

# Análisis del Programa "Tic-Tac-Toe"

Para el programa “Tic-tac-toe” el autor utiliza las librerías utilizadas en los ejemplos anteriores para redes feedforward, de tal manera que por esta parta el autor no realiza ninguna modificación a dichas clases.

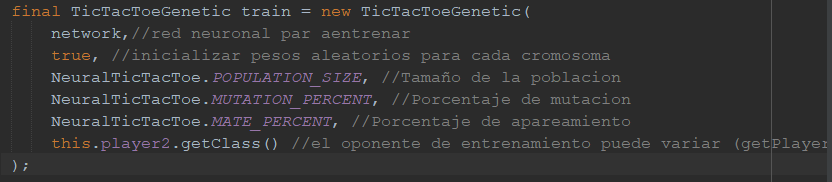
Iniciando el programa el autor establece una red neuronal feedforward con las características siguientes:

* Capa de entrada: 9 neuronas
* Capa oculta: 10 neuronas
* Capa de salida: 1 neurona

Con ayuda de la función estática *createNetwork()* construye la red neuronal utilizando la clase feedforward y los parámetros antes mencionado. Los pesos son seteados de manera aleatoria.

Después de construir la red se inicia el entrenamiento. En esta parte se tiene en cuenta a los jugadores: uno de ellos es la red neuronal por entrenar y el oponente es otro método de juego que el programa tiene integrado. Además de los jugadores es bueno rescatar que ahora se utiliza un entrenamiento no supervisado, en donde no proporcionamos salidas ideales y es compatible con el algoritmo genético, profundizare mas de esto en el cálculo de error.

En el ejemplo el autor utilizar el algorimo genético, y para ello establece los parámetros:



* Tamaño de población = 200
* Porcentaje de mutación = 0.10
* Porcentaje de apareamiento = 0.25

Con estos valores se inicializa los valores de la población la cual cuento con 200 cromosomas con 120 genes cada una, en donde sus valores son seteados de manera aleatoria.

Para optimizar los tiempos que tomara entrenar la red, el autor utiliza los hilos y la característica multicore de los procesadores actuales, de esta manera asigna tareas a los hilos y asegura de llevar mas de una tarea a la vez. Y el entrenamiento consiste en una cantidad de tiempo definido, lo que hace que la calidad y precisión de la red se con respecto al tiempo que se entrene y la capacidad del ordenador donde se realice dicha acción.

Una vez inicializados los valores para los genes de los 200 cromosomas iniciamos la iteración para el entrenamiento de la red con el algoritmo genético. La aplicación del algoritmo no tiene mucha variación. Mas adelante explico las diferencias.

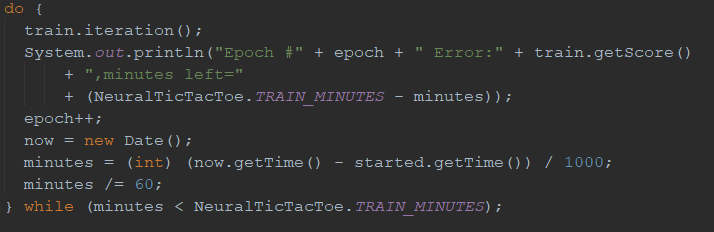


Ilustración Ciclo de entrenamiento

## Clase TicTacToeGenetic

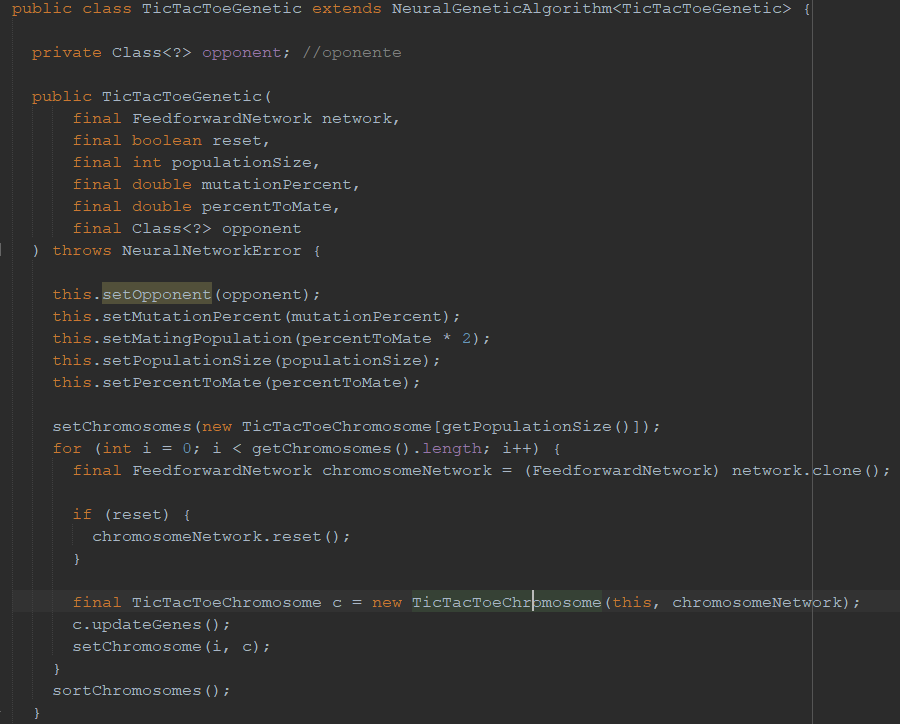


Ilustración Fragmento de la clase TicTacToeGenetic

Esta es la clase que hereda de *NeuralGeneticAlgorithm* y es una de las clases principales, en ella se establecen los parámetros especificados en las secciones anteriores, como en el ejemplo de xor la clase inicializa un array para las cromosomas e inicializa uno a uno los valores de este estableciendo una red neural feedforward con pesos aleatorios (dependiendo de la variable de control), y además obtiene los valores de los genes a partir de los pesos de conexión. Entre las diferencias significativas esta la existencia del método

*setOpponent(final Class<?> opponent) {}*

en donde se establece una instancia del metodo seleccionado para ser oponente durante el entrenamiento de la red neuronal.

## Clase TicTaeToeChromosome

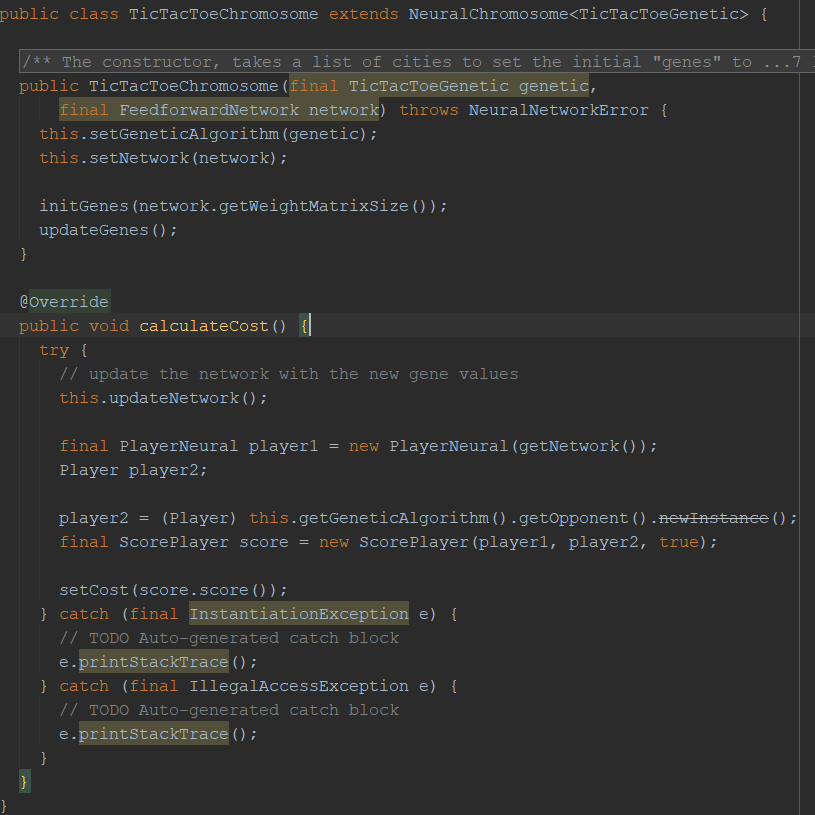


Ilustración Fragmento de la clase TicTaeToeChromosome

La instancia de esta clase se realiza en la clase *TicTacToeGenetic.* Extiende de la clase *NeuralChromosome*. En esta clase en particular es donde aparece la primera implementación que difiere del caso del xor; en el cálculo del costo la dinámica que se había venido estudiando cambia. Ahora en vez de obtener el valor del error cono representante del costo, el autor propone que el costo sea representado con el puntaje obtenido del jugar 100 partidas contra su oponente, el marcador generado con esas jugadas es sometido a un calculo y el resultado es el costo.

Mas a detalle esto consiste en los siguiente:

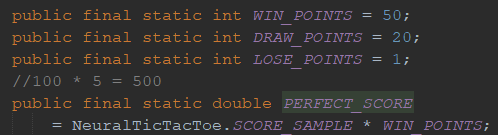


Ilustración Valores estáticos que propone el autor

Con el valore de estas variables el autor realizar 100 iteraciones (correspondientes al valore de la variable estática *SCORE\_SAMPLE*) en donde los dos jugadores juegan y se establece si la red neuronal gano, perdió o empato; dichos valores son acumulados. Al finalizar las 100 interacciones el autor realizar el cálculo:

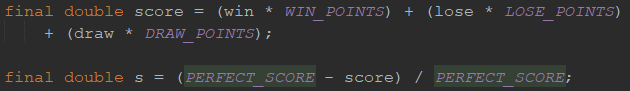


Ilustración Calculo del score

Ejemplo

*Score = (68 \* 50) + (21 \* 1) + (11 \* 20 ) = 3,641*

*S = (5000 - 3,641) / 5000 = 0.2718*

Si s en menor a 0 se retorna un -1 de lo contrario retorna el valore de s.

Y así es como se obtiene el valore del costo de la red neuronal, el resto del proceso de selección, apareamiento, mutación se realiza de la misma manera que en el ejemplo del xor.

Así es como se entrena la red neuronal feedforward para aprender a jugar al tic-tac-toe. EL procedimiento con un método de entrenamiento no supervisado cambia con respecto al método supervisado, pero también es óptimo según sea la situación en la que se quiere aplicar. Además, cabe resaltar que la compatibilidad y versatilidad que el algoritmo genético tiene con los diversos métodos de entrenamiento es sorprendente, tiene una capacidad de acoplamiento muy grande y se puede aplicar a muchos problemas no solo para redes neuronales.

## ¿Cómo juega la Red neuronal Feedforward?

El programa tiene una interface llamada *Player* y luce de la siguiente manera:

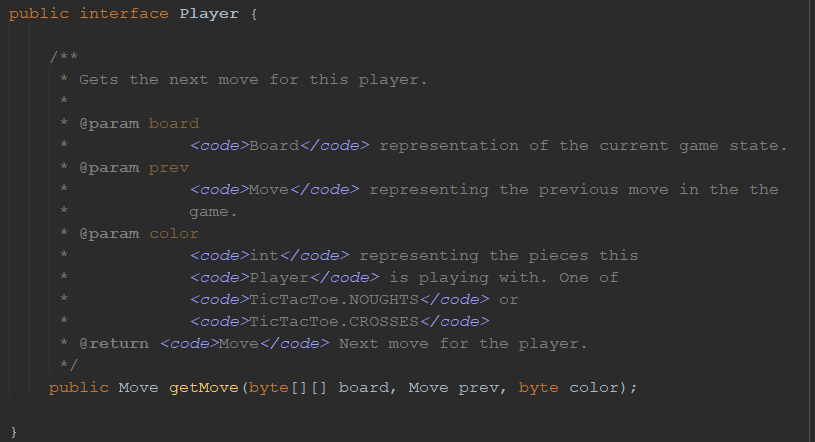


Ilustración interfaz Player

Aquella clase que la implemento debe de tener el método *getMove* y es la forma en que el programa puede implementar diferentes tipos e jugadores, mientras implementen este método de manera correcta se podrá jugar sin problemas.

Para el caso de la red neuronal, el autor propone la clase *PlayerNeural* con dos métodos:

* *getMove*
* *tryMove*

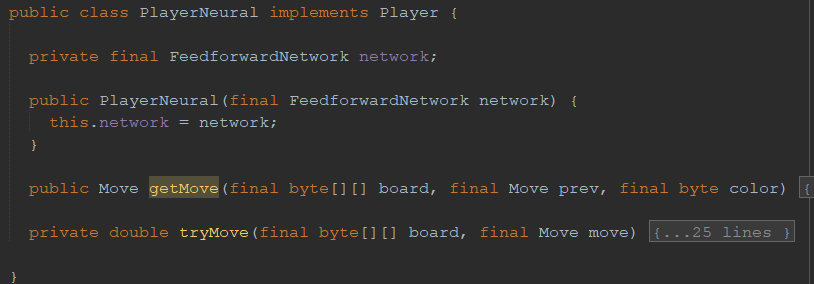


Ilustración implementación de la interfaz Player

### Metodo *getMove*

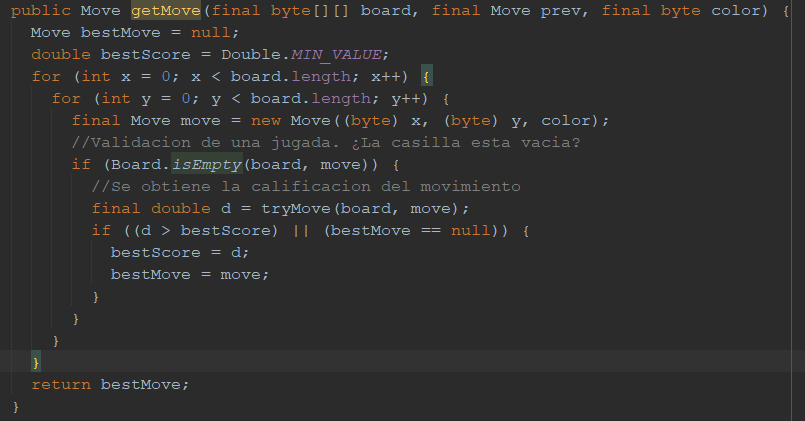


Ilustración Metodo getMove

En este método se captura el movimiento de la red neuronal, estableciendo las coordenadas x, y además del dibujo que le corresponde realizar (tachas o círculos). Se verifica se la casilla del tablero esta vacía y hace un llamado al segundo método (*tryMove)* para obtener la calificación del movimiento, este valor se va sobrescribiendo cuando el actual supera al establecido y captura la instancia del movimiento. Al final el movimiento con mayor calificación es el que la red neuronal decide realizar.

### Metodo *tryMove*

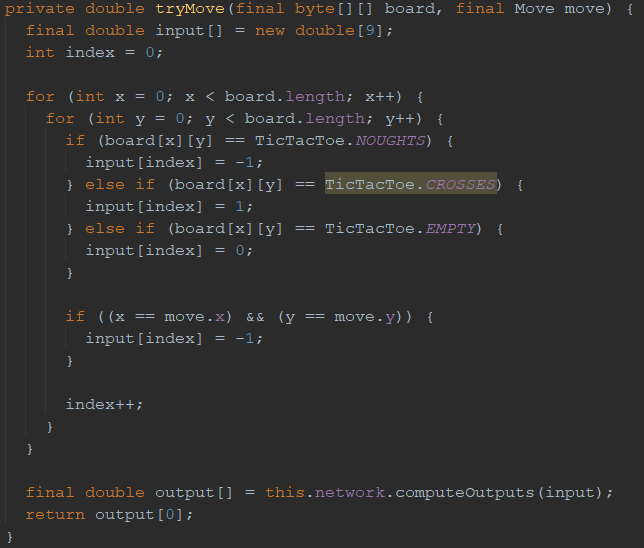


Ilustración Método tryMove

En este método es donde se calcula la entrada para un movimiento en específico y una disposición del tablero. Recordado un poco la arquitectura de la capa de entrada de la red neuronal, el autor estableció 9 neuronas de entrada correspondientes a la cantidad de casillas del tablero, por lo tanto, el autor propone la estatización de los valores de la casilla según su estado:

* Si la casilla tiene un circulo esta tiene un valor de -1
* Si la casilla tiene una tacha esta tiene un valor de 1
* Si la casilla tiene esta vacía esta tiene un valor de 0

Con estos valores establecidos el autor construye la entrada para la red neuronal a partir del estado actual del tablero, agrega el estado de la casilla del movimiento que se está evaluando. A partir de este punto el calculo de la salida se realiza de la manera en que hemos estado revisando hasta el momento.

De esta forma es en la que el jugador correspondiente a la red neuronal decide realizar una jugada. Básicamente convierte el estado del tablero en una entrada, con base a una estandarización y con esta evalúa las diferentes jugadas y la que obtenga el mayor puntaje es la que se realizara.