Tarea 3

Javier Espinoza
November 11, 2017

CC5114-1 Redes Neuronales y Programación Genética
Departamento de Ciencias de la Computación
Facultad de Ciencias Físicas y Matemáticas

1 Cómo ejecutar

Para ejecutar el script necesario, se debe correr el script *Genetic.py* que contiene una sola idea. Se crea un objeto/clase de tipo GeneticAlg, y luego se ejecuta leyendo el archivo *tictactoe.txt* para llevar a cabo el análisis y procesamiento de los datos.

2 Red Neuronal

La construcción de la red neuronal se da por una layer de inputs, 1 hidden layer y 1 ouput layer. La capa de inputs posee 9 inputs que corresponden a los 9 símbolos que pueden haber en la tabla de gato, con los primeros tres inputs siendo la primera fila, y así para los otros trios. La layer de output contiene 2, que corresponden a si gana o no el jugador X en la jugada. La hidden layer contiene 12 nodos. La visualización de esta red es:

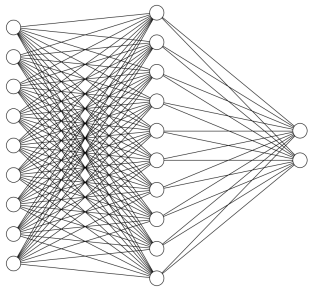

Figure 1: Red neuronal usada

2.1 Datos para entrenar

Los datos para entrenar la red neuronal se encuentran en el archivo tictactoe.py. Este archivo, al igual que el que se usó en la tarea 1, posee el siguiente formato:

```
b,b,o,o,o,x,x,x,x,positive
b,b,o,o,b,b,x,x,x,positive
b,b,o,b,o,b,x,x,x,positive
b,b,o,b,b,o,x,x,x,positive
b,b,b,x,x,x,o,o,b,positive
b,b,b,x,x,x,o,b,o,positive
b,b,b,x,x,x,b,o,o,positive
b,b,b,o,o,b,x,x,x,positive
b,b,b,o,b,o,x,x,x,positive
b,b,b,b,o,o,x,x,x,positive
x,x,o,x,x,o,o,b,o,negative
x,x,o,x,x,o,b,o,o,negative
x,x,o,x,x,b,o,o,o,negative
x,x,o,x,o,x,o,o,b,negative
x,x,o,x,o,x,o,b,o,negative
x,x,o,x,o,o,o,x,b,negative
x,x,o,x,o,o,o,b,x,negative
```

Figure 2: Ejemplo de formato de archivo. Se ve que se tiene 1 clase muchas veces, y luego se tienen los inputs de la otra. Esto hace que el aprendizaje sea incorrecto.

Se tienen los inputs en las casillas correspondientes del tictactoe, y luego se evalua si es que el jugador X gana el juego con esa jugada.

3 Algoritmo Genético

El algoritmo genético comienza con una población de 10 habitantes (extensible a más), y comienza a entrenar las redes que se encuentran en la población. Luego de este proceso (que se hace alrededor de 50 epochs), la red prueba sus resultados con los datos de prueba y guarda su precisión dentro de la red. Luego, la función fitness se encarga de revisar si es que la precisión es mayor a 0.7 (u otro valor aplicable). Si es que existe una red con esa accuracy, entonces el algoritmo termina y se entrega la red. Si no, se hacen los procesos de algoritmos genéticos correspondientes: Se seleccionan las redes que posean el mayor accuracy acumulado, y luego se mezclan estas redes formando una nueva: Con el learning rate del padre y la topología de la madre. Y usando el mutation rate, el peso de la nueva red se cambia a un valor correspondiente (la mitad del valor, el promedio entre el peso correspondiente del padre y la madre, entre otros). Luego se entrega una nueva población de redes neuronales, y se ejecuta la función fitness de nuevo hasta encontrar una red con precisión mayor a 0.7.

4 Resultados

Los resultados de esta tarea se dividen en dos: Primero el aprendizaje de las redes correspondientes a medida que el algoritmo genético se lleva a cabo, y segundo, el número de generaciones y tiempo que se demora el algoritmo para encontrar la solución que se pide.

4.1 Aprendizaje

Para mostrar los resultados del aprendizaje de la red, se tiene la curva de aprendizaje de las primeras 2 redes de la población, que luego son modificadas para tener una red más avanzada.

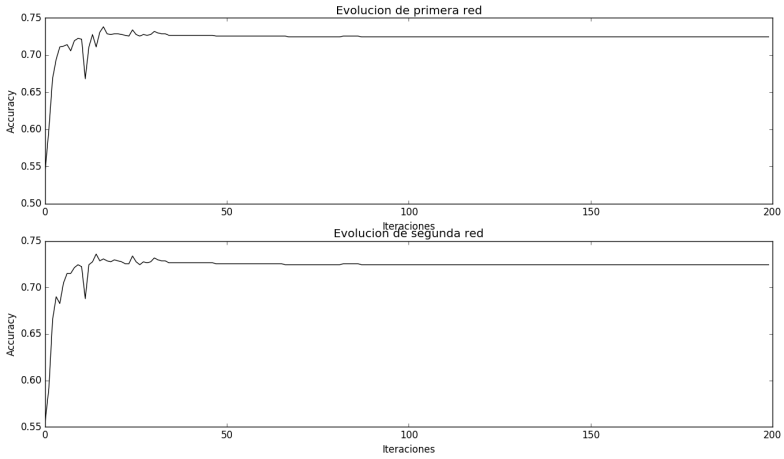

Figure 3: Curva de aprendizaje de las primeras dos redes de la población

El aprendizaje deja de crecer en el punto de 0.73, y esto se debe a que la red neuronal se queda 'estancada' y no logra aprender más allá, quedándose en un loop, lo que puede perjudicar la performance del algoritmo genético. Esto sin embargo, concuerda con las pruebas que se hicieron en la tarea 1, donde la precisión de la red alcanzaba niveles de 0.75 máximo.

4.2 Generaciones

Para evaluar la ejecución del algoritmo genético, se muestran los gráficos de evolución de generaciones para distintas precisiones pedídas además del tiempo necesitado para encontrar la solución. Esto se ve en el siguiente gráfico: Estos resultados se pueden atribuir al hecho de que las redes poseen pesos randomizados

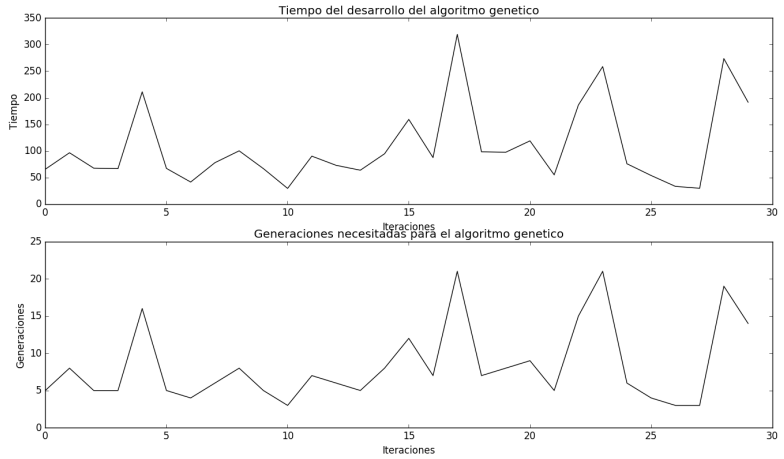

Figure 4: Tiempo en segundos y número de generaciones para 30 iteraciones del algoritmo genético

al principio, lo que hace que algunos aprendizajes sea distinto a otros (precisión puede ser muy baja al principio, causando que el algoritmo se demore mucho más en ejecutar). Otro factor puede ser la cantidad

de epochs que se le da al entrenar la red neuronal después de hacer el fitness.

5 Conclusiones

Los resultados son los esperados para lo que se había ya implementado. Era visto que en la tarea 1, la red neuronal desarrollada solo llegaba a un accuracy de 0.7, por lo que la tendencia a 0.75 es un avance de lo que ya se tenía. El tiempo y generaciones son variables debido a que en momentos donde el algoritmo genético queda estancado (tiene miembros de la población con accuracy alrededor de 0.69) se causa un gran salto en los gráficos vistos y un decremento de performance. Sin embargo, los resultados obtenidos son satisfactorios para la experiencia dada.