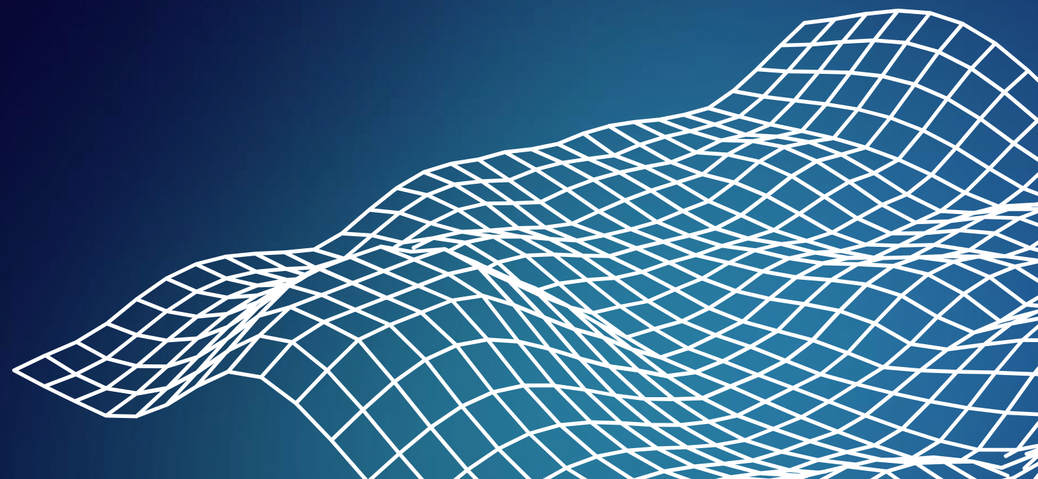
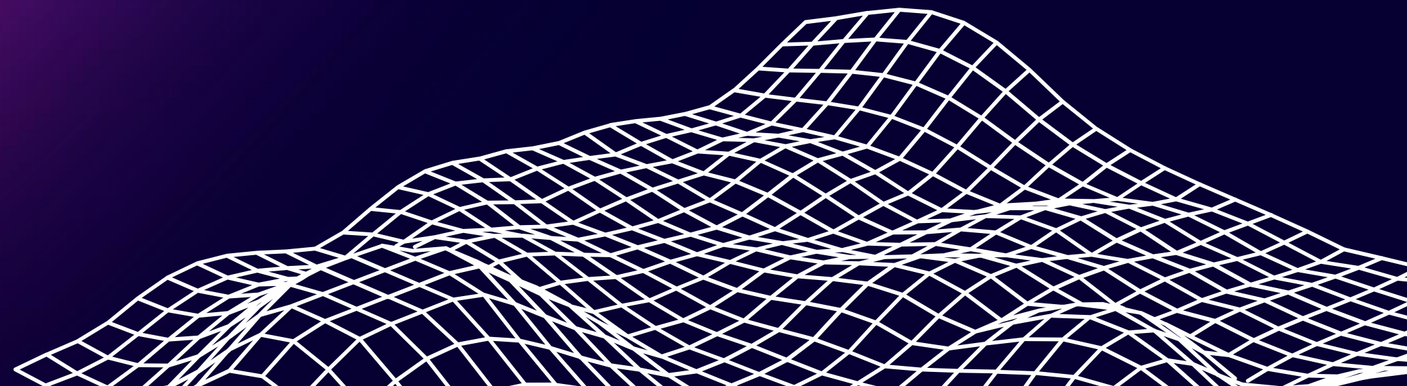
FakeNEAT:
Neuroevolución

mediante AG y Enfriamiento Simulado

HUGO ALBERT BONET



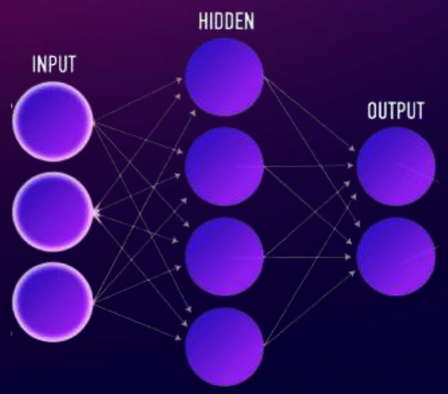
1 DESCRIPCIÓN DEL PROBLEMA



Entrenamiento de Redes Neuronales

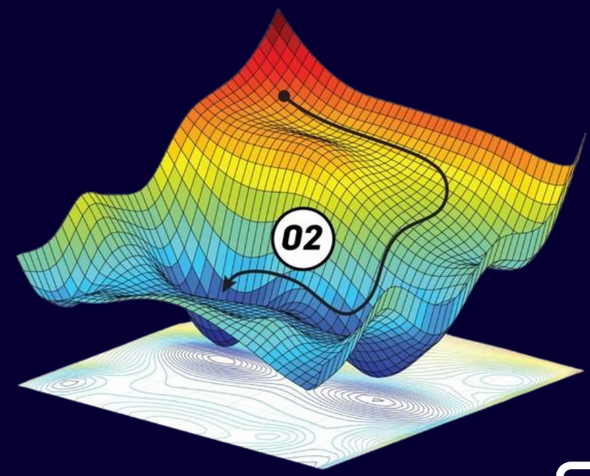
Búsqueda de arquitecturas

¿N° capas?



¿N° neuronas?

Entrenamiento de la red



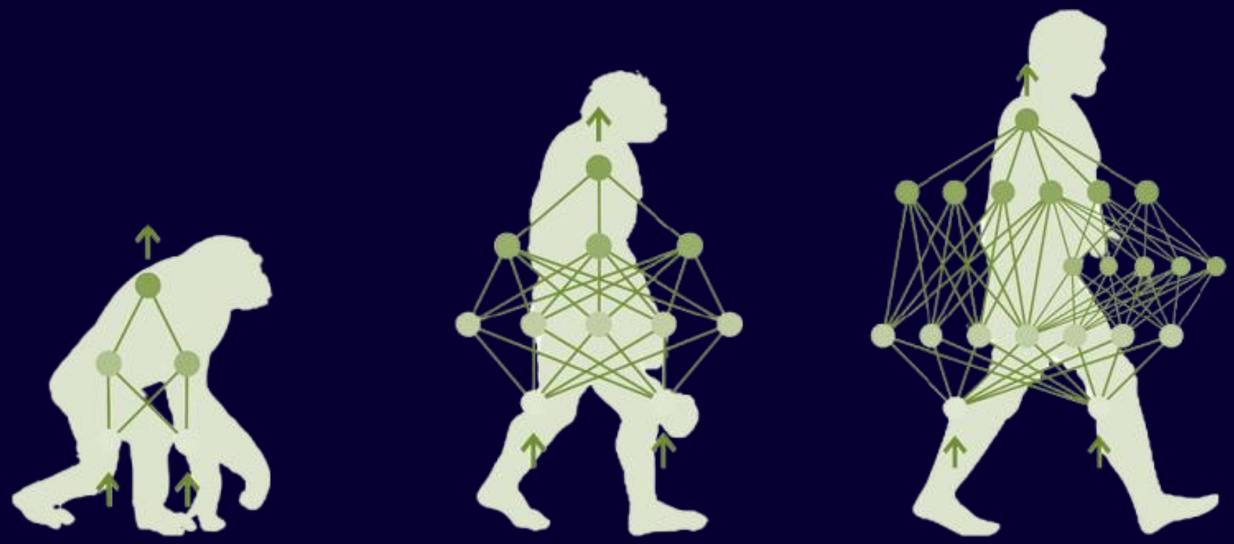
Neuroevolución

¿Qué es?

Algoritmos evolutivos para entrenar redes neuronales buscando al mismo tiempo la mejor arquitectura

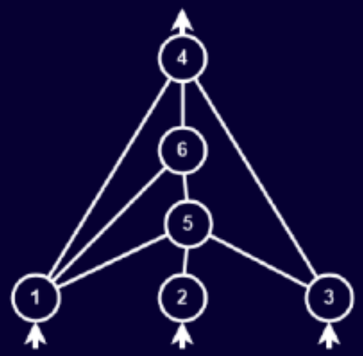
Estado del arte:

- NeuroEvolution of Augmented Topologies (NEAT)
- HyperNEAT



NEAT vs FakeNEAT

NEAT



Principal problema: Poco óptimo para inferencia

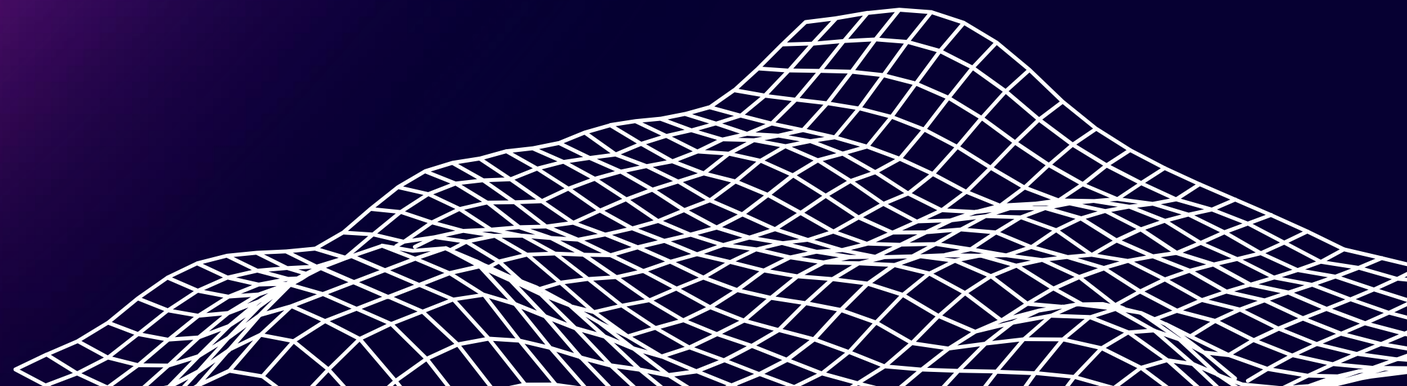
FakeNEAT

Centrado en optimizar:
- Neuronas de cada capa
- Pesos de la red

Limitaciones:
- Número de capas
- Funciones de activación
- Capas especiales (Convolucionales, Batch Normalization...)

2

ALGORITMO GENÉTICO



Representación de la solución

Solución = Arquitectura + Pesos

Aquitectura:

$[N_1, N_2, \dots, N_H]$ → N_h: Número de neuronas en la capa h.
H: Número de capas de la red (hiperparámetro).

Pesos:

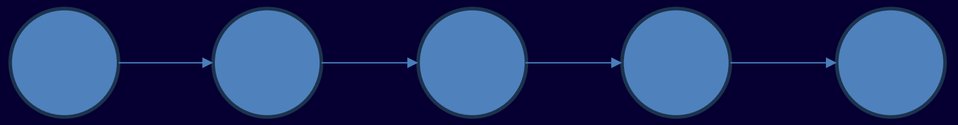
Implementados sobre la librería PyTorch

Población inicial

Búsqueda de simplicidad

Una neurona por capa:

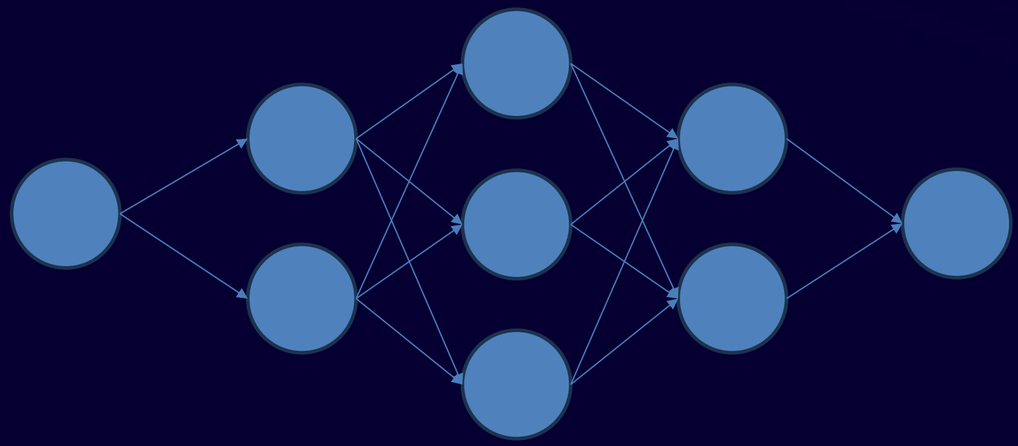
- Parte de la solución más simple y eficiente computacionalmente
- Genera redes “grandes” si produce un beneficio



Búsqueda de variabilidad

Número aleatorio de neuronas por capa:

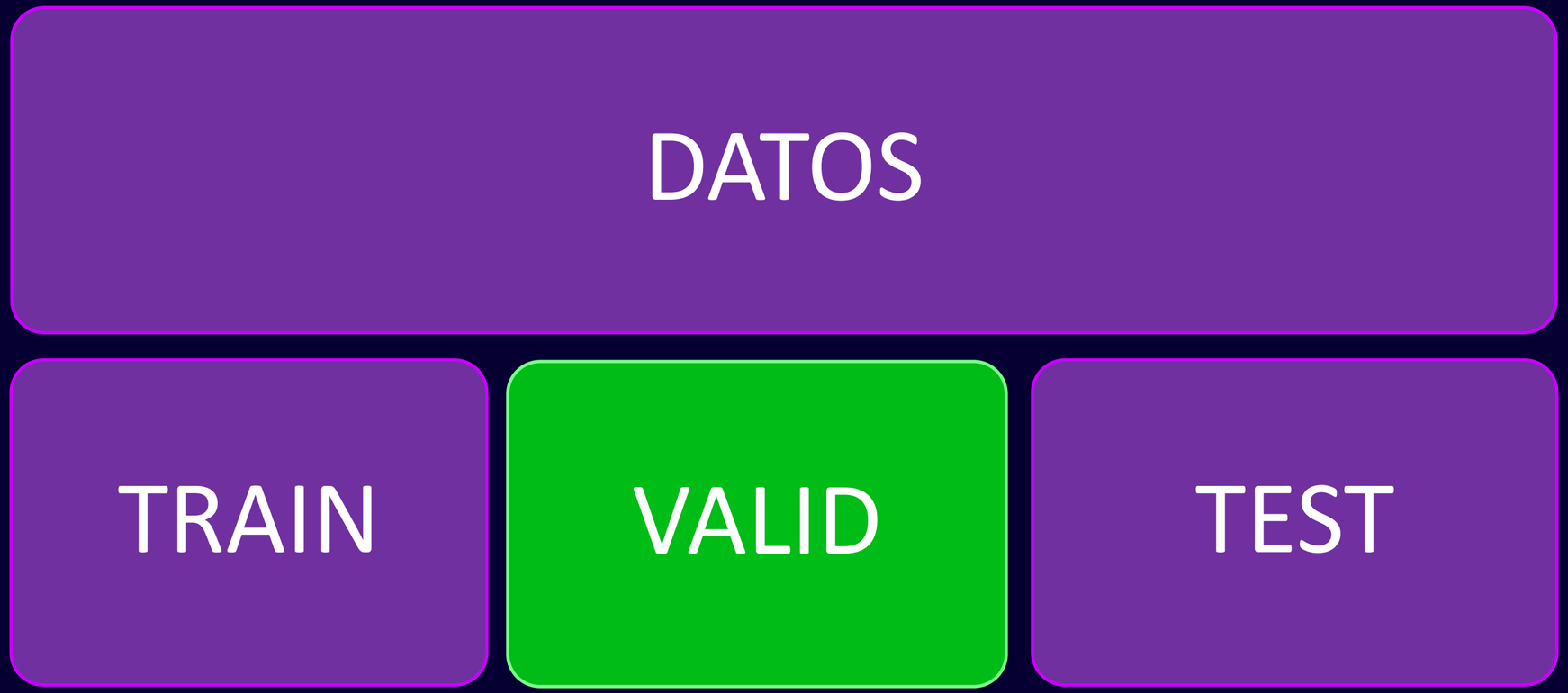
- Se establece un máximo de neuronas iniciales por capa
- Incrementa la variabilidad inicial (Exploración)



Función fitness

DATOS

Función fitness



Función fitness

Clasificación

Fitness:

$$Fitness = 1 - \frac{True\ Positives}{N} = 1 - Accuracy$$

Función de pérdidas:

$$L_1(\hat{y}) = 1 - Accuracy$$

$$L_2(\hat{y}) = CrossEntropy(\hat{y})$$

Regresión

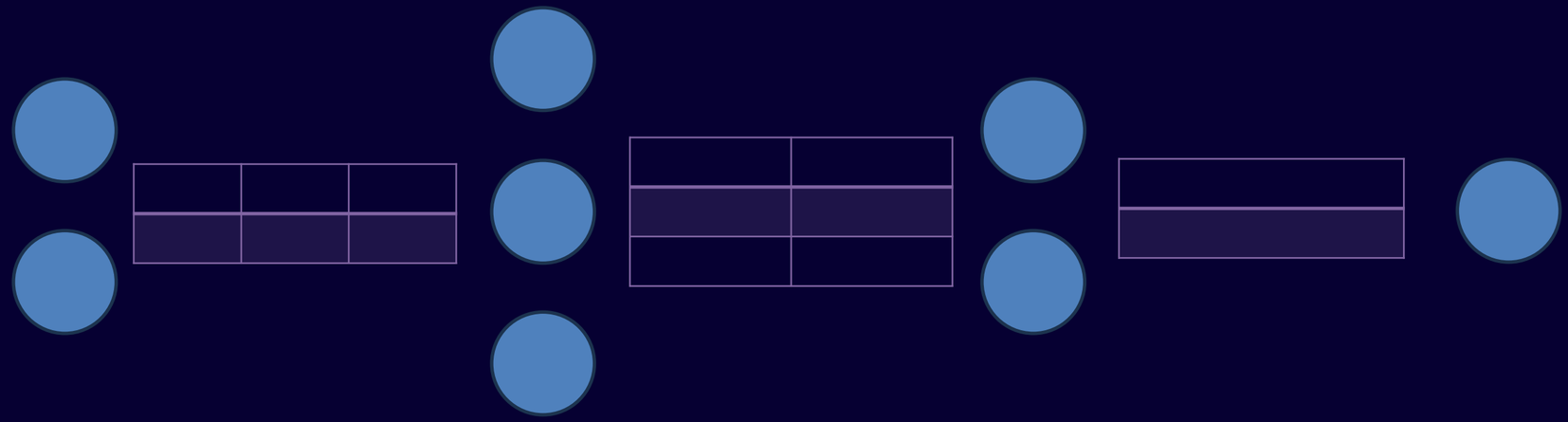
Fitness:

$$Fitness = L(\hat{y}) = MSE(y, \hat{y}) = (y - \hat{y})^2$$

Mutaciones

Eliminar neuronas:

Elimina un número aleatorio de neuronas a una capa aleatoria (dejando siempre al menos una neurona restante en la capa).



Añadir neuronas:

Añade un número aleatorio de neuronas (en un rango establecido) a una capa aleatoria.

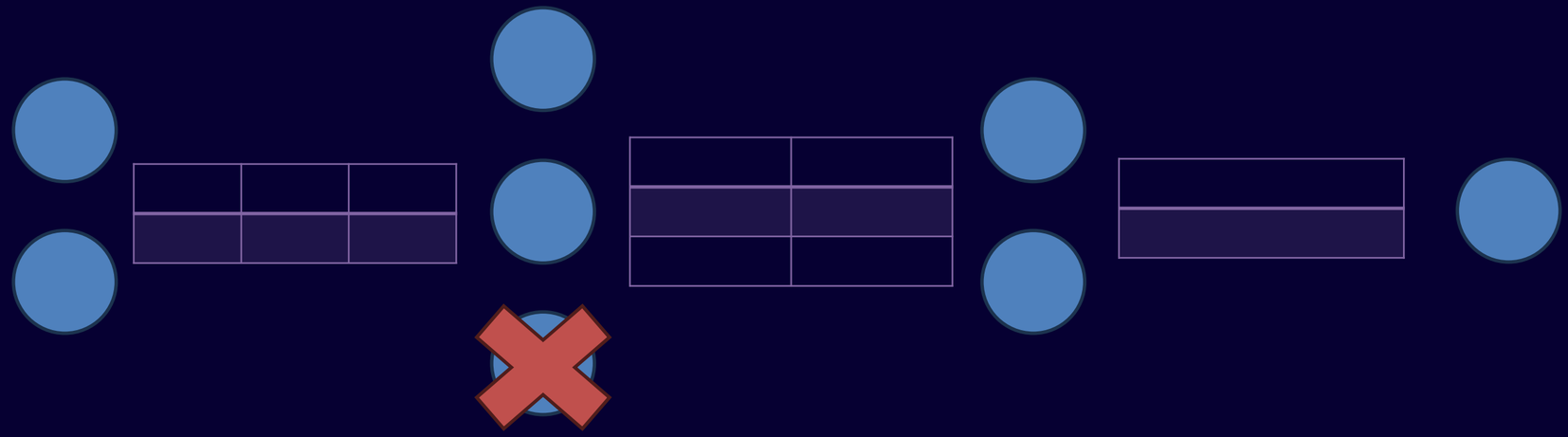
Mini-train:

Ejecuta una iteración del algoritmo de backpropagation en un minibatch.

Mutaciones

Eliminar neuronas:

Elimina un número aleatorio de neuronas a una capa aleatoria (dejando siempre al menos una neurona restante en la capa).



Añadir neuronas:

Añade un número aleatorio de neuronas (en un rango establecido) a una capa aleatoria.

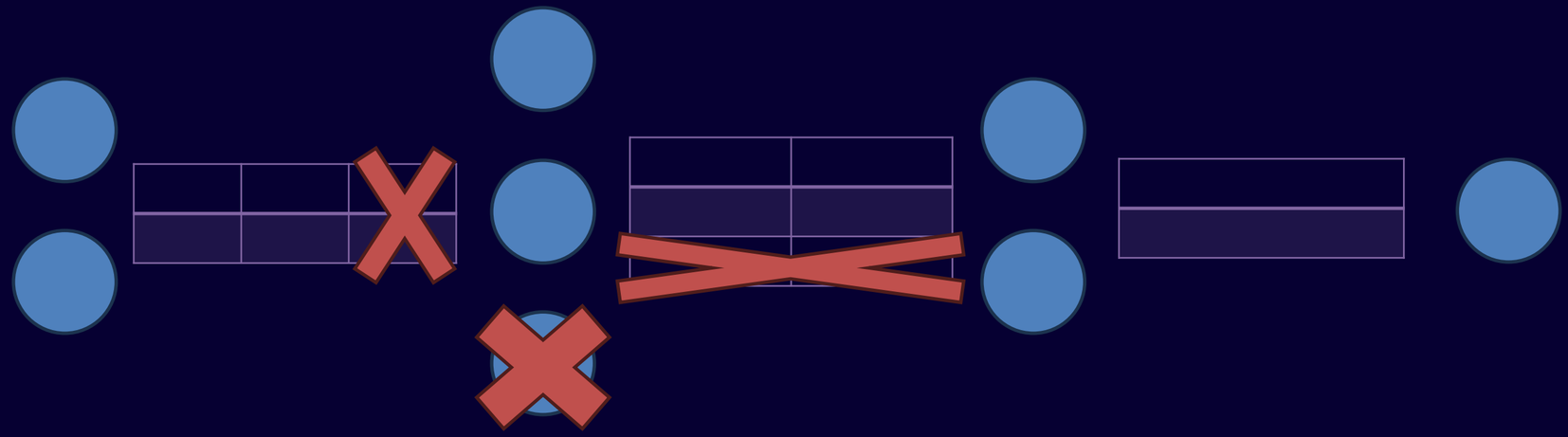
Mini-train:

Ejecuta una iteración del algoritmo de backpropagation en un minibatch.

Mutaciones

Eliminar neuronas:

Elimina un número aleatorio de neuronas a una capa aleatoria (dejando siempre al menos una neurona restante en la capa).



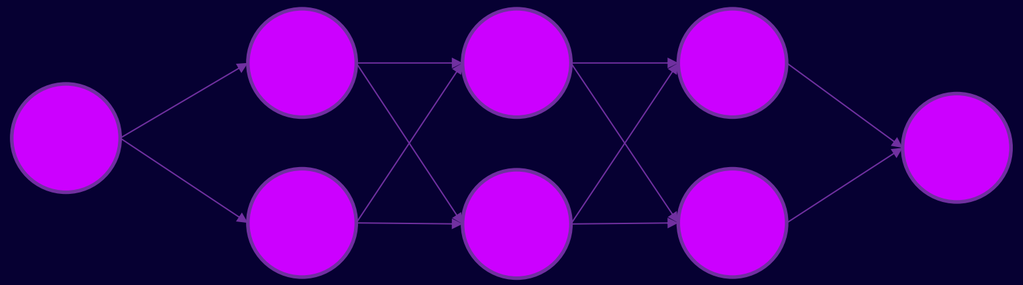
Añadir neuronas:

Añade un número aleatorio de neuronas (en un rango establecido) a una capa aleatoria.

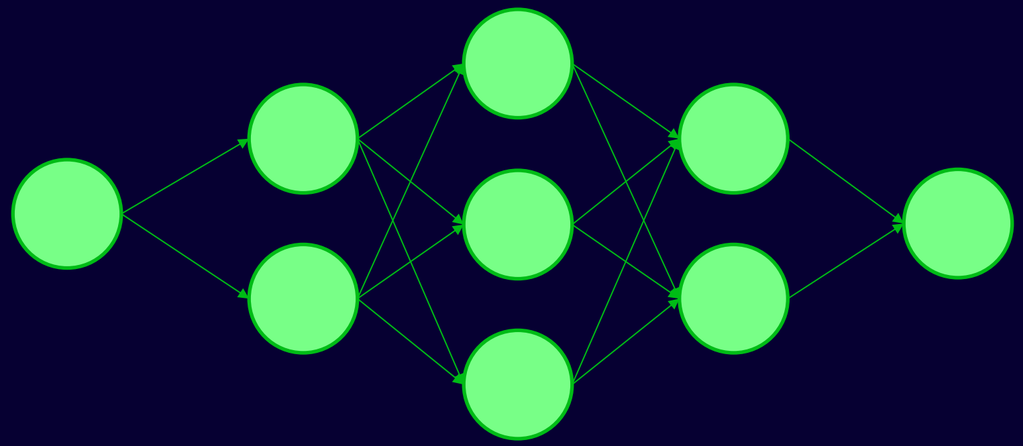
Mini-train:

Ejecuta una iteración del algoritmo de backpropagation en un minibatch.

Crossover



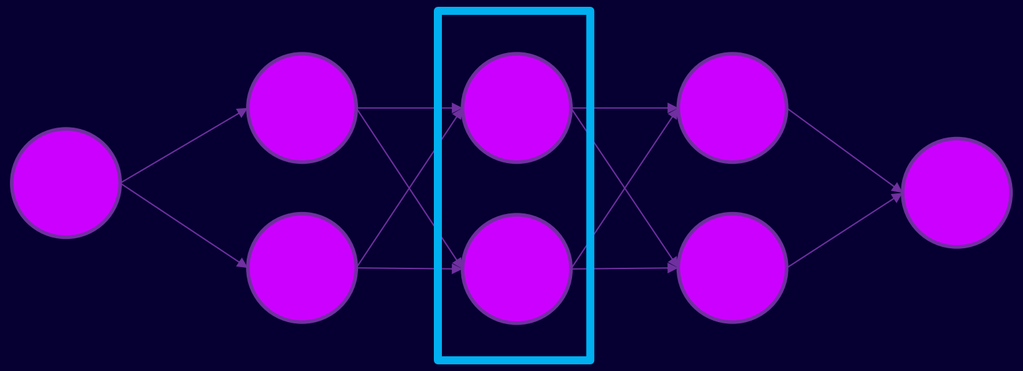
PARENT 1



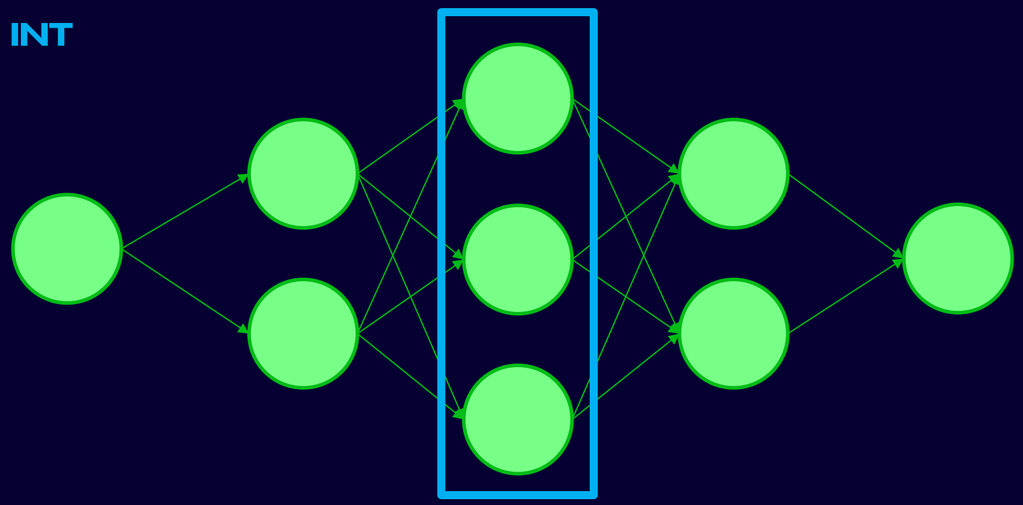
PARENT 2

Crossover

IDX = RANDOM INT

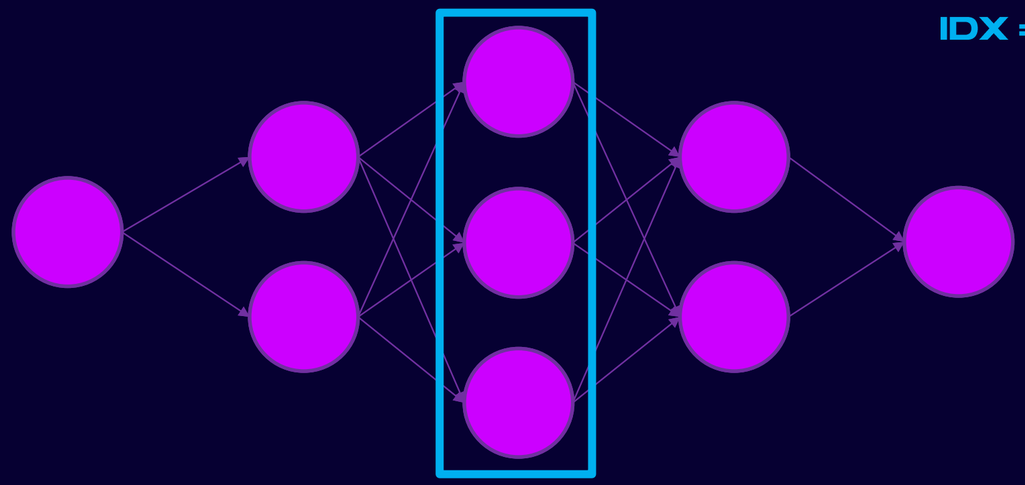


PARENT 1



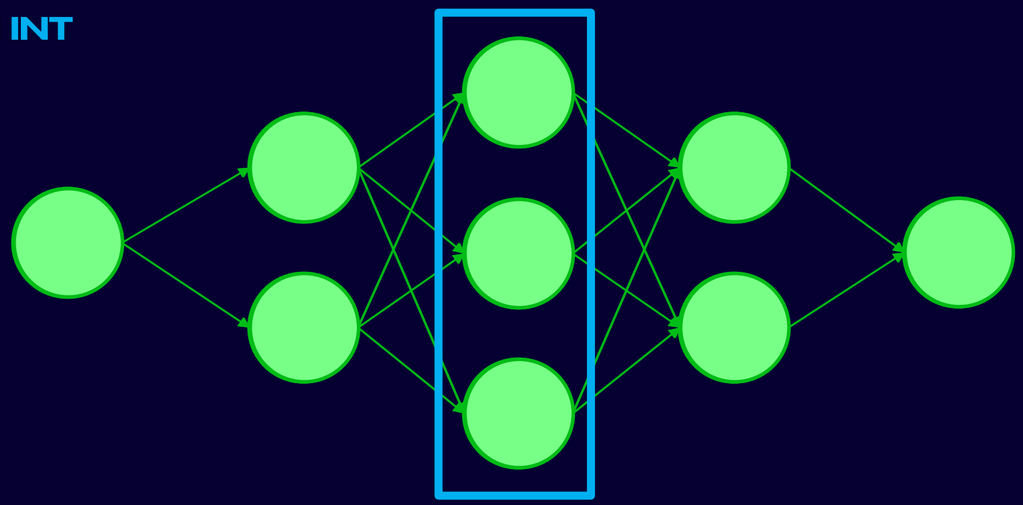
PARENT 2

Crossover



PARENT 1

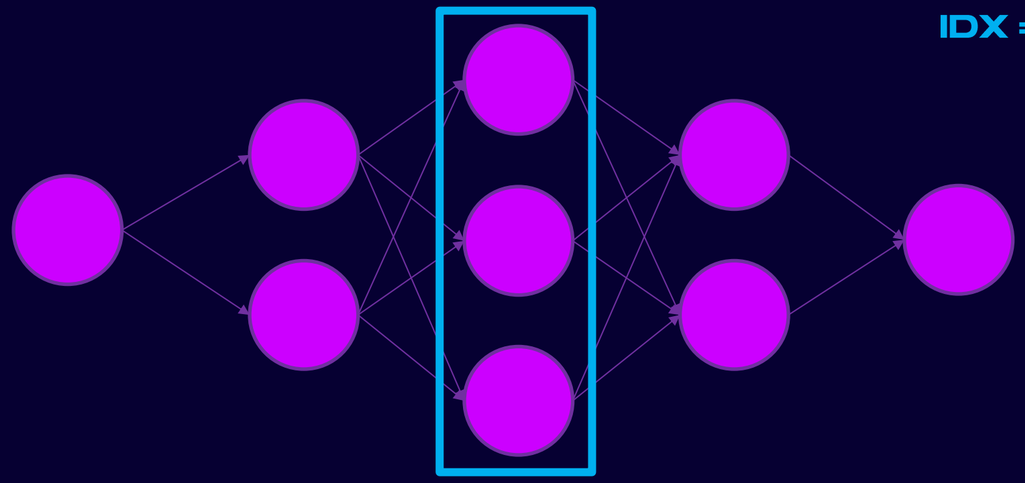
IDX = RANDOM INT



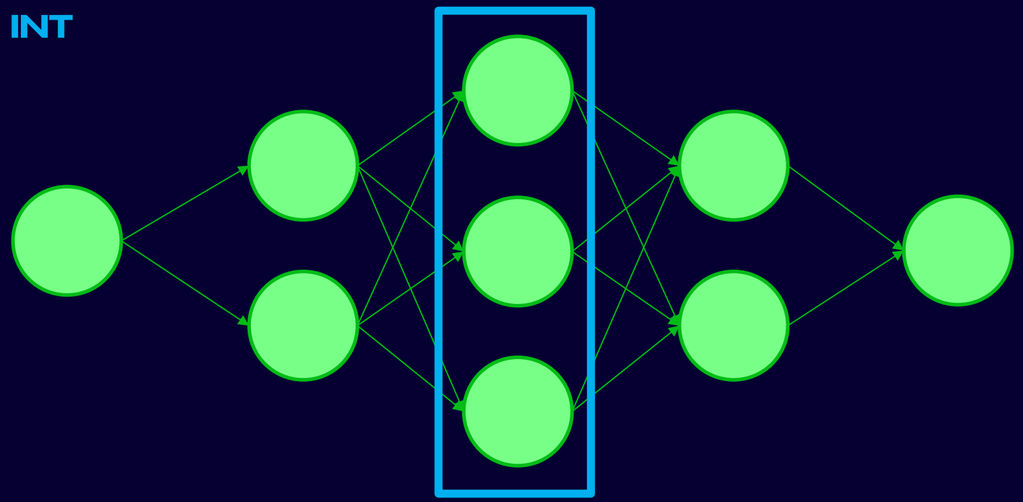
PARENT 2

Crossover

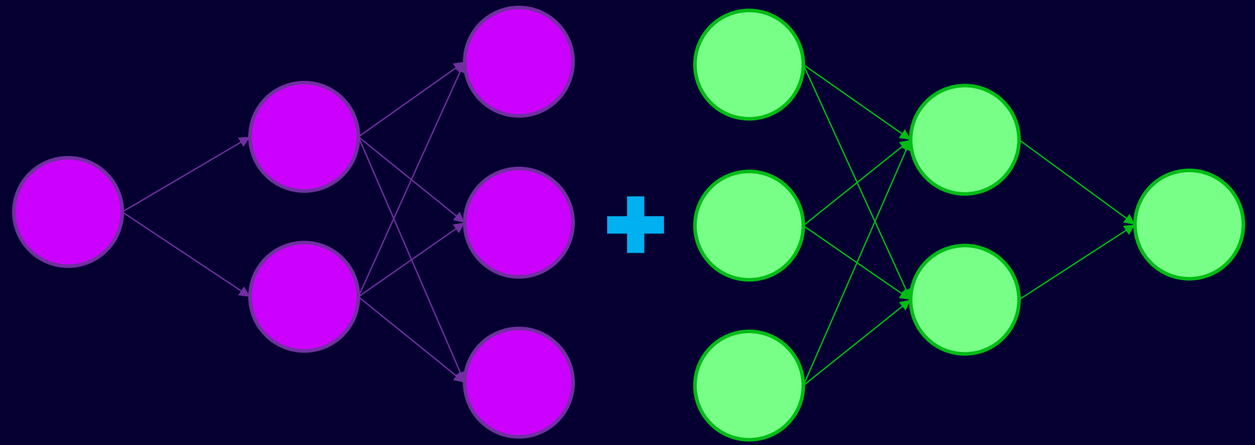
IDX = RANDOM INT



PARENT 1



PARENT 2



Selección y reemplazo

Selección

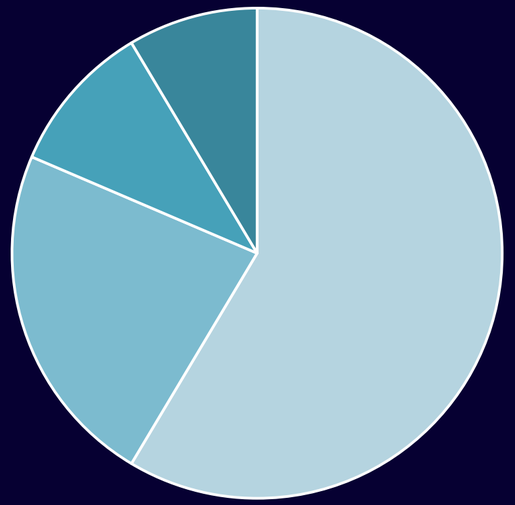
Proporcional al Accuracy:

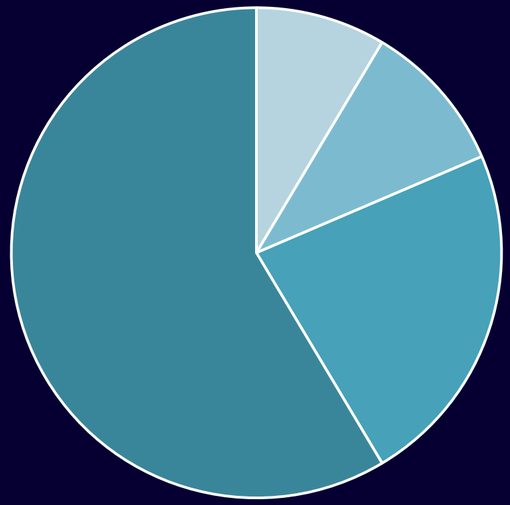
$$P(i) = \frac{Accuracy_i}{\sum_{j=1}^{N} Accuracy_j}$$

Reemplazo

Proporcional al fitness:

$$P(i) = \frac{f_i - \min\_fitness}{\sum_{j=1}^{N}(fj\ \_\ \min\_fitness)}$$





Experimentos

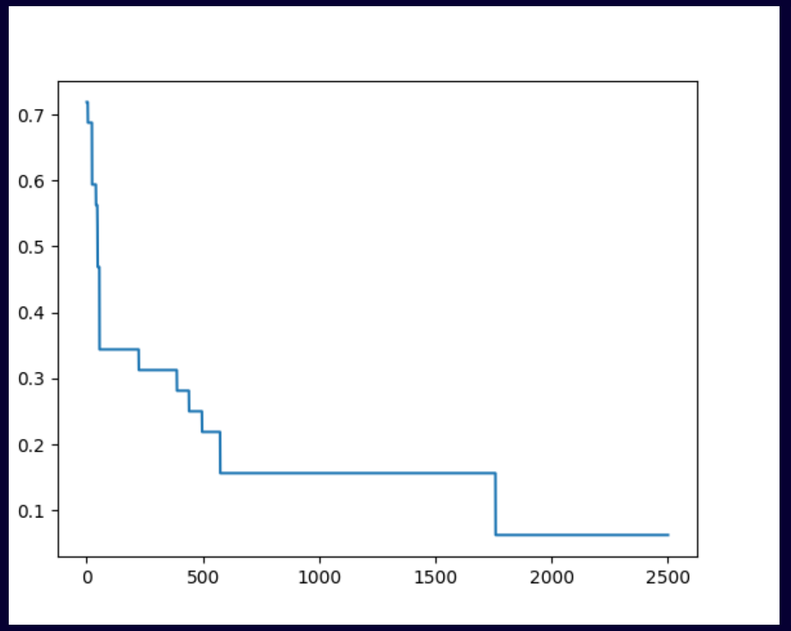
Resultados generales (Dataset Iris)

Población inicial	Capas ocultas	Max. Nº Neuronas	Tamaño Población	Ratio de Mutación	Crossovers por generación	Max. Iteraciones	Tiempo de ejecución (s)	Fitness (1-Accuracy)	Fitness test
Simple	5	500	100	0.1	1	2000	45	0.36	0.37
Variable	5	500	100	0.1	1	2000	45	0.03	0.0
Variable	5	500	200	0.1	1	2000	404	0.16	0.13
Variable	5	500	100	0.2	1	2000	479	0.33	0.40
Variable	5	1000	100	0.1	1	2000	330	0.18	0.13
Variable	10	500	100	0.1	1	2000	75	0.02	0.0

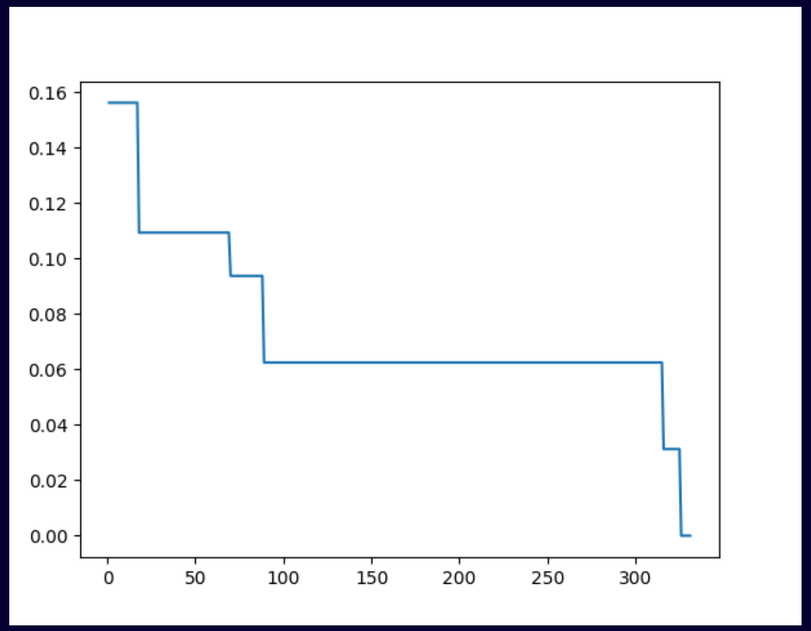
Experimentos

Otros datasets

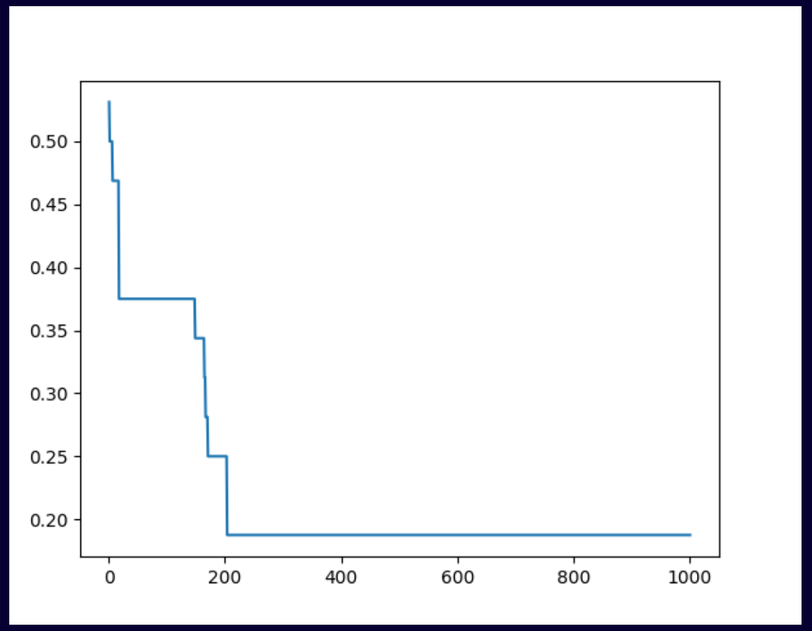
Digits



Breast Cancer

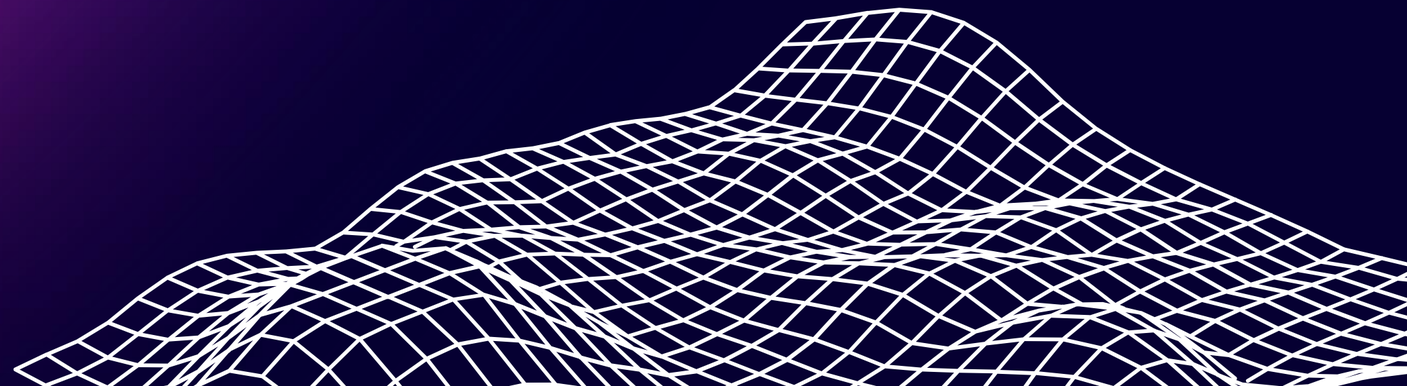


Wine



Generación

3 ENFRIAMIENTO SIMULADO



Diseño

Solución inicial:

Solución *temprana* del AG.

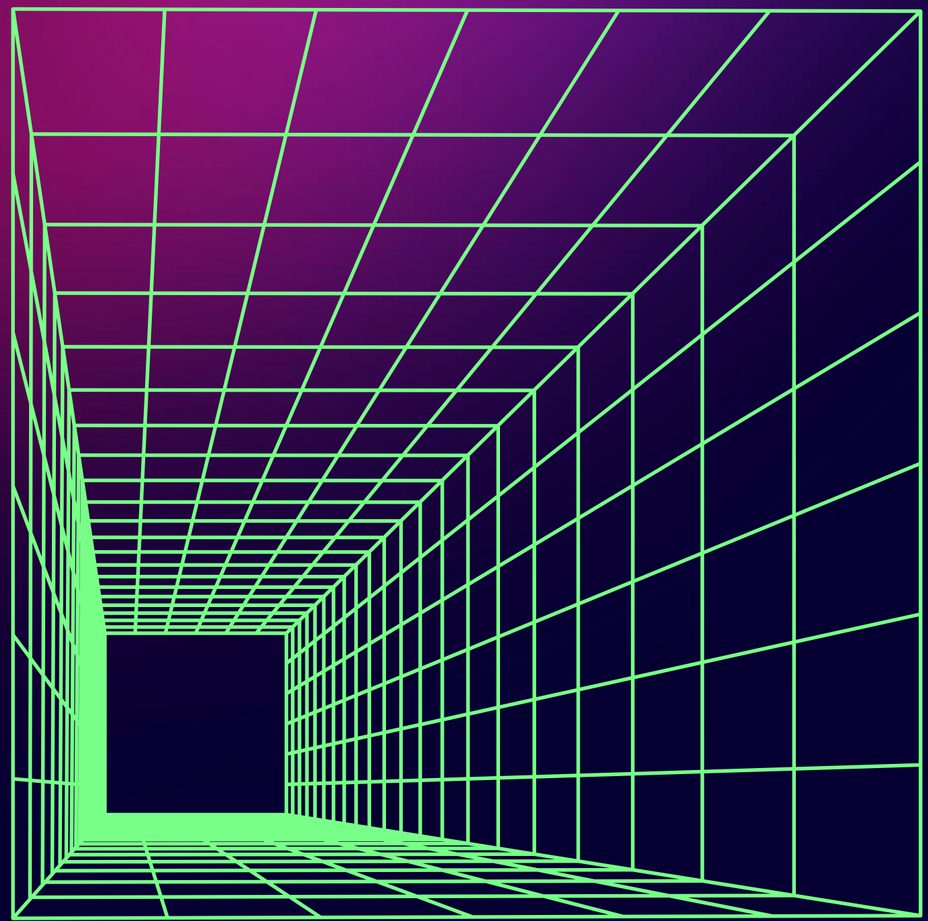
Generación del vecindario:

Una mutación del AG aleatoria.

- Añadir neuronas.
- Eliminar neuronas.
- Mini-train.

Temperatura:

Reducción lineal.



Experimentos

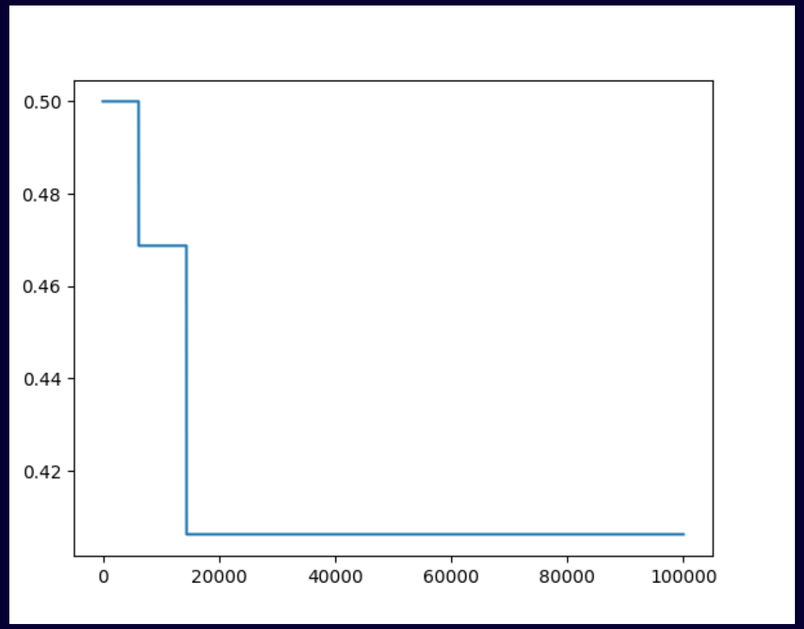
Resultados generales (Dataset Iris)

Min. Delta	Tamaño vecindario	Temperatura inicial	Temperatura final	Max. Iteraciones	Tiempo de ejecución (s)	Fitness (1-Accuracy)	Fitness test
0.01	50	0.01	0	3000	309	0.20	0.30
0.01	50	0.1	0	3000	235	0.64	0.63
0.01	50	0.05	0	3000	317	0.36	0.36
0.01	100	0.01	0	3000	688	0.20	0.30
0.01	20	0.01	0	3000	133	0.18	0.13
0.1	20	0.01	0	3000	133	0.33	0.40

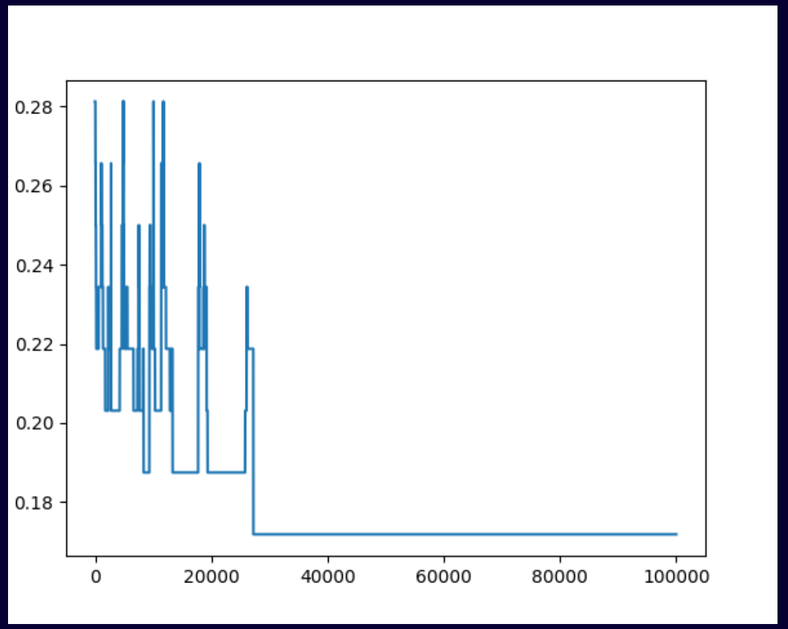
Experimentos

Otros datasets

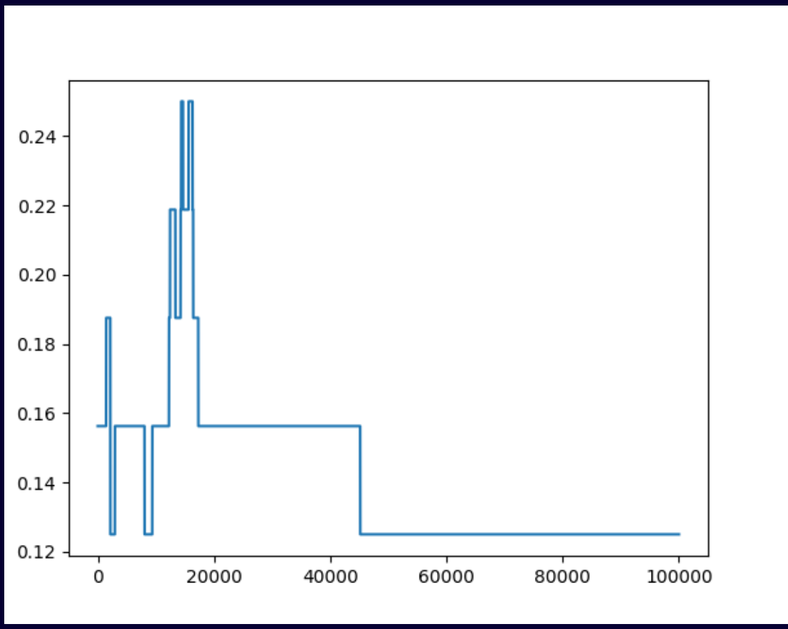
Digits



Breast Cancer



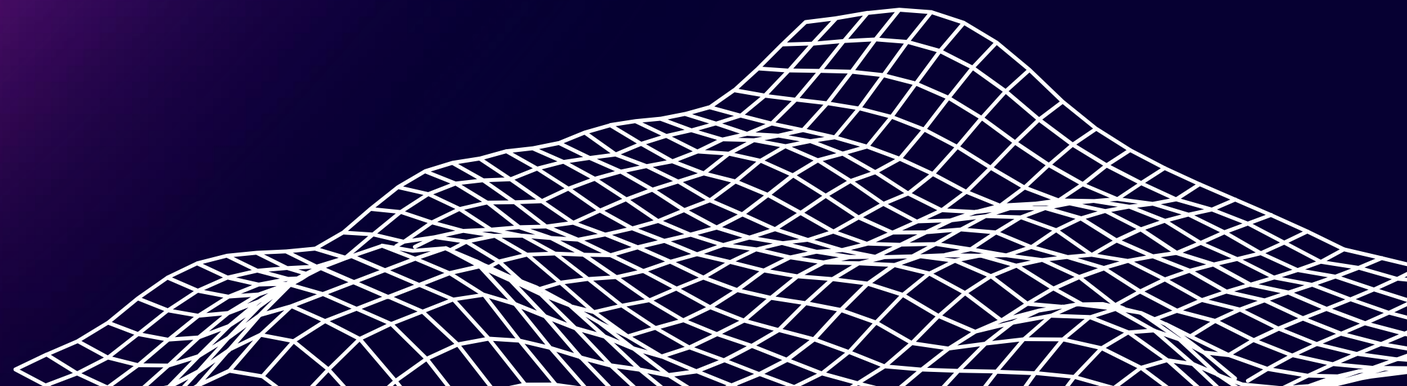
Wine



Fitness

Iteración

4 DETALLES DE IMPLEMENTACIÓN



Detalles de implementación

Implementación propia en Python.

Librerías utilizadas:

- PyTorch: Tratamiento de redes neuronales
- Scikit-Learn: Obtención de datasets
- NumPy, random, math: Operaciones matemáticas
- Copy, pandas, dataclasses: Tratamiento de EDAs
- Time: Control del tiempo de ejecución
- Matplotlib: Generación de gráficos

Repositorio disponible en GitHub.

Estructura orientada a objetos y modular.