# 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:

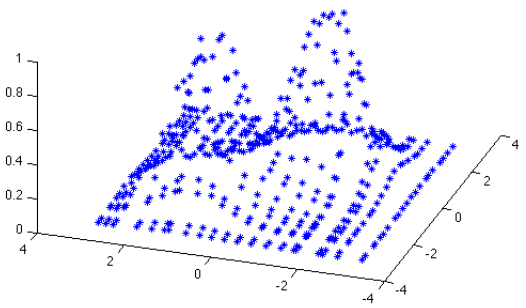


**Figura 1: Distribución de puntos dada**

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. DESARROLLO Y PROBLEMAS ENCONTRADOS

### 2.1 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.

### 2.2 Distintas arquitecturas

Mediante el archivo de configuración se puede seleccionar la arquitectura que se desea utilizar para realizar las pruebas. Las arquitecturas a elegir 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]

### 2.3 Representación del individuo

La representación más óptima del individuo fue representar a toda la red como un vector de *doubles*. 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 el vector de doubles a una matriz y viceversa.

#### *2.3.1 Fidelidad del individuo*

- **Completitud**: Es completa ya que se puede representar todo el dominio del problema.
- **Coherencia**: Representa únicamente el dominio del problema, ya que ninguna representación puede pertenecer a un conjunto fuera del dominio.
- **Uniformidad**: Se puede concluir que esta representación es uniforme ya que es imposible representar dos individuos distintos con la misma cadena, en caso que esto sucediera esa representación sería exactamente la misma que la otra.
- **Sencillez**: Convertir matriz de pesos a un vector de doubles es una operación sencilla a nivel operacional, por lo tanto se concluye que la representación es sencilla.
- **Localidad**: Es local dada que un cambio en un elemento del vector representa un cambio en el peso de la conexión que representa.

## 3. RESULTADOS Y CONCLUSIONES

## 3.1 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 selección 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* (500) y/o *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.

## 3.2 Elección del operador de Cruza

## 3.3 Elección del operador de Mutación

## 3.4 Utilización de backpropagation

## 3.5 Elección del operador de Selección y reemplazo

## 3.6 Posibles mejoras

Como posible mejora se concluye que el operador *backpropagation* debe disminuir su probabilidad de acción a medida que se alcanza un fitness deseado. El motivo por el cual se desea disminuir esa probabilidad es para que comience a tener mas influencia los operadores de mutación y cruce. El deseo de que la mutación y cruce tenga más influencia a lo largo de las generaciones es porque se observó que el fitness en un punto se asintotiza al error cuadrático medio obtenido por *backpropagation* en el Trabajo Práctico Especial 2.

| Fitness | Generación | Corte | $p_m$ |
|---|---|---|---|
| 0.22501570790437642 | 59 | Contenido | 0.05 |
| 0.21842640558999668 | 59 | Contenido | 0.1 |
| 0.22113522454399956 | 66 | Contenido | 0.3 |
| 0.22050756611447994 | 71 | Contenido | 0.5 |

**Tabla 1: Configuración simple variando la $p_m$**

| Fitness | Generación | Corte | $p_m$ | decrecimiento (%) | dec. gen. |
|---|---|---|---|---|---|
| 0.21222349890474348 | 49 | Content | 0.6 | 0.05 | 30 |
| 0.22795744299992385 | 76 | Content | 0.6 | 0.1 | 30 |
| 0.21740760693242447 | 64 | Content | 0.6 | 0.15 | 30 |

**Tabla 2: Configuración simple método de mutación No Uniforme**

| Fitness | Generación | Corte | $p_m$ | Cruce |
|---|---|---|---|---|
| 0.22501570790437642 | 59 | Contenido | 0.05 | Clásico |
| 0.20662153685787393 | 49 | Contenido | 0.05 | Gene |
| 0.21418198155450324 | 49 | Contenido | 0.05 | Multiples puntos con 2 |
| 0.21674758534799746 | 69 | Contenido | 0.05 | Uniforme con 0.3 prob |
| 0.20662153685787393 | 49 | Contenido | 0.05 | Anular |

**Tabla 3: Configuración simple variando el método de cruce**