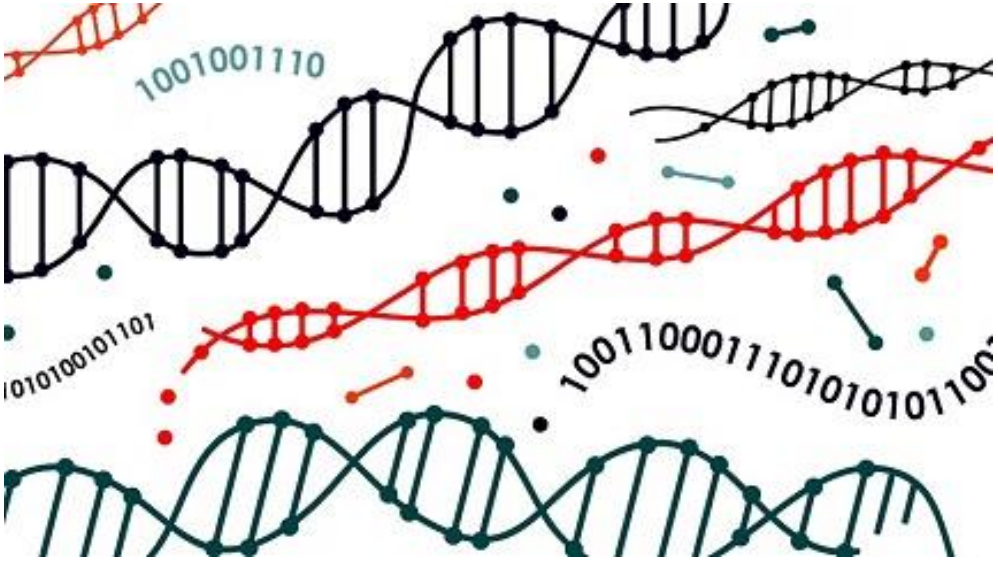
Inteligencia Artificial

Unidad V: Algoritmos Bioinspirados

ESCOM-IPN Daniel Aguilar Velázquez - Rodolfo Romero Herrera – Gabriela de Jesús López Ruiz

5.1 Algoritmos genéticos

Los Algoritmos Genéticos (AAGG) son un tipo de algoritmos evolutivos usados para resolver problemas de búsqueda y optimización. Se basan en la imitación del proceso evolutivo que se produce en la naturaleza para resolver problemas de adaptación al medio.

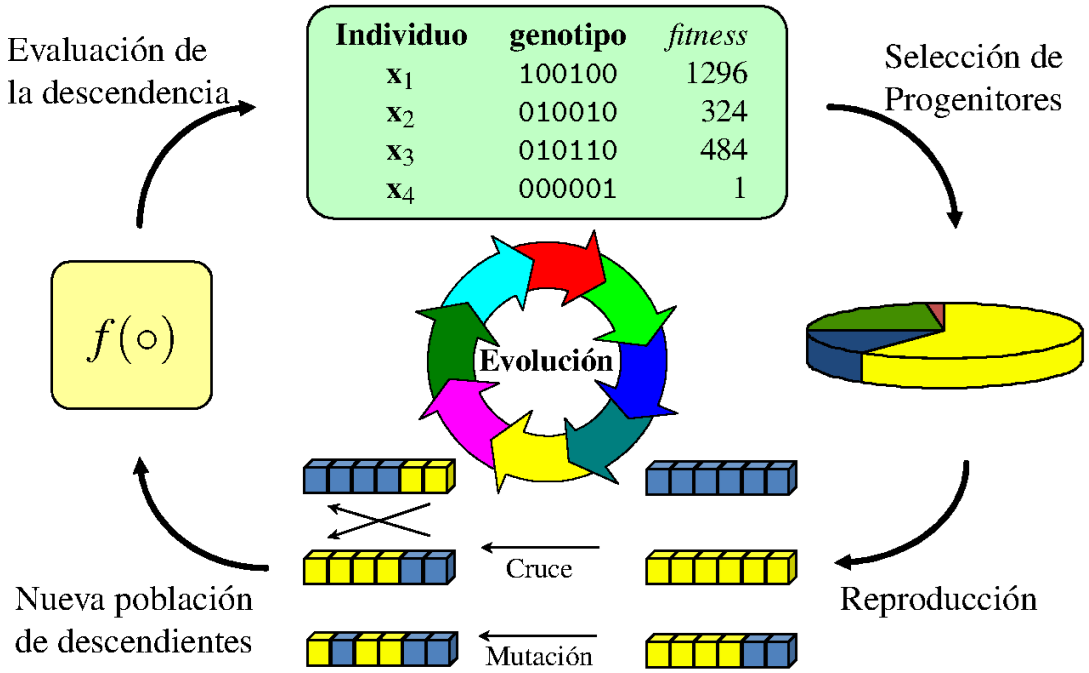


5.1.1 Trasfondo biológico

- Los AAGG simulan mediante poblaciones de individuos la evolución sufrida a través de diferentes operadores. En la naturaleza, los individuos compiten entre ellos por diferentes recursos necesarios para la supervivencia como el agua y la comida, y los miembros de una misma especie también compiten por aparearse. De todos ellos, los que tienen más éxito en sobrevivir y aparearse conseguirán un mayor número de descendientes. Esto significa que los genes de los individuos de mayor adaptación al medio serán heredados por un número creciente de individuos en cada generación . La combinación de buenas características de diferentes antepasados puede generar una descendencia altamente adaptada al medio.

5.1.1 Trasfondo biológico

- De la misma forma, es posible también que la combinación de genes de individuos bien adaptados generen otro con peores características, con lo que estos individuos tenderán a desaparecer. Es posible, además, que se produzcan cambios en el propio medio de vida, por lo que las especies evolucionarán hacia las nuevas condiciones de vida

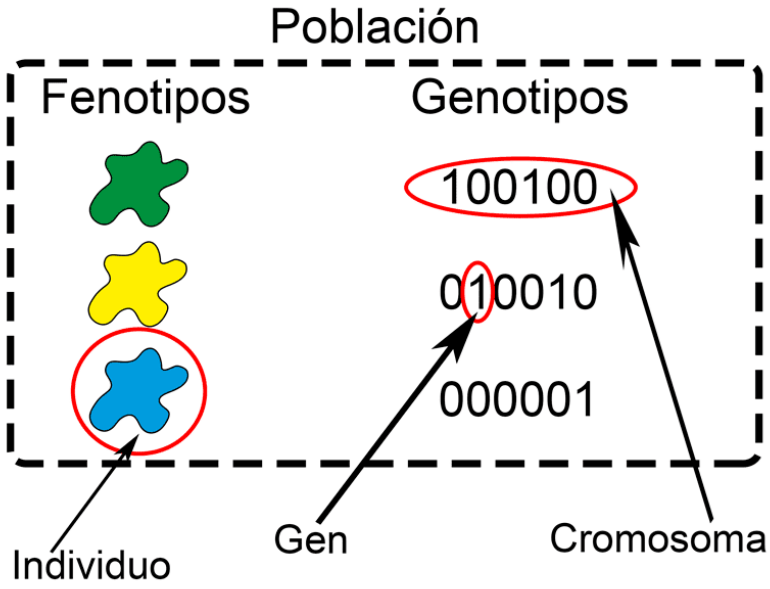


5.1.1 Trasfondo biológico (Cont.)

- La potencia de los AAGG se debe a que la técnica que usan es robusta y puede tratar con éxito un gran número de tipos de problemas, incluyendo y destacando aquellos que son difícilmente solucionables utilizando otro tipo de métodos clásicos. Los AAGG no garantizan la localización exacta de la solución óptima del problema, pero proporcionan una solución alternativa aceptablemente buena. Este hecho es especialmente interesante en el caso de resolución de problemas donde todavía no se ha desarrollado una técnica específica.

5.1.2 Algoritmo genético con representación binaria

- Inicialmente se empezó a trabajar con codificaciones binarias de las cadenas que forman la población. Es decir, el alfabeto habitualmente utilizado en los algoritmos genéticos era {0,1}. Para realizar el estudio de las consecuencias de la aplicación de cualquiera de los operadores, así como la posibilidad de obtener o no la solución o soluciones esperadas, se introduce el concepto de *esquema* o *patrón* . Paralelo, se añade al alfabeto binario un tercer carácter, denominado *carácter añadido*, obteniendo el alfabeto {0,1,*}. Se denomina entonces patrón o esquema y se representa por H a todo elemento del conjunto {0,1,*}l.



5.1.2 Algoritmo genético con representación binaria

- De este modo, un patrón o esquema se puede representar mediante una *t*upla (p1, p2,..., p*l*) donde cada pi∈{0,1,*}. Un patrón p1p2...p*l* representa un conjunto de cadenas o elementos de {0,1}*l*. Para ello, si pi=*, entonces las cadenas del conjunto pueden tomar cualquiera de los valores del alfabeto original en la posición i. Se dice que las cadenas contenidas en el esquema H, como elementos del conjunto H, son *ejemplos* o *representantes* de dicho esquema. La cantidad de caracteres añadidos de cada patrón determina sus grados de libertad, de modo que aquellos patrones con más caracteres añadidos tienen, como conjuntos, mayor cardinal. Concretamente, si el patrón H aparece el carácter añadido k veces , dicho patrón es un conjunto formado por 2k cadenas.

5.1.2 Algoritmo genético con representación binaria (Cont.)

- Se denomina *orden* de esquema H, denotado por O(H), a la cantidad de posiciones que no están ocupadas, en su representación en cadena, por caracteres añadidos. Se trata por tanto del número de posiciones fijas en el esquema. De este modo, la cantidad de grados de libertad se obtiene como la diferencia entre la longitud de cadena y el orden.
- Se denomina *longitud* de un patrón a la distancia máxima entre dos posiciones de dicho patrón no ocupadas por el carácter añadido. La longitud del patrón H se representa por *l*(H).

5.1.2 Algoritmo genético con representación binaria (Cont.)

- Por otra parte, ξ(H,t) denota el número de cadenas que en una determinada población en el tiempo t, se asocian con el esquema H. Se denomina *adaptación* de un esquema H en el tiempo t, denotada por eval(H,t), a la media de la función objetivo de todas las cadenas de la población que pertenecen al esquema H.
- Dada una población en el instante t compuesta por λ individuos I1,..,Iλ, se denota por F(t) a la media de la función objetivo extendida a toda la población en el tiempo t. Por tanto, se tiene:

$$F(t) = \frac{1}{\lambda} \sum_{i=1}^{\lambda} \mathrm{f}(\mathrm{I}_i),$$

5.1.2 Algoritmo genético con representación binaria (Cont.)

- siendo $f(I_i)$ el valor de la función objetivo del individuo Ii de la población en el instante t.
- Con todos estos conceptos y notación introducida hasta el momento, es posible enunciar el teorema de los esquemas de la siguiente manera :
- Sea $E_{sel,cru,mut}(\xi(H,t))$. El número esperado de individuos que se asocian con el esquema H, en el tiempo
- t+1. Después de efectuar la reproducción, el cruce y la mutación en un algoritmo genético con alfabeto binario, se tiene que:

$$E_{sel,cru,mut}\left(\xi(H,t)\right) \geq \xi(H,t) \cdot \frac{eval(H,t)}{F(t)} \cdot \left[1 - p_c \frac{l(H)}{l-1} - O(H) \cdot p_m\right]$$

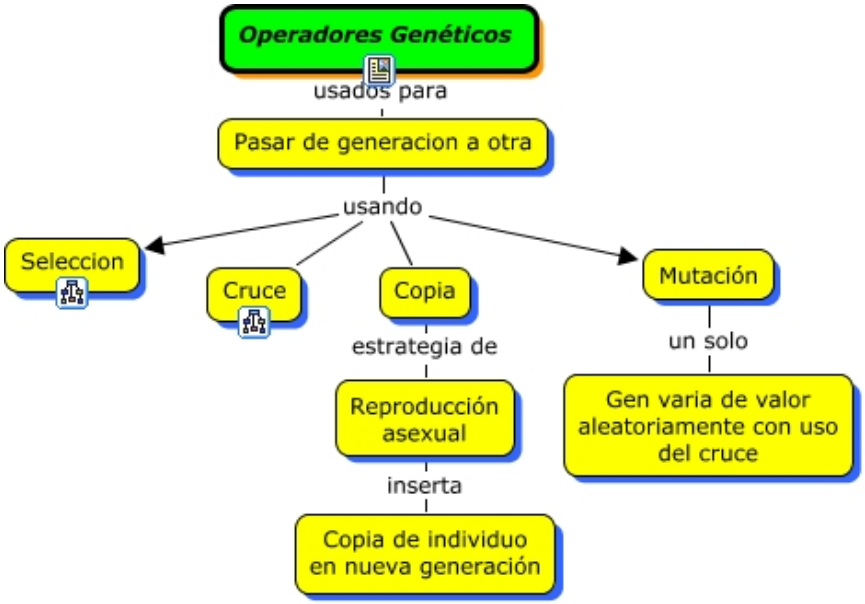
siendo pc y pm la probabilidad de cruce y mutación respectivamente.

5.1.2 Algoritmo genético con representación binaria (Cont.)

- De la fórmula anterior, se desprende como conclusión que esquemas cortos, de bajo orden y longitud, y con una adaptación al problema superior a la adaptación media, se espera que a medida que evoluciona el AG, obtengan un rápido incremento en el número de individuos que se asocian con los mismos. La importancia de este teorema queda de manifiesto cuando se trata de predecir la convergencia de un algoritmo, en cierto problema. La solución o soluciones obtenidas en cada generación se ajustarán a los patrones que sobrevivan hasta ese momento, a la acción se los operadores genéticos.

5.1.3 Operadores genéticos para representaciones reales y permutaciones

- Entre los operadores genéticos, los considerados fundamentales por la mayoría de los estudiosos del tema, son los de reproducción, mutación y cruce. Cada uno de ellos desarrolla una labor específica, bien diferenciada de las de los demás, como se verá seguidamente.



5.1.3 Operadores genéticos para representaciones reales y permutaciones (Cont.)

Operadores de reproducción

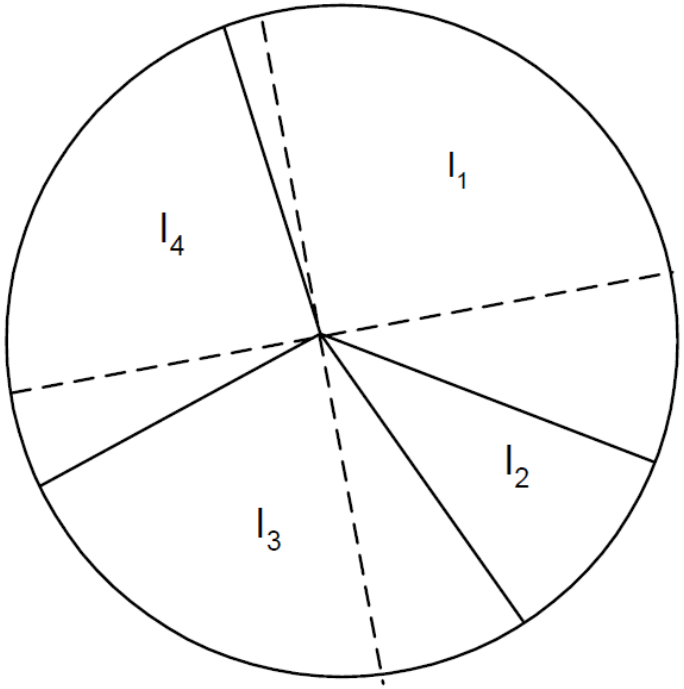
- El operador de reproducción se encarga de seleccionar a los individuos destinados a tener descendencia agrupándolos en un lugar intermedio denominado lugar de apareamiento (mating pool). Los individuos mejor adaptados serán aquellos que tendrán mayor probabilidad de ser escogidos, por lo que la mayoría de los operadores de reproducción se basan en el valor proporcionado por la función objetivo para cada individuo. Todos los operadores de reproducción ofrecen un rendimiento similar si se utilizan los parámetros adecuados.
- La función de selección más utilizada, es la denominada *función de selección proporcional a la función objetivo,* en la cual cada individuo tiene una probabilidad de ser seleccionado que es proporcional a su valor en la función objetivo. Denotando por pj^{prop} la probabilidad de que el individuo I_j sea seleccionado de entre una población de λ individuos y por $f(I_j)$ el valor de la función objetivo, se tiene que:

5.1.3 Operadores genéticos para representaciones reales y permutaciones (Cont.)

$$p_j^{prop} = \frac{f(I_j)}{\sum_{j=1}^{\lambda} f(I_j)}$$

Otro de los métodos más utilizados es el *muestreo universal estocástico o método de la ruleta* . Este método emplea un círculo (la ruleta) divido en sectores circulares que son proporcionales a la función objetivo. De esta forma, los sectores correspondientes a individuos mejor adaptados ocupan un área mayor dentro de la ruleta, mientras que los peor adaptados se corresponden con sectores circulares con una superficie muy pequeña. Los individuos son seleccionados a partir de los marcadores igualmente espaciados, siendo la colocación del primero de ellos aleatoria. La figura representa un ejemplo de este tipo de métodos en el que el individuo I_1 (el mejor adaptado) se escoge dos veces; los individuos I_3 el$_4$ se escogen una vez cada uno, mientras que el individuo I_2 no es seleccionado.

5.1.3 Operadores genéticos para representaciones reales y permutaciones (Cont.)



Método de la ruleta para la selección de los individuos.

5.1.3 Operadores genéticos para representaciones reales y permutaciones (Cont.)

Operadores de mutación

- El operador de mutación actúa sobre un determinado individuo con una probabilidad pm. Cuando una cadena resulta afectada por el operador, se selecciona aleatoriamente un lugar de mutación. Para ello, todos los lugares de la cadena son equiprobables. El valor correspondiente a dicho lugar cambia por otro de los posibles, eligiendo éste, también aleatoriamente, entre todos los del alfabeto que se esté utilizando.
- El operador de mutación encuentra su más amplia justificación como método para obtener nuevos puntos en los que recomenzar la búsqueda, evitando de esta forma la pérdida de diversidad genética en la población, lo cual, como se verá más adelante, tiene implicaciones directas en el proceso de convergencia del algoritmo genético. Tanto es así que, según ciertos estudios, el operador de mutación va ganando en importancia a medida que la población de individuos va convergiendo.

5.1.3 Operadores genéticos para representaciones reales y permutaciones (Cont.)

- La elección de una adecuada probabilidad de mutación es un elemento crucial. Si la probabilidad de mutación es muy elevada, el algoritmo se convierte en una búsqueda aleatoria; mientras que si se elige una probabilidad de mutación muy pequeña, entonces la pérdida de diversidad genética producida cuando los individuos de la población están convergiendo, puede llevar al algoritmo a un subóptimo. La determinación de la probabilidad de mutación es, de hecho, mucho más crucial que la probabilidad de cruce.

5.1.3 Operadores genéticos para representaciones reales y permutaciones (Cont.)

- La búsqueda del valor óptimo para la probabilidad de mutación, es una cuestión que ha sido motivo de muchos trabajos de investigación. Así, una de las recomendaciones es la utilización de una probabilidad de mutación de 1// (/ es la longitud de la cadena). Otros estudios, a partir de ciertos resultados experimentales, obtuvieron que la mejor probabilidad de mutación era proporcional al tamaño y a la longitud según la fórmula

$$1/\lambda^{0.9318} \cdot l^{0.4535}$$

5.1.3 Operadores genéticos para representaciones reales y permutaciones (Cont.)

- Si bien en la mayoría de las implementaciones de algoritmos genéticos se asume que la probabilidad de mutación permanece constante, algunos estudios realizados revelan que se obtienen mejores resultados modificando esta probabilidad a lo largo del proceso de convergencia del algoritmo, lo cual ha sido corroborado en algunos de los últimos estudios realizados, donde a intervalos regulares de tiempo, se obtiene la diversidad genética existente en la población a través del cálculo de la desviación típica. Si el valor obtenido es menor que un cierto límite predefinido, lo cual significa que la población está convergiendo, entonces se incrementa la probabilidad de mutación. Si, por el contrario, el valor calculado de la desviación típica como medida de la diversidad genética de la población es mayor que este límite, entonces se reduce la probabilidad de mutación.

5.1.3 Operadores genéticos para representaciones reales y permutaciones (Cont.)

Operadores de cruce con alfabeto finito

- El trabajo con algoritmos genéticos cuya codificación de los puntos de la función a optimizar se realiza empleando un alfabeto finito se puede modelizar, como ya se ha indicado anteriormente, suponiendo que se emplea un alfabeto A de cardinal m. Asimismo, se supone que la cadena $c_1,c_2,...,c_l$ corresponde a la codificación de un punto del espacio de búsqueda, donde $c_k \in A$, para cada $k \in \{1,2,...,l\}$, utilizando cadenas de longitud *l*. El operador de cruce actúa sobre cadenas correspondientes a la población actual, generalmente dos, denominadas *cadenas progenitoras* o *padres*. Actuando sobre las cadenas progenitoras, un operador de cruce produce generalmente otras dos cadenas que corresponden a nuevos puntos del espacio de búsqueda, denominadas *cadenas descendientes*. Este operador afecta a cada par de puntos de la población con probabilidad pc, siendo el valor de pc característico de cada problema concreto.

5.1.3 Operadores genéticos para representaciones reales y permutaciones (Cont.)

Los operadores de cruce con alfabeto finito más importantes son:

1. Operador de cruce ordinario u operador basado en un punto. Se seleccionan dos padres y se cortan sus cadenas por un punto elegido al azar, denominado *lugar de cruce*, de entre los *l* posibles. Con esto quedan determinadas dos subcadenas en cada cadena del par. Sea el lugar de cruce k, y las cadenas a cruzar a1,a2,...,ak,ak+1,...,a*l* y b1,b2,...,bk,bk+1,...,b*l* , se generarán dos nuevas cadenas a1,a2,...,ak,bk+1,...,b*l* y b1,b2,...,bk,ak+1,...,a*l*. De esta forma, ambos descendientes heredan genes de cada uno de los padres.

5.1.3 Operadores genéticos para representaciones reales y permutaciones (Cont.)

Los operadores de cruce con alfabeto finito más importantes son:

2. Operador de cruce basado en dos puntos. Se han realizado diferentes investigaciones en cuanto al comportamiento del operador de cruce basado en múltiples puntos, concluyéndose que el basados en dos puntos representa una mejora respecto al operador basado en uno solo. Sin embargo, el añadir más puntos de cruce no beneficia al comportamiento del algoritmo. Este operador actúa de la siguiente forma: dadas dos cadenas a cruzar a1,a2,...,ak,ak+1,...,ak',ak'+1,...,a*l* y b1,b2,...,bk,bk+1,...,bk',bk'+1,...,b*l*, se determinan aleatoriamente, de entre los *l* posibles, los dos lugares de cruce k y k', y se intercambian las subcadenas, quedando las cadenas descendientes: a1,a2,...,ak,bk+1,...,bk',ak'+1,...,a*l* y b1,b2,...,bk,ak+1,...,ak', bk'+1,...,b*l*.

5.1.3 Operadores genéticos para representaciones reales y permutaciones (Cont.)

3. Operador de cruce uniforme. Se genera una *máscara de cruce* de longitud *l* cuyos valores (0 o 1) se escogen de forma aleatoria. El operador genera dos descendientes para cada par de padres de la siguiente forma: dadas dos cadenas padres a1,a2,...,ai,...,a*l* y b1,b2,...,bi,...,b*l*, la posición i de la cadena del primer descendiente será ai si el valor de la máscara en la posición i es 0; y será bi en caso contrario. Para el caso del segundo descendiente, se aplica la máscara inversa.

5.1.3 Operadores genéticos para representaciones reales y permutaciones (Cont.)

4. Operador de cruce generalizado. Parte de dos cadenas padres codificadas con un alfabeto binario para obtener dos nuevas cadenas descendientes de las anteriores. Sea S = {0,1}^{l} el conjunto de todas las posibles cadenas de longitud l. Sea g: S → {0,1,...,2^l-1}la función de transformación que permite, a partir de una cadena binaria, obtener su decodificación usual como número entero. Puesto que g es una función biyectiva, se puede definir el *conjunto entero equivalente* como el conjunto g(S). De esta forma es posible trabajar con el espacio de búsqueda o el conjunto entero equivalente indistintamente.

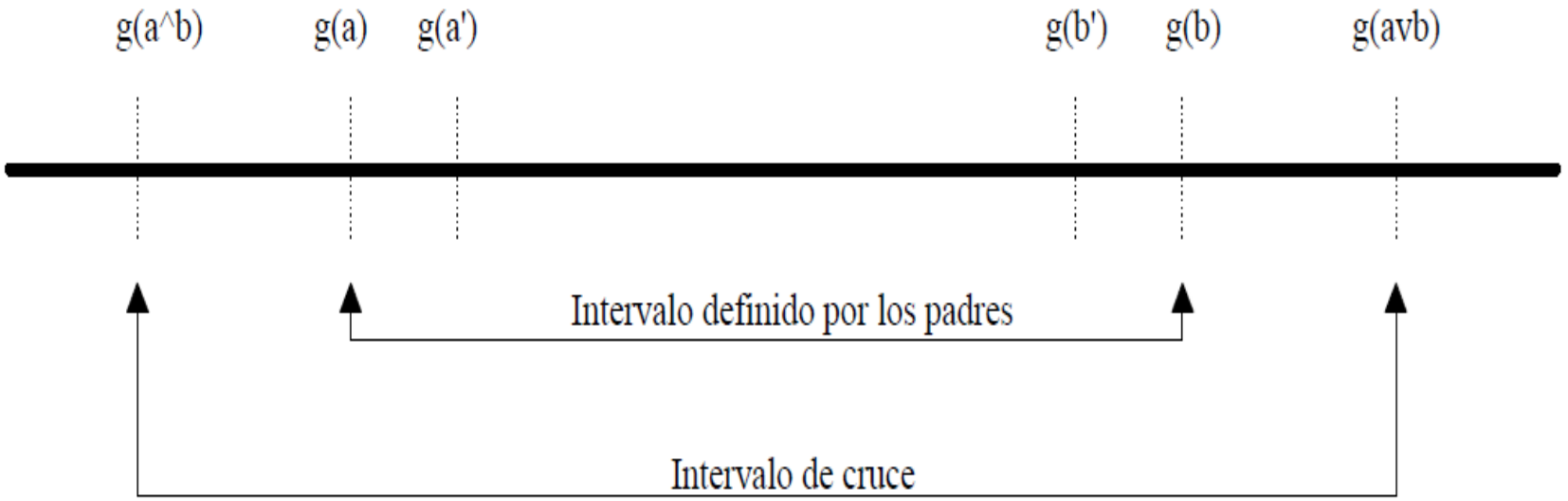
5.1.3 Operadores genéticos para representaciones reales y permutaciones (Cont.)

5. Dadas dos cadenas binarias a y b ∈ S, los descendientes mediante el operador de cruce generalizado serán los puntos a' y b' tales que:

$$a' \in g^{-1}\left(\left[g(a \wedge b), g(a \vee b)\right]\right) \qquad b' = g^{-1}\left(g(a) + g(b) - g(a')\right)$$

siendo el operador lógico *and* bit a b $\wedge$ bit, y $\vee$ el operador lógico *or* bit a bit.

A la cadena a $\wedge$ b se le denomina *cadena mínima*, mientras que a la cadena $a \wedge b$ la denomina *cadena máxima* ya que siempre se cumple que si $g(a) \leq g(b)$ entonces $g(a \wedge b) \leq g(a) \leq g(b) \leq g(a \vee b)$. Al intervalo $[g(a \wedge b), g(a \vee b)]$ se le denomina *intervalo de cruce* ya que siempre se cumple que $g(a')$ y $g(b') \in [g(a \wedge b), g(a \vee b)]$. La figura localiza de forma esquemática los progenitores y descendientes dentro del intervalo de cruce.



Progenitores y descendientes en el intervalo de cruce

5.1.3 Operadores genéticos para representaciones reales y permutaciones (Cont.)

Operadores de cruce con codificación real

- Los individuos con los que se trabaja en este apartado son de la forma a=a1,a, ∀i∈{1,..., / 1,a2,...,a/ donde ai ∈ ℜ . Si únicamente se recombinaran las cadenas asociadas a los individuos, nunca obtendríamos genes nuevos. Es decir los números reales que representan los genes serían siempre los mismos. De este modo, se dependería completamente de las mutaciones para generar nuevos genes. Se hace necesario por tanto el uso de nuevos operadores de cruce más apropiados para trabajar con números reales.

5.1.3 Operadores genéticos para representaciones reales y permutaciones (Cont.)

Operadores de cruce con codificación real

- Uno de los primeros operadores de cruce para cadenas codificadas con números reales es el *operador de cruce plano* de Radcliffe [RADC90]. Este operador toma dos cadenas de números reales a=a1,a2,...,a*l* y b=b1,b2,...,b*l* y genera dos descendientes c=c1,c2,...,c*l* y d=d1,d2,...,d*l* donde cada uno de los genes descendientes ci y di se obtiene de forma aleatoria del intervalo de cruce Ci , obtenido como Ci=[min(ai,bi),max(ai,bi)], para cada i∈{1,....*l*}.

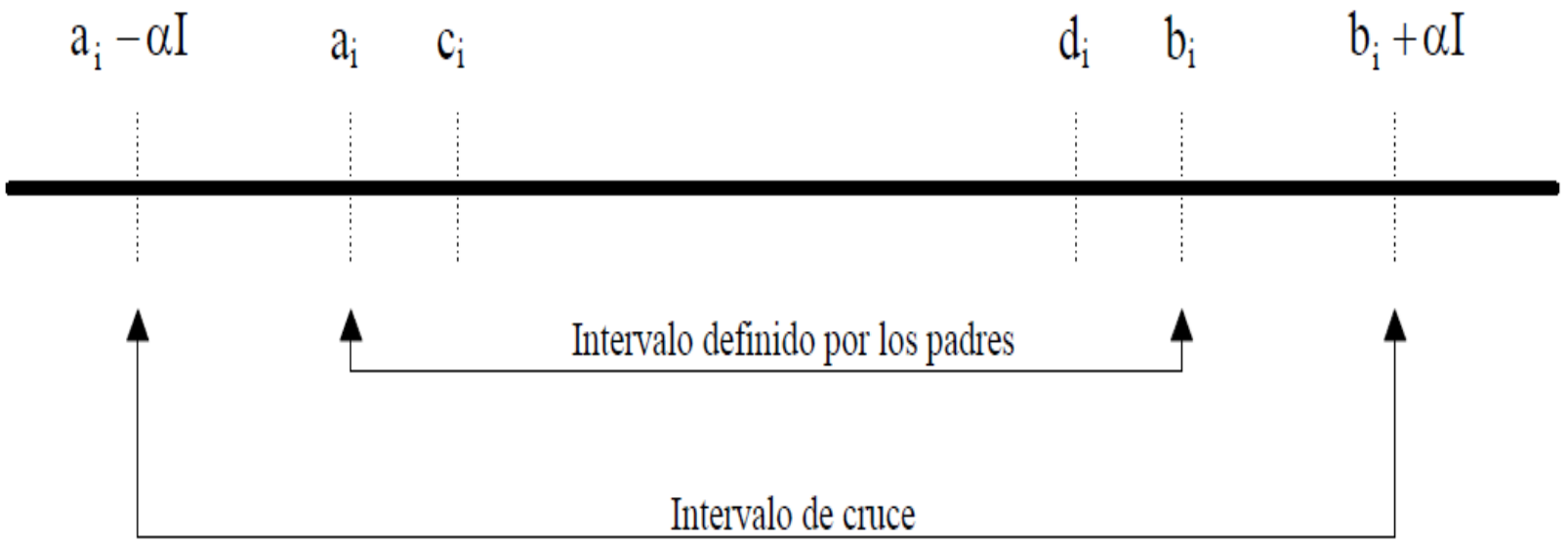
5.1.3 Operadores genéticos para representaciones reales y permutaciones (Cont.)

- El *operador de cruce lineal* de Wright [WRIG91] toma dos cadenas a y b como progenitores para generar tres descendientes c, d y e, donde para el caso del gen ci se calcula como el punto medio de ai y bi, es decir, ci = 2 *ai* + *bi* . Por otro lado el gen di se calcula como di = 1.5·ai – 0.5·bi , mientras que ei = -0.5·ai + 1.5·bi.

5.1.3 Operadores genéticos para representaciones reales y permutaciones (Cont.)

- Una generalización del operador de cruce plano es el operador de cruce combinado . Este operador toma dos progenitores a y b para generar dos nuevos descendientes c y d. Para cada par de genes ai y bi procedentes de las cadenas padres, y suponiendo *ai* ≤ *bi* , se crea el intervalo de cruce Ci calculado como Ci = [ai-αI , bi+αI], donde I = bi-ai y α∈[0,1] es un parámetro predeterminado de antemano. La figura muestra de forma esquemática el intervalo de cruce formado por los genes ai y bi de los progenitores. El descendiente ci se toma de forma aleatoria de este intervalo de cruce, siendo di = ai + bi –ci. A partir de la figura 3.4 se puede observar que este operador de cruce guarda un cierto paralelismo en su funcionamiento con el operador de cruce generalizado definido para cadenas binarias. Por otro lado, se puede ver que el valor del parámetro α determina cuánto se ensancha el intervalo de cruce con respecto al intervalo [ai,bi] determinado por los genes de las cadenas de los padres. Si α = 0, entonces el funcionamiento de este operador es el mismo que en el caso del operador de cruce plano. Por el contrario, si α = 1, la longitud del intervalo de cruce es la máxima posible, siendo de tres veces la longitud del intervalo determinado por los genes padres.

5.1.3 Operadores genéticos para representaciones reales y permutaciones (Cont.)



Operador de cruce combinado.

5.1.3 Operadores genéticos para representaciones reales y permutaciones (Cont.)

- Otro operador, desarrollado recientemente, es el *operador de cruce morfológico (MMX)*, que trabaja con poblaciones de individuos codificados con números reales normalizados. Este operador parte de un número n impar de cadenas progenitoras de longitud *l* a1,...,a*l* con ai = (ai0,...,ai*l*-1) i = 1..n, obtenidas sin repetición de la población actual, para generar dos nuevos individuos descendientes. Los valores de los genes de las cadenas progenitoras se representan mediante la denominada *matriz progenitora G*.

$$G = \begin{pmatrix} a_{10} & a_{11} & \dots & a_{1l-1} \\ a_{20} & a_{21} & \dots & a_{2l-1} \\ \dots & \dots & \dots & \dots \\ a_{n0} & a_{n1} & \dots & a_{nl-1} \end{pmatrix}$$

Para cada una de las *l* columnas de G se define el vector unidimensional fi como:

$$f_i = (a_{1i}, a_{2i}, \dots, a_{ni}) \text{ con } i = 0, \dots, l\text{-}1$$

Partiendo de estos *l* vectores columna: f0, ...,f*l*-1, el cruce morfológico lleva a cabo los siguientes pasos:

5.1.3 Operadores genéticos para representaciones reales y permutaciones (Cont.)

1. Cálculo de la medida de la diversidad genética.
2. Cálculo de los intervalos de cruce.
3. Obtención de la descendencia.

Se define como *medida de la diversidad genética* del gen i, el valor $gi \in [0,1]$, calculado aplicando el operador gradiente morfológico g sobre cada vector columna fi:

$$gi = g\,(E(n/2)+1)$$

donde E(x) es la función parte entera de x.

- Es decir, se toma como medida de la diversidad genética del gen número i al valor del gradiente morfológico aplicado sobre la componente situada en la posición media del vector columna fi de la matriz progenitora.

- Se procede entonces al cálculo de los *intervalos de cruce* C0,...,*Cl*-1. Para ello se hace uso de la denominada *función de exploración / explotación* o *función* ϕ , para calcular g_{imax} y g_{imin}:

5.1.3 Operadores genéticos para representaciones reales y permutaciones (Cont.)

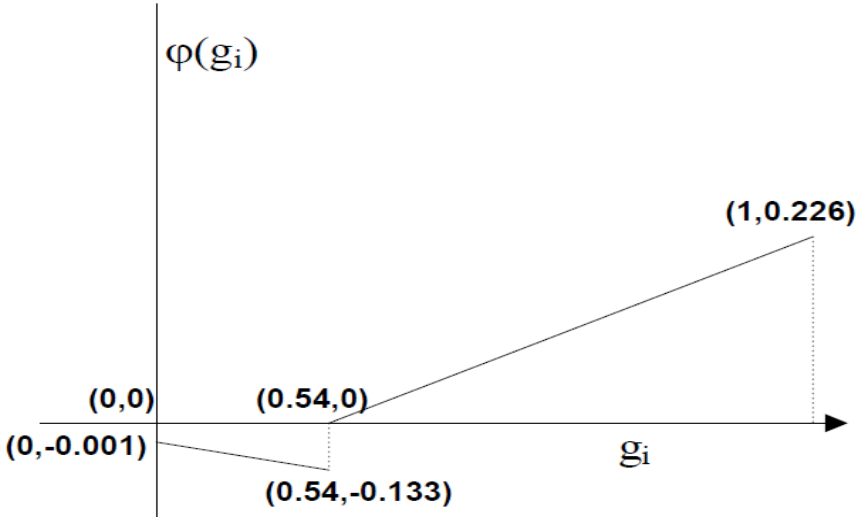
$$g_{imin} = mín(fi) + \phi(gi)$$
$$g_{imax} = máx(fi) - \phi(gi)$$

- Donde mín(fi) es el menor valor para el gen i (el menor valor del vector fi) y máx(fi) el mayor valor para el gen i (el mayor valor del vector fi). Cada intervalo de cruce estará definido entonces como $Ci=[g_{imin},g_{imax}]$, con i = 0,...,*l*-1.

- La función ϕ de exploración / explotación, es crucial en el funcionamiento del cruce morfológico, ya que determina cómo van a ser los intervalos de cruce. La figura 3.5. muestra una representación gráfica de la función ϕ, y su expresión analítica es la siguiente:

$$\varphi(g_i) = \begin{cases} -(0.25 \cdot g_i) - 0.001 \text{ si } g_i \leq 0.54 \\ (0.5 \cdot g_i) - 0.265 \text{ si } g_i > 0.54 \end{cases}$$

5.1.3 Operadores genéticos para representaciones reales y permutaciones (Cont.)



Función ϕ

Cuando los valores que toma algún determinado gen i en los individuos de la población son muy similares entre sí, éste está convergiendo hacia algún punto. En estos casos, el valor obtenido de gi será muy cercano a cero y por tanto, la función ϕ debe expandir el intervalo [min(fi), max(fi)], para permitir la exploración de nuevos puntos dentro del espacio de búsqueda, con el fin de evitar la convergencia hacia un punto que no sea el óptimo.

5.1.3 Operadores genéticos para representaciones reales y permutaciones (Cont.)

- Existe también la posibilidad de que los valores de un determinado gen sean muy diferentes unos de otros en las cadenas que forman la población actual. O bien, el caso general de que los individuos de la población estén muy dispersos dentro de todo el espacio de búsqueda. Este hecho se produce, por ejemplo, en la población inicial o en las primeras etapas del proceso de convergencia del AG. En estos casos, el valor de gi será muy elevado (más cercano a 1) y, por tanto, la función ϕ debe estrechar el intervalo de referencia. De esta forma, se evita el problema de la no convergencia por la continua exploración de nuevos puntos cuando la población está muy dispersa, consiguiendo, por el contrario, una convergencia muy rápida hacia algún punto. Si resultara ser el óptimo global, el AG finalizaría en muy pocas iteraciones. Si por el contrario, este proceso no conduce hacia la solución, sería entonces cuando se volverían a aplicar técnicas de exploración ensanchando el intervalo de cruce con respecto al de referencia, para explorar nuevas regiones del espacio de búsqueda. lado o'i se obtiene a partir de la expresión:

5.1.3 Operadores genéticos para representaciones reales y permutaciones (Cont.)

- A partir de los intervalos de cruce se generan las *cadenas descendientes* del operador, denotadas por o = (o0,...,o*l*-1) y o' = (o'0,...,o'*l*-1). oi es un valor aleatorio escogido dentro del intervalo de cruce Ci. Por otro lado o'i se obtiene a partir de la expresión:

$$o' = (\text{mín}\,(f_i) + \text{máx}\,(f_i) - o_i)$$

Este operador no solo tiene un coste computacional bajo, sino que mejora en la mayoría de los casos a otros operadores de cruce reales como el BLX-0.5 y el RFX. Utilizado para el entrenamiento de redes de neuronas artificiales proporciona muy buenos resultados, mejorando incluso el método de retro propagación del gradiente con factor momento.

5.1.4 Aplicaciones

Hay una gran cantidad de problemas que pueden ser resueltos mediante los algoritmos genéticos. Algunos de estos problemas son:

Caso 1: Optimización de rutas. Aplicado en la optimización de rutas, permite encontrar la ruta más corta o más rápida entre ciudades o zonas en muy poco tiempo. También resulta muy útil en ciudades inteligentes, para tratar de reducir las emisiones de CO2.

Caso 2: Optimización de tareas. Cuando se trata de realizar tareas en el tiempo más breve posible, optimizarlas es fundamental. Con los algoritmos genéticos, además el cálculo se hace rápido y eficaz.

Caso 3: Gestión automatizada de equipamiento industrial. Es posible hacer un cálculo en tiempo real para optimizar un proceso automatizado de equipamiento industrial.

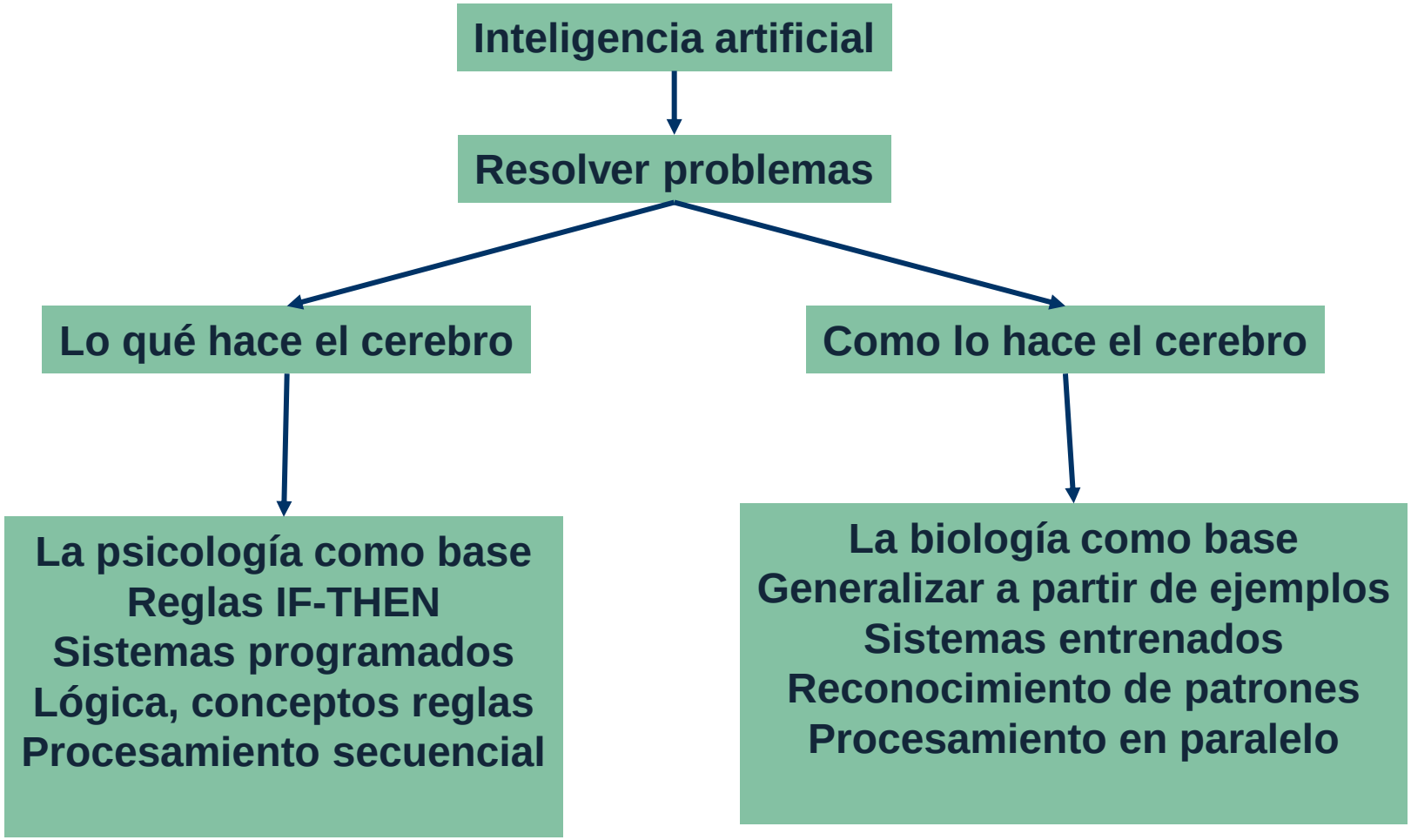
Caso 4: Aprendizaje de comportamiento de robots. El aprendizaje de robots es posible respecto a una función de coste. Con estos algoritmos el aprendizaje se realiza de un modo más rápido y eficaz.

5.1.4 Aplicaciones

Caso 5: Sistemas del sector financiero. Una aplicación interesante en finanzas es que estos algoritmos permiten descubrir reglas de inversión que indican cuándo entrar y salir de un mercado para obtener los máximos beneficios. Por otro lado, en métodos de Splitwise, se puede optimizar el método de compartir gastos entre distintos usuarios.

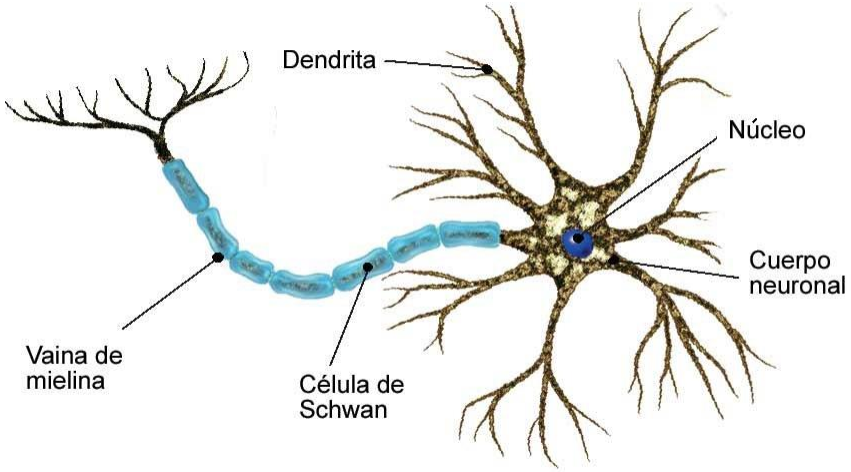
Caso 6: Encontrar errores en programas. Permite detectar errores en los programas de los desarrolladores, lo que ayuda a ahorrar tiempo y dinero en su implementación.

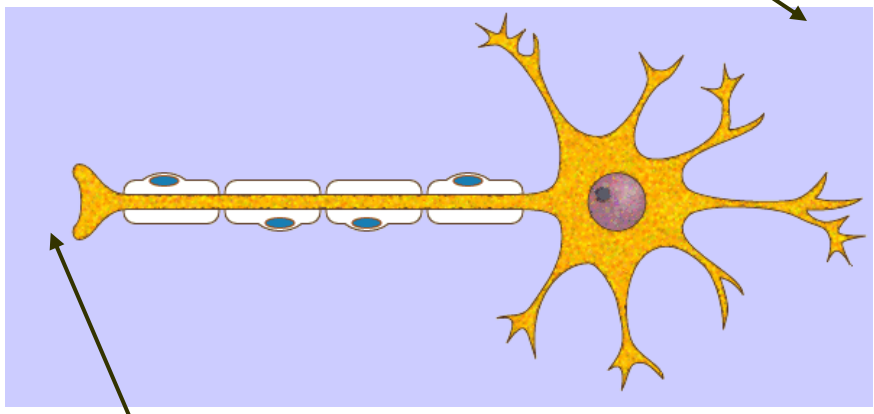
5.2 Redes neuronales



5.2.1 Modelo biológico y modelo artificial

	Cerebro	Computadora
velocidad de proceso	100 Hz	4GHz
tipo de proceso	paralelo	secuencial
número de procesadores	10E11-10E12	pocos
conexiones	10000	pocas
almacenamiento de conocimiento	distribuido	direcciones fijas
tolerancia a fallos	amplia	nula
tipo de contron del proceso	auto-organizado	centralizado





5.2.2 Modelo de McCulloch-Pitts

En los años 50 dos neurocientíficos, Warren McCulloch y Walter Pitts, propusieron un modelo básico de neurona. Aunque McCullcoch y Pitts no fueron los primeros en considerar las neuronas como sistemas para realizar cálculos, fueron los pioneros en el intento de definir formalmente a las neuronas como elementos computacionales y explorar las consecuencias de las propiedades neuronales.

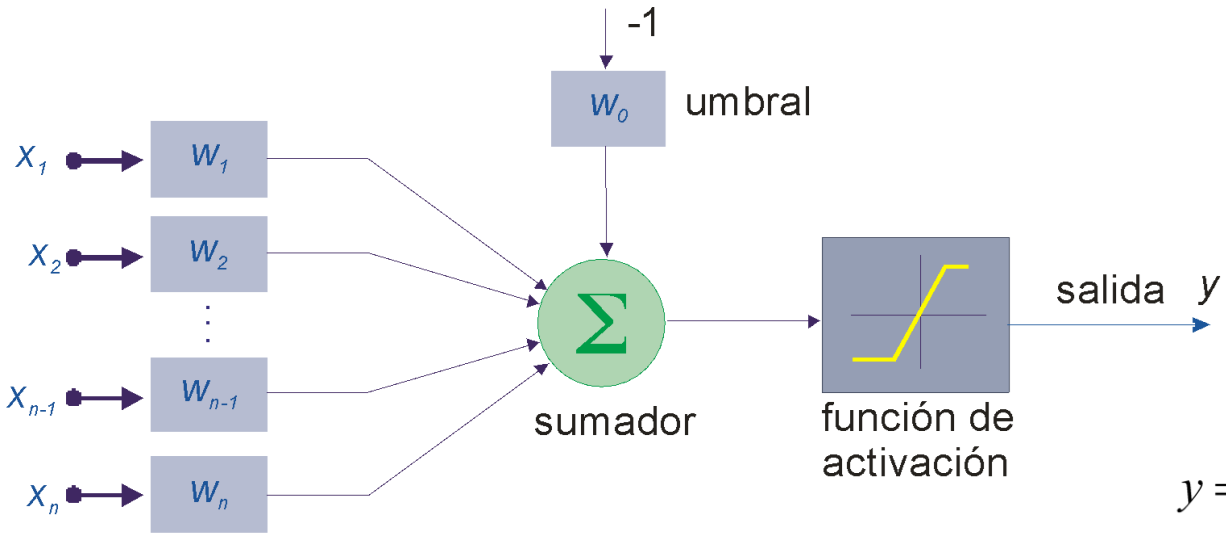
Modelo de una RNA

Entradas x_i, que representan señales que vienen de otras neuronas o señales del exterior al sistema.

Pesos w_i, que corresponden a intensidades de una sinapsis.

Umbral w_0 con valor de -1, que es un peso que se aplica a una señal de entrada, el cual se debe sobrepasar para activar la neurona.

Función de activación *f( )*.



en la literatura el umbral w_0 también se conoce como θ

$$y = f\left(\sum_{i=1}^{n} w_i \cdot x_i - w_0\right)$$

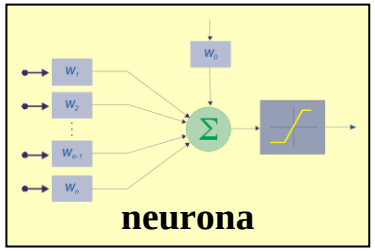
- Por la topología de la red:
  - Monocapa
  - Multicapa

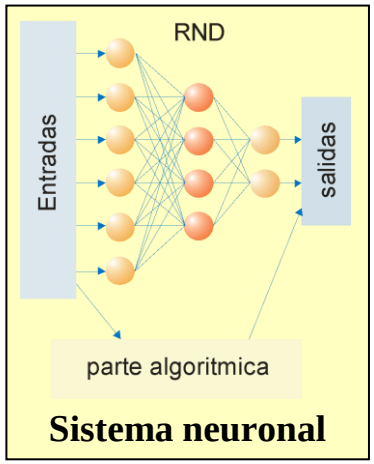
Topología de una RNA

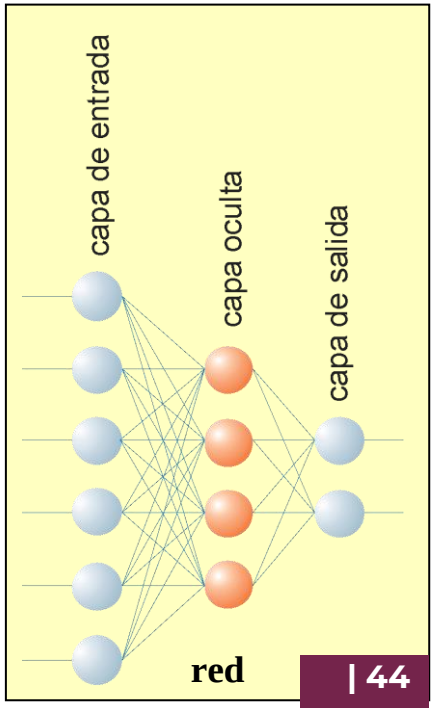
La topología hace referencia a:

- Organización de la RN
- Número de neuronas
- La forma en que se conectan estas neuronas.
- Cada neurona puede tener una función de activación diferente

- Los elementos simples de una red se agrupan en capas.
- Se pueden distinguir tres tipos de capas:
  - Capa de entrada
  - Capa oculta (puede existir o no)
  - Capa de salida







Topología de una RNA (cont.)

- Se pueden restringir tres tipos de redes:
  - Redes unidireccionales o redes de propagación hacia delante.
  - Redes recurrentes o redes de propagación hacia atrás.
  - Redes autoorganizadas.

Redes unidireccionales

- Todas las conexiones entre neuronas van en una sola dirección.
- Son usadas para convertir un conjunto de datos específicos en otro también especifico.
- El aprendizaje es supervisado a través de un proceso de ajuste de pesos.
- Tiene una arquitectura multicapa.
- La información que recibe cada neurona se transmite a la siguiente capa.
- La conexión entre neuronas puede ser total o parcial
- Con este tipo de red con una capa oculta puede aproximar cualquier función continua.
- Una red con dos capas ocultas puede aproximar cualquier función.
- El número de neuronas crece exponencialmente con respecto al número de entradas.

Redes recurrentes

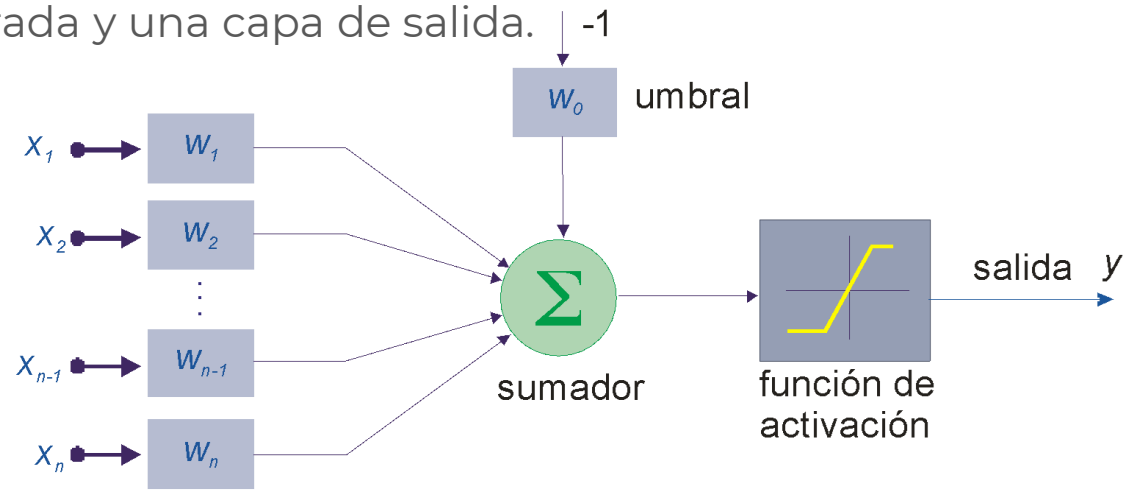
- Pueden almacenar información.
- La red funciona como un sistema dinámico cuyos puntos de equilibrio son la información almacenada.
- Arquitectura monocapa con retroalimentación
- La retroalimentación puede ser de una misma neurona, de una neurona a otra de la misma capa y de una capa a otra.

Redes autoorganizadas

- Entrenamiento con aprendizaje no supervisado, denominado competitivo.
- Se desconocen las salidas.
- Sirven para clasificar datos sin saber la clasificación.
- Cada dato clasificado forma parte de un subconjunto o cluster que posee atributos comunes o clustering.
- Se necesita un método de clustering para realizar la clasificación.
- Tiene una arquitectura de una sola capa donde cada neurona representa un cluster.
- Cada neurona tiene una cierta conexión con neuronas colaterales.

5.2.3 Perceptrón simple y multicapa

El perceptrón simple consta de una sola neurona o bien de Una capa de neuronas de entrada y una capa de salida.
El modelo simple es:



Se tiene que $y=f(\ )$ en R en un intervalo {-1, 1}
Un patrón de entrenamiento puede integrarse con:
Entrada x y Salida z

$$x = (x_1, x_2, \ldots, x_n)^T \quad \in R$$

$$z \in \{1,0\}$$

se desea obtener

$$z = f(x)$$

$$u = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$$

En general una red neuronal se puede entrenar con un conjunto de parejas de patrones:

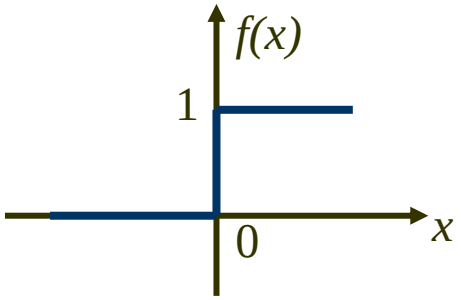
$$\{\{x^1, z^1\}, \{x^2, z^2\}, \ldots, \{x^p, z^p\}\}$$

w: pesos sinápticos
w_0: umbral
u: potencial sináptico que es la suma ponderada de los valores de entrada x_i

5.2.3 Perceptrón simple y multicapa (cont.)

Si se tiene una función de transferencia binaria:

$$f(x)=\begin{cases}1 & \text{si } w_1x_1+w_2x_2+\ldots+w_nx_n \geq w_0\\ 0 & \text{si } w_1x_1+w_2x_2+\ldots+w_nx_n < w_0\end{cases}$$



determinar pesos

1. **Inicia con *w* aleatorios.**
2. **Modificar los pesos si la salida no coincide con la deseada.**

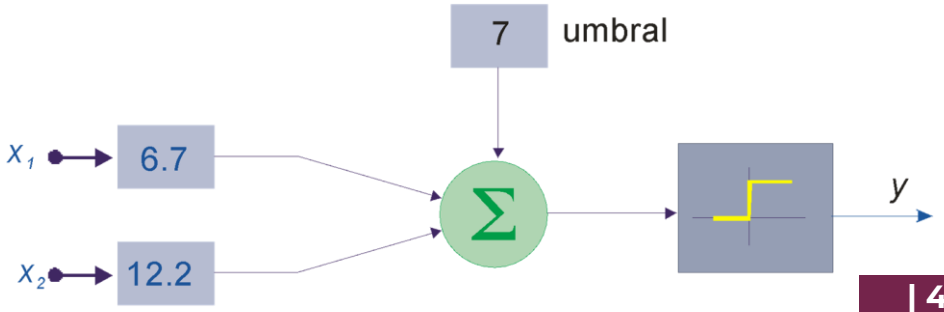
$$w_j(k+1)=w_j(k)+\Delta w_j(k)$$
$$\Delta w_j(k)=\eta\,(k)[\text{error}]x_j$$

***k*: es la iteración de entrenamiento**
***η*: constante de aprendizaje**
Como datos se tiene:
Un patrón de entrada *x*={…}
Una salida *z*
Una función de transferencia *y=f(x)*

Ejemplo:
si el patrón es *x*={3,5} y *z*=1, w_0=7
si la función de transferencia es binaria:

$$f(x)=\begin{cases}1 & \text{si } w_1x_1+w_2x_2 \geq w_0\\ 0 & \text{si } w_1x_1+w_2x_2 < w_0\end{cases}$$

se inicia con valores arbitrarios w_1=0.1, w_2=1.2 y η=2.2
$w_1x_1+w_2x_2$=0.3+6=6.3<(w_0=7) ∴ *y*=0
se modifican los pesos con:
$w_1(2)=w_1(1)+\eta(z-y)x_1$=0.1+2.2(1-0)(3)=6.7
$w_2(2)$=1.2+2.2(1-0)(5)=12.2
$w_1x_1+w_2x_2$=6.7(3)+12.2(5)=20.1+61=81.1
81.1≥7 ∴ *y*=1



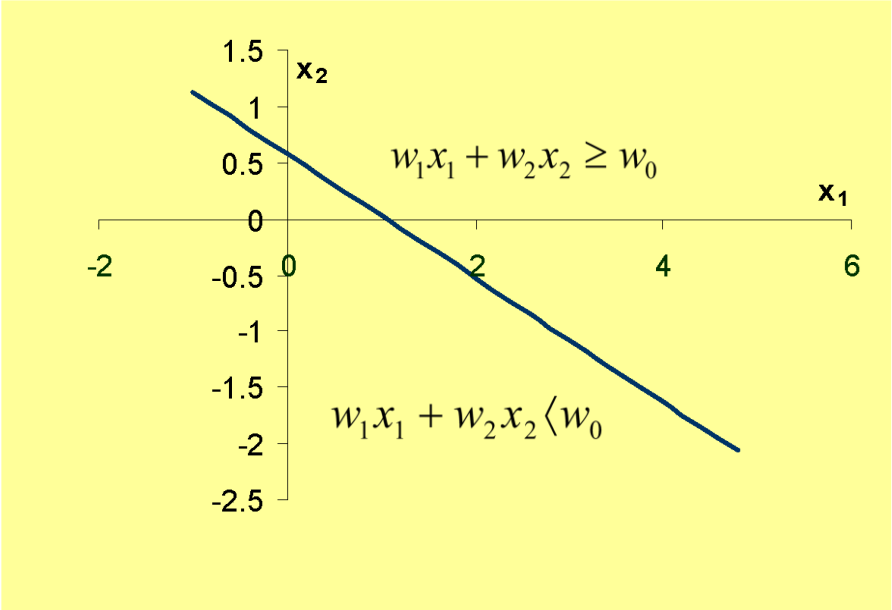
5.2.3 Perceptrón simple y multicapa (cont.)

si se introduce x={5,7}
$w_1x_1+w_2x_2=118.9 \geq 7 \therefore y=1$

si se introduce x={-8,3}
$w_1x_1+w_2x_2=-17 < 7 \therefore y=0$

el perceptrón esta funcionando como un clasificador binario. Dice que patrones son 1 y cuales 0.

Con el patrón de entrenamiento x={3,5} y z=1, w_0=7,se obtuvieron los pesos: w_1=6.7 y w_2=12.2
y la función de activación para valor 0

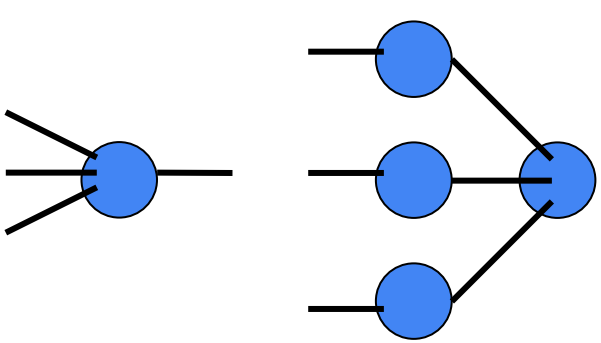


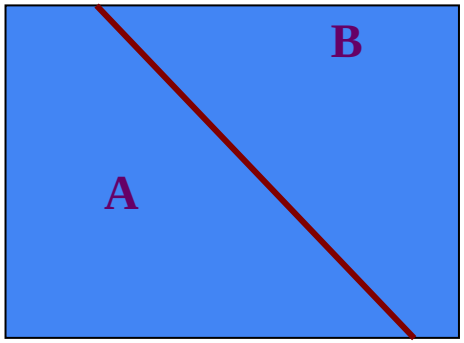
$$w_1x_1 + w_2x_2 \geq w_0$$

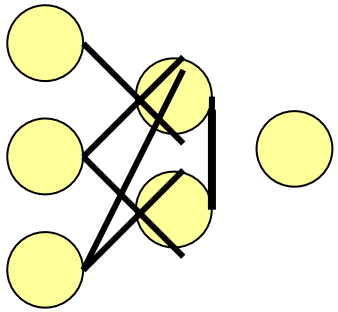
$$x_2 \geq \frac{w_0}{w_2} - \frac{w_1}{w_2}x_1 \qquad x_2 \geq 0.574 - 0.55x_1$$

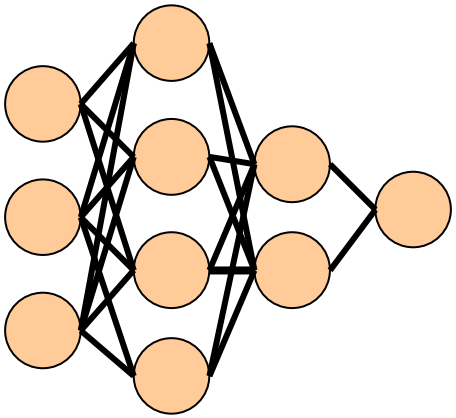
5.2.3 Perceptrón simple y multicapa (cont.)

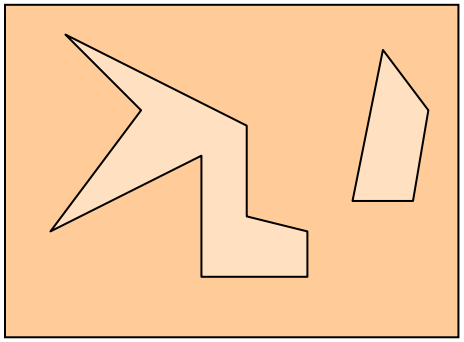
Tipos de regiones de clasificación del perceptrón.

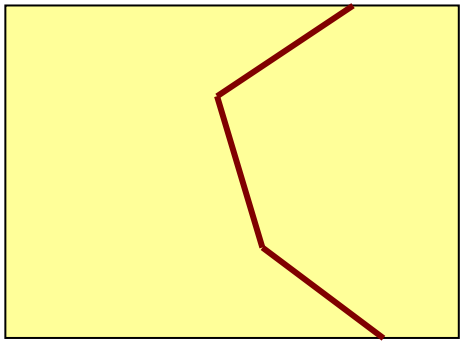








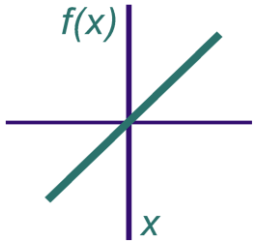




5.2.3 Perceptrón simple y multicapa (cont.)
Funciones de activación

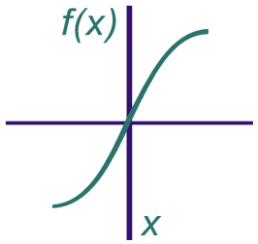
- La salida de la red está determinada por una función de activación o función de transferencia.
- Es una función matemática que relaciona la información que llega a la neurona y el estado de activación.
- La función de activación puede ser lineal o no lineal y puede ser elegida para cada tipo de problema a resolver.
- Existen diferentes funciones de activación cuya elección depende de:
  - Tipo de red empleada
  - La función que realiza la neurona.
  - La interpretación de la salida de la red neuronal

5.2.3 Perceptrón simple y multicapa (cont.)
Funciones de activación (cont.)



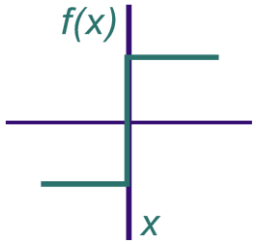
$y=x$

[-∞,+∞]



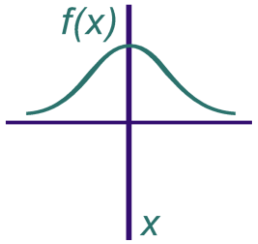
$y = \frac{1}{1+e^{-x}}$ [0,+1]

$y = tgh(x)$ [-1,+1]

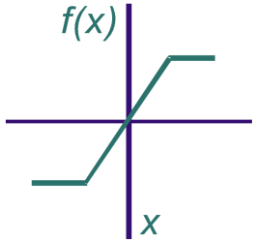


$y=\text{signo}(x)$ {-1,+1}

$y=H(x)$ {0,+1}

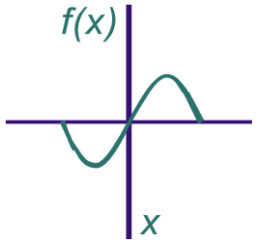


$y = Ae^{-Bx^2}$ [0,+1]



$$y = \begin{cases} -1, & \text{si } x\langle -1 \\ x, & \text{si } +1 \le x \le -1 \\ +1, & \text{si } x\rangle +1 \end{cases}$$

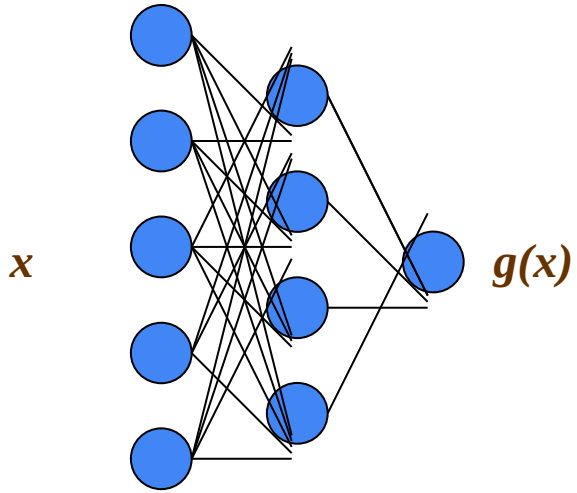
[-1,+1]



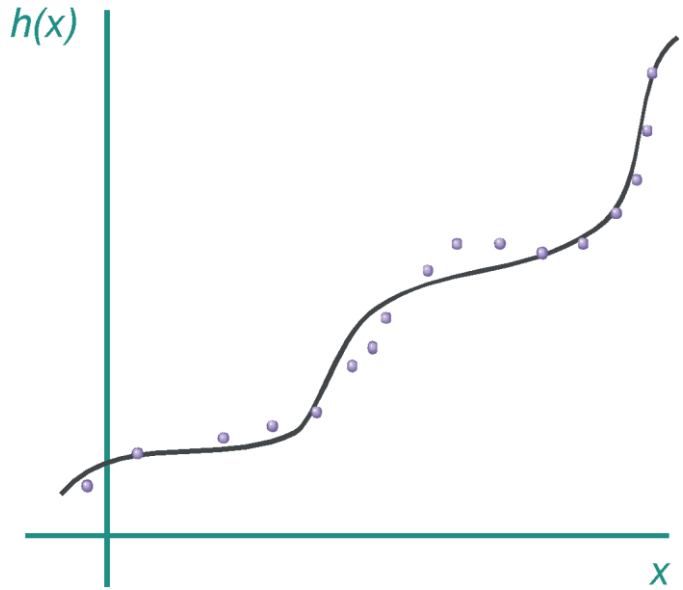
$y = Asen(\omega x + \varphi)$ [-1,+1]

5.2.3 Perceptrón simple y multicapa (cont.)

Un perceptrón multicapa con una capa oculta y función de activación lineal en la salida puede hacer una aproximación arbitrariamente cercana a cualquier función continua dado el número suficiente de neuronas ocultas.



No existe una técnica formal para determinar el número de neuronas y número de capas ocultas para resolver determinados problemas.



$$g(x,w) = \sum_{j=0}^{N_s} w_{s,1,j} f\left(\sum_{i=0}^{N_e} w_{e,j,i} x_i \right)$$

5.2.3 Perceptrón simple y multicapa (cont.)

- Entrenamiento con propagación hacia atrás.
  - Minimizar una función de coste de suma de errores cuadráticos.
  - El entrenamiento implica utilizar derivadas parciales.
  - La inicialización de los pesos es tomando valores aleatorios y pequeños.
  - El parámetro de aprendizaje suele ser igual para todas las neuronas.
- Para una clasificación binaria, con una capa oculta

$$y = g(x) \quad \in \{-1,+1\}$$

M: número de neuronas ocultas.
W: número de pesos en la red.
N: número de ejemplos de entrenamiento.
ε: fracción de errores permitidos.

$$N \geq \frac{32W}{\varepsilon} \ln\left[\frac{32M}{\varepsilon}\right]$$

$$N \rangle \frac{W}{\varepsilon}$$

Ejemplo. Reconocimientos de los números {0,1,2,...,9} escritos a mano.

Entrada *x* de una matriz de imagen 16x16,

si M=12;

W=((16x16+1)+12)x10=2690

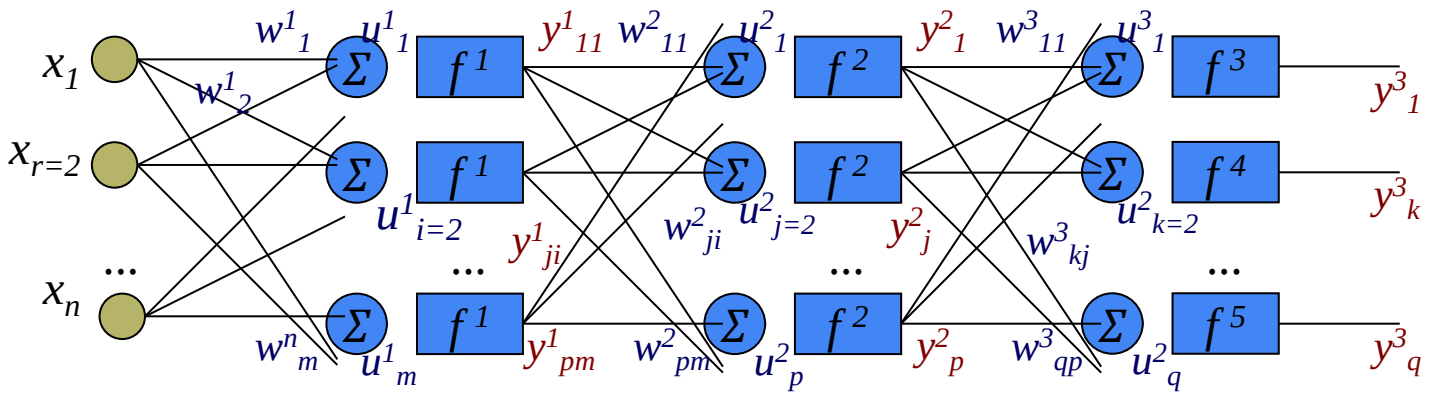
Si se permite un error del 10%

se necesitan más de 26900 ejemplos

5.2.4 Propagación hacia atrás

- El entrenamiento de una red neuronal se refiere a encontrar los pesos de cada neurona que se tienen con patrones de entrada específicos.
- El entrenamiento de propagación hacia atrás es un entrenamiento supervisado, es decir, existen parejas de conjuntos de datos entrada-salida.
- El algoritmo de entrenamiento de propagación hacia atrás aprovecha la naturaleza paralela de las redes neuronales para reducir el tiempo requerido por un procesador secuencial para determinar la correspondencia entre unos patrones dados y se basa en un error medio cuadrático para el cálculo de pesos.

De manera general una red neuronal multicapa tiene su salida como evaluaciones en las funciones de transferencia de cada capa. Por ejemplo para una red de 3 capas:



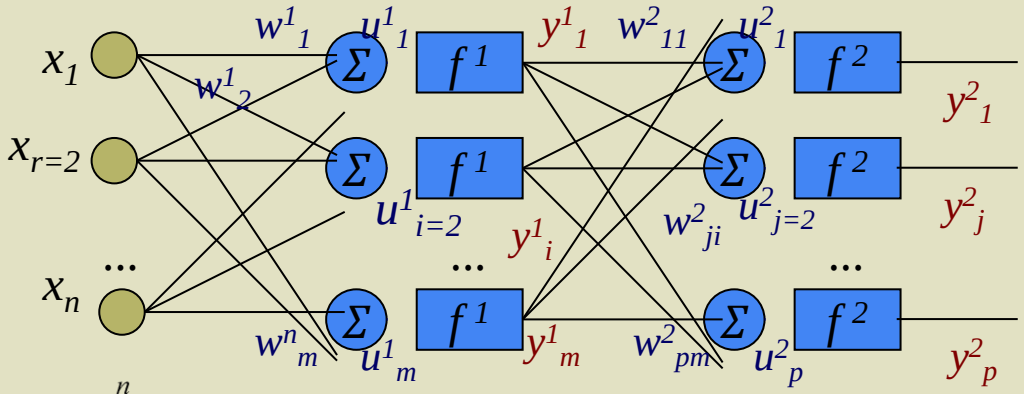
$$y^1_i = f^1(w^i_r x_r + \theta^1_i) \qquad y^2_j = f^2(w^j_{ji} y^1_i + \theta^2_j) \qquad y^3_k = f^3(w^k_{kj} y^2_j + \theta^2_k)$$

$$y^3_k = f^3(w^k_{kj} f^2(w^j_{ji} f^1(w^i_r x_r + \theta^1_i) + \theta^2_j) + \theta^3_k)$$

5.2.4 Propagación hacia atrás (cont.)

- El proceso general de entrenamiento es:
  - Un patrón de entrenamiento **x** se propaga hacia adelante hasta obtener una salida.
  - La salida se compara con la respuesta deseada y se calcula un error cuadrático medio.
  - Este error marca el camino adecuado para actualizar pesos y acercarse a la salida deseada.
  - El error cuadrático medio se minimiza en cada iteración del entrenamiento.

Si se tiene una red neuronal multicapa de 3 capas, entrada, oculta y salida:



$$u_i^1 = \sum_{r=1}^{n} w_r^i \cdot x_r + \theta_i^1 \qquad u_j^2 = \sum_{i=1}^{m} w_{ji}^2 \cdot y_i^1 + \theta_j^2$$

$$y_i^1 = f^1\left(\sum_{r=1}^{n} w_r^i \cdot x_r + \theta_i^1\right) \qquad y_j^2 = f^2\left(\sum_{i=1}^{m} w_{ji}^2 \cdot y_i^1 + \theta_j^2\right)$$

m salidas en la capa oculta

p salidas en la capa de salida

El conjunto de un patrón A de entrenamiento es:

$$^{A}\{\{x_1,x_2,..,x_n\},\{z_1,z_2,...,z_p\}\}$$

comparando cada salida y_j con la salida deseada z_j $\quad \delta_j = \left(z_j - y_j^2\right)$

se obtiene el error δ en la neurona de salida *j*. El error debido al patrón A es

$$ex_A^2 = \frac{1}{2}\sum_{j=1}^{p}\left(\delta_j\right)^2$$

Con una cantidad N de patrones de entrenamiento, el error total en una iteración de entrenamiento es:

$$e^2 = \sum_{t=1}^{N} ex_t^2$$

5.2.4 Propagación hacia atrás (cont.)

El error total de la RNA con respecto a los pesos *w* de la neurona genera un espacio de *W* dimensiones.

Siendo *W* en número total de pesos

Al evaluar el gradiente del error en algún punto de esta superficie se obtiene la dirección en la cual la función del error crece.

Como se quiere minimizar el error se toma la dirección negativa del gradiente y la actualización en la iteración *k* de la matriz **w** de pesos de la RN se da con:

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \eta \ \nabla ex^2$$

$$-\nabla ex^2 = -\frac{\partial (ex^2)}{\partial (w_{ji}^2)}$$

Siendo la derivada parcial del error de producido por el patrón **x** con respecto a todos los pesos w_{ji} de la capa oculta.

$$-\frac{\partial (ex^2)}{\partial (w_{ji}^2)} = -\frac{\partial}{\partial (w_{ji}^2)}\left(\frac{1}{2}\sum_{j=1}^{p}\left(z_j - y_j^2\right)^2\right) = \left(z_j - y_j^2\right)\frac{\partial (y_j^2)}{\partial (w_{ji}^2)}$$

esta derivada parcial resultante se calcula con la regla de la cadena y queda

$$\frac{\partial (ex^2)}{\partial (w_{ji}^2)} = \left(z_j - y_j^2\right)\frac{\partial (y_j^2)}{\partial (u_j^2)}\frac{\partial (u_j^2)}{\partial (w_{ji}^2)}$$

Reemplazando las derivadas parciales resultantes:

$$\frac{\partial (ex^2)}{\partial (w_{ji}^2)} = \left(z_j - y_j^2\right) f'^{2}(u_j^2)\, y_i^1$$

la sensitividad de la capa de salida está dada por:

$$\delta_j^2 = \left(z_j - y_j^2\right) f'^{2}(u_j^2)$$

de una manera análoga se encuentra el gradiente del error para la capa oculta:

$$-\frac{\partial (ex^2)}{\partial (\mathbf{w}_i^1)} = \sum_{j=1}^{p} \delta_j^2 \cdot \mathbf{w}_{ji}^2 \cdot f'^{1}(u_i^1) \cdot \mathbf{x}_i$$

la sensitividad de la capa oculta es:

$$\delta_i^1 = f'^{1}(u_i^1) \cdot \sum_{j=1}^{p} \delta_j^2 \cdot w_{ji}^2$$

5.2.4 Propagación hacia atrás (cont.)

Una vez calculadas las sensitividades, tanto para la capa oculta como para la de la salida, se actualizan los pesos de la capa de salida con:

$$\mathbf{w}_{ji}^{2}(k+1) = w_{ji}^{2}(k) - 2\eta\,\delta_{j}^{2}$$

$$\theta_{j}^{2}(k+1) = \theta_{j}^{2}(k) - 2\eta\,\delta_{j}^{2}$$

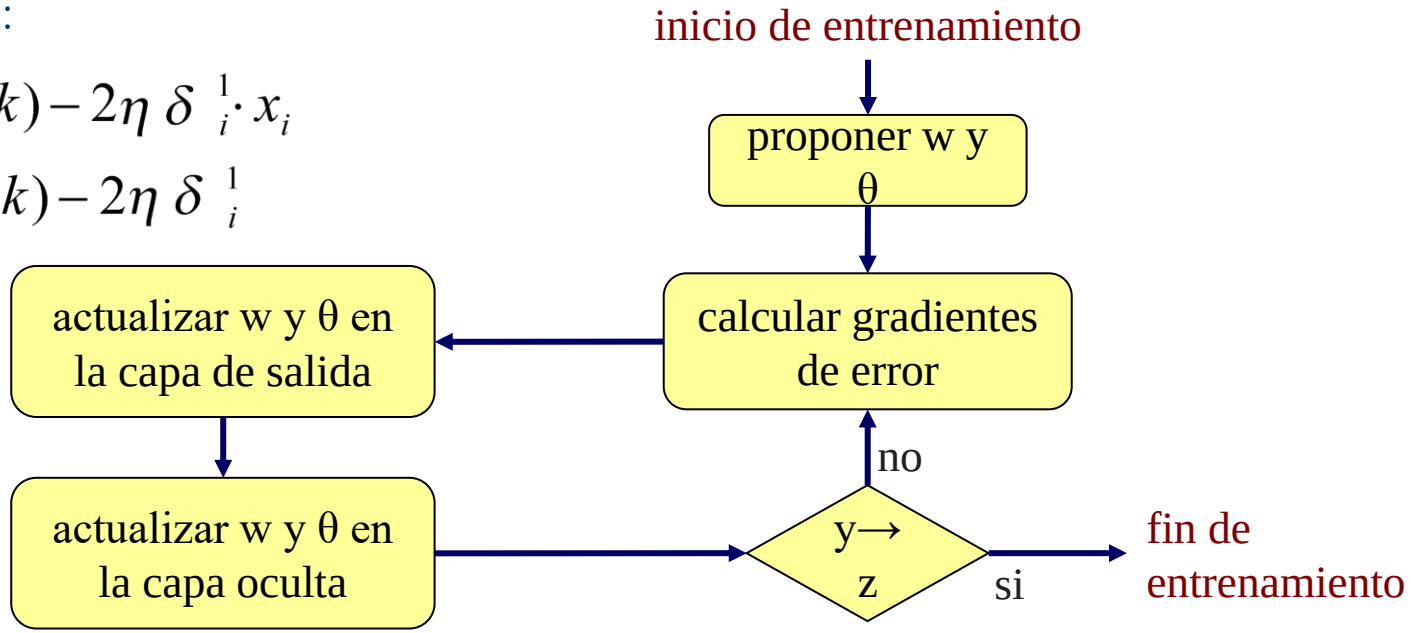
siendo η la constante de aprendizaje de la neurona, cuyo valor esta entre 0 y 1.

Después de actualizar los pesos de la capa de salida se actualizan los pesos de la capa oculta con:

$$\mathbf{w}_{i}^{1}(k+1) = \mathbf{w}_{i}^{1}(k) - 2\eta\,\delta_{i}^{1}\cdot x_{i}$$

$$\theta_{i}^{1}(k+1) = \theta_{i}^{1}(k) - 2\eta\,\delta_{i}^{1}$$

algoritmo de entrenamiento hacia atrás



5.2.5 Aplicaciones

Economía y Finanzas

- Minado de datos
- Descubrimiento de fraudes
- Análisis de mercado
- Pronóstico de existencias
- Aplicaciones de créditos
- Selección de personal
- Análisis de mercados
- Localización de evasores fiscales
- Predicción

HNC software.- Ubicación de publicidad en www.

Britvic.- Predicción de ventas en bebidas refrescantes

- Bienes Raíces

Day & Zimmerman, Inc.- Tasación de bienes inmuebles

- Negocios

Citybank.- Análisis de tendencias de mercados

John Deere & Company.- Fondo de pensiones

5.2.5 Aplicaciones (Cont.)

- Detección de fraudes

Dunn and Bradstreet.- Aprobación de cheques

Mastercard.- Identificación de hábitos de compra

Bank of América, Canadian Imperial Bank of Commerce

First Bank USA

- Solvencia de crédito

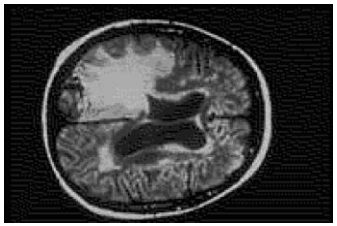
Chase Financial Technologies.- Predicción del valor de un crédito.

5.2.5 Aplicaciones (Cont.)

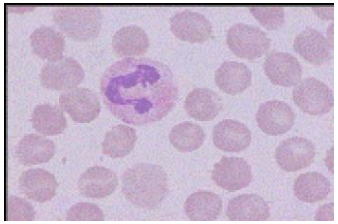
Medicina y salud

- Diagnóstico de enfermedades
- Neuro prótesis
- Conteo en microscopía
- Diagnóstico psiquiátrico
- Diagnóstico en electroencefalogramas
- Optimización en entrenamiento atlético
- Diferenciación de soplos cardiacos

Diferenciación de soplos cardiacos



Análisis de electroencefalogramas



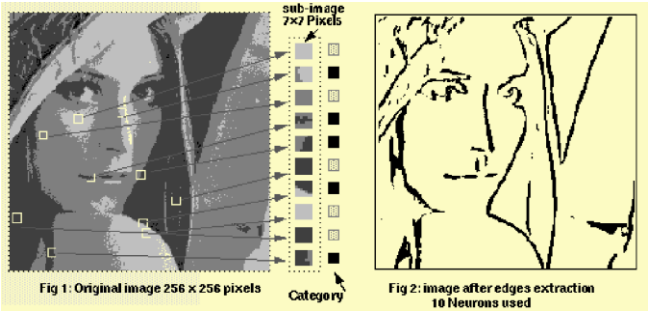
Conteo en microscopía

5.2.5 Aplicaciones (Cont.)

Ciencia e Ingeniería

- Reconocimiento de partículas
- Reconocimiento de blancos
- Detección de contorno
- Supresión de ruidos
- Reconocimiento de personas por el iris
- Predicción de carga eléctrica
- Control en presas
- Análisis de datos de vuelo de helicóptero

Eliminación de ruido

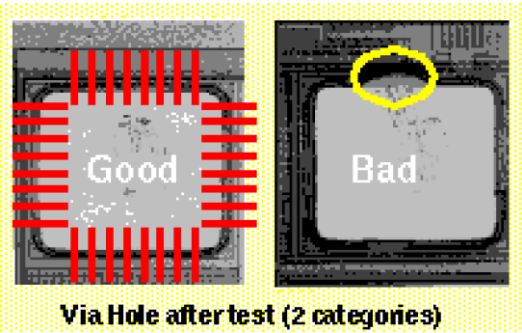


Detección de contorno

5.2.5 Aplicaciones (Cont.)

Industria

- Diagnóstico en tarjetas de CI
- Juguetes
- Control de molinos
- Control de proceso
- Identificación de productos dañados
- Optimización de horario de producción
- Análisis de fabricación y venta
- Predicción de temperatura
- Análisis de aroma y olor
- Desarrollo de productos
- Garantía de calidad (pruebas de pureza)
- Sistemas de reconocimiento visual
- Control de calidad en papel, plásticos, soldadura, pintura, bocinas, chips

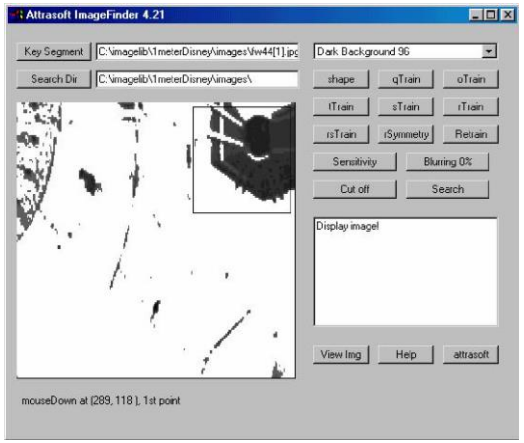


Detección de errores en CI

5.2.5 Aplicaciones (Cont.)

Medios de Comunicación

- Reconocimiento óptico de caracteres impresos por máquinas
- OCR de caracteres hechos a mano
- Reconocimiento de gráficas
- Reconocimiento de imágenes
- Reconocimiento de rostros
- Reconocimiento de huellas digitales
- Reconocimiento satelital
- Lectura de labios
- Reconocimiento de gestos
- Monitoreo del movimiento ocular
- Localización acústica
- Detección de fallas en autotransportes
- Reconocimiento óptico de estampillas
- Cancelación de eco en señales



Reconocimiento de imágenes satelital

Reconocimiento de huellas digitales