Algoritmos Genéticos

Sistemas de Inteligencia Artificial - ITBA

Carlos Sessa Lucas Pizzagalli Nicolás Purita

1. INTRODUCCIÓN

Se implementó un motor de algoritmo genéticos para obtener los pesos para la red neuronal construida en el Trabajo Práctico número 2. La red neuronal resuelve la función que se puede observar en la figura 1:

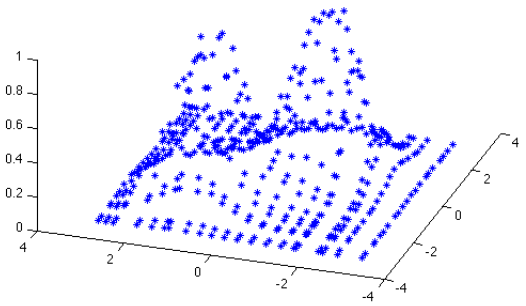

Figura 1: Distribución de puntos dada

En la figura 1 se puede observar que los puntos de la entrada pertenecen al intervalo $[-3.5, 3.5]$ y la salida se encuentra en el intervalo $(0, 1)$.
El algoritmo genético se implementó en *Java* y la red neuronal fue realizada en *Matlab*. Utilizando el *MATLAB Compiler Runtime* se realizan las pruebas pertinentes para verificar el funcionamiento de la red.

2. NUEVO OPERADOR DE CRUCE

- *Gene*: Funciona eligiendo las capas que se desean cambiar, a diferencia del clásico que toma un *locus* y a partir del mismo en adelante los cruza.

3. DESARROLLO Y PROBLEMAS ENCONTRADOS

3.1 Distintas arquitecturas

Mediante el archivo de configuración se puede seleccionar la arquitectura que se desea utilizar para realizar las pruebas. Las arquitecturas elegidas son las siguientes:

1. [30 20 10 1]
2. [10 10 1]
3. [10 1]
4. [10 10 10 10 1]
5. [40 20 1]
6. [10 10 10 1]
7. [5 10 20 1]

3.2 Representación del individuo

La representación mas óptima del individuo fue representar a toda la red como un vector de números flotantes (Double). Esta representación no se adapta a la representación de la matriz en *MATLAB*, por lo tanto se realiza una transformación del individuo. Esta transformación consiste en cambiar la cadena de Doubles a una matriz y viceversa.

3.2.1 Fidelidad del individuo

- **Completitud**: Es completa ya que se puede representar todo el dominio del problema. Esto quiere decir que se pueden utilizar todas las cadenas de números para representar la red.
- **Coherencia**: Representa unicamente el dominio del problema, ya que ninguna representación puede pertenecer al un conjunto fuera del dominio.
- **Uniformidad**: Podemos concluir que esta representación es uniforme ya que es imposible representar dos redes con la misma arquitectura de la misma forma, en caso que esto sucediera esa red es exactamente la misma.
- **Sencillez**: Es sencilla ya el individuo sabe que cantidad de neuronas que tiene en cada capa por lo tanto dividir un vector es sencillo del mismo modo que transformar una matriz en un vector.
- **Localidad**: Es local esta representación ya que al hacer pequeños cambios sobre la red, instantaneamente cambia la red.

4. RESULTADOS Y CONCLUSIONES

4.1 Contexto general

Con el fin de obtener resultados comparables entre sí se define un contexto en base para todas las pruebas. Luego en cada prueba se reemplaza el caso base con lo que se desea. A continuación se muestra el contexto base:

- Tamaño de la población es de 52 individuos.
- La brecha generacional (G) es de 0.5
- La arquitectura elegida para realizar las pruebas es la de [10 10 1].
- El operador de mutación inicial es *Clásico* con una probabilidad de 0.5 de mutar.
- El operador de cruce inicial es *Clásico* con una probabilidad de 1 de cruce.
- El operador *Backpropagation* se encuentra desactivado es decir que se ejecute con probabilidad 0.
- El método de reemplazo es *Elite*
- El método de reemplazo es *Elite* al igual que el de selección
- Hay dos condiciones de corte, ya sea por *Máxima cantidad de generaciones* y *Contenido*, donde la última verifica que si en 50 épocas no mejoro mas de un 0.01 el fitness del individuo finaliza su ejecución.

Debe tenerse en cuenta que el hecho de prefijar algunos parámetros puede hacer que los distintos métodos y operadores a comparar se vean beneficiados o perjudicados por su funcionamiento en conjunto.

4.2 Elección del operador de Cruza

4.3 Elección del operador de Mutación

4.4 Elección del operador de Selección y reemplazo