Práctica 2: Aprendizaje Evolutivo con NEAT
Control de Robot Robobo

Marcelo Ferreiro Sánchez
Pepe Romero Conde

26 de noviembre de 2025

1. Práctica 2.1: Seguimiento de Objeto Estático

1.1. Definición del Problema

El objetivo de esta subpráctica es desarrollar un controlador neuronal mediante el algoritmo NEAT (NeuroEvolution of Augmenting Topologies) que permita al robot Robobo seguir y acercarse a un cilindro rojo estático en el entorno de simulación.

1.2. Espacios de Estados y Acciones

Espacio de Estados: El estado del sistema se representa mediante un diccionario con cuatro componentes principales:

- **blob xy:** Posición del objeto en la imagen, valores en el rango $[-1, 102]$, dimensión $\mathbb{R}^2$. El valor $[-1, -1]$ indica que el objeto no es visible.
- **IR:** Lecturas de sensores infrarrojos frontal y trasero, valores en $[0, 10000]$, dimensión $\mathbb{R}^2$.
- **tamaño blob:** Tamaño aparente del objeto en píxeles, rango $[0, 1000]$, dimensión $\mathbb{R}^1$.
- **velocidad:** Velocidad actual de las ruedas izquierda y derecha, rango $[-2, 2]$, dimensión $\mathbb{R}^2$.

El vector de observación completo tiene dimensión 7 y se obtiene mediante la concatenación: $\mathbf{o} = [\text{blob\_xy}, \text{IR}, \text{tamano\_blob}, \text{velocidad}]$.

Espacio de Acciones: El espacio de acciones es continuo bidimensional en $[-2, 2]^2$, donde cada acción $\mathbf{a} = [a_1, a_2]$ representa:

- a_1: Incremento de avance recto
- a_2: Incremento de giro hacia la derecha

Las velocidades de las ruedas se calculan como:

$$v_{\text{izq}}(t+1) = \text{clip}(v_{\text{izq}}(t) + a_1 + a_2, -2, 2) \tag{1}$$
$$v_{\text{der}}(t+1) = \text{clip}(v_{\text{der}}(t) + a_1 - a_2, -2, 2) \tag{2}$$

1.3. Función de Fitness

La función de fitness diseñada combina múltiples objetivos mediante una suma ponderada:

$$F = \alpha_1 \cdot e^{-(x-50)^2} + \alpha_2 \cdot e^{-\left(\frac{d}{\sigma}\right)^2} - \alpha_3 \cdot \text{máx}(0, \text{IR}_{\text{atras}} - 58) + \alpha_4 \cdot s \tag{3}$$

donde:

- x: Posición horizontal del blob en la imagen (centrado en 50)
- d: Distancia euclidiana 3D entre robot y objeto
- IR_{atras}: Lectura del sensor infrarrojo trasero
- s: Tamaño del blob en píxeles
- $\alpha_1, \alpha_2, \alpha_3, \alpha_4$: Pesos de ponderación
- σ: Parámetro de escala para la distancia

Justificación: Los dos primeros términos (gaussianos) premian mantener el objeto centrado en la visión y reducir la distancia al objetivo. El tercer término penaliza el movimiento hacia atrás. El cuarto término recompensa cuando el objeto se ve más grande (está más cerca). Los valores utilizados fueron: $\alpha_1 = 0{,}5$, $\alpha_2 = 0{,}5$, $\alpha_3 = 0{,}00001$, $\alpha_4 = 0{,}1$, $\sigma = 15$.

1.4. Implementación y Resultados

La implementación utiliza la librería NEAT-Python con los siguientes parámetros clave:

- Población: [Falta]
- Generaciones: [Falta]
- Función de activación: [Falta]
- Pasos por episodio: [Falta]

Calidad de la Solución: El robot converge consistentemente hacia el objetivo, manteniéndolo centrado en su campo de visión. El aprendizaje se completa en aproximadamente [X] generaciones. La solución muestra robustez ante perturbaciones menores en la posición inicial.

2. Práctica 2.2: Objeto en Movimiento Aleatorio

2.1. Modificaciones y Desafíos

En esta subpráctica, el cilindro rojo se mueve aleatoriamente mediante un random walk con pasos de $\pm$`velocidad_blob` metros en las direcciones x y z. Este comportamiento se implementa en la función `mover_blob_random_walk` del módulo `RoboboAPI`.

El principal desafío es que el robot debe adaptarse continuamente a un objetivo móvil e impredecible, requiriendo una política de control más reactiva y robusta que en el caso estático.

2.2. Resultados y Análisis

Faltan los resultados

3. Práctica 2.3

3.1. Diseño del Experimento

Esta aún no la quiero ni mirar.

3.2. Función de Fitness

$$F = \alpha_1 \cdot e^{-(x-50)^2} + \alpha_2 \cdot e^{-\left(\frac{d}{\sigma}\right)^2} - \alpha_3 \cdot \text{máx}(0, \text{IR}_{\text{atras}} - 58) + \alpha_4 \cdot s \tag{4}$$

Se mantiene la misma forma funcional, pero se ajustan los hiperparámetros para favorecer la exploración inicial cuando el robot está lejos del objetivo.

3.3. Conclusiones

Non hai conclusións lo de