

# Neuroevolución

Algoritmos genéticos para redes profundas (2018)

Verónica E. Arriola-Rios

Facultad de Ciencias, UNAM

11 de noviembre de 2020



# Antecedentes

- 1 Antecedentes
- 2 La red neuronal
- 3 Algoritmo genético de entrenamiento
- 4 Conclusiones

# Temas

- 1 Antecedentes
  - Aprendizaje por refuerzo
  - Juegos

# Aprendizaje por refuerzo

- Un agente reactivo recibe recompensas  $r_t$  por sus acciones  $a_t$ .
- Una política  $\pi$  le indica al agente qué hacer dado un estado  $s$ . En su versión probabilista, esta política se ve como:

$$\pi : S \times A \rightarrow [0, 1] \quad (1)$$

$$\pi(a|s) = P(a_t = a | s_t = s) \quad (2)$$

- La recompensa acumulada es el total de recompensas recibidas al seguir la política  $\pi$ , penalizando aquellas que se obtienen más lejos en el futuro por un factor  $\gamma$  con  $\gamma \in (0, 1]$ :

$$V_{\pi}(s) = E[R] = E[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s] \quad (3)$$

- Se desea que el agente aprenda a maximizar su recompensa al encontrar una política  $\pi$  tal que, dado un estado  $s$ , le indique la acción  $a$  que más le conviene.

# Escenarios de prueba: Atari

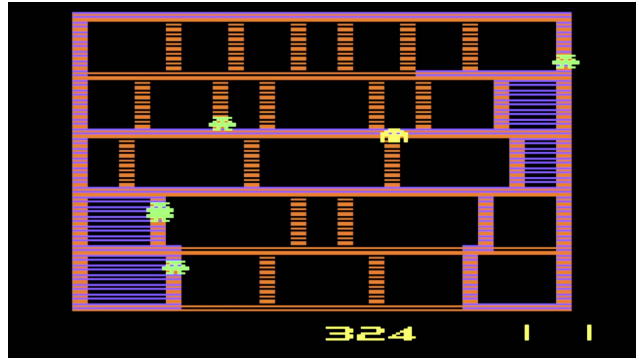
- Los videojuegos otorgan recompensas en la forma de puntuaciones.

# Temas

- 1 Antecedentes
  - Aprendizaje por refuerzo
  - Juegos

# Amidar

**Objetivo:** Recorrer los cuatro lados de cada cuadrado, sin ser tocado por los enemigos.



## Assault

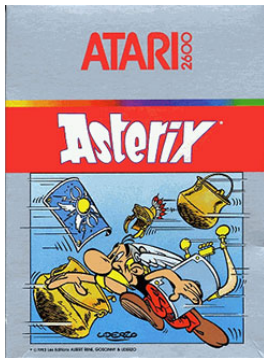
**Objetivo:** Una pequeña nave en la base de la pantalla dispara contra naves extraterrestres que van descendiendo.





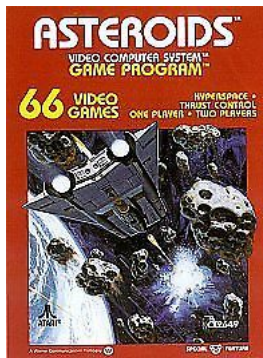
# Asterix

**Objetivo:** Asterix debe tomar las ovas en los canales, evitando las liras.



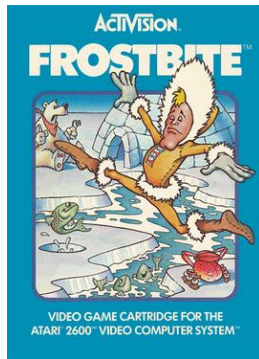
# Asteroids

**Objetivo:** Una pequeña nave triángulo debe destruir asteroides de diferentes tamaños a su alrededor usando un láser, evitando chocar con ellos.



# Frostbite

**Objetivo:** Construir el iglú, brincando sobre plataformas de hielo, sin caer al agua o ser tocado por animales peligrosos, pero pudiendo comer pescado.



# Otros juegos

- Atlantis
- Enduro
- Gravitar
- Kangaroo
- Seaquest
- Skiing
- Venture
- Zaxxon

# La red neuronal

- 1 Antecedentes
- 2 La red neuronal
- 3 Algoritmo genético de entrenamiento
- 4 Conclusiones

# Temas

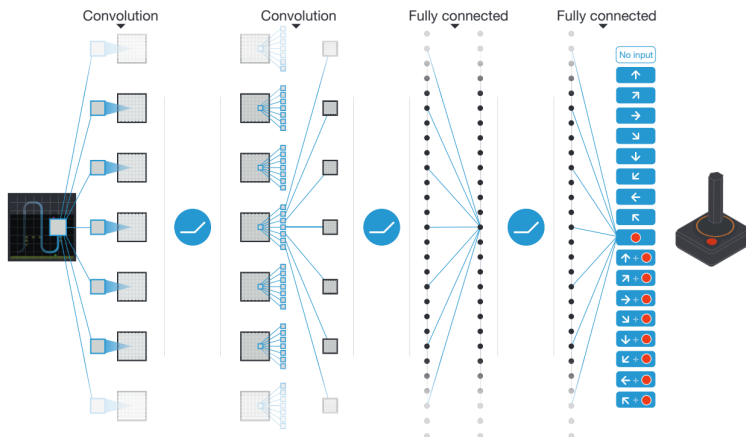
- 2 La red neuronal
  - Arquitectura

# Arquitectura de la red

Descrita en Mnih y col. 2015:

- ❶ 1 imagen como entrada.
- ❷ 3 capas convolucionales:
  - ❶ 32 canales(filtros) de  $8 \times 8$  con salto de 4.
  - ❷ 64 canales de  $4 \times 4$  con salto de 2.
  - ❸ 64 canales de  $3 \times 3$  con salto de 1.
- ❸ 1 capa oculta con 512 unidades.
- ❹ 1 capa de salida con tantas neuronas como acciones válidas había en el videojuego (entre 4 y 18) en *1 hot encoding*.

Todas las capas ocultas (cada convolución incluida) utilizaron  $\text{ReLU} = \max(0, x)$  como función de activación.



**Figura:** 3 capas convolucionales, 1 capa oculta con 512 unidades y una capa de salida. Todas las capas ocultas con ReLU  $\max(0, x)$ , la última es sólo lineal  $x$ .



# Algoritmo genético de entrenamiento

- 1 Antecedentes
- 2 La red neuronal
- 3 Algoritmo genético de entrenamiento
- 4 Conclusiones

# Temas

- 3 Algoritmo genético de entrenamiento
  - Algoritmo de Felipe Petroski et al.
  - Búsqueda con novedad

# Algoritmo genético

Versión de Such y col. 2017:

- Población  $\mathcal{P}$  con  $N$  individuos.
- El *genotipo* de cada red neuronal es su vector de parámetros  $\theta$ .
- En cada generación, cada ejemplar  $\theta_i$  es evaluado, obteniéndose su *aptitud*  $F(\theta_i)^{[1]}$ , en este caso, la *recompensa*.
- *Selección por truncamiento*, los mejores  $T$  individuos se convierten en los padres de la siguiente generación.
- *Elitismo* se conserva al mejor elemento de la generación anterior. Para determinar quién es este elemento se evalúan a los 10 mejores individuos en un conjunto de validación que consiste en 30 episodios adicionales.
- Código disponible en: <https://github.com/uber-research/deep-neuroevolution>

---

<sup>[1]</sup> *fitness* en inglés.

# Generación

---

## Algoritmo 1

 Siguiente generación.

---

```
1: function SELECCIONA(tMejores)
2:   población  $\leftarrow \{\}$ 
3:   for  $N - 1$  veces do
4:     padre  $\leftarrow$  seleccionado uniformemente al azar con reemplazo de
       tMejores.
5:     Mutar padre con ruido Gaussiano  $\theta' = \theta + \sigma\epsilon$  con  $\epsilon \sim \mathcal{N}(0, I)$ 
6:     población  $\leftarrow$  padre'
7:   end for
8:   población  $\leftarrow$  mejorIndividuo
9:   return población
```

---

Donde  $\sigma$  es un parámetro experimental.

# Entrenamiento

- Episodio: hasta morir o máximo 20k cuadros.
- Cada episodio puede incluir múltiples vidas.
- La aptitud es la suma de las recompensas en todos los episodios (i.e. puntaje final otorgado por Atari).

# Inicialización

- Todos los sesgos inician en cero.
- Los pesos se muestrean con una distribución normal estándar.
  - Varianza  $\frac{1}{N_{in}}$  con  $N_{in}$ , el número de conexiones que entran a la neurona.

# Hiperparámetros

Seleccionados de entre 36 en los juegos:

- Asterix, Enduro, Gravitar, Kangaroo, Seaquest, Venture.

Los hiperparámetros usados para los juegos de Atari fueron:

Tamaño de la población ( $N$ ):  $1,000 + 1$

Potencia de la mutación ( $\sigma$ ): 0.002

Truncamiento ( $T$ ): 20

Intentos: 1

# Codificación del individuo

- Cada vector de parámetros es representado como una *semilla* para la inicialización, más la lista de semillas aleatorias que produce la serie de mutaciones que produjo a cada  $\theta$ , a partir de la cual es posible reconstruir  $\theta$ .

$$\theta^n = \psi(\theta^{n-1}, \tau_n) = \theta^{n-1} + \sigma \epsilon(\tau_n) \quad (4)$$

$$\theta^0 = \phi(\tau_0) \quad (5)$$



# Temas

- 3 Algoritmo genético de entrenamiento
  - Algoritmo de Felipe Petroski et al.
  - Búsqueda con novedad

# Búsqueda con novedad

- Otorga recompensa a aquellos agentes que hayan realizado comportamientos que no habían sido vistos anteriormente, en lugar de la función de aptitud tradicional.
- Requiere una *característica* (BC), que describa el comportamiento de una política  $\pi$ :

$$BC(\pi) \quad (6)$$

- Requiere una función de distancia entre los comportamientos de dos políticas cualesquiera:

$$\text{dist}(BC(\pi_i), BC(\pi_j)) \quad (7)$$

Esta función de evaluación depende del dominio.

# Problema de navegación en un laberinto

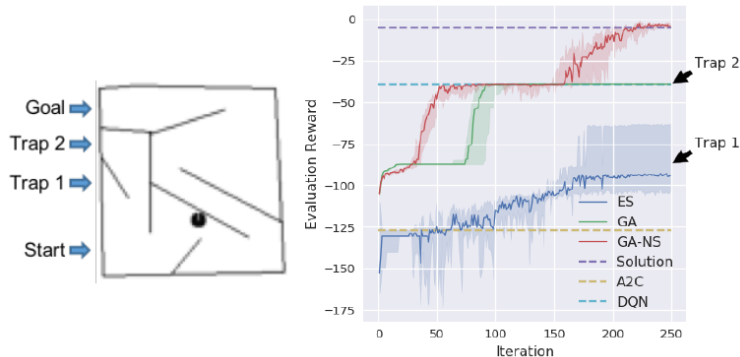


Figura: GA-NS eventualmente logra evadir la trampa. GA se queda atorado en el mínimo local de Trap2



## Conclusiones

- 1 Antecedentes
- 2 La red neuronal
- 3 Algoritmo genético de entrenamiento
- 4 Conclusiones

# Resultados

- Lograron entrenar la red con AG en  $\sim 4$ hrs en una computadora de escritorio o en  $\sim 1$ hr distribuyendo el trabajo entre 720CPUs.

# Referencias I

-  Mnih, Volodymyr y col. (2015). «Human-level control through deep reinforcement learning». En: *Nature* 518, págs. 529-533. URL: <https://web.stanford.edu/class/psych209/Readings/MnihEtAlHassibis15NatureControlDeepRL.pdf>.
-  Such, Felipe Petroski y col. (2017). «Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning». En: *CoRR* abs/1712.06567. arXiv: 1712.06567. URL: <http://arxiv.org/abs/1712.06567>.

# Licencia

Creative Commons  
Atribución-No Comercial-Compartir Igual

