Universidad de Buenos Aires
Facultad De Ingeniería
Año 2012 - 1er Cuatrimestre

Sistema de Programación No Convencional de Robots (75.70)

Trabajo Práctico Final
Utilización de Redes Neuronales en Juegos de Ingenio

Integrantes

Apellido, Nombre	**Nro. Padrón**	**E-mail**
Bukaczewski, Verónica	86954	vero13@gmail.com
Rivero, Hernán	88455	riverohernanj@gmail.com

Índice

1. Objetivo

El objetivo del presente trabajo práctico es descubrir y analizar diferentes posibilidades para entrenar una red neuronal de forma tal que sea capaz de jugar de manera eficaz a un juego de ingenio. Se utilizará una red neuronal de tipo backpropagation, para generar un método de aprendizaje a medida que se desarrollan las partidas. El juego elegido para los ensayos es el Ta-Te-Ti.

2. El juego Ta-Te-Ti

Por lo general, el Ta-Te-Ti se juega en una cuadrícula de tres por tres (ver Figura [1]). Cada jugador, a su vez se mueve mediante la colocación de un marcador en un casillero vacío. El marcador de un jugador es "X"(cruz) y el del otro es "O"(círculo). El juego termina tan pronto como un jugador tiene tres marcadores en una fila: horizontalmente, verticalmente, o en diagonal (un ejemplo se muestra en la Figura [2]). El juego puede también terminan en empate (ver Figura [3]), si no hay posibilidad de ganar para alguno de los jugadores. Un jugador que conozca unas pocas estrategias básicas puede asegurarse al menos un empate.

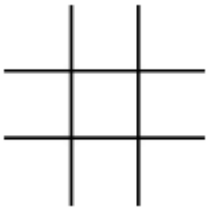

Figura 1: Grilla vacía TaTeTi.

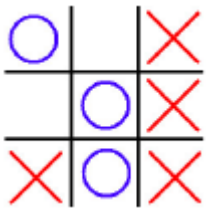

Figura 2: El jugador cruz gana la partida.

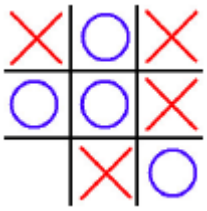

Figura 3: La partida terminó empatada.

3. Estructura general de la solución propuesta

Se realizó un programa en lenguaje Java y se utilizó el framework de red neuronal Joone. A continuación se presenta el diagrama de clases.

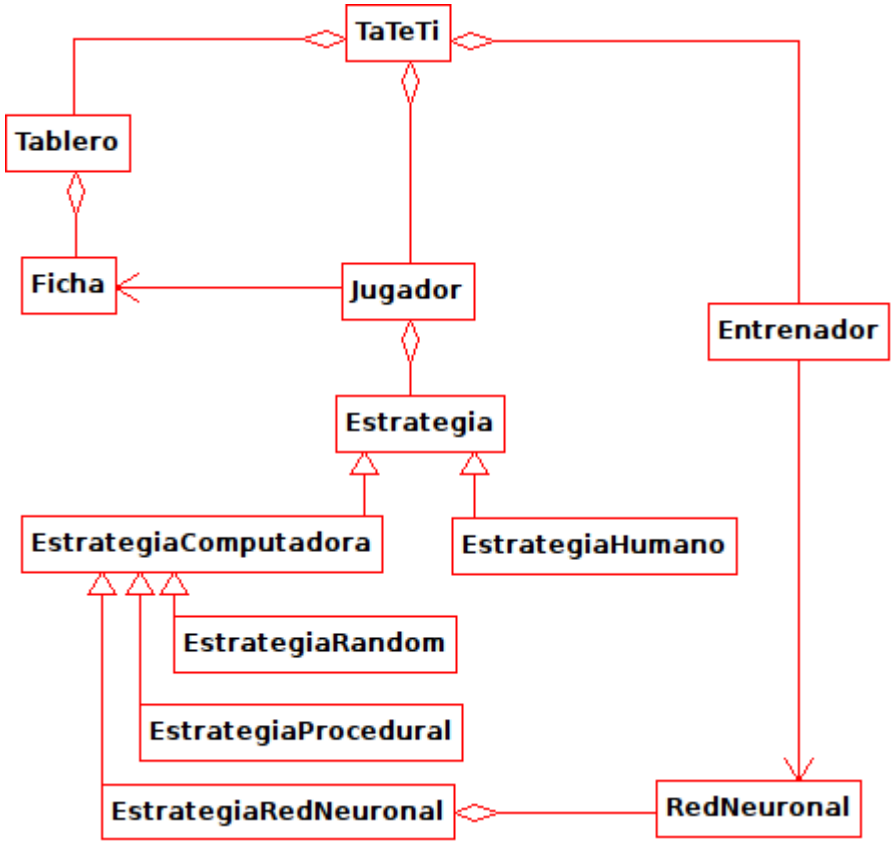

Figura 4: Diagrama de clases de la solución propuesta

4. Soluciones propuestas

Se armaron distintas estructuras de redes neuronales, con el objetivo de poder entrenarlas para que jueguen al Ta-Te-Ti. El primer intento fue la realización de una red con nueve inputs y nueve outputs, al cual se le variaron la cantidad de capas ocultas. Los inputs representan los nueve casilleros del tablero y se especifican con:

- 1: ficha de la red neuronal.
- -1: ficha rival.
- 0: casillero vacío.

Los outputs representan los nueve casillero del tablero, pero esta vez se le asocia un puntaje que nos indica que tan apropiado es jugar en dicho casillero.
Luego de varios intentos de aprendizaje, quedó descartada porque no mostraba cambios significativos al variar los inputs.

5. Solución utilizada

5.1. Estructura de la Red Neuronal

Se armó una red con dieciocho valores de input, organizados de la siguiente manera:

- del 1 al 9 valen 1 si la posición está ocupada con una ficha de la red, 0 en caso contrario.
- del 10 al 18, valen 1 si hay una ficha del rival, 0 en caso contrario.

Los outputs son nueve y representan los puntajes para los casilleros del tablero. Se utilizó una capa oculta de dieciocho filas.

5.2. Entrenando la Red Neuronal

El entrenamiento de la red neuronal se realizó mediante el juego contra otros algoritmos que juegan Ta-Te-Ti. Con el avance de las partidas, se le fue enseñando a la red neuronal los movimientos a realizar. Para ellos se utilizó un sistema de puntajes donde luego de cada partida se busca que la red imite los movimientos del ganador y evite los del perdedor. De esta forma se le fue marcando un camino y con el pasar de las partidas, la red iba adquiriendo una mayor inteligencia a la hora de realizar sus jugadas.

Para analizar diferentes alternativas de entrenamiento, se construyó un algoritmo que sigue una estrategia metódica que resulta repetitiva pero efectiva y otro que simplemente coloca marcas en lugares al azar.

6. Conclusiones

Una primera conclusión nos indica que el armado de una red neuronal lleva un tiempo considerablemente mayor al del entrenamiento de la misma. Esto se debe a que el entrenamiento es un proceso automatizado, que no requiere más que la conguración de algunos parámetros. Mientras que para el armado de una red neuronal, que nos permita resolver el problema, es necesario un conocimiento sobre la situación que estamos enfrentando, de manera de poder crear una red que nos permita resolver el problema de manera eciente. Por otro lado, observamos que el entrenamiento prolongado de una red mal armada no signicaba llegar a los resultados deseados. Es decir, que lo más importante es concentrarnos en crear una red acorde al problema; de manera de asegurarnos llegar a los resultados buscados; lo cual puede estar acompañado de un tiempo de entrenamiento más corto de lo esperado.

En cuanto a lo que se observa del entrenamiento de la red, la conclusión más importante que se obtiene es que no es posible alcanzar un desarrollo óptimo si se utiliza un único algoritmo. La red es capaz de aprender a defenderse del algoritmo bueno, pero esto no le va a servir al momento de enfrentarse a otros. La red aprende también a ganarle con mucha facilidad al algoritmo aleatorio, pero utilizando estrategias muy básicas que no le sirven ante un rival de mayor inteligencia.

La mejor forma de entrenar a la red es alternando los algoritmos que la enfrentan, de manera tal de que se encuentre con la mayor cantidad de variantes de jugadas posibles. La estrategia de jugar en posiciones aleatorias no resulta conveniente para el entrenamiento inicial. Esto es debido a que la red de esta manera, luego de una cierta cantidad de partidas, comienza a repetir siempre la misma jugada, que no necesariamente es buena pero resulta eficaz ante la nula inteligencia del rival. Prolongar un entrenamiento de estas características conduce a que la red refuerce cada vez más estas prácticas poco recomendables.

El algoritmo aleatorio sí resulta de utilidad para reforzar una red ya entrenada, ya que puede ser capaz de descubrirle vulnerabilidades. Cuando se utiliza es recomendable no premiar las victorias, por el fenómeno descripto anteriormente. Tan solo se deben penar los movimientos que conducen a derrotas.

Referencias

[1] **Documentación Joone**
http://sourceforge.net/projects/joone/files/Documentation/DTE/JooneDTEGuide.pdf

[2] **Tutorial Básico Joone**
http://ubuntuone.com/p/1dB/

[3] **Training an artificial neuronal network to play tic-tac-toe**
http://users.auth.gr/kehagiat/GameTheory/12CombBiblio/TicTacToe.pdf

[4] **An artificial neural network (Tic-tac-toe)**
http://stackoverflow.com/questions/761216/how-to-code-an-artificial-neural-network-tic-tac-toe

[5] **Neural Net Training for Tic-Tac-Toe**
www.cs.virginia.edu/~bmb5v/cs660/Project.doc

[6] **TD Learning of Game Evaluation Functions with Hierarchical Neural Architectures**
http://webber.physik.uni-freiburg.de/~hon/vorlss02/Literatur/reinforcement/GameEvaluationWithNeuronal.pdf

[7] **A Neural Network that plays TicTacToe**
www.tropicalcoder.com/NeuralNetwork.htm

[8] **Tic Tac Toe Neural Network as Evaluation Function**
http://stackoverflow.com/questions/10680049/tic-tac-toe-neural-network-as-evaluation-function

[9] **Tic Tac Toe Neural Networks**
http://es.appbrain.com/app/tic-tac-toe-neural-networks/com.samuel.tictactoe

[10] **Playing tic-tac-toe using a modified neural network and an improved genetic algorithm**
http://repository.lib.polyu.edu.hk/jspui/handle/10397/1364

[11] **Probabilistic Neural Network Playing a Simple Game**
www.dsi.unifi.it/ANNPR/CR/23.pdf

[12] **Using Evolutionary Programming to Create Neural Networks that are Capable of Playing Tic-Tac-Toe**
http://65.44.200.132/Library/1993/EP_NN_tic_tac_toe.pdf

[13] **Tic Tac Toe Time**
challenge.nm.org/FinalReports/93.pdf

[14] **Evolution, Neural Networks, Games, and Intelligence**
http://red.cs.nott.ac.uk/~gxk/courses/g5baim/papers/evolve-nn-001/ChellapillaAndFogelProcIEEEText.PDF