Índice

1. MPI

2. Aprendizaje por refuerzo

Reinforcement Learning

Se basa en experiencias y simulaciones, con prueba y error, recibiendo recompensas con las acciones tomadas (también pueden ser negativas). Nadie le dice al agente que hacer, toma las decisiones con diferentes estrategias. En la etapa de entrenamiento suele ser de forma aleatoria. Una vez tiene un feedback del entrenamiento se toma las decisiones maximizando las recompensas obtenidas en experiencias pasadas.

Algoritmo Q-Learning: Mezcla entre programación dinámica y Monte Carlo.

R=Matriz de recompensas.

Q=Matriz (Estados x Acciones). Que acción elegir en cada estado. (mayor valor)

Aprende el camino si es una buena acción, Back Propagation (Como en redes neuronales).

S=estado actual. A=acción tomada. S'=Estado siguiente. A_i=una acción.

$$\mathbf{Q(S, A) = (1-\alpha)*Q(S, A) + \alpha*(R(S, A) + \gamma*\max_i Q(S', A_i))}$$

Hiperparametros:

• *α (tasa de aprendizaje):*

Debería disminuir a medida que continúa adquiriendo una base de conocimientos cada vez mayor.

• *γ (factor de descuento):*

A medida que se acerca cada vez más al valor límite, su preferencia por la recompensa a corto plazo debería aumentar, ya que no estará el tiempo suficiente para obtener la recompensa a largo plazo, lo que significa que su gamma debería disminuir.

• *ϵ:* Evita que la acción siga siempre la misma ruta.

A medida que desarrollamos nuestra estrategia, tenemos menos necesidad de exploración y más explotación para obtener más utilidad de nuestra política, por lo que en vez de utilizar un valor fijo, a medida que aumentan los ensayos, épsilon debería disminuir. Al principio un épsilon alto genera más episodios de exploración y al final un épsilon bajo explota el conocimiento aprendido.

Mejoras

Paralelización del entorno:

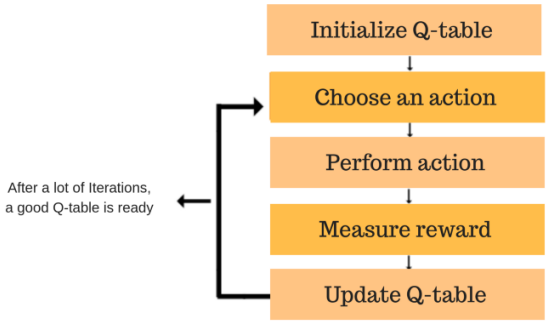
- Ejecutar en varios “workers“ el programa en la misma celda.
- Ejecutar en varios “workers” el programa en diferentes celdas.
- Ejecutar varios “workers” asignando secciones del mapa.
- Recorrer la matriz Q e ir actualizando los valores. Varios workers toda la matriz, dividir la matriz.

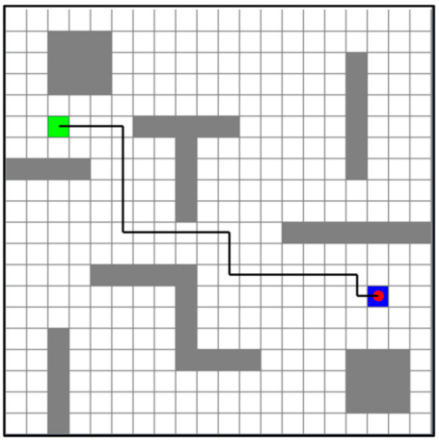
Optimizar los hiperparámetros:

- Ejecutar en varios “workers” el mismo programa, pero con diferentes hiperparámetros

Estrategias de exploración:

DQN (Deep Q-Network): tipo de algoritmo de aprendizaje por refuerzo que combina Deep Learning con Q-Learning.





(a)

https://www.freecodecamp.org/news/an-introduction-to-q-learning-reinforcement-learning-14ac0b4493cc/

https://www.mdpi.com/2073-8994/13/6/1057

3. Programación Evolutiva

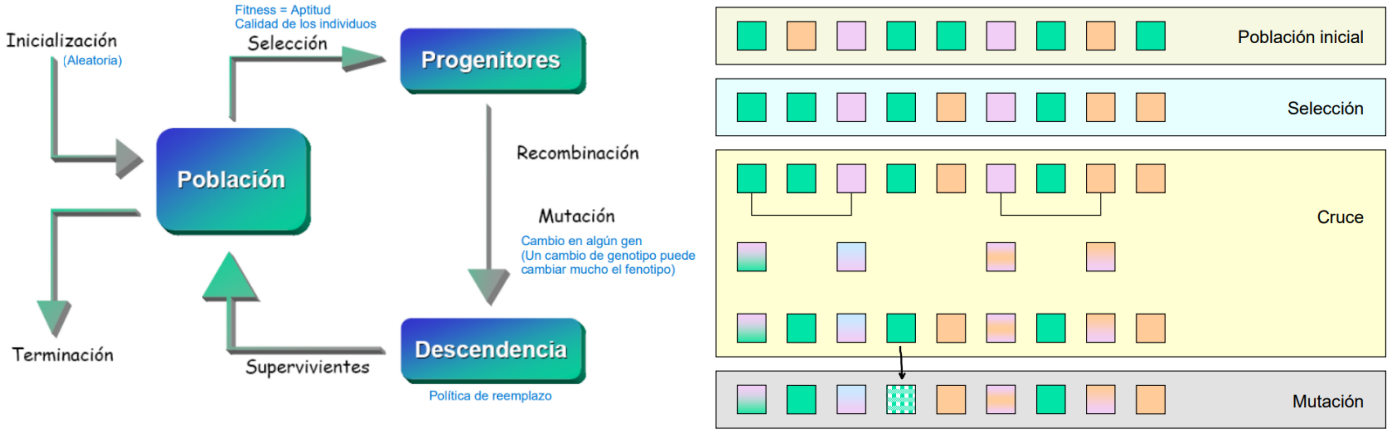
La programación evolutiva es una técnica de optimización inspirada en la teoría de la evolución biológica. Se basa en el concepto de selección natural y evolución de las poblaciones para encontrar soluciones a problemas complejos.

Una población está compuesta por individuos. Un individuo tiene un cromosoma, que tiene uno o varios genes, que a su vez cada gen tiene 1 o varios alelos. Los individuos se pueden representar:

- Binarios: Cada alelo es un bit. Los rangos de números naturales en binario son de 2^N, para codificar un 127 en binario se necesitan 2^7 bits. Y si queremos añadir números reales con una cierta precisión este número de bits aumenta.
- Reales: Números reales, este es más fácil de manejar.

El fenotipo (Decodificación) de un individuo son los rasgos observables, es decir el valor numérico.

El genotipo (Codificación) es la composición genética de un individuo.



Inicialización: Hay que inicializar los individuos de la población.

Tipos: Aleatoria, Elección de población inicial.

Métodos de selección: Como elegir a los individuos de la población que se van a cruzar y mutar.

Tipos: Ruleta, torneo, estocástico, truncamiento, restos. Otros menos conocidos, Boltzmann, roulette-wheel selection (parecido a ruleta, pero es más optimizable), exponential ranking selection.

Métodos de cruce: Individuos de la población que van a cruzarse, tiene una probabilidad, suele ser 0.6.

Tipos: Un Punto, Dos Puntos, Uniforme, Aritmetico, SBX, PMX, BLX, Mask-based crossover, Order crossover. Otros menos conocidos. Cycle crossover, Position crossover, Tree-based crossover.

Métodos de mutación: Individuos de la población que una vez cruzados (o no) van a ser mutados, con una población (Suele ser baja en binarios 0.05 reales 0.3)

Tipos: Único, Múltiple, Uniforme, Intercambio, Inserción, Desplazamiento, No uniforme, Duplicación, Adaptativo.

Elitismo: Asignar una parte de la población a los mejores, así garantizando la supervivencia de los más aptos.

Modificar aptitudes/Desplazamiento: Para maximización y minimización. Devuelve valores positivos.

Convertir problemas de maximización a minimización y viceversa.

Presión Selectiva: La mayor o menor tendencia a favorecer a los individuos más aptos.

Bajar, aumentar o mantener.

Escalado de la adaptación: Permite controlar la diversidad de las aptitudes. Tipos: Lineal, Sigma, Potencial, basado en orden.

Codificación Gray: En muchos problemas de optimización numérica se ha comprobado la utilidad de utilizar un cromosoma representado como un vector de números reales

Criterios de terminación: Búsqueda del mejor criterio de terminación para un problema.

Tipos: Generaciones, llegar al optimo

Mutación variable

Tasa de mutación variable que se vaya reduciendo a medida que avanza la evolución [FOG89].

Modificando los parámetros de entrada:

Tamaño de población, numero de generaciones, probabilidad de cruce, mutación.

Tamaño de individuo, precisión.

Todo esto hace que el Algoritmo Genético pueda llegar a ser bastante lento. Por lo que aplicar paralelización en estos procesos reduce el tiempo de ejecución.

4. Aprendizaje No Supervisado

5. Aprendizaje Supervisado – Redes Neuronales