dentro de esta posición se le llama "alelo"



Operadores: cruza de un punto



1. Se elige una posición en el cromosoma de forma aleatoria (cross-point)
2. El primer hijo tendrá los elementos hasta el cross-point del primer padre y del cross-point en adelante del segundo.
3. El segundo hijo tendrá los elementos hasta el cross-point del segundo padre y del cross-point en adelante del primero.

También existen cruza de dos puntos, tres, hasta de n puntos

Operadores: mutación

- Para cada alelo del cromosoma, generar un número aleatorio y si es menor a P_m cambiar el valor de 0 a 1 y de 1 a 0 según corresponda
- Normalmente el valor de P_m es muy bajo, se ha sugerido $P_m = \frac{1}{L}$ donde L es el tamaño del cromosoma

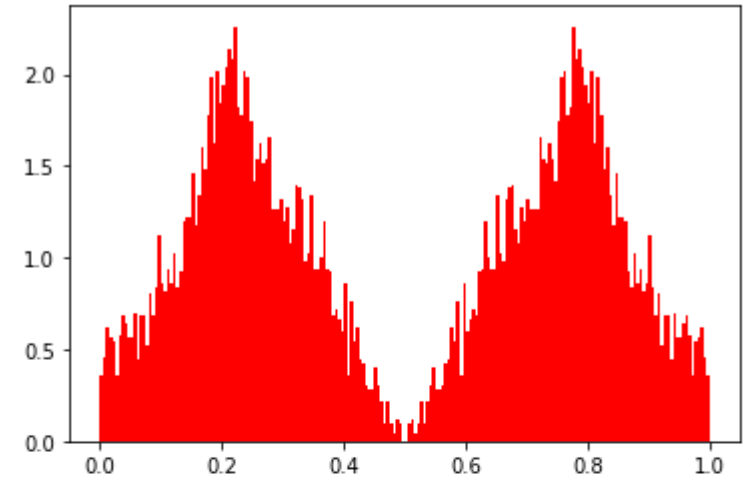
Simulated Binary Crossover (SBX)

- Propuesta por Deb y Agrawal en 1995
- Intenta emular el efecto de la cruce de un punto de
- Algoritmo
 - $u = X \sim U(0,1)$

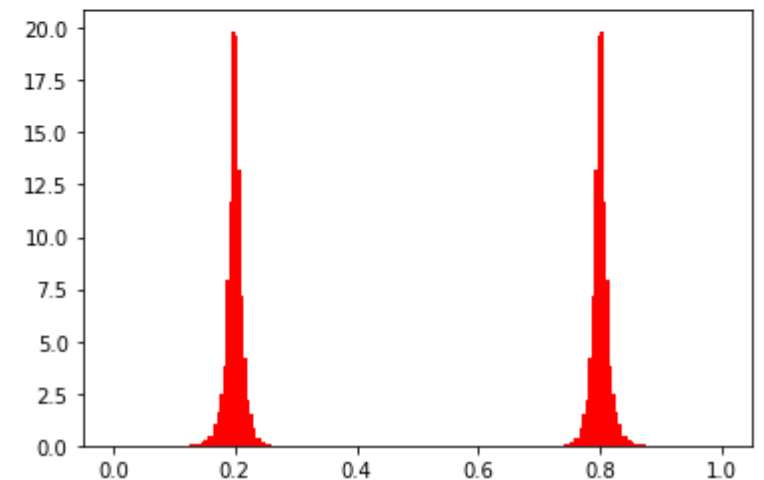
$$\beta = \begin{cases} (2u)^{\frac{1}{\eta_c+1}} & \text{si } u \leq 0.5 \\ \frac{1}{2(1-u)^{\frac{1}{\eta_c+1}}} & \text{de lo contrario} \end{cases}$$

$$h_1 = \frac{1}{2} (p_1 + p_2 - \beta |p_2 - p_1|)$$

$$h_2 = \frac{1}{2} (p_1 + p_2 + \beta |p_2 - p_1|)$$



[3]: show(30)



Parameter-Based Mutation

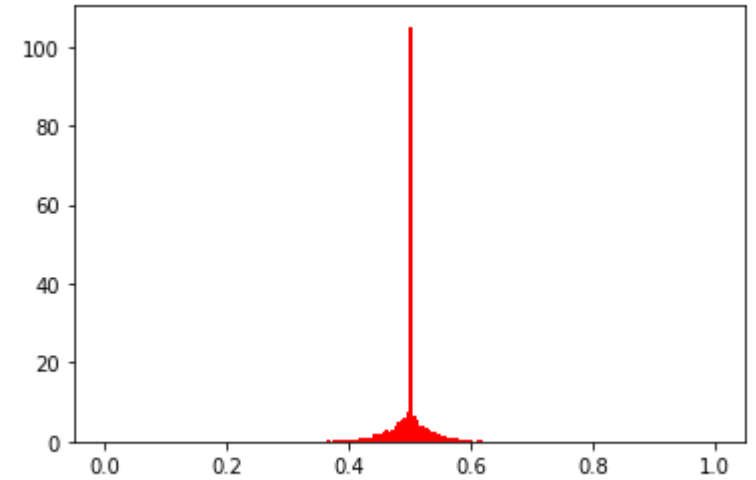
- Propuesto por Deb en 1995
- Algoritmo
 - Dado $p = \langle V_1, \dots, V_m \rangle$
 - Regresar $p' = \langle V_1, \dots, V'_k, \dots, V_m \rangle$
 - Donde

$$V'_k = V_k + \delta_q(ub_k - lb_k)$$

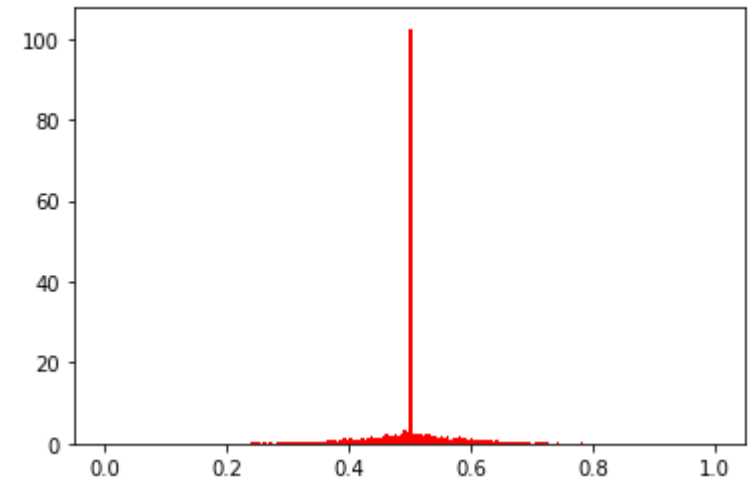
$$\delta_q = \begin{cases} [2u + (1 - 2u)(1 - \delta)^{\eta_m + 1}]^{\frac{1}{\eta_m + 1}} - 1 & \text{si } u \sim U(0,1) \leq 0.5 \\ 1 - [2(1 - u) + 2(u - 0.5)(1 - \delta)^{\eta_m + 1}]^{\frac{1}{\eta_m + 1}} & \text{de lo contrario} \end{cases}$$

$$\delta = \frac{\min[V_k - lb_k, ub - V_k]}{ub_k - lb_k}$$

$\eta_m \in \mathbb{N}$



[2]: show(10)




Estrategias evolutivas

- Ingo Rechenber, lidera el grupo que propone el algoritmo en 1964 para problemas hidrodinámicos
- La versión original (1+1)-EE solo usaba un padre y generaba un hijo
- El hijo se mantenía si era mejor que el padre
- El nuevo individuo era generado con
$$x^{t+1} = x^t + \mathcal{N}(0, \sigma)$$
- Rechenberg propone $(\mu + 1)$ -EE
- Schwefel propone el esquema $(\mu + \lambda)$ -EE y (μ, λ) -EE
- CMA-ES es uno de los métodos más utilizados para optimización

Regresemos al grid world

Policy i :

s_1	s_1	s_3		s_N
a_1	a_1	a_3	...	a_N



	a	b	c	d	e
1	0	2	1	-1	1
2	1	1	2	0	2
3	3	-5	4	3	1
4	1	-2	4	1	2
5	1	1	2	1	1

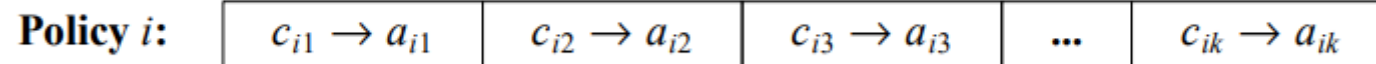
Algoritmo evolutivo simple para aprendizaje por refuerzo

- Representación
 - Un cromosoma por política con cada gen asociado con los estados observados y el valor correspondiente a una acción
- Cruza, mutación y selección
 - Como normalmente se realizan

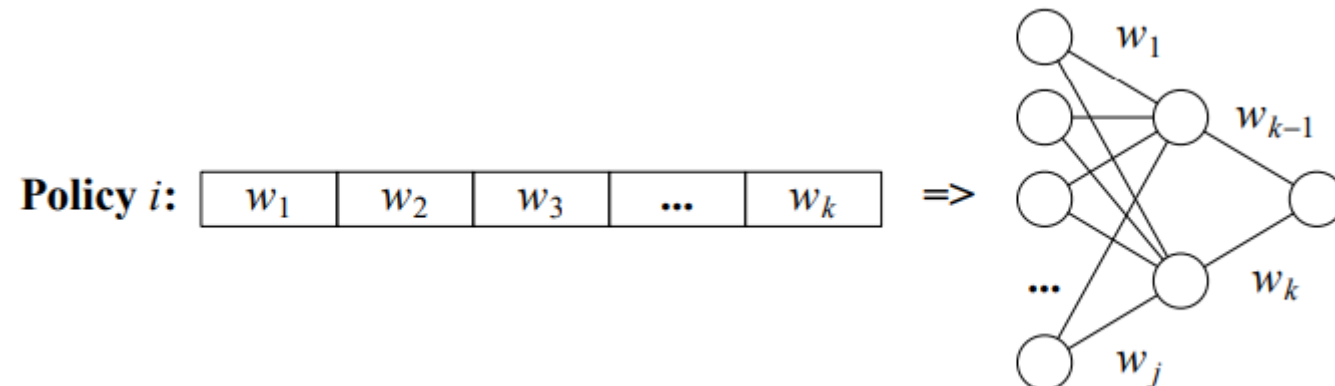
Policy	a1	a2	a3	a4	a5	b1	b2	b3	b4	b5	c1	c2	c3	c4	c5	d1	d2	d3	d4	d5	e1	e2	e3	e4	e5	Fitness
1	D	R	D	D	R	R	R	R	R	R	D	R	D	D	R	R	D	R	R	R	D	R	R	D	R	8
2	D	D	D	D	R	R	R	R	R	R	D	D	R	R	D	R	D	R	R	R	D	R	D	D	R	9
3	R	D	D	R	R	D	R	D	R	R	D	D	D	R	D	R	D	R	R	R	D	R	D	D	D	17
4	D	D	D	D	R	D	R	R	R	R	R	D	R	R	R	D	R	R	D	R	D	R	D	D	R	11
5	R	D	D	D	R	D	R	R	D	R	R	D	R	R	D	R	D	R	R	D	D	R	D	D	D	16

Representación de un cromosoma

- Estado-acción

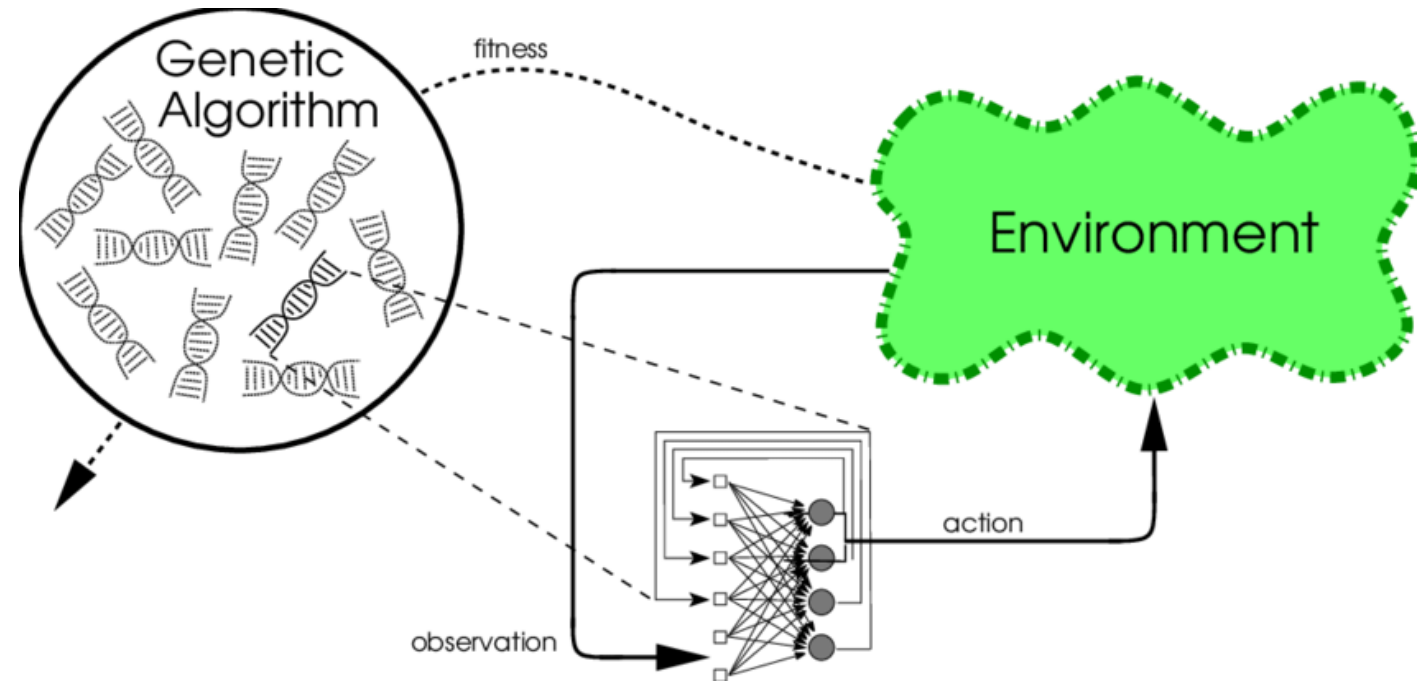


- Redes neuronales



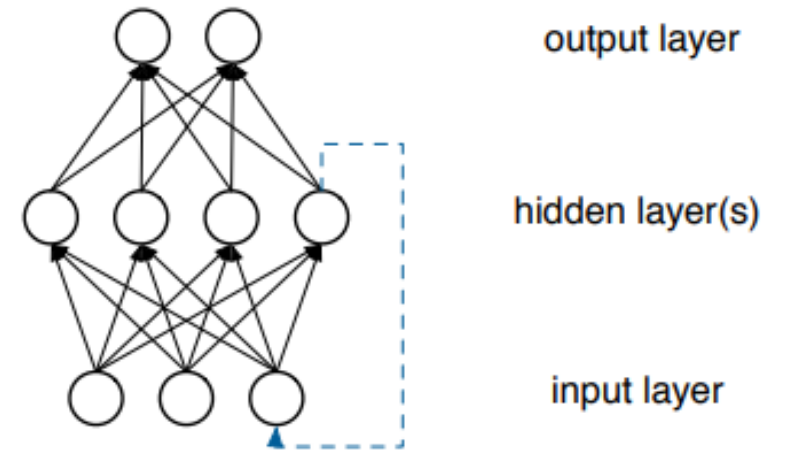
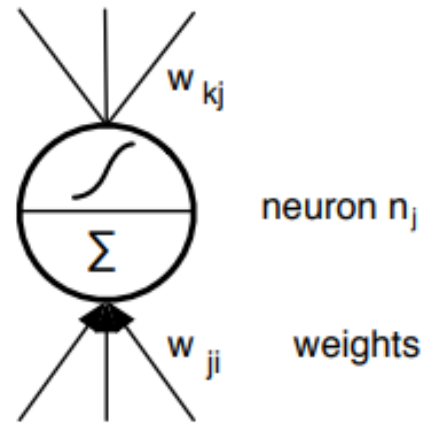
Neuroevolución

- La idea es evolucionar redes neuronales (recurrentes) junto con sus pesos
- Área iniciada en los 90s



Representación

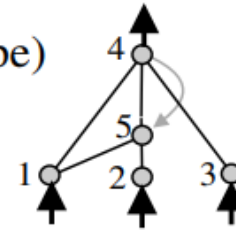
- Redes neuronales (recurrentes)
- Existen dos variantes con respecto a que puede evolucionar:
 - Estructura definida, pesos variables
 - Estructura y pesos variables



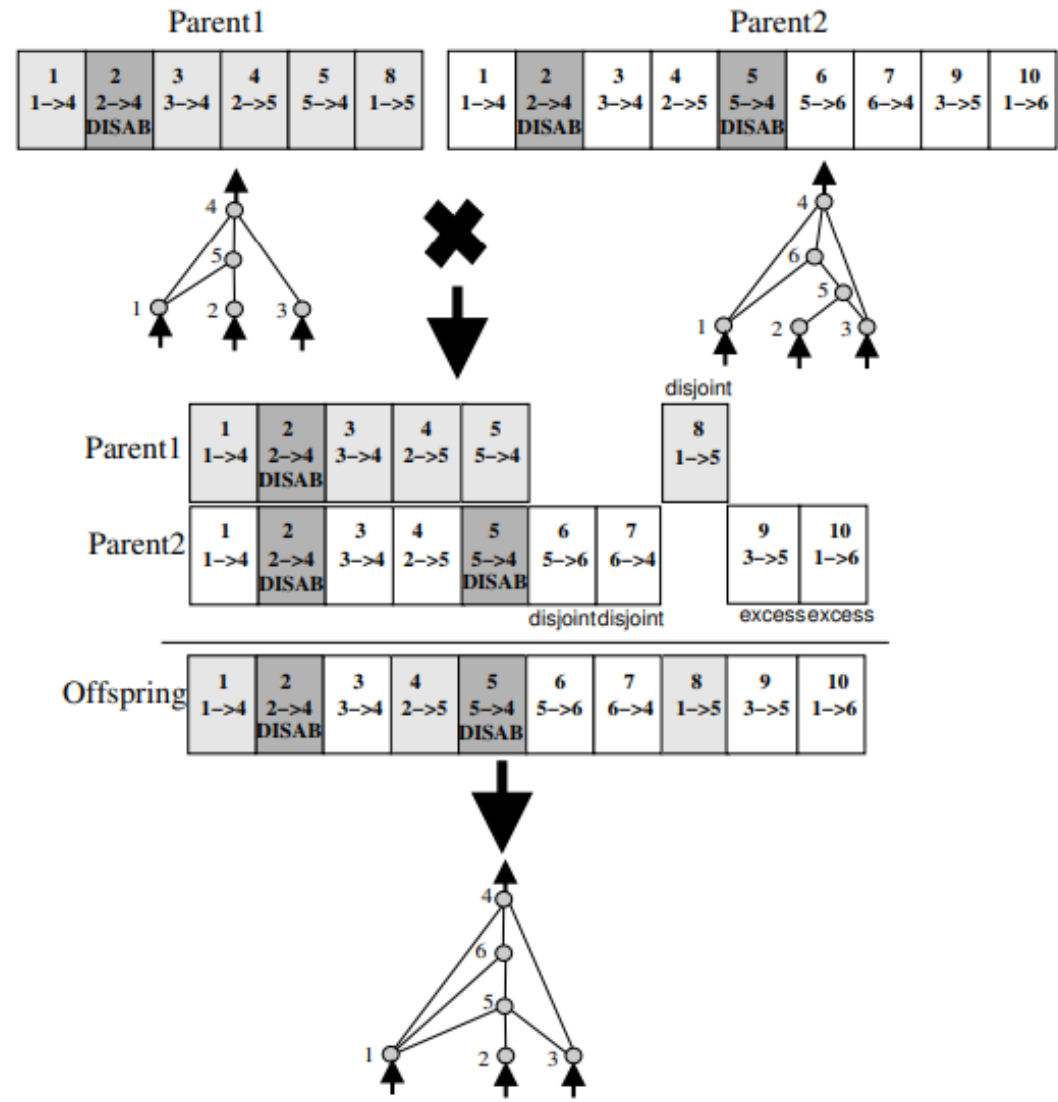
Un ejemplo

Genome (Genotype)							
Node Genes	Node Genes					Connect. Genes	
	Node 1	Node 2	Node 3	Node 4	Node 5		
Connect. Genes	Sensor	Sensor	Sensor	Output	Hidden		
	In 1	In 2	In 3	In 2	In 5	In 1	In 4
	Out 4	Out 4	Out 4	Out 5	Out 4	Out 5	Out 5
	Weight 0.7	Weight-0.5	Weight 0.5	Weight 0.2	Weight 0.4	Weight 0.6	Weight 0.6
	Enabled	DISABLED	Enabled	Enabled	Enabled	Enabled	Enabled
	Innov 1	Innov 2	Innov 3	Innov 4	Innov 5	Innov 6	Innov 11

Network (Phenotype)

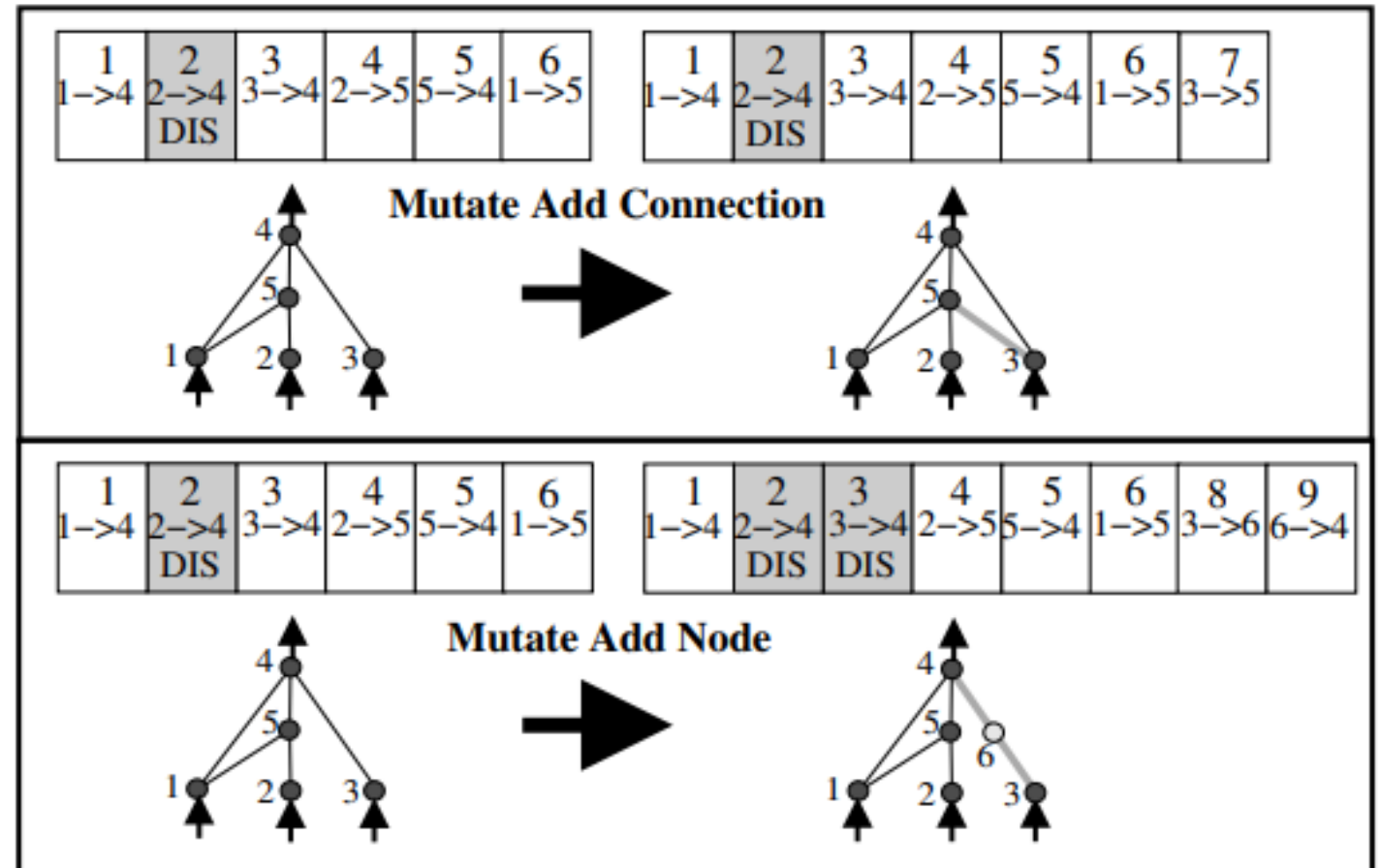


Cruza



Mutación

- Cambios aleatorios en los pesos
- Agregar una conexión entre neuronales aleatorias
- Agregar una neurona aleatoria



NEAT: NeuroEvolution of Augmented Topologies

- Evolucionar pesos y estructura
- Se inicia con redes completamente conectadas con solo capa de entrada y capa de salida
- Utiliza: subespecies y compartición de aptitud
- Existen otros enfoques como: SANE, ESP de Risto Miikkulainen; ENS³ de Frank Pasemann o EANT(2) de Siebel y Sommer

Natural Evolutionary Strategies (NES)

Algorithm 3: Canonical Natural Evolution Strategies

input: f, θ_{init}

repeat

for $k = 1 \dots \lambda$ **do**

 draw sample $\mathbf{z}_k \sim \pi(\cdot|\theta)$

 evaluate the fitness $f(\mathbf{z}_k)$

 calculate log-derivatives $\nabla_{\theta} \log \pi(\mathbf{z}_k|\theta)$

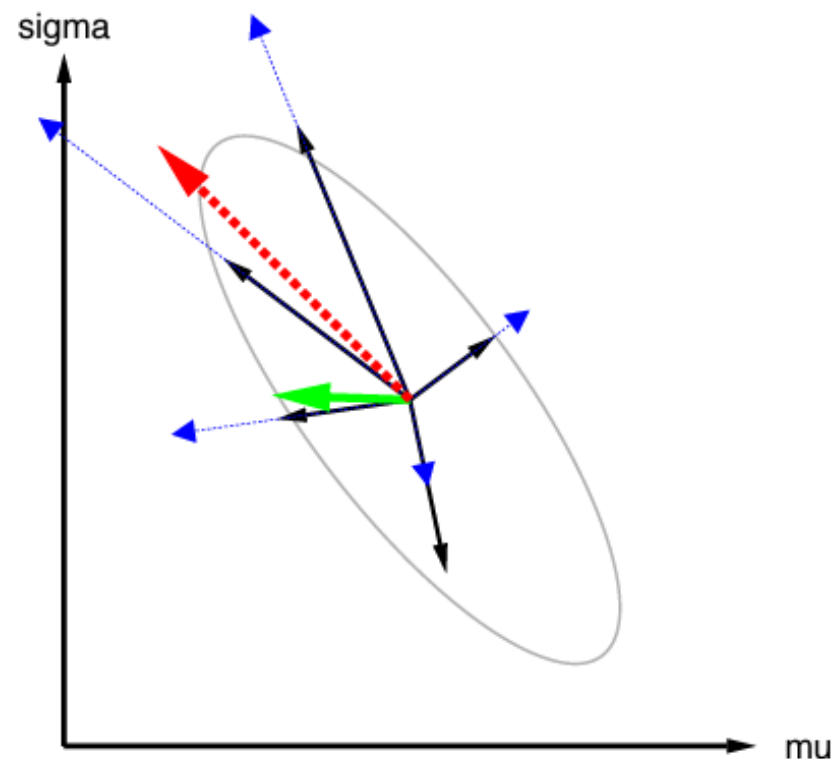
end

$$\nabla_{\theta} J \leftarrow \frac{1}{\lambda} \sum_{k=1}^{\lambda} \nabla_{\theta} \log \pi(\mathbf{z}_k|\theta) \cdot f(\mathbf{z}_k)$$

$$\mathbf{F} \leftarrow \frac{1}{\lambda} \sum_{k=1}^{\lambda} \nabla_{\theta} \log \pi(\mathbf{z}_k|\theta) \nabla_{\theta} \log \pi(\mathbf{z}_k|\theta)^{\top}$$

$$\theta \leftarrow \theta + \eta \cdot \mathbf{F}^{-1} \nabla_{\theta} J$$

until *stopping criterion is met*

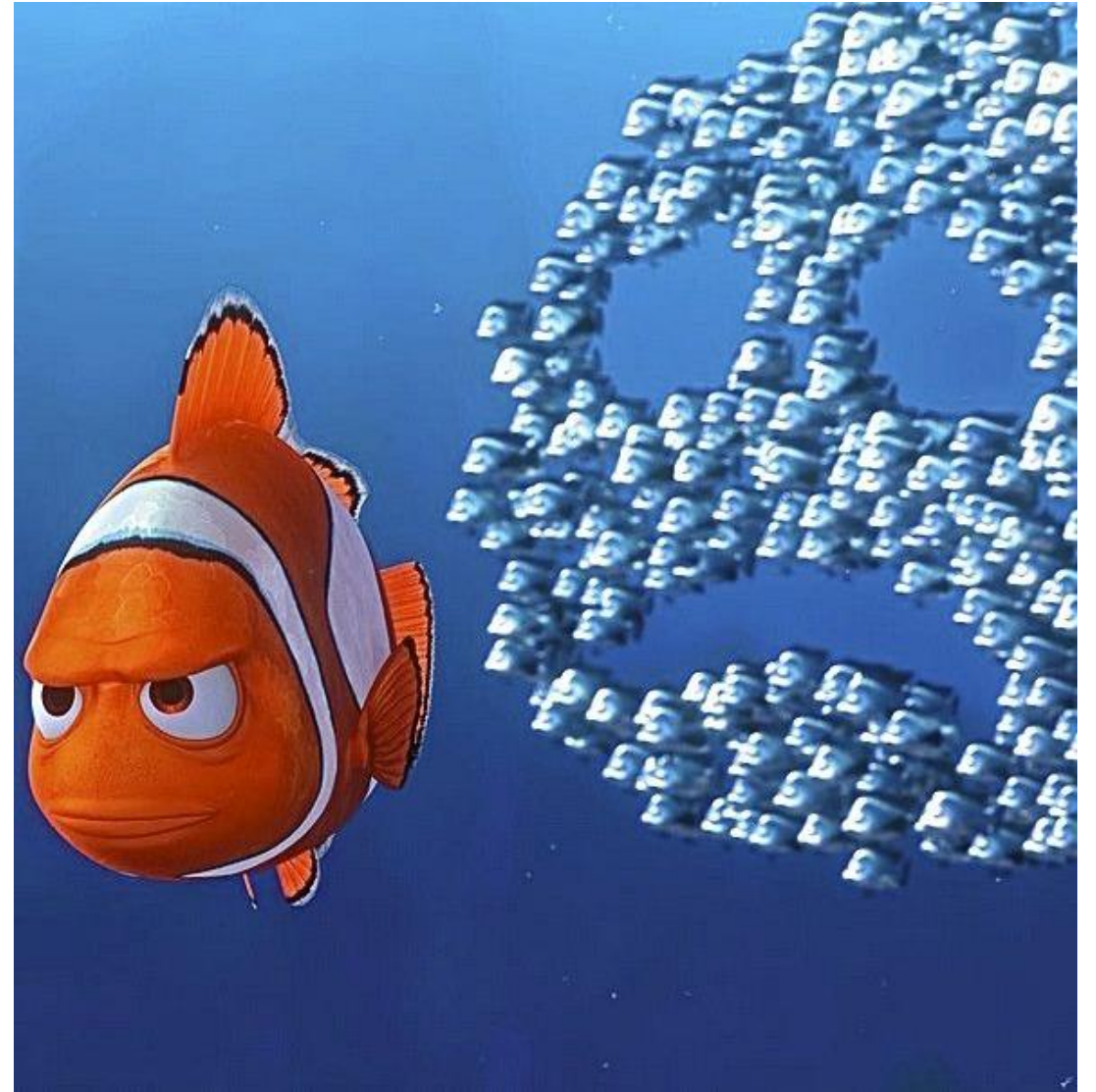


Ventajas de las estrategias evolutivas

- No necesita backpropagation
- Altamente paralelizable
- Robusto
- Exploración estructurada
- Efectivo para asignación de crédito en largos tiempos

Otras metaheurísticas

- Bio inspirados
 - Cumulo de partículas
 - Colonia de hormigas
 - Colonia de abejas
 - Búsqueda Cuckoo
 - Algoritmos genéticos
- Programación matemática
 - Evolución diferencial
 - Algoritmos de estimación de distribución



Para la otra vez...

- Otros temas avanzados

The End.