Universidad de Sevilla
Escuela Técnica Superior de Ingeniería

Ampliación de Robótica

Trabajo de curso
Algoritmo Genético

Julio José López Paneque y Gonzalo Mier Muñoz

4º GIERM

Sevilla - 15 de junio de 2016

Índice

1. Objetivos

En este trabajo se ha tiene como objetivo realizar una implementación básica de un algoritmo genético. Para ello, se ha propuesto desarrollar un ejemplo basado en la idea mostrada en este vídeo, donde dos "luchadores" compiten para eliminarse mutuamente y conseguir una puntuación acorde a su número de disparos acertados. La Figura 1. Muestra un instante de dicha pelea.

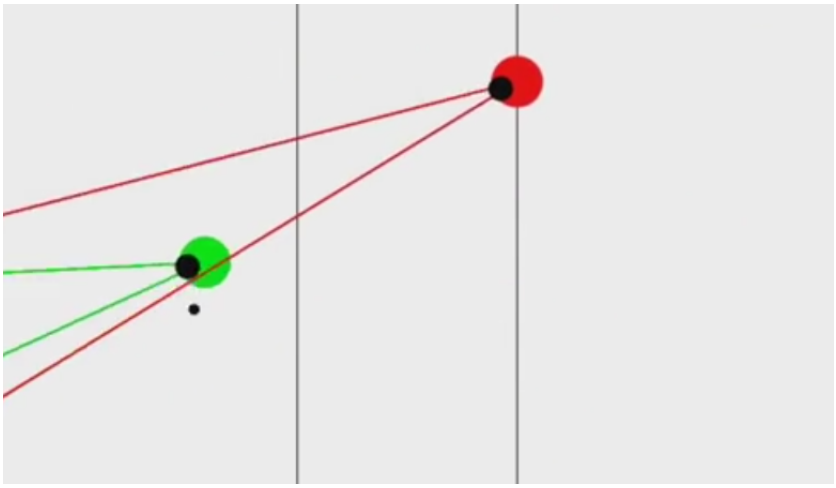

Figura 1: Combate del vídeo de referencia.

El algoritmo genético, en este caso, será utilizado para entrenar los pesos de las redes neuronales que controlarán a los "luchadores" mencionados. La meta de éstos será ganar los combates a los cuales serán sometidos por parejas (o en grupos de 4), por lo que las redes neuronales serán entrenadas de manera que mejore el resultado en estos combates.

2. División del Trabajo

Sólo hay dos partes de este trabajo que se han realizado de manera "independiente". Estas han sido:

- La base del motor gráfico fue realizada por Julio.
- La inclusión de peleas de 4 "luchadores" fue realizada por Gonzalo.

El resto del trabajo, así como las correcciones necesarias de dichas dos partes, se realizó en conjunto por los dos alumnos, al igual que la memoria.

3. Explicación de las Técnicas Usadas

3.1. Red Neuronal

La red neuronal (Figura 2) es una técnica de aprendizaje automático inspirada en el sistema nervioso humano, que establece relaciones no lineales entre sus entradas con el fin de obtener una salida. Estas relaciones se obtienen mediante alguna técnica de "aprendizaje". Las redes neuronales están formadas por una capa de entrada o *"input layer"*, un número arbitrario de capas ocultas o *"hidden layers"* y una capa de salida u *"output layer"*.

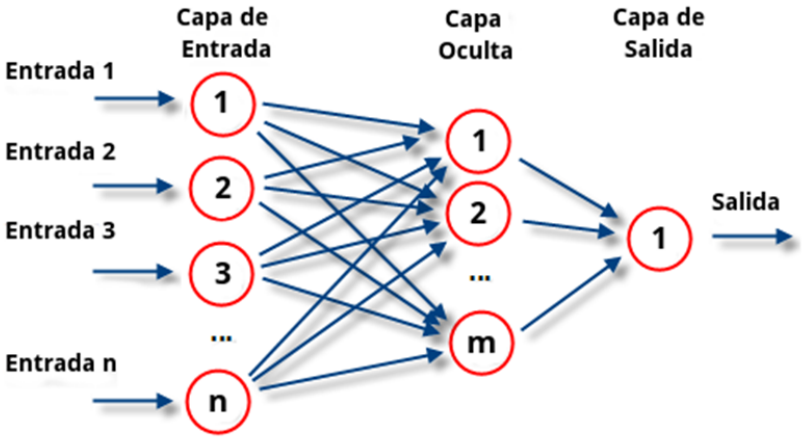

Figura 2: Red neuronal.

Cada neurona (Figura 3) está compuesta por N entradas, $N+1$ pesos (ya que en cada capa se añade una entrada de valor unidad, para obtener términos independientes de las entradas exteriores), una función que multiplica los pesos por sus respectivas entradas y luego los suma (función de red), y una función de activación.

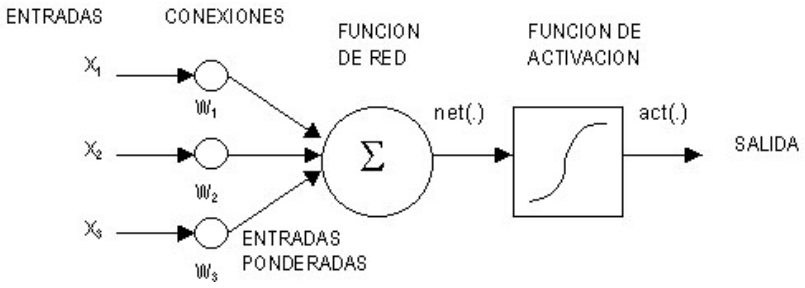

Figura 3: Neurona.

La función de activación (Figura 4) es una transformada que añade el carácter no lineal a las redes neuronales. En este trabajo se ha optado porque esta función sea una sigmoidea con la fórmula:

$$f(x) = \frac{1}{1 + e^{-8(x-0{,}5)}} \tag{1}$$

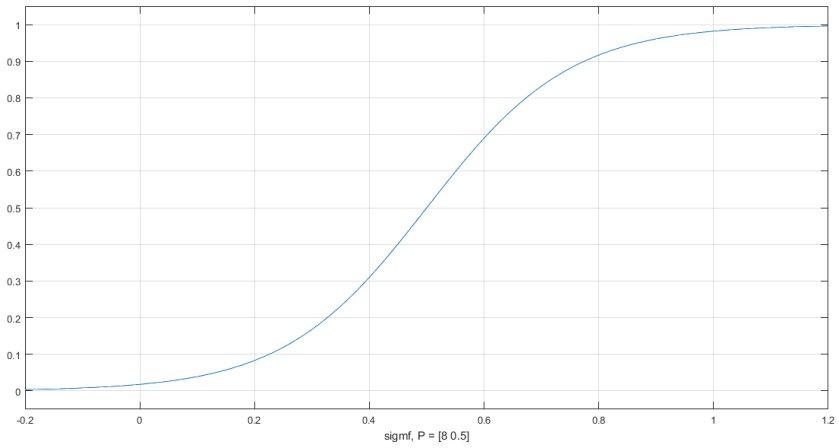

Figura 4: Función sigmoidea utilizada.

3.2. Algoritmo Genético

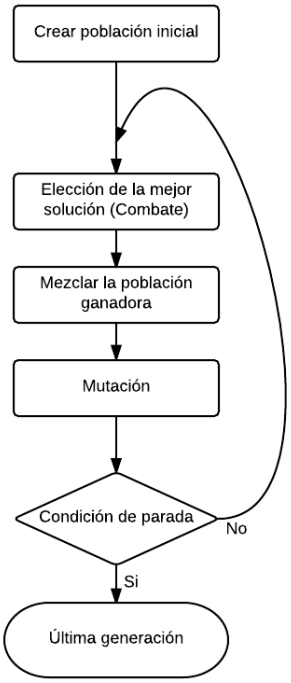

Figura 5: Algoritmo genético.

Los algoritmos genéticos son algoritmos de inteligencia artificial basados en la evolución. Siguen el principio de desaparición de los resultados más "débiles" (que funcionan peor), mientras que aquellos más "fuertes" (que dan un mejhor resultado) sobreviven y dan lugar a una nueva generación. Estos algoritmos buscan una solución que funciona bien para el problema que se intenta solucionar, aunque eso no implica que sea la solución más óptima. Cada vez que se ejecutan se genera una solución diferente a la anterior, en aquellos casos en que es posible.

Los pasos a seguir (Figura 5) son:

1. Se crea una población inicial con n individuos aleatorios.
2. Se evalúan los individuos de la población actual y se separan los más fuertes (menor error o mejor resultado). Formarán parte de la nueva población.

3. Se mezclan los individuos de la nueva población para obtener nuevos individuos (cruzamiento). Al final de esta fase, tiene que haber el mismo número de individuos que en la ronda anterior.

4. Se añade un gen de mutación a algunos individuos, para abrir posibles nuevos resultados.

5. Si la última generación cumple los objetivos propuestos (por ejemplo, ruta más corta a un punto), el resultado es el mejor individuo de ésta, que es el que haya obtenido mejor resultado en dicho paso.

6. Si no se cumple el objetivo se vuelve al paso 2.

Elección de Mejores Individuos

Contando con una población, para realizar la elección de los mejores individuos hay que centrarse en saber cual es el objetivo del problema que queremos solucionar.

Las posibilidades son:

- Menor error: Se evalúa cada individuo con la función objetivo, se calcula el error cuadrático medio y se elimina aquella parte de la población que menos se aproxima o que tiene mayor error. Este uso solo es aplicable cuando queremos hacer aproximaciones de funciones matemáticas, se puede calcular el error o la prueba a la que es sometida participa solo un individuo a la vez.

- Competición: Se hace que compitan los individuos en grupos. El que se aproxime más al objetivo propuesto es el que sobrevivirá. También recibe el nombre de selección por torneo. En este caso la prueba de evaluación no atiende unicamente a un único individuo evaluado, si no también a los contrincantes.

Cruce

Cada individuo tiene un ADN que lo caracteriza. En el caso de este trabajo, pesos que se quieren hallar puestos en un vector. Cuando se cruzan dos padres lo que se hace es combinar sus cadenas de ADN para obtener el ADN del hijo. Se puede hacer mediante un corte (hasta X gen son todos del primer padre, el resto son del segundo), dos cortes (entre X e Y gen son todos del primer padre y el resto del segundo) o con una máscara (se elige

aleatoriamente de que padre se coge cada gen).

En este trabajo se ha utilizado el cruce por máscara, y se han usado 2 padres por cada hijo.

Mutación

Para evitar que la población se estanque en una solución, se añade aleatoriamente un gen que mute (un peso que adopte otro valor aleatorio) a varias de las nuevas soluciones.

Se ha comprobado que, sin cruce, el algoritmo genético sigue funcionando, mientras que sin mutación llega un momento en que se estanca.

Se puede elegir el número de mutaciones como:

- Número bajo de mutaciones: Poca variación a lo largo de las rondas. Esto es útil cuando se ha obtenido una solución satisfactoria y se quiere afinar más.
- Número alto de mutaciones: La población varia mucho, haciendo que su comportamiento diverja más. Esto resulta útil para acercarse más rápidamente a una solución final.
- Creciente con el número de rondas: Hace que la población no se estanque con el paso de las generaciones.
- Decreciente con el número de rondas: Hace una búsqueda inicial grande, para luego ir decreciendo el número de variaciones. Esta combinación de los dos primeros enfoques permite ampliar rápidamente el número de soluciones, pero luego afinar la búsqueda en torno a una combinación correcta de pesos.

4. Implementación

4.1. Desarrollo de la Partida

El mapa en el que se disputarán los combates esta formado por una región de igual tamaño para cada combatiente, separadas por una región central por la cual los individuos no pueden pasar. Un requisito al crear al mapa es que fuera de igual tamaño para cada jugador, de tal manera que ninguno de los dos lados tuviera ventaja sobre el otro. Además, los jugadores comienzan girados $\frac{\pi}{2}$ radianes con respecto al contrincante, lo cual es equivalente a mirar hacia arriba, de manera que los luchadores que no aprendan a moverse y apuntar no podrán conseguir ninguna puntuación.

En la Figura 6 se muestra un ejemplo típico de combate. Cada luchador tiene un campo de visión dentro del cual puede ver las balas del enemigo y al propio jugador enemigo. A mayor campo de visión, más cosas se pueden ver, pero menos precisa será la información (el luchador sabe que ve una bala, pero no dónde).

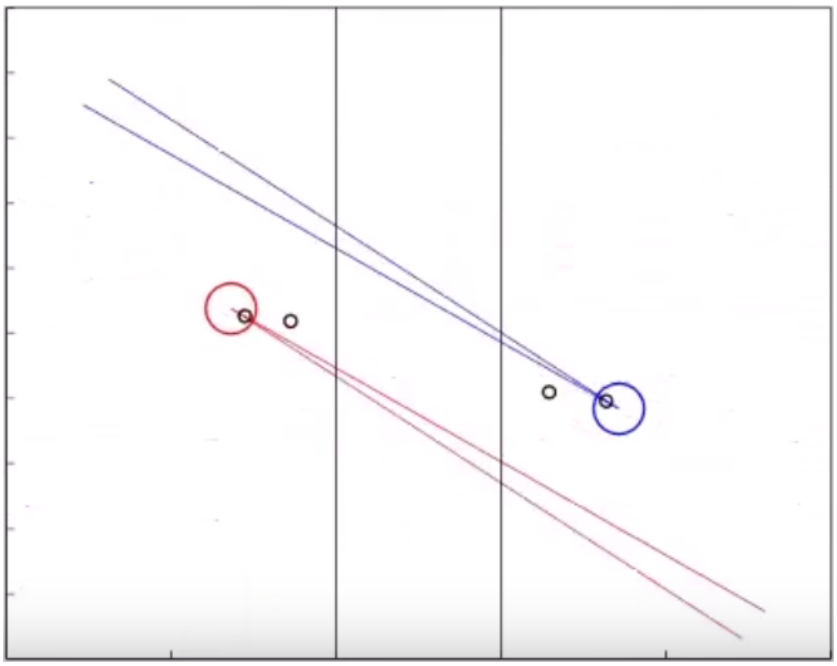

Figura 6: Ejemplo de combate.

El combate se ha implementado por turnos: en cada ciclo, los luchadores reciben información de su entorno, toman una decisión y ejecutan una acción.

Por ejemplo, si un luchador decide moverse hacia adelante, avanzará unos pocos píxeles en ese turno. Al ejecutarse todos los turnos de manera continua, la sensación es de que el combate es en tiempo real (siempre que se use un ordenador moderno).

Una vez se completan todos los combates de una generación (la mitad que el número de luchadores), el algoritmo genético se actualiza para generar a los nuevos combatientes.

4.2. Decisiones del Luchador

Las acciones de cada luchador son decididas por un red neuronal de la siguiente estructura:

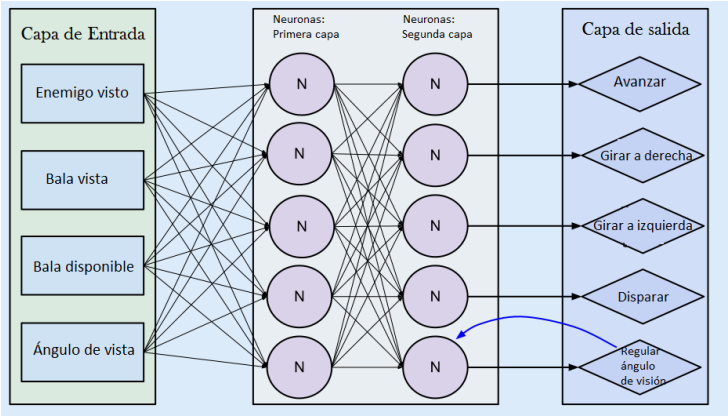

Figura 7: Individuo

La cual se compone de 4 entradas y 5 salidas.

Entradas:

1. Enemigo visto: ¿Hay algún enemigo en mi campo de visión?
2. Bala vista: ¿Hay alguna bala enemiga en el campo de visión? (En caso de que haya 4 jugadores no se distingue de quién es la bala).
3. Bala disponible: ¿Puedo disparar? (Esto se cumple cuando no hay una bala de este luchador en el campo de juego)
4. Ángulo de vista: ¿Qué tamaño tiene mi campo de visión?

Salidas:

1. Avanzar: Ir hacia adelante.

2. Girar a la derecha.

3. Girar a la izquierda.

4. Disparar una bala.

5. Regular ángulo de vista: Aumentar o reducir campo de visión.

Los 70 pesos de esta red neuronal serán los 70 elementos del ADN de cada luchador, que serán entrenados por el algoritmo genético.

4.3. Ejecución del Algoritmo Genético

Para hacer el cruzamiento se ha optado por usar una máscara de cruce, ya que con uno o dos cortes se hace necesario aumentar mucho el número de generaciones de entrenamiento, y tampoco reduce demasiado la complejidad.

La mutación se ha optado por hacerse en un gen cada vez a todos los nuevos hijos.

La manera de distribuir a la población en cada ronda es la siguiente:

1. Se seleccionan los padres que van a sobrevivir y se colocan como primeros de la población.

2. Se rellena la población restante con hijos de estos padres, tomando pares de padres aleatorios.

3. Se mutan los hijos.

Con esto se consigue que el individuo más fuerte se mantenga el primero de la población, ya que si en algún momento aparece un individuo más fuerte, irá subiendo posiciones hasta quedarse en la primera posición. Además, hace que los hijos peleen con otros hijos y los padres con padres, por lo que los combates están en principio más nivelados.

4.4. Tipos de Combate Implementados

Por último, como casos a probar se han puesto:

1. Combate de 2 jugadores. Cada vez que una bala da al enemigo se suma uno al marcador. Gana quien puntúe más.

2. Combate de 2 jugadores con vidas. Cada bala que golpee a un enemigo aumenta en un punto tu marcador y reduce el del contrincante en uno.

3. Combate de 4 jugadores. Cada bala que golpee a un enemigo suma un punto. Gana quien más puntuación obtenga. En la figura 8 se ve un ejemplo de un combate a 4.

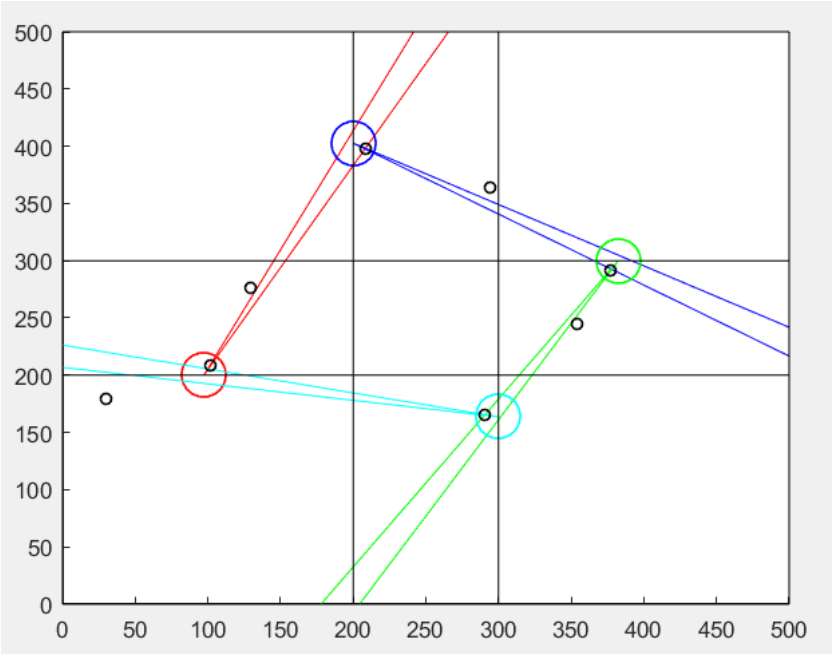

Figura 8: Ejemplo de lucha entre cuatro jugadores.

5. Uso del Código

En el .rar hay dos carpetas, una llamada .ᴬlg2pz otra .ᴬlg4p". La primera contiene la implementación con dos jugadores, y la segunda el caso de 4.

Para ejecutar el algoritmo, sólo se necesita usar el archivo "Game.m‘" en MATLAB. En él, se ve de manera fácil cómo cambiar el número de generaciones y la cantidad de luchadores de cada una. Para ver una lucha, el archivo "Last_gen.m" se puede editar para elegir qué generación y lucha visualizar. Este archivo se debe ejecutar con el entrenamiento ya hecho.

El resto de archivos ("seen.m" y "nn.m") son llamados automáticamente por el código. En concreto, el archivo "seen.m" se encarga de decirle a cada

luchador si ha visto algún elemento en ese turno (una bala o un enemigo) mediante unas reglas simples de productos vectoriales, y el archivo "nn.m" es el que ejecuta la red neuronal de cada jugador a partir de los pesos de que disponen.

6. Resultados

Los resultados han sido recopilados en este vídeo, donde se muestran todos los casos mencionados y se dan explicaciones del comportamiento del algoritmo.

7. Mejoras Posibles

Sería interesante probar distintos tipos de reproducción entre los padres de las nuevas generaciones, por ejemplo:

- El primer padre aporta la mayor fracción de pesos al nuevo hijo, el siguiente padre aporta algo menos y así sucesivamente con todos los padres.
- Otra posibilidad es que cada padre (de todos los supervivientes) aporte exactamente el mismo número de genes al nuevo hijo, aunque los genes aportados sean distintos para cada.

También se podría hacer un estilo de combate diferente (movimientos más rápidos, balas más rápidas, estimación de la velocidad de las balas, movimientos hacia atrás...) para ver cómo evolucionan los luchadores en dicho caso.