# CAPITULO 2

# 2. Teoría de los algoritmos genéticos

* 1. **Introducción**

En este capítulo encontramos la base teórica y matemática de los algoritmos genéticos, es decir porqué funcionan, así como el algoritmo genético básico, y varias técnicas de representación para la programación de los algoritmos genéticos con todas sus posibles representaciones y sus operadores genéticos comunes y especiales.

Luego se realiza una explicación detallada del funcionamiento de las redes neuronales, autómatas celulares y los algoritmos meméticos, así como sus posibles aplicaciones, dando lugar a una comparación entre los algoritmos genéticos y otras técnicas de programación evolutiva.

* 1. **Marco teórico**
		1. **El teorema de esquemas**

Este teorema proporciona el fundamento teórico de porqué los algoritmos genéticos pueden resolver diversos problemas, y porqué se adaptan a dichos problemas, acercándose rápidamente a una posible solución. En su análisis se considera el proceso de selección y los operadores de cruce y mutación.

Un esquema se construye utilizando un nuevo símbolo ( \* ) para representar un comodín que puede aparear ambos valores, cero o uno (0 o 1). Por ejemplo, el esquema 11\*00\* representa las cadenas: 111001, 111000, 110001, 110000.

* + - 1. **El orden de los esquemas**

El orden de un esquema es el número de elementos que no son (\*) dentro del esquema. Supongamos los siguientes esquemas:

Si o(Qi) es el orden del esquema Qi, se tiene los siguientes órdenes:

El concepto de orden del esquema se utiliza para calcular la probabilidad de supervivencia del esquema con relación al operador mutación.

* + - 1. **La longitud de los esquemas**

La longitud que define a un esquema es la distancia entre la primera posición fija y la última posición fija, considerando como posiciones fijas a aquellas que contienen un elemento diferente de ( \* ). La longitud del esquema Qi, se denota por medio de ó(Qi),

El concepto anterior se utiliza para calcular la probabilidad de supervivencia del esquema frente al operador de cruce.

**El teorema dice que:** “Los esquemas pequeños de bajo orden arriba del promedio reciben un incremento exponencial de representantes en las siguientes generaciones de un algoritmo genético”. [Holland 92].

Entonces, supongamos un alfabeto binario S = {0, 1}. Para construir un esquema necesitamos ampliar el alfabeto anterior por medio del símbolo \*. Un esquema será cualquier ristra de caracteres formada a partir de elementos del alfabeto ampliado S', siendo S’ = {0,1, \*}. Si la longitud de la ristra es l, el número de esquemas existentes es 3l, ya que cada posición puede ocuparse por cualquier de los tres elementos del alfabeto extendido S’. Un esquema representa por tanto todas aquellas ristras que se asocian con él, en todas las posiciones distintas de las ocupadas por él.

Ejemplo:

Si se considera ristras de longitud 4, entonces se tiene que el esquema:

se empareja con los cromosomas:

Desde un punto de vista geométrico un esquema equivale a un hiperplano en el espacio de búsqueda.

Ahora veamos algunos conceptos que son necesarios para comprender el Teorema:

1. **n:** Numero total de individuos de una población.
2. **f(h):** Valor de la función de adaptación para un individuo concreto denominado h.
3. **f(t):** Valor medio de la función de adaptación de todos los individuos de una población en una generación t.
4. **p(t):** Representa la población en una generación t.
5. **hÎ sÇ p(t):** Expresión que indica que un individuo h pertenece tanto a la población en un instante de tiempo como al conjunto de individuos representados por el esquema s.
6. **û(s,t):** Valor medio de la función de adaptación de los individuos del esquema s en la generación t.
7. **m(s,t):** Numero de individuos del esquema s en la generación t.
8. **m(s,t+1):** Numero de individuos del esquema s en la siguiente generación.
9. **E[m(s,t+1)]:** Significa lo mismo que la expresión anterior.
10. **Pr(h)=f(h)/nf(t):** Probabilidad de que un individuo sea seleccionado en la generación t
11. **Pr(hÎ s)=û(s,t)m(s,t)/ nf(t):** Probabilidad de que un individuo del esquema s sea seleccionado
12. **E[m(s,t+1)]=û(s,t)m(s,t)/f(t):** Considerando solamente el efecto de la selección sobre la población, esta expresión nos da el valor del numero de individuos del esquema s en la siguiente generación, que será la probabilidad de que seleccionemos un individuo por el número de individuos totales de la población.

El análisis hecho arriba considera sólo el paso de selección del algoritmo genético, la combinación y la mutación deben ser considerados también. El Teorema fundamental considera únicamente la posibilidad de influencia negativa de estos operadores genéticos. El Teorema completo se enuncia a continuación, tras haber definido antes algunos conceptos:

1. **L:** Longitud de un individuo.
2. **pc:** Probabilidad de que el operador de combinación se aplique a un individuo.
3. **pm:** Probabilidad de que un bit de un individuo sea mutado.
4. **o(s):** Numero de bits definidos en el esquema s.
5. **d(s):** Distancia máxima entre dos bits definidos en el esquema s.

El primer factor de la expresión representa, como ya hemos visto, el efecto del operador de selección en la siguiente generación. El factor medio y último describen el efecto de los operadores de combinación (de un solo bit) y mutación, en particular describen la probabilidad de que un individuo concreto que es representado por el esquema s pueda seguir siendo representado por dicho esquema tras haber sido aplicado sobre el operador de combinación y mutación respectivamente.

La fórmula anterior nos proporciona una cota inferior del número esperado de individuos que se asocian con el esquema Q en la siguiente generación, expresado en función del número de individuos que se asocian con el esquema en la actual generación, así como de la adaptación del esquema relativa a la adaptación media de la población, el orden y la longitud del esquema, la longitud de las ristras, así como las probabilidades de cruce y mutación. Tal y como se desprende de la fórmula anterior, esquemas cortos, de bajo orden y con una adaptación al problema superior a la adaptación media, se espera que a medida que evoluciona el Algoritmo Genético, obtengan un incremento exponencial en el número de individuos que se asocian con los mismos.

* + 1. **Paralelismo implícito**

El hecho de que los algoritmos genéticos efectúen búsquedas robustas, se debe a que implícitamente muestrean hiperplanos. Es decir cada individuo, representado por medio de una codificación binaria, constituye un vértice del hipercubo que representa el espacio de búsqueda, y es a su vez un miembro de 2l – 1 diferentes hiperplanos, donde l es la longitud de la codificación binaria. Por otra parte existen un total de 3l – 1 hiperplanos particiones sobre el espacio de búsqueda total.

Cada individuo no proporciona gran información si se examina por separado, es por ello que la búsqueda basada en una población es crítica para los algoritmos genéticos. Una población de individuos, proporciona información sobre numerosos hiperplanos, estando los hiperplanos de bajo orden hechos un muestreo por numerosos puntos de la población. El hecho de que se muestreen muchos hiperplanos cuando se evalúa una población de ristras, se conoce como el paralelismo implícito o intrínseco de los algoritmos genéticos. Dicho paralelismo implícito, significa que muchas competiciones entre hiperplanos se efectúan simultáneamente.

* + 1. **Teoremas de convergencia**

Dado que el algoritmo genético opera con una población en cada iteración, se espera que el método converja de modo que al final del proceso la población sea muy similar, y en el infinito se reduzca a un sólo individuo.

Se ha desarrollado toda una teoría para estudiar la convergencia de estos algoritmos en el caso de cadenas binarias. Esta teoría se basa principalmente en considerar que una cadena es un representante de una clase de equivalencia o esquema, reinterpretando la búsqueda en lugar de entre cadenas, entre esquemas.

A partir de este resultado el teorema de esquemas (o teorema fundamental), prueba que la población converge a unos esquemas que cada vez son más parecidos, y en el límite a una única cadena.

En el caso de cadenas no binarios se introducen los conceptos de forma y conjunto de similitud que generalizan al de esquema. Se consideran una serie de condiciones sobre los operadores de manera que se garantice la convergencia. Básicamente se exige que al cruzar dos cadenas de la misma clase se obtenga otro dentro de ésta. Además hay que respetar ciertas condiciones sobre selección de los progenitores. Bajo toda esta serie de hipótesis se prueba la convergencia del algoritmo.

En la práctica no se suelen respetar las condiciones vistas ya que son difíciles de seguir y probar, encontrándonos con que, en ocasiones los algoritmos genéticos resuelven satisfactoriamente un problema de optimización dado y otras se quedan muy alejadas del óptimo. Los estudiosos del tema han tratado de caracterizar lo que han denominado problemas algoritmos genéticos fáciles (aquellos en los que los algoritmos genéticos proporcionan buenos resultados) y algoritmos genéticos difíciles con el objetivo de saber de antemano, al estudiar un nuevo problema, si los algoritmos genéticos son una buena elección para su resolución.

Se han tratado de caracterizar estas clases mediante el concepto de engaño considerando que si el algoritmo converge al mejor esquema (aquel con mejor promedio de aptitud de sus cadenas) y en éste se encuentra el óptimo, entonces es fácil que se resuelva satisfactoriamente. En caso de que el óptimo esté en un esquema con bajo promedio se denomina engaño y se pensaba que en estos casos es cuando el problema es Algoritmo genético difícil. Sin embargo se ha visto que esta caracterización mediante el engaño no es siempre cierta y no constituye un criterio fiable.

Rudolph demuestra la no convergencia hacia el óptimo global del algoritmo genético canónico, así como la garantía, de convergencia expresada en términos probabilísticos, del algoritmo genético que mantiene a la mejor solución en la población.

Davis y Principe extrapolan los fundamentos teóricos del algoritmo simulado de alineación a un modelo de algoritmo genético basado en cadenas de Markov. Se efectúa un estudio de las matrices de transición de estados teniendo en cuenta en primer lugar tan sólo la reproducción, a continuación la reproducción y la, mutación y finalmente la reproducción, la mutación y el cruce.

Suzuki efectúa un estudio de la convergencia de los algoritmos genéticos por medio de cadenas de Markov. Los algoritmos genéticos estudiados presentan un criterio de reducción elitista modificado, según el cual se genera una población de A individuos, incluyendo en ella al mejor individuo de la población en la generación anterior, obteniéndose los A – l individuos restantes por medio de las operaciones genéticas normales.

Liepins demuestra la convergencia del algoritmo genético hacia poblaciones que contienen al óptimo, en el caso de algoritmos genéticos sin operador de mutación; pero en los cuales el reemplazo de individuos es elitista - el mejor individuo no se pierde nunca – y además se efectúa de tal manera que en cada paso cualquier punto del espacio sea potencialmente alcanzable por medio de la operación de cruce.

Chakraborty y Dastidar presentan un modelo de fiabilidad estocástica de un esquema para el algoritmo genético binario con longitud de representación fija, y obtienen una estimación para el número de generaciones necesarias hasta obtener la convergencia.

Eiben y Col modelan la evolución del algoritmo genético por medio de una cadena de Markov, obteniendo condiciones suficientes para la convergencia en probabilidad del proceso evolutivo hacia el óptimo global.

* 1. **La estructura de un algoritmo genético**

Un algoritmo genético está descrito por tres módulos de programación:

1. **Modulo evolutivo,** Es un mecanismo de decodificación que interpreta la información de un cromosoma y tiene una función de evaluación que mide la calidad del cromosoma. Este es el único punto donde existe información del dominio.
2. **Modulo poblacional,** Este módulo tiene una representación poblacional y técnicas para manipularla. Se utilizan técnicas de representación, técnicas de arranque, criterios de selección y de reemplazo. Aquí también se define el tamaño de la población y la condición de terminación.
3. **Modulo reproductivo,** Este módulo contiene los operadores genéticos así como lo son el crossover, la mutación la selección, entre otros.
	1. Tipos de algoritmos genéticos

Los algoritmos genéticos se han aplicado a diversos problemas dentro del aprendizaje automático solucionando tareas que van de la planificación al desarrollo de topologías neuronales. De entre todas estas aplicaciones la que ha sido, a nuestro entender, más prometedora es la del aprendizaje supervisado, es decir, a partir de una serie de ejemplos (y posiblemente contraejemplos) de uno o varios conceptos, generar un conjunto de reglas que permitan, dado un individuo, decidir la clase a la cual pertenece. Las soluciones propuestas para resolver este tipo de problemas pueden caracterizarse básicamente según dos criterios:

* ¿Qué representa un individuo en la población?

En general ha habido dos respuestas a la primera de las preguntas, cada una de las cuales representa una escuela diferente en el uso de algoritmos genéticos de aprendizaje. Dichas escuelas son:

**Michigan:** En este enfoque los individuos de la población representan reglas individuales, por lo que la solución del problema consiste en toda la población.

**Pittsburgh**: La población consiste en conjuntos de reglas de cardinalidad variable, por tanto, cada individuo representa una posible solución del problema a resolver.

Dentro del primer enfoque cabe destacar los sistemas clasificadores, cuyo estudio y análisis por si mismo constituye una disciplina aparte. Todos los sistemas que siguen este enfoque tienen un problema adicional cuya resolución es crítica para su funcionamiento: como la solución del problema es toda la población ¿cómo se evalúa la bondad de sus individuos?, pues la bondad de la solución hace referencia a todos ellos.

Los sistemas que siguen la segunda tendencia siguen muy fielmente el esquema clásico de algoritmo genético ya que la población consiste en diferentes soluciones posibles que compiten entre sí.

* ¿Cómo se representan los objetos?

Existen básicamente dos tipos de representaciones, las cuales a su vez influyen en la definición de los operadores genéticos a utilizar.

**Clásicas:** Se mantiene la representación de cadenas binarias, aunque algunas veces de longitud variable.

**Ad hoc:** En este caso la representación es mucho más cercana al problema a resolver, y por ejemplo, en tareas de clasificación suele utilizarse la representación atributo valor.

En el primer caso, que podría denominarse minimalista, no deben hacerse apenas cambios sobre un algoritmo genético estándar. El escollo principal consiste en encontrar una representación binaria adecuada a nuestro problema. La gran ventaja radica en que la mayor parte de la teoría de los algoritmos genéticos se ha realizado para este tipo de representaciones.

Para el otro tipo de representaciones se deben diseñar también los operadores genéticos de forma particular (lo cual puede ser complicado) pero, al estar mucho más cercanos al problema a resolver, suelen ser más eficientes. Además, al utilizar representaciones de más alto nivel, la hibridación con otro tipo de sistema de aprendizaje es mucho más simple.

Existen varios tipos de algoritmos genéticos, cada uno basado en una metáfora distinta de la naturaleza:

* Algoritmos genéticos generacionales.
* Algoritmos genéticos de estado fijo.
* Algoritmos genéticos paralelos.
	+ 1. Algoritmos genéticos generacionales

Se asemeja a la forma de reproducción de los insectos, donde una generación pone huevos, se aleja geográficamente o muere y es substituida por una nueva. En este modelo se realizan cruces en una piscina de individuos, los descendientes son puestos en otra, al final de la fase reproductora se elimina la generación anterior y se pasa a utilizar la nueva. Este modelo también es conocido como algoritmo genético canónico.

* + 1. Algoritmos genéticos de estado fijo

Utilizan el esquema generacional de los mamíferos y otros animales de vida larga, donde coexisten padres y sus descendientes, permitiendo que los hijos sean educados por sus progenitores, pero también que a la larga se genere competencia entre ellos.

En este modelo, no sólo se debe seleccionar los dos individuos a ser padres, si no también cuales de la población anterior serán eliminados, para hacer espacio a los descendientes.

La diferencia esencial entre el modelo de reemplazo generacional y el modelo de estado fijo es que las estadísticas de la población son recalculadas luego de cada cruce y los nuevos descendientes están disponibles inmediatamente para la reproducción. Esto permite al modelo utilizar las características de un individuo prometedor tan pronto como es creado.

Algunos autores dicen que este modelo tiende a evolucionar mucho más rápido que el modelo generacional [BD93], sin embargo entre las investigaciones de Goldberg y Deb (1993) se encontró que las ventajas parecen estar relacionadas con la alta tasa de crecimiento inicial, ellos dicen que los mismos efectos pueden ser obtenidos en rangos de adaptación exponencial o selección por competencia. No encontraron evidencia que este modelo sea mejor que el generacional.

* + 1. Algoritmos genéticos paralelos

Estos algoritmos son denominados también algoritmos meméticos. Parte de la metáfora biológica que motivó a utilizar la búsqueda genética consiste en que es inherentemente paralela, donde al evolucionar se recorren simultáneamente muchas soluciones, cada una representada por un individuo de la población. Sin embargo, es muy común en la naturaleza que no sólo sea una población evolucionando, si no varias poblaciones, normalmente aisladas geográficamente, que originan respuestas diferentes a la presión evolutiva. Esto origina dos modelos que toman en cuenta esta variación, y utilizan no una población como los anteriores sino múltiples concurrentemente.

Una tendencia relativamente nueva en el campo de los algoritmos genéticos es la incorporporación de técnicas de búsqueda local. Esta aplicación tras una increíble mejora en algunos resultados que fueron desalentadores utilizando los algoritmos genéticos cuando han sido aplicados en problemas específicos, especialmente en los más difíciles de optimización combinatoria, por esta razón los investigadores comenzaron a buscar nuevas maneras para extender los algoritmos genéticos y obtener mejores resultados en períodos de tiempo menores.

Un paso importante en esta dirección fue dado por Muhlenbein con un desenvolvimiento de algoritmos genéticos paralelos (PGA’s) que permiten que los individuos en una población mejoren sus disposiciones mediante mejoras iterativas (conocidas como alpinismo de la colina). Los algoritmos genéticos paralelos también permiten que los individuos se seleccionen entre ellos por procesos locales, o que faciliten su implantación en sistemas locales.

De cierto modo, los algoritmos genéticos simples con búsqueda local pueden ser vistos cómo métodos sofisticados de descenso múltiple. El proceso de reiniciación está en este caso comandado por reglas genéticas, y la fase de descenso está encerrada cómo de traje. El éxito de los métodos puede ser atribuido a su equilibrio entre tener una búsqueda rápida y mantener una diversidad para evitar la convergencia prematura. Otro método puede consistir en desenvolver una búsqueda iterativa llamada codificación Delta propuesta por D. Whitley y sus colaboradores. La codificación Delta introdujo diversidad, generando aleatoriamente una nueva población, no en tanto preserva la información de las generaciones previas basándose en las nuevas codificaciones y en soluciones parciales previas.

* + - 1. **Modelos de islas**

Si se tiene una población de individuos, esta se divide en subpoblaciones que evolucionan independientemente como un algoritmo genético normal. Ocasionalmente, se producen migraciones entre ellas, permitiéndoles intercambiar material genético.

Con la utilización de la migración, este modelo puede explotar las diferencias en las subpoblaciones; esta variación representa una fuente de diversidad genética. Sin embargo, si emigra un gran número de individuos en cada generación, ocurre una mezcla global y se eliminan las diferencias locales, y si la migración es infrecuente, es probable que se produzca convergencia prematura en las subpoblaciones.

* + - 1. **Modelo celular**

Coloca cada individuo en una matriz, donde cada uno sólo podrá buscar reproducirse con los individuos que tenga a su alrededor (más cerca de casa) escogiendo al azar o al mejor adaptado. El descendiente pasará a ocupar una posición cercana.

No hay islas en este modelo, pero hay efectos potenciales similares. Asumiendo que el cruce esta restringido a individuos adyacentes, dos individuos separados por 20 espacios están tan aislados como si estuvieran en dos islas, este tipo de separación es conocido como aislamiento por distancia.

Luego de la primera evaluación, los individuos están todavía distribuidos al azar sobre la matriz. Posteriormente, empiezan a emerger zonas con cromosomas y adaptaciones semejantes. La reproducción y selección local crea tendencias evolutivas aisladas, luego de varias generaciones, la competencia local resultará en grupos más grandes de individuos semejantes.

# El algoritmo genético básico

Inicio

 Genere la población inicial

Halle su genotipo

Halle la función de adaptación

Mientras < no converge >

 Seleccione los de padres

 Realice el crossover

 Realice la mutación

 Actualice la población

Halle su genotipo

 Halle la función de adaptación

Fin\_Mientras

Muestre la solución optima

Fin

El algoritmo genético básico

Cuadro 2.1

El algoritmo es realmente sencillo y fácil de comprender, y debe recordarse que la parte más importante se desea llegar a una solución es la función de adaptación que se escoja, ya que de esta dependerá también una correcta selección de padres.

***Diagrama de flujo del algoritmo genético básico***

***Figura 2.1***

* + 1. **La población inicial**

La población inicial consta de n individuos, que conforman una solución inicial. Por lo general esta población es generada mediante cadenas de símbolos o caracteres, para luego ser procesados.

Los tipos más comunes de codificación son:

* Codificación Binaria
* Codificación por permutaciones
* Codificación por símbolos
* Codificación mediante árboles
	+ - 1. **Codificación binaria**

La codificación en base binaria es la más común de todas, ya que luego es sencillo llevarlo a base decimal, para obtener una función de adaptación.

Evidentemente los cromosomas estarán formados por cadenas de ceros y unos. Esto permitirá representar un intervalo [a, b), en donde se tomarán n puntos que serán números binarios de una longitud ideal de logaritmo en base dos de n (log 2 n).

|  |  |
| --- | --- |
| Cromosoma A | 101100101100101011100101 |
| Cromosoma B | 111111100000110000011111 |

Ejemplo de dos cromosomas en codificación binaria

Cuadro 2.2

Estos cromosomas tienen una longitud 24, por lo que con ellos se puede representar 224 números, es decir 16’777.216 números.

* + - 1. **Codificación por permutaciones**

Este tipo de codificación puede ser utilizado en problemas de ordenamiento, y sus números representan una secuencia.

|  |  |
| --- | --- |
| Cromosoma A | ( 1 5 3 2 6 4 7 9 8 ) |
| Cromosoma B | ( 8 5 6 7 2 3 1 4 9 ) |

Ejemplo de dos cromosomas en la representación por permutaciones

Cuadro 2.3

Un problema en el que se puede aplicar esta técnica de codificación es el TSP, que consiste en la optimización de una ruta.

* + - 1. **Codificación por valores**

En este caso se atribuyen diferentes valores a cada cromosoma, que pueden ser reales, alfabéticos, o símbolos especiales.

|  |  |
| --- | --- |
| Cromosoma A | 1.2324 5.3243 0.4556 2.3293 2.4545 |
| Cromosoma B | ABDJEIFJDHDIERJFDLDFLFEGT |
| Cromosoma C | (abajo), (abajo), (derecha), (arriba), (izquierda) |

Ejemplo de tres cromosomas en codificación por valores

Cuadro 2.4

En realidad este tipo de codificación suele ser bastante buena en algunos problemas.

* + - 1. **Codificación mediante árboles**

En este tipo de codificación cada árbol representará un cromosoma, y las ventajas es que con un buen manejo de árboles, la programación se facilitará muchísimo.

Por ejemplo:

***Ejemplo de un cromosoma de la codificación por árboles***

***Figura 2.2***

* + 1. **El genotipo**

El genotipo de cada cromosoma, representa el valor decimal de la cadena de símbolos o caracteres generados. Para cierto tipo de problemas, este genotipo debe ser llevado a un intervalo [a, b) mediante un pequeño cambio de variable, de tal manera que se representen solamente los valores que pertenecen a dicho intervalo.

* + 1. **La función de adaptación.**

Una gran ventaja de los algoritmos genéticos sobre las otras técnicas de optimización es que la función de adaptación puede ser la misma función que deseamos optimizar, y no se necesita utilizar ningún otro conocimiento matemático tal como derivadas, inversas, o cualquier tipo de artificio para luego tener una adaptación al problema.

Dos aspectos que resultan cruciales en el comportamiento de los algoritmos genéticos son la determinación de una adecuada función de adaptación o función objetivo, así como la codificación utilizada.

La función de adaptación es de muchísima importancia para el algoritmo genético, ya que de esta depende que el problema pueda o no resolverse, y aunque parece sencillo manipular una expresión que será la misma que se ha planteado, se presentará un inconveniente cuando no se pueda utilizar una función de adaptación, porque esta no se conoce.

Idealmente nos interesaría construir funciones objetivo con ciertas regularidades, es decir funciones objetivo que verifiquen que para dos individuos que se encuentren cercanos en el espacio de búsqueda, sus respectivos valores en las funciones objetivo sean similares. Por otra parte una dificultad en el comportamiento del algoritmo genético puede ser la existencia de gran cantidad de óptimos locales, así como el hecho de que el óptimo global se encuentre muy aislado.

La regla, general para construir una buena función objetivo es que ésta debe reflejar el valor del individuo de una manera real; pero en muchos problemas de optimización combinatoria, donde existe gran cantidad de restricciones, buena parte de los puntos del espacio de búsqueda representan individuos no válidos.

Para este planteamiento en el que los individuos están sometidos a restricciones, se han propuesto varias soluciones. La primera sería la que se podría denominar absolutista, en la que aquellos individuos que no verifican las restricciones, no son considerados como tales, y se siguen efectuando cruces y mutaciones hasta obtener individuos válidos, o bien, a dichos individuos se les asigna una función objetivo igual a cero.

Otra posibilidad consiste en reconstruir aquellos individuos que no verifican las restricciones. Dicha reconstrucción suele llevarse a cabo por medio de un nuevo operador que se acostumbra a denominar reparador.

Otro enfoque está basado en la penalización de la función objetivo. La idea general consiste en dividir la función objetivo del individuo por una cantidad (la penalización) que guarda relación con las restricciones que dicho individuo viola. Dicha cantidad puede simplemente tener en cuenta el número de restricciones violadas ó bien el denominado costo esperado de reconstrucción, es decir el coste asociado a la conversión de dicho individuo en otro que no viole ninguna restricción.

Otra técnica que se ha venido utilizando en el caso en que la computación de la función objetivo sea muy compleja es la denominada evaluación aproximada de la función objetivo. En algunos casos la obtención de n funciones objetivo aproximadas puede resultar mejor que la evaluación exacta de una única función objetivo (supuesto el caso de que la evaluación aproximada resulta como mínimo n veces más rápida que la, evaluación exacta).

Un problema habitual en las ejecuciones de los algoritmos genéticos surge debido a la velocidad con la que el algoritmo converge. En algunos casos la convergencia es muy rápida, lo que suele denominarse convergencia prematura, en la cual el algoritmo converge hacia óptimos locales, mientras que en otros casos el problema es justo el contrario, es decir se produce una convergencia lenta del algoritmo. Una posible solución a estos problemas pasa por efectuar transformaciones en la función objetivo. El problema de la convergencia prematura, surge a menudo cuando la selección de individuos se realiza de manera proporcional a su función objetivo. En tal caso, pueden existir individuos con una adaptación al problema muy superior al resto, que a medida que avanza el algoritmo dominan a la población. Por medio de una transformación de la función objetivo, en este caso una comprensión del rango de variación de la función objetivo, se pretende que dichos superindividuos no lleguen a dominar a la población.

El problema de la lenta convergencia del algoritmo, se resolvería de manera análoga, pero en este caso efectuando una expansión del rango de la función objetivo.

La idea de especies de organismos, ha sido imitada en el diseño de los algoritmos genéticos en un método propuesto por Goldberg y Richardson, utilizando una modificación de la función objetivo de cada individuo, de tal manera que individuos que estén muy cercanos entre sí devalúen su función objetivo, con objeto de que la población gane en diversidad.

* + 1. **La selección de padres**

La selección de padres es un proceso donde se escogerá mediante algún método los padres más aptos, simulando la ley de supervivencia de los individuos más aptos.

Este proceso es de gran importancia, ya que de una buena selección de padres, bajo un criterio lo suficientemente bueno, se realizará una convergencia más rápida, hacia una solución más óptima.

Los tipos más comunes de métodos o funciones de selección son:

* Modelo de selección proporcional a la función objetivo.
* Modelo de selección proporcional al rango.
* Modelo de selección del valor esperado.
* Modelo de muestreo con remplazo del resto.
* Modelo universal estocástico.
* Modelo de la rueda de la ruleta.
* Modelo del Torneo.
* Modelo de mantener el estado.
	+ - 1. **Modelo de la selección proporcional a la función objetivo.**

En esta función cada individuo tiene una, probabilidad de ser seleccionado como padre que es proporcional al valor de su función objetivo.

Denotemos por P( j ) la probabilidad de que el individuo j sea seleccionado como padre y llamemos g( j ) a la función de adaptación del problema para cualquier individuo j. Entonces podemos decir que:

Esta función de selección es invariante ante un cambio de escala, pero no ante una traslación.

* + - 1. **Modelo de la selección proporcional al rango.**

Se produce una repartición más uniforme de la probabilidad de selección. Denotemos por R(g( j ) ) al rango de la función objetivo para el individuo j. Entonces se puede decir que:

Donde el denominador es la suma de los rangos de todos los individuos. La función de selección basada en el rango es invariante frente a la translación y al cambio de escala.

* + - 1. **Modelo de la selección del valor esperado.**

Para cada individuo SI, se introduce un contador, inicialiazado en g( j )/gt, donde, gt denota la media, de la función objetivo en la generación t. Cada vez que el individuo ( j ) es seleccionado para el cruce, dicho contador decrece en una cantidad c (c pertenece a (0, 5;.1)). El individuo en cuestión dejará de poder ser seleccionado en esa generación, cuando su contador sea negativo.

* + - 1. **Modelo de muestreo con remplazo del resto.**

Cada individuo es seleccionado un número de veces que coincide con la parte entera del número esperado de ocurrencias de dicho suceso, compitiendo los individuos por los restos. Es decir que si denotamos por m(j) el número de veces que el individuo j es seleccionado para el cruce, tenemos que:

* + - 1. **Modelo del muestreo universal estocástico.**

Efectuando un paralelismo con los métodos de muestreo estadísticos, este último tipo de selección de padres se relaciona con el muestreo sistemático, mientras que la selección proporcional a la función objetivo, está basada en el muestreo estratificado con fijación proporcional al tamaño. También el procedimiento de selección que hemos denominado muestreo estocástico con reemplazo del resto, mantiene un paralelismo con el muestreo estratificado con fijación de compromiso.

* + - 1. **Modelo de la rueda de la ruleta.**

Es un método muy utilizado, ya que puede realizárselo mediante pocos pasos muy fáciles de comprender y aplicar:

Gráfico de la selección de la rueda de la ruleta

Figura 2.3

INICIO

1. Halle la suma de las funciones de aptitud
2. Halle la frecuencia relativa a la suma
3. Halle la frecuencia acumulada
4. Genere un número aleatorio con distribución U(0,1)
5. I=1
6. Si el Número es menor que la I-ésima frecuencia acumulada, entonces seleccionamos el I-ésimo cromosoma.

Ir a i.

1. I=I+1
2. Ir a f.
3. FIN

Algoritmo para realizar el método de selección de la rueda de la ruleta

Cuadro 2.5

El problema es en cuanto a la disposición cuando ésta difiere mucho. Por ejemplo, si la aptitud del cromosoma mejor es 90% entonces de toda la rueda de la ruleta los otros cromosomas tendrán muy pocas oportunidades ser seleccionado.

* + - 1. **Modelo del torneo**

Se asigna un rango a las líneas de la población y entonces cada cromosoma recibe aptitud de esta clasificación jerárquica. El peor tendrá aptitud 1, segundo peor 2, etc. y los más buenos tendrán aptitud N (el número de cromosomas en población).

Después de esto todos los cromosomas tienen una oportunidad similar de ser seleccionados. Pero este método puede llevar a la convergencia más lenta, porque los cromosomas más buenos no difieren tanto de otro.

* + - 1. **Modelo de mantener el estado.**

Éste no es un método particular de seleccionar a los padres. La idea principal de esta selección es que una gran parte de cromosomas debe sobrevivir a la próxima generación.

Los algoritmos genéticos entonces trabajan de la siguiente manera. En cada generación se selecciona unos (bueno - con aptitud alta) los cromosomas por crear una nueva descendencia. Entonces algunos (malo - con aptitud baja) los cromosomas están alejados y la nueva descendencia se pone en su lugar. El resto de población sobrevive a la nueva generación.

* + - 1. **Elitismo**

La idea de elitismo ya se ha introducido. Al crear nueva población por crossover y mutación, nosotros tenemos una gran oportunidad, de soltar el mejor cromosoma.

Elitismo es nombre del método que primero copia el mejor cromosoma (o unos cromosomas más buenos) a la nueva población. El resto se hace de manera clásica.

El elitismo encausa aumento de eficiencia del Algoritmo Genético, porque previene la pérdida de la solución que se encontró.

Una posible clasificación de procedimientos de selección de padres consistirá en: métodos de selección dinámicos, en los cuales las probabilidades de selección varían de generación a generación, (por ejemplo la selección proporcional a la función objetivo), frente a métodos de selección estáticos, en los cuales dichas probabilidades permanecen constantes (por ejemplo la selección basada en rangos).

Si se asegura que todos los individuos tienen asignada una probabilidad de selección distinta de cero el método de selección se denomina preservativo. En caso contrario se acostumbra a denominarlo extintivo.

* 1. **Crossover**

El crossover y la mutación son dos operadores básicos de los algoritmos genéticos. La actuación de los algoritmos genéticos depende mucho de ellos. El tipo y la aplicación de estos operadores dependen de la codificación en un problema.

* + 1. **Crossover de la codificación Binaria**

Las formas más conocidas de crossover para codificación binaria son:

* Crossover en un solo punto.
* Crossover en n puntos.
* Crossover Uniforme.
* Crossover matemático.
* Máscara de crossover
	+ - 1. **Crossover en un punto**

|  |  |  |
| --- | --- | --- |
|  | **Cromosoma 1** | **Cromosoma 2** |
| **Padres** | **11001010** | 10011011 |
| **Hijos** | **1100**1011 | 1001**1010** |

***Ejemplo del crossover en un punto de la representación binaria***

***Cuadro 2.6***

En este tipo de Crossover se elige aleatoriamente un punto, y se toman los bits desde el inicio hasta dicho punto de un cromosoma, y los demás del otro cromosoma. El crossover en un punto puede generalizarse a n puntos.

* + - 1. **Crossover Uniforme**

|  |  |  |
| --- | --- | --- |
|  | **Cromosoma 1** | **Cromosoma 2** |
| **Padres** | **11001010** | 10011011 |
| **Hijos** | **1**00**01**011 | 1**10**11**010** |

***Ejemplo del crossover uniforme en la representación binaria***

***Cuadro 2.7***

En este Crossover cada bit recibe igual probabilidad de ser seleccionado de los padres en cualquiera de los dos hijos, de manera similar a utilizar una máscara de crossover.

* + - 1. **Crossover matemático**

En el siguiente ejemplo de crossover matemático, el primer hijo nace a partir del operador de conjunción ( ∧ ), mientras que el segundo hijo nace a partir del operador de disyunción ( ∨ ).

|  |  |  |
| --- | --- | --- |
|  | **Cromosoma 1** | **Cromosoma 2** |
| **Padres** | **11001010** | 10011011 |
| **Hijos** | **1**00**01**011 | 1**10**11**010** |

Ejemplo de crossover matemático en la representación binaria

***Cuadro 2.8***

En este tipo de Crossover, cada hijo será el resultado de una operación de lógica matemática, la que puede ser: ∧,∨,⇒,⇔,⇐.

* + - 1. **Máscara de crossover.**

Cuando se utiliza el denominado crossover binario uniforme, se puede trabajar con una máscara de crossover, la cual atribuye de antemano las posiciones a tomar para la descendencia, es decir, que la máscara de crossover también será una cadena binaria, con la condición que si en esta aparece el valor de uno, entonces el gen se tomará del primer padre, mientras que si aparece el valor de cero, el gen se tomará del segundo padre.

* + 1. **Crossover de la codificación por permutaciones**

Básicamente se trabaja con el crossover en un solo punto, aunque podría ser diferente que en la representación binaria, la selección de la segunda parte del cromosoma.

* + - 1. **Crossover en un solo punto**

Se selecciona un punto del crossover, entonces hacemos que hasta este punto de la permutación se copie del primer padre, mientras que el segundo padre se examina y si el número no está todavía en la descendencia que se agrega.

Por ejemplo:



***Ejemplo de crossover en un solo punto de la representación por permutaciones***

***Cuadro 2.9***

* + 1. **Crossover de la codificación por valores**

Para este tipo de Crossover, pueden utilizarse todos los métodos del crossover binario, es decir crossover en un punto, crossover en n puntos o el crossover uniforme.

* + - 1. **Crossover en un solo punto**

|  |  |  |
| --- | --- | --- |
|  | **Cromosoma 1** | **Cromosoma 2** |
| **Padres** | (**1.29 5.68 2.86 4.11 5.55**) | (1.32 5.24 2.75 4.22 5.55) |
| **Hijos** | (**1.29 5.68 2.86** 4.22 5.55) | (1.32 5.24 2.75 **4.11 5.55**) |

Ejemplo de crossover en un solo punto de la representación por valores

Cuadro 2.10

En este tipo de Crossover se elige aleatoriamente un punto, y se toman los bits desde el inicio hasta dicho punto de un cromosoma, y los demás del otro cromosoma.

* + - 1. **Crossover en n puntos**

Este Crossover es una generalización del Crossover en un solo punto, se selecciona n posiciones para realizar el intercambio de bits.

|  |  |  |
| --- | --- | --- |
|  | **Cromosoma 1** | **Cromosoma 2** |
| **Padres** | (**1.29 5.68 2.86 4.11 5.55**) | (1.32 5.24 2.75 4.22 5.55) |
| **Hijos** | (**1.29 5.68** 2.754.22 **5.55**) | (1.32 5.24 **2.86 4.11** 5.55) |

***Ejemplo de crossover en n puntos de la representación por valores***

***Cuadro 2.11***

* + - 1. **Crossover uniforme**

En este tipo de Crossover, cada posición tiene igual probabilidad de ser seleccionada de uno u otro cromosoma, es un crossover muy similar al de utilizar la denominada máscara de crossover.

|  |  |  |
| --- | --- | --- |
|  | **Cromosoma 1** | **Cromosoma 2** |
| **Padres** | (**1.29 5.68 2.86 4.11 5.55**) | (1.32 5.24 2.75 4.22 5.55) |
| **Hijos** | (**1.29** 5.682.754.22 **5.55**) | (1.32 **5.24** **2.86 4.11** 5.55) |

***Ejemplo del crossover uniforme en la representación por valores***

***Cuadro 2.12***

* + 1. **Crossover de la codificación mediante árboles**

Se realiza un corte en alguna arista de los árboles, y se toma la parte izquierda de un padre, y la derecha del otro.

Por ejemplo:

 + =

Ejemplo de crossover en la representación mediante arboles

Figura 2.4

* 1. **Mutación**

En la Evolución, una mutación es un suceso poco común, como ya se ha visto anteriormente. En la mayoría de los casos las mutaciones son letales, pero en promedio, contribuyen a la diversidad genética de la especie. En un algoritmo genético tendrán el mismo papel, y la misma frecuencia (es decir, muy baja).

Una vez establecida la frecuencia de mutación, por ejemplo, uno por mil, se examina cada bit de cada cadena cuando se vaya a crear la nueva criatura a partir de sus padres (normalmente se hace de forma simultánea al crossover).

La gran ventaja real de la mutación es que no permitirá estancarse al algoritmo genético en una solución local, sino que permitirá dar saltos hacia otros puntos de la función, permitiendo de este modo una búsqueda global.

Además, no hace falta decir que no conviene abusar de la mutación. Es cierto que es un mecanismo generador de diversidad, y, por tanto, la solución cuando un algoritmo genético está estancado; pero también es cierto que reduce el algoritmo genético a una búsqueda aleatoria, lo cual es bastante lento. Siempre es más conveniente usar otros mecanismos de generación de diversidad, como aumentar el tamaño de la población, o garantizar la aleatoriedad de la población inicial.

* + 1. **Mutación en codificación binaria.**

Los tipos más comunes de mutación en este caso son:

* Mutación en un punto.
* Mutación en n puntos.
* Mutación uniforme.
	+ - 1. **Mutación en n puntos.**

Debe tenerse claro el número de puntos a mutar, y si siempre mutará o no para cada punto. La mutación transformará el bit de 0 a 1 o de 1 a 0 según el caso.

|  |  |  |
| --- | --- | --- |
| Cromosoma antes de mutar |  | Cromosoma después de mutar |
| 1**1**0**0**1**0**01 | 🡺 | 1**0**0**1**1**0**01 |

***Ejemplo de mutación en n puntos de la representación binaria***

***Cuadro 2.13***

* + - 1. **Mutación Uniforme**

Cada bit tendrá una diferente probabilidad para mutar; pero en todos los casos se comparará con la misma que se ingresó.

|  |  |  |
| --- | --- | --- |
| Cromosoma antes de mutar |  | Cromosoma después de mutar |
| 1**1**001001 | 🡺 | 1**01**0100**0** |

***Ejemplo de mutación uniforme de la representación binaria***

Cuadro 2.14

* + 1. **Mutación en codificación por permutaciones.**

La más habitual es la de cambio de orden de las posiciones.



***Ejemplo de mutación en la representación por permutaciones***

Cuadro 2.15

* + 1. **Mutación en codificación por valores.**

Una forma muy utilizada es la adición y substracción de una pequeña cantidad real a dos posiciones seleccionadas.



***Ejemplo de mutación en la representación por valores***

Cuadro 2.16

* + 1. **Mutación en codificación mediante árboles**

Una forma de realizar la mutación en árboles es cambiando el operador o un número de éste.

 🡺

Antes de mutar Después de mutar

# *Ejemplo de la mutación en la representación por árboles*

# *Figura 2.5*

* 1. **Otros operadores**

No se usan en todos los problemas, sino sólo en algunos, y en principio su variedad es infinita. Generalmente son operadores que exploran el espacio de soluciones de una forma más ordenada, y que actúan más en las últimas fases de la búsqueda, en la cual se pasa de soluciones "casi buenas" a "buenas" soluciones.

* + 1. **Cromosomas de longitud variable**

Hasta ahora se han descrito cromosomas de longitud fija, donde se conoce de antemano el número de parámetros de un problema; pero hay problemas en los que esto no sucede. Por ejemplo, en un problema de clasificación, donde dado un vector de entrada, queremos agruparlo en una serie de clases, puede ser que no sepamos siquiera cuantas clases existen.

En diseño de redes neuronales, puede que no se sepa (de hecho, nunca se sabe) cuántas neuronas se van a necesitar. Por ejemplo, en un perceptrón hay reglas que dicen cuantas neuronas se deben de utilizar en la capa oculta; pero en un problema determinado puede que no haya ninguna regla heurística aplicable; tendremos que utilizar los algoritmos genéticos para hallar el número óptimo de neuronas.

En estos casos, necesitamos dos operadores más: añadir y eliminar. Estos operadores se utilizan para añadir un gen, o eliminar un gen del cromosoma. La forma más habitual de añadir un locus es duplicar uno ya existente, el cual sufre mutación y se añade al lado del anterior. En este caso, los operadores del algoritmo genético simple (selección, mutación, crossover) funcionarán de la forma habitual, salvo, claro está, que sólo se haga el crossover en la zona del cromosoma de menor longitud.

Estos operadores permiten, además, crear un algoritmo genético de dos niveles:

* Al nivel de cromosomas.
* Al nivel de genes.

Supongamos que, en un problema de clasificación, hay un gen por clase. Se puede asignar una puntuación a cada gen en función del número de muestras que haya clasificado correctamente. Al aplicar estos operadores, se duplicarán los aleles con mayor puntuación, y se eliminarán aquellos que hayan obtenido menor puntuación, o cuya puntuación sea nula.

* + 1. **Operadores de nicho (ecológico).**

Otros operadores importantes son los operadores de nicho. Estos operadores están encaminados a mantener la diversidad genética de la población, de forma que cromosomas similares sustituyan sólo a cromosomas similares, y son especialmente útiles en problemas con muchas soluciones; un algoritmo genético con estos operadores es capaz de hallar todos los máximos de manera global y local, dedicándose cada especie a un máximo. Más que operadores genéticos, son formas de enfocar la selección y la evaluación de la población.

Los operadores de nicho más utilizados son:

* **Crowding o apiñamiento.**

Este operador trata de mantener la diversidad genética de la población, combinándose solamente entre cromosomas similares.

* **Discriminación o sharing.**

Consiste en introducir una función de discriminación que indica cuán similar es un cromosoma al resto de la población. La puntuación de cada individuo se dividirá por esta función de discriminación, de forma que se facilita la diversidad genética y la aparición de individuos diferentes.

* + 1. **Operadores especializados**

En una serie de problemas hay que restringir las nuevas soluciones generadas por los operadores genéticos, pues no todas las soluciones generadas van a ser válidas, sobre todo en los problemas con restricciones. Por ello, se aplican operadores que mantengan la estructura del problema. Otros operadores son simplemente generadores de diversidad, pero la generan de una forma determinada.

Los operadores especializados más conocidos son:

* **Zap**

**E**n lugar de cambiar un solo bit de un cromosoma, lo cambia completamente.

* **Creep**

Este operador aumenta o disminuye en 1 el valor de un gen; sirve para cambiar suavemente y de forma controlada los valores de los genes.

* **Transposición**

**Es** similar al crossover y a la recombinación genética, pero dentro de un solo cromosoma. Dos genes intercambian sus valores, sin afectar al resto del cromosoma. Similar a este es el operador de **eliminación-reinserción**, en el que un gen cambia de posición con respecto a los demás.

* 1. La población
		1. **Técnicas basadas en población**

Este tipo de técnicas pueden ser versiones de cualquiera de las anteriores, pero en vez de tener una sola solución, que se va alterando hasta obtener el óptimo, se persigue el óptimo cambiando varias soluciones; de esta forma es más fácil escapar de los mínimos locales tan temidos. Entre estas técnicas se hallan la mayoría de los algoritmos evolutivos.

* + 1. Tamaño de la población

Una cuestión que uno puede plantearse es la relacionada con el tamaño idóneo de la población. Parece intuitivo que las poblaciones pequeñas corren el riesgo de no cubrir adecuadamente el espacio de búsqueda, mientras que el trabajar con poblaciones de gran tamaño puede acarrear problemas relacionados con el excesivo costo computacional.

Goldberg efectuó un estudio teórico, obteniendo como conclusión que el tamaño óptimo de la población para ristras de longitud I, con codificación binaria, crece exponencialmente con el tamaño de la ristra.

Este resultado traería como consecuencia que la aplicabilidad de los algoritmos genéticos en problemas reales sería muy limitada, ya que resultarían no competitivos con otros métodos de optimización combinatoria. Alander, basándose en evidencia empírica sugiere que un tamaño de población comprendida entre l y 21 es suficiente para atacar con éxito los problemas considerados por él.

* + 1. La población inicial

Habitualmente la población inicial se escoge generando ristras al azar, pudiendo contener cada gen a uno de los posibles valores del alfabeto con probabilidad uniforme. Nos podríamos preguntar que es lo que sucedería si los individuos de la población inicial se obtuviesen como resultado de alguna técnica heurística o de optimización local. En los pocos trabajos que existen sobre este aspecto, se constata que esta inicialización no aleatoria de la población inicial, puede acelerar la convergencia del algoritmo genético. Sin embargo en algunos casos la desventaja resulta ser la prematura convergencia del algoritmo, queriendo indicar con esto la convergencia hacia óptimos locales.

* 1. **Otras técnicas de programación evolutiva**
		1. **Redes neuronales**

Considero importante comparar el funcionamiento de los algoritmos genéticos con otras técnicas informáticas robustas en optimización, por lo que en esta etapa se explica un poco sobre las redes neuronales.

##### La neurona

A finales del siglo 19 se logró una mayor claridad sobre el trabajo del cerebro debido a los trabajos de Ramón y Cajal en España y Sherrington en Inglaterra. El primero trabajó en la anatomía de las neuronas y el segundo en los puntos de conexión de las mismas o sinápsis.

El tejido nervioso es el más diferenciado del organismo y está constituido por células nerviosas, fibras nerviosas y la neuroglia, que está formada por varias clases de células. La célula nerviosa se denomina neurona, que es la unidad funcional del sistema nervioso. Hay neuronas bipolares, con dos prolongaciones de fibras y multipolares, con numerosas prolongaciones. Estas pueden ser neuronas sensoriales, motoras y de asociación.

Se estima que en cada milímetro del cerebro hay cerca de 50.000 neuronas. El tamaño y la forma de las neuronas es variable, pero con las mismas subdivisiones que muestra la figura. El cuerpo de la neurona o Soma contiene el núcleo. Se encarga de todas las actividades metabólicas de la neurona y recibe la información de otras neuronas vecinas a través de las conexiones sinápticas.

Las dendritas son las conexiones de entrada de la neurona y por su parte el axón es la "salida" de la neurona y se utiliza para enviar impulsos o señales a otras células nerviosas. Cuando el axón esta cerca de sus células destino se divide en muchas ramificaciones que forman sinápsis con el soma o axones de otras células. Esta unión puede ser inhibidora o excitadora según el transmisor que las libere. Cada neurona recibe de 10.000 a 100.000 sinápsis y el axón realiza una cantidad de conexiones similar.

La transmisión de una señal de una célula a otra por medio de la sinápsis es un proceso químico. En él se liberan substancias transmisoras en el lado del emisor de la unión. El efecto es elevar o disminuir el potencial eléctrico dentro del cuerpo de la célula receptora. Si su potencial alcanza el umbral se envía un pulso o potencial de acción por el axón. Se dice, entonces, que la célula se disparó. Este pulso alcanza otras neuronas a través de las distribuciones de los axones.

#### **La red neuronal**

El sistema de neuronas biológico esta compuesto por neuronas de entrada (censores) conectados a una compleja red de neuronas calculadoras (neuronas ocultas), las cuales a su vez están conectadas a las neuronas de salidas que controlan, por ejemplo, los músculos.

Los censores pueden ser señales emitidas por los oídos, los ojos, etc. Las respuestas de las neuronas de salida activan los músculos correspondientes. En el cerebro hay una gigantesca red de neuronas calculadoras u ocultas que realizan la computación necesaria. De esta manera similar, una red neuronal artificial debe ser compuesta por censores del tipo mecánico o eléctrico.

* + - 1. **Las redes neuronales artificiales**
				1. **Sistema experto.**

Un método más avanzado para representar el conocimiento, es el sistema experto. Típicamente está compuesto por varias clases de información almacenada: Las reglas Si *-* Entonces le dicen al sistema como se debe reaccionar ante los estados del mundo. Una regla del sistema experto puede ser Si *Y* es un hombre, Entonces *Y* es mortal. Los hechos describen el estado del mundo. Por ejemplo: Juan es mortal. Por último, una máquina de inferencia relaciona los hechos conocidos con las reglas Si - Entoncesy genera una conclusión. En el ejemplo: Juan es mortal, esta nueva conclusión se añade a la colección de hechos que se almacena en los medios ópticos o magnéticos del computador digital. De esta forma, un sistema experto sintetiza nuevo conocimiento a partir de su entendimiento del mundo que le rodea, y de esta forma, un sistema experto es un método de representación y procesamiento del conocimiento mucho más rico y poderoso que un simple programa de computador. Sin embargo, con respecto a la manera en que opera el cerebro humano, las limitaciones son múltiples. Los problemas planteados en términos difusos o ambiguos, por ejemplo, son muy complejos de analizar o de conocer con sistemas de procesamiento simbólico, tales como los sistemas expertos o programas de computador.

#### **Interpretación de la neurona por computadora.**

Una neurona se puede comparar con una caja negra compuesta por varias entradas y una salida. La relación de activación entre la salida y la entrada, o en términos circuitales o de teoría de control.

La variable **f** es la frecuencia de activación o emisión de potenciales y **u** es la intensidad del estímulo de la soma.

#### **La neurona artificial**

Es un circuito eléctrico que realice la sume ponderada de las diferentes señales que recibe de otras unidades iguales y produzca en la salida un **uno** o un **cero** según el resultado de la suma con relación al umbral o nivel de disparo, conforma una buena representación de lo que es una neurona artificial. La función de transferencia para la activación o disparo de la neurona puede ser:

* de umbral lógico
* de limitación dura, o
* de función tipo s.

W representa el peso o ponderación de la conexión a través de una entrada.

La neurona artificial es un dispositivo eléctrico que responde a señales eléctricas. La respuesta la produce el circuito activo o función de transferencia que forma parte del cuerpo de la neurona. Las dendritas llevan las señales eléctricas al cuerpo de la misma. Estas señales provienen de censores o son salidas de neuronas vecinas. Las señales por las dendritas pueden ser voltajes positivos o negativos; los voltajes positivos contribuyen a la excitación del cuerpo y los voltajes negativos contribuyen a inhibir la respuesta de la neurona.

***Función de transferencia de las redes neuronales de tipo umbral lógico***

***Figura 2.6***

***Ejemplo de una función de transferencia de las redes neuronales de tipo S***

***Figura 2.7***

***Ejemplo de una función de transferencia de las redes neuronales de tipo limitación dura***

***Figura 2.8***

**Entrenamiento**

Cuando el sistema humano de neuronas, los ojos captan un objeto A, por ejemplo, algunos de los censores de la visión se activan y envían señales a las neuronas ocultas. Las neuronas que se disparan con la señal de entrada aumentan el grado de conexión de ellas. Si el mismo objeto A se presenta una y otra vez, la interconexión de neuronas se refuerza y, por lo tanto, el conocimiento del objeto.

Si se le presenta a la red anterior el objeto A modificado, la unión de las neuronas para el conocimiento de tal objeto, es débil. Las neuronas deben entrenarse para reconocer el objeto A en esta nueva presentación. Luego de algunas sesiones de entrenamiento, el sistema neuronal es capaz de reconocer el objeto A en todas sus formas. Si el objeto cambia nuevamente la red de neuronas y el conocimiento se actualizan.

Este entrenamiento, repetido para todos los valores de entrada y salida que se quiera, origina una representación interna del objeto en la red, que considera todas las irregularidades y generalidades del mismo.

* + - 1. **Método de transmisión de la información en el cerebro**

Antes conviene saber que en los primeros tiempos de la informática a los ordenadores se los llamaba calculadoras de cifras electrónicas o simplemente calculadoras digitales. Los sistemas digitales trabajan con cifras en código binario que se transmiten en formas de impulsos (bits). Los sistemas analógicos procesan señales continuamente cambiantes, como música o palabra hablada.

Por suerte para nuestro propósito de imitar con un ordenador el cerebro este también codifica la información en impulsos digitales. En los humanos las sensaciones se generan digitalmente y se transmiten así a través del sistema nervioso. Con otras palabras cuando la luz se hace más intensa, el sonido mas alto o la presión mas fuerte, entonces no es que fluya mas corriente a través de los nervios, sino que la frecuencia de los impulsos digitales aumenta.

En principio los ordenadores trabajan de manera semejante. Así una sensación mas fuerte corresponde en un equipo informático a una cifra más alta (o en una palabra mas larga). Sin embargo en un ordenador los datos se transmiten siempre a un mismo ritmo; la frecuencia base es inalterable. Por eso las cifras mas altas tardan mas tiempo en ser transmitidas. Como por lo general el ordenador o trabajan en tiempo real, esto no tiene mayor importancia, pero cuando se trata de un procesador en tiempo real, como son los empleados en proceso industrial, hace falta de ampliar él numero de canales de transmisión para que en el mismo espacio de tiempo pueda fluir mayor cantidad de datos.

* + - * 1. **Compuertas lógicas**

Sabemos que los elementos básicos de un ordenador son las compuertas lógicas, en el cerebro también existen aunque no son idénticas a las de un ordenador.

En un ordenador las compuertas de conjunción, disyunción, etc. tienen una función perfectamente determinada e inalterable. En el cerebro también hay elementos de conexión parecidos, las llamadas sinapsis, donde confluyen en gran numero las fibras nerviosas.

**Funcionamiento de las sinapsis**

Cientos de datos fluyen por los nervios hasta cada sinapsis, donde son procesados. Una vez analizada y tratada la información esta sale ya transformada por los canales nerviosos.

En los seres vivos no pueden permitirse el lujo de la especialización ya que si algo se rompe otro elemento debe hacerse cargo de la función. Por eso cada sinapsis es simultáneamente una compuerta de conjunción, disyunción, negación, etc.

Una sinapsis suma las tensiones de los impulsos entrantes. Cuando se sobrepasa un determinado nivel de tensión; el llamado umbral de indicación; esta se enciende, esto es deja libre el camino para que pasen los impulsos. Si el umbral de indicación de tensión es demasiado bajo, la sinapsis actúa como una puerta lógica del tipo disyuntivo, pues en tal caso pocos impulsos bastan para que tenga lugar la conexión. En cambio cuando el umbral de indicación es alto, la sinapsis actúa como una puerta de conjunción, ya que en ese caso hace falta que lleguen la totalidad de los impulsos para que el camino quede libre. También existen conducciones nerviosas que tienen la particularidad de bloquear el paso apenas llegan los impulsos. Entonces la sinapsis hace la función de una compuerta inversora. Esto demuestra la flexible del sistema nervioso.

* + - 1. **Diferencias entre el cerebro y un ordenador**

La diferencia más importante y decisiva el cómo se produce el almacenamiento de información en el cerebro y en el ordenador.

**Ordenador**: Los datos se guardan en posiciones de memoria que son celdillas aisladas entre sí. Así cuando se quiere acceder a una posición de memoria se obtiene el dato de esta celdilla. Sin que las posiciones de memoria aldeanas sé de por aludidas.

* Sistemas altamente especializados con capacidad para procesar información muy concreta, siguiendo unas instrucciones dadas.
* La frecuencia de transmisión es inalterable y esta dada por el reloj interno de la maquina.
* Las compuertas lógicas tienen una función perfectamente determinada e inalterable.
* La información se guarda en posiciones de memoria de acceso directo por su dirección.
* En el interior de una computadora los impulsos fluyen a la velocidad de la luz.

**Cerebro:** La gestión es totalmente diferente. Cuando buscamos una información no hace falta que sepamos donde se encuentra almacenada y en realidad no lo podemos saber ya que nadie sabe donde guarda hasta hoy en ida el cerebro los datos.

* Sistema de datos de múltiple propósito capaz de tratar gran cantidad de información en poco tiempo pero no necesariamente con exactitud.
* La frecuencia de los impulsos nerviosos puede variar.
* Las llamadas sinapsis cumple en el cerebro la función simultánea de varias compuertas (and, or, not etc.)
* La memoria es del tipo asociativo y no se sabe dónde quedara almacenada.
* Los impulsos fluyen a 30 metros por segundo.

Pero tampoco es necesario ya que basta con que pensemos en el contenido o significado de la información para que un mecanismo, cuyo funcionamiento nadie conoce, no nos proporcione automáticamente solo la información deseada sino que también las informaciones vecinas, es decir, datos que de una u otra manera hacen referencia a lo buscado.

Los expertos han concebido una serie de tecnicismos para que lo incomprensible resulte algo más comprensible. Así a nuestro sistema para almacenar información se lo llama memoria asociativa. Esta expresión quiere dar a entender que los humanos no memorizan los datos direccionándolos en celdillas, sino por asociación de ideas; esto es, interrelacionando contenidos, significados, modelos.

En todo el mundo pero sobre todo en Estados Unidos y Japón, científicos expertos tratan de dar con la clave de la memoria asociativa. Si se consiguiera construir un chip de memoria según el modelo humano, la ciencia daría un paso gigante en la fascinante carrera hacia la inteligencia artificial. Y además el bagaje del saber humano quedaría automáticamente enriquecido.

* + - 1. **Similitudes entre el cerebro y una computadora**
* Ambos codifican la información en impulsos digitales.
* Tanto el cerebro como la computadora tienen compuertas lógicas.
* Existen distintos tipos de memoria.
* Los dos tienen aproximadamente el mismo consumo de energía.
	+ - 1. **Un superordenador llamado cerebro**

El hombre necesita un sistema de proceso de datos de múltiple propósito capaz de tratar gran cantidad de información muy distinta y en muy poco tiempo y con el mayor sentido practico(pero no necesariamente con exactitud), para inmediatamente poder actuar en consecuencia. Los ordenadores, en cambio, son altamente especializados con capacidad para procesar con exactitud información muy concreta(en principio solo números) siguiendo unas instrucciones dadas.

El cerebro humano posee mas de diez millones de neuronas las cuales ya están presentes en el momento del nacimiento conforme pasa el tiempo se vuelven inactivas, aunque pueden morir masivamente.

Nuestro órgano de pensamiento consume 20 Patios/hora de energía bioquímica, lo que corresponde a una cucharada de azúcar por hora. Los ordenadores domésticos consumen una cantidad semejante. Las necesidades de oxigeno y alimento es enorme en comparación con el resto del cuerpo humano: casi una quinta parte de toda la sangre fluye por el cerebro para aprovisionar de oxigeno y nutrieres. La capacidad total de memoria es difícil de cuantificar, pero se calcula que ronda entre 1012 y 1014 bits.

La densidad de información de datos de un cerebro todavía no se ha podido superar artificialmente y en lo que se refiere a velocidad de transmisión de datos, a pesar de la lentitud con que transmite cada impulso aislado, tampoco esta en desventaja, gracias a su sistema de proceso en paralelo: la información recogida por un ojo representa 106 bits por segundo.

Según todos los indicios el cerebro dispone de dos mecanismos de almacenamiento de datos: la memoria intermedia acepta de cinco a diez unidades de información, aunque solo las mantiene durante algunos minutos. La memoria definitiva guarda las informaciones para toda la vida, lo que no significa que nos podamos acordar siempre de todo. La memoria inmediata trabaja como una espacie de cinta continua: la información circula rotativamente en forma de impulsos eléctricos por los registros. El sistema es comparable a la memoria dinámica de un ordenador, en la que la información tiene que ser refrescada continuamente para que no se pierda. En cambio, la memoria definitiva parece asemejare mas bien a las conocidas memorias de celdillas de los ordenadores. Se cree que esta memoria funciona gracias a formaciones químicas de las proteínas presentes en el cerebro humano.

* + - 1. **Aplicaciones de las redes neuronales**

Las redes neuronales son una gran herramienta de optimización, porque aplican el conocimiento que acumulan por medio de premisas, por lo que se han desarrollado diversas aplicaciones tales como:

* Optimización de rutas
* Aplicaciones de inferencias lógicas
* Sistemas de toma de decisiones
* Juegos de tablero

De igual manera se ha llegado a simular partes del cuerpo humano que actualmente se utilizan, y otras que se utilizarán en el futuro, así como:

* Retina artificial
* Oído artificial
* Motrocidad asistida
* Minibomba para diabéticos
* Minidesfibrilador
* Corazón Artificial
* Descargas eléctricas contra la epilepsia
	+ - 1. **Redes neuronales y algoritmos genéticos**

Las redes neuronales y los algoritmos genéticos se utilizan para extraer automáticamente conocimiento útil para el proceso de modelaje y previsión.

Las redes neuronales son una potente herramienta de modelaje:

1. Aprenden los patrones de un conjunto de datos de entrenamiento
2. Poseen un buen comportamiento en presencia de ruido
3. Pueden modelar relaciones complejas
4. Son capaces de generalizar para predecir

Los algoritmos genéticos se utilizan para buscar el mejor modelo cuando la cantidad de combinaciones de modelos posibles es demasiado grande. Utilizan una función de evaluación para determinar la calidad de los individuos de la población.

En nuestro caso, para la selección del mejor modelo de forma automática se utiliza como criterio un estadístico que tiene en cuenta la calidad de los residuos y la cantidad de parámetros.

* 1. **Autómatas Celulares**

Los autómatas celulares son redes de autómatas simples conectados localmente. Cada autómata simple produce una salida a partir de varias entradas, modificando en el proceso su estado según una función de transición. Por lo general, en un autómata celular, el estado de una célula en una generación determinada depende única y exclusivamente de los estados de las células vecinas y de su propio estado en la generación anterior.

Los autómatas celulares son herramientas útiles para modelar cualquier sistema en el universo. Pueden considerarse como una buena alternativa a las ecuaciones diferenciales y han sido utilizadas para modelar sistemas físicos, como interacciones entre partículas, formación de galaxias, cinética de sistemas moleculares y crecimiento de cristales, así como diversos sistemas biológicos a nivel celular, multicelular y poblacional.

La Vida Artificial es un mecanismo para la representación de sistemas complejos. Una vez representado un sistema, podemos experimentar con él en situaciones hipotéticas.

En toda ciencia existe simplificación o generalización; esto es lo que nos permite razonar, extraer conclusiones o crear hipótesis sobre nuevos problemas gracias a los ejemplos del pasado. Una simulación nunca es perfecta, lo contrario sería realizar una copia idéntica, átomo por átomo (átomo como unidad indivisible, si existiera tal concepto). Se trata de no tener que construir una central nuclear para experimentar y prever comportamientos en una central nuclear. Los sistemas complejos más completos de todos serían aquellos cuya única representación aceptable serían ellos mismos.

En los sistemas complejos puede producirse lo que se denomina el comportamiento emergente. La gran ventaja de la Vida Artificial es que permite observar fenómenos emergentes. Las propiedades emergentes son aquellas que aparecen en los sistemas como resultado de la interacción entre sus partes y que no pueden explicarse a partir de las propiedades de los elementos que lo componen.

Toda ciencia tiende al reduccionismo, a explicar fenómenos complejos por su descomposición en otros más simples. Sin embargo existen ciencias como la ecología (que se dedica al estudio de cómo los organismos, estudiados individualmente por otras ramas de la biología, se insertan en el mundo real) no puede dejar de observarse el sistema en conjunto, no puede olvidarse el punto de vista hollístico, es decir, la convergencia de comportamiento que resulta cuando diversos elementos del sistema se condicionan mutuamente. No es posible, o no basta, estudiar los elementos por separado.

En estos sistemas, una acción simple se puede propagar exponencialmente provocando efectos de gran envergadura. Por esta razón se dice que estos sistemas se encuentran en la frontera entre el orden y el desorden.

Por ejemplo, un montón de arena sobre el que dejamos caer granos lentamente. Ocurre que a partir de cierto ángulo la caída de un sólo grano provoca avalanchas de gran tamaño. De la misma forma, la caída de un copo de nieve puede provocar la ruptura de la rama de una árbol sobre la que se ha acumulado gran cantidad de copos.

Una colonia de hormigas naturales puede llevar a cabo tareas de gran complejidad como explorar el entorno, construir galerías, o decidir si es adecuado un tipo de alimento, con la lamentable pérdida de algunas hormigas en el proceso. Pero las hormigas, consideradas una por una, son incapaces de realizar ningunas de estas tareas.

El comportamiento del hormiguero emerge a partir de interacciones entre elementos simples (hormigas) y no se puede explicar únicamente basándonos en las propiedades de un individuo. El hormiguero podría considerarse como un ser vivo de un nivel superior a la hormiga.

* + 1. Comparación entre los algoritmos genéticos y los autómatas celulares

En un algoritmo genético es difícil saber cuándo una estructura de datos es adecuada; sin embargo, sí existe una forma de saber por anticipado cuándo una estructura de datos no es adecuada.

Para que el algoritmo funcione como deseamos, la reproducción y las mutaciones deberían producirse de tal forma que mantengan, en la mayoría de los casos, las cualidades positivas de los individuos. Pensemos en el caso de las mutaciones. Dada una cadena, una mutación consistirá en modificar al azar uno de los elementos de esa cadena. Para que todo marche bien deberá ocurrir que los valores de los genes sean buenos o malos, independientemente de los valores de otros genes, o al menos, que sean lo más independientes posible.

Un buen método para saber cuándo una estructura de datos no es adecuada, es el siguiente: si al modificar un elemento cualquiera de una cadena seleccionada (buena) existe una probabilidad alta de que la cadena resultante sea pésima, entonces la estructura de datos elegida no es correcta, ya que los componentes de la cadena por sí solos no contienen la información que hace que esa cadena sea seleccionada.

Es decir, *"Cuando el objetivo (peso) de una entidad se puede calcular como una función en la que variaciones muy pequeñas de los valores de sus variables (valores de los genes) producen con una probabilidad alta, variaciones muy grandes en el resultado de la función, la estructura de datos es incorrecta"*. Una variación muy pequeña sería cambiar el valor de un gen.

El ejemplo por excelencia de este tipo de problemas es la lotería. El cambio de uno sólo de los dígitos de nuestro billete puede producir cambios enormes en la cantidad que se recibe como premio. Un algoritmo genético que trate de predecir el número premiado de esta forma nunca tendrá demasiado éxito.

La descripción que se acaba de dar de los problemas que hay que evitar suena mucho a funciones discontinuas; sin embargo, se habla de estos problemas como funciones no lineales. ¿Qué es lo que ocurre aquí? Vamos a verlo con detenimiento. Una función puede ser continua o discontinua, lineal o no lineal. En la siguiente figura se ilustran los cuatro casos.

La confusión de la que se está tratando se hizo patente en el artículo "Algoritmos Genéticos Avanzados" publicado en la revista Solo Programadores (Nº 37 de Septiembre de 1997) donde se identificaron erróneamente las funciones no lineales con aquellas en la que variaciones pequeñas en las variables producen variaciones grandes en el valor de la función. La afirmación era errónea. Lo que llevó al error es que ambas definiciones hacen referencia al mismo problema en el caso de los algoritmos genéticos.

|  |  |  |
| --- | --- | --- |
| Funciones | LINEAL | NO LINEAL |
| CONTINUA | FCL.gif | FCNL.gif |
| DISCONTINUA | Fdl.gif | FDNL.gif |

Tipos de funciones en cuanto a linealidad y continuidad

Figura 2.12

Al ser los valores de los genes discretos, el valor de la función también lo será, y podemos representar los posibles valores del peso como una nube de puntos. Dado que la función sólo puede saltar de un valor a otro, podemos identificar dos tipos de funciones: aquellas en las que al realizar un pequeño cambio en sus variables, se producen siempre saltos pequeños (en cuanto a peso) o aquellas en las que es probable que el salto sea muy grande, correspondiendo el segundo caso como un problema que de difícil solución.

En las gráficas de ejemplo, los tres primeros casos corresponden con problemas que pueden ser resueltos por algoritmos genéticos. El último de ellos (función discontinua y no lineal) corresponde con un problema de difícil solución, ya que no es posible acercarse gradualmente al objetivo buscado, o lo que es lo mismo, no hay una presión selectiva que dirija a los individuos hacia el máximo.

Los problemas en los que encontramos este tipo de inconvenientes han sido identificados por John H. Holland (el "padre" de los algoritmos genéticos) como problemas no lineales, y ciertamente son así. En el artículo "Algoritmos Genéticos" publicado en la Revista Investigación y Ciencia, en el número de Septiembre 1992. En las páginas 41 y 42 aparece:

*"El paralelismo implícito del algoritmo genético le permite ensayar un gran número de regiones del espacio de búsqueda manipulando un número relativamente reducido de ristras. También le facilita afrontar problemas no lineales, problemas en las que la adecuación de una cadena que contenga dos bloques constructivos concretos puede ser mucho mayor (o mucho menor) que la suma de adecuaciones atribuibles por separado a cada bloque constructivo individual"*.

Es decir, que si ocurre que un gen es bueno o malo en función del valor de otros genes, el problema se nos complica. ¿Por qué no habla Holland de funciones discontinuas y además no lineales?. La función gausiana es no lineal y sin embargo sí puede ser resuelta con un algoritmo genético. La explicación a esto se encuentra en lo que ya se ha mencionado. Como se trata de variables discretas, la función siempre será discontinua, por lo que estrictamente podemos ignorar las funciones continuas, y referirnos a este tipo de problemas simplemente como funciones no lineales.

Una vez aclarado esto, se van a puntualizar otras cosas que pueden no quedar claras. Aunque se están haciendo hipótesis sobre una determinada función, dicha función es desconocida, precisamente se trata de hallar el máximo (o varios máximos) sin saber cómo es realmente la función. Si la conociésemos, bastaría derivarla e igualar a cero para hallar el máximo, es decir, la mejor entidad. Solo sabemos que:

peso = f (valor\_de\_gen\_1, valor\_de\_gen\_2,...)

Curiosamente, lo que sí somos capaces es de asignar un peso a un individuo o conjunto de valores de genes. Es decir, sabemos reconocer en que grado se cumple aquello que buscamos, aunque no sabemos cómo encontrarlo.

Esto es evidente y no hay otra opción. Si no podemos reconocer aquello que buscamos, es imposible encontrarlo. Cuanta más exactitud tengamos en identificar el grado de bondad de cada solución, tanto mejor funcionará el algoritmo.

Volviendo a la no-linealidad, en nuestros problemas ojalá siempre ocurriera que:

objetivo = f1 (valor\_de\_gen\_1) + f2 (valor\_de\_gen\_1) + f3 (valor\_de\_gen\_1)

O lo que es lo mismo: ciertos valores para ciertos genes son buenos, y otros son malos, pero la bondad o maldad de cada uno es independiente de los valores de los otros.

Por ejemplo, tenemos la función Y = 3X + 4Z + 17

En esta función variaciones pequeñas de X ó Z producen variaciones pequeñas de Y.

#### En cambio en la función

Ocurre que, por ejemplo, Y tiene el valor 400 si Z=1, pero para Z=1.000000000001 tenemos que Y = 3 \* X, que no tiene porqué estar cercano al valor 400.

En un algoritmo genético, si el patrón posee 4 elementos A, B, C y D cuyos valores pueden oscilar entre 0 y 10, una función objetivo adecuada podría ser:

objetivo = (2 \* A) + (4 \* B) - (2 \* C)

En este caso se observa que valores altos de A producen siempre una gran proximidad al objetivo, lo mismo ocurre con los valores de B, pero con un efecto del doble que A. En cambio, los valores de C deberán ser lo más pequeños posibles, ya que disminuyen el objetivo, siendo indiferentes los de D.

Así, por ejemplo, al reproducir una solución, como por ejemplo:

A = 5; B = 4; C =1; D = 9 contra otra, es muy probable que las cualidades positivas (valores altos de A y B; valores bajos de C) se transmitan a la descendencia.

El tres en raya con reglas es bastante lineal: cada regla, o es buena, o es mala. Si una entidad seleccionada posee 40 o 50 reglas, probablemente podamos cambiar una sin que se convierta en un pésimo jugador.

Pero en casi todos los problemas existe una cierta no-linealidad inevitable. Por ejemplo, pueden producirse por azar reglas muy interesantes, pero que no son seleccionadas porque necesitan de otras para demostrar su utilidad y producir un peso alto. Es decir, las no linealidades son trozos de cadena cuya utilidad depende de los valores de otros segmentos, y que sólo son útiles cuando todos ellos contienen determinados datos, desapareciendo la utilidad al fallar uno de ellos.

Cuanta más no-linealidad exista, más lentamente convergerá nuestro algoritmo genético. Esto es completamente inevitable, ya que no es posible encontrar algo si no sabemos qué es lo que estamos buscando. Sin embargo, la naturaleza parece que sí lo consigue. Así cómo han podido generarse gradualmente órganos complejos, tales como los ojos o las plumas para el vuelo, que no dependen de una única mutación, si solamente el órgano completo es útil al individuo. No siempre parece fácil explicar todos los rasgos que presenta un organismo con un mecanismo adaptativo.

En mi opinión, este planteamiento es erróneo. Los seres vivos sacan beneficio de cualquier órgano a medio formar. Un pato puede huir de un depredador terrestre alzando el vuelo tan sólo un metro. El sistema visual de la rana está sintonizado solamente a cuatro clases de estímulo, estos son: el contraste de luz y oscuridad; un borde de luz u oscuridad; una repentina disminución de la luminosidad; y el constante movimiento de un pequeño objeto de color negro.

Es habitual considerar nuestra inteligencia o la visión de un halcón como perfectas, en vez de simples estadios intermedios, que llegarán a niveles mucho más avanzados gracias a la evolución. La naturaleza probablemente sea mucho más lineal de lo que parece, y nosotros podemos ser la rana.

Entonces, los algoritmos genéticos tal vez logren llegar a una solución óptima pero tardarán muchísimo en este tipo de funciones discontinuas y no lineales.

* + 1. **Aplicaciones de los autómatas celulares**

Al mencionar los autómatas celulares, como se escribió anteriormente, se simularan y resolverán situaciones en conjunto, que como elementos independientes no se pueden evaluar conocimientos. Entonces puede mencionares como aplicaciones de los autómatas celulares:

* Inteligencia artificial basada en el comportamiento.
* Ecosistemas artificiales.
* Vida artificial en la construcción de un espacio cibernético.
	1. **Observaciones**

Con este capítulo he dado una explicación completa de las diferentes formas de representar los algoritmos genéticos, y he explicado no solamente la parte que necesito en este trabajo, sino todo lo referente a los mismos, y además he realizado una comparación con las redes neuronales y con los autómatas celulares que son otras técnicas de programación evolutiva.