SCALAB

Universidad Carlos III de Madrid

Técnicas bioinspiradas

En este tema

Técnicas bioinspiradas

- Redes de neuronas
  - Introducción
  - Perceptrón Simple
  - Perceptrón Multicapa
  - Aplicaciones
- Algoritmos Evolutivos
  - Introducción
  - Algoritmo General
  - Implementaciones

En este tema

Técnicas bioinspiradas

Redes de neuronas

- El sistema nervioso de muchos animales es capaz de realizar tareas de las que hemos identificado en Aprendizaje Automático
- La información está representada en una red de elementos simples (enfoque conexionista) con un elevado grado de paralelismo
- Más que la configuración física de las conexiones, el aprendizaje se produce gracias al efecto facilitador de los neurotransmisores en algunas sinapsis nerviosas

Redes de neuronas Artificiales.

- Surgieron como modelo biológico de inteligencia, aunque hoy en día están considerados una forma más de programar
- Son útiles para aprender **funciones** de valores reales, aunque pueden aprender también funciones de valores discretos
- Basadas en unidades de cómputo que:
  - reciben un conjunto de señales de entrada (otras neuronas o del exterior)
  - se reciben a través de arcos (conexiones) que tienen un peso
  - procesan la información recibida → según los pesos
  - emiten una señal de salida
- Aprendizaje (supervisado o no): modificación de los pesos de la red

↳ Lo importante

↳ Lo Para determinar los pesos.

Historia

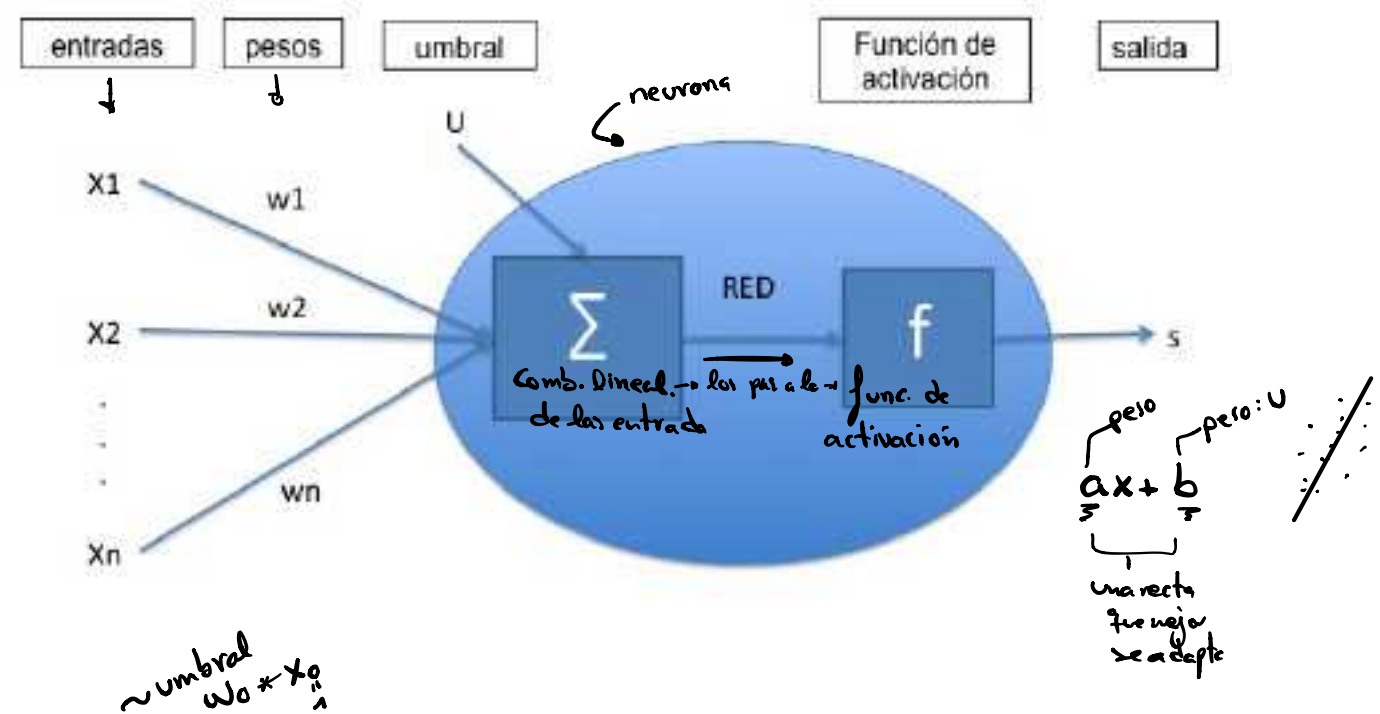
- McCulloch y Pitts (1949) mostraron el concepto de conexión y la capacidad de un grupo de células de M-P conectadas para llevar a cabo la implementación de ciertas funciones lógicas
- Rosenblatt (1958) introdujo la primera arquitectura de red de neuronas artificial con capacidad de aprendizaje: perceptrón simple
- Minsky y Papert en 1969 y demuestran las serias limitaciones de dicha red, y la mayor parte de los investigadores en este área abandonaron su trabajo
- A principios de la década de los 80, las redes de neuronas artificiales renacen
- Rumelhart, McCelland and Williams (1986) propusieron el perceptrón multicapa y el algoritmo de aprendizaje por retropropagación

2010 → Deep Learning.

Computación con Redes de Neuronas

- Propiedades:
  - Procesamiento colectivo, asíncrono y paralelo
  - Capacidad de aprendizaje
  - Capacidad de aproximación a partir de ejemplos
  - Tolerancia a fallos
- Se utilizan en:
  - aproximación, predicción, clasificación
  - reconocimiento de patrones (imagen, voz, caracteres)
  - compresión y análisis de datos
  - robótica

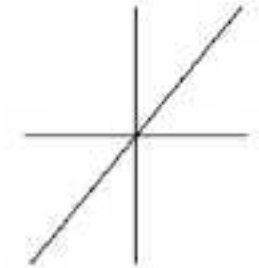
Estructura de una neurona



- $RED = U + w_1 * X_1 + w_2 * X_2 + \ldots + w_n * X_n = U + \sum_{i=1}^{n} w_i x_i$
- $S = f(RED)$

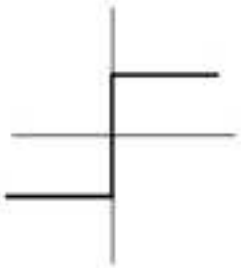
Tipos de funciones de activación : Dada una entrada devuelve una salida.

Funciones de activación



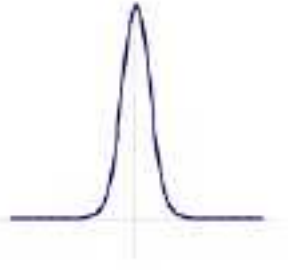
Función lineal

$f(x)=x$



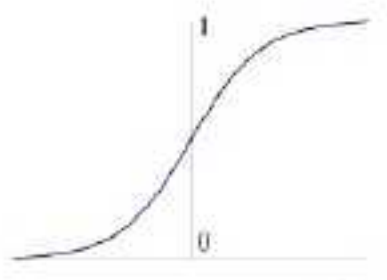
Función umbral o escalon

$$f(x) = \begin{cases} f1 & / \quad x > 0 \\ -f1 & / \quad x \leq 0 \end{cases}$$



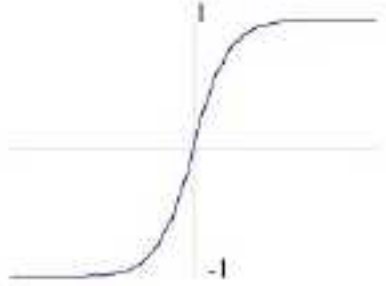
Función gaussiana

$$f(x) = e^{\frac{-x^2}{2}}$$



Función en (0,1)

$$f(x) = \frac{1}{1+e^{-x}}$$



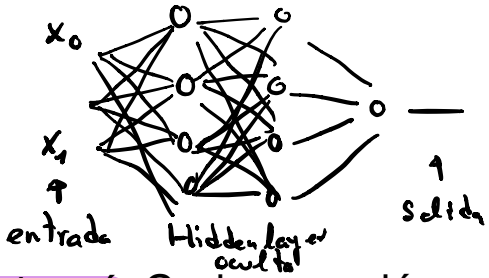
Función en (-1,1)

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Funciones sigmoidales

Redes de neuronas artificiales

fully connected



- Conjunto de neuronas artificiales conectadas entre sí. Cada conexión tiene un número real asociado, llamado peso.
- Las neuronas generalmente se distribuyen en capas de distintos niveles. Hay conexiones entre neuronas de capas distintas, así como conexiones entre neuronas de la misma capa.
- Aprendizaje de la red: es el proceso mediante el cual la red modifica sus respuestas ante las entradas para que sus pesos se vayan adaptando de manera paulatina al funcionamiento que se considera correcto

Determinar los pesos que más se adaptan...
↳ Entrenamiento de la red, mediante ejemplos

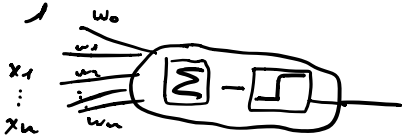
Back propagation ⇒ Retropropagación.

En este tema

Técnicas bioinspiradas

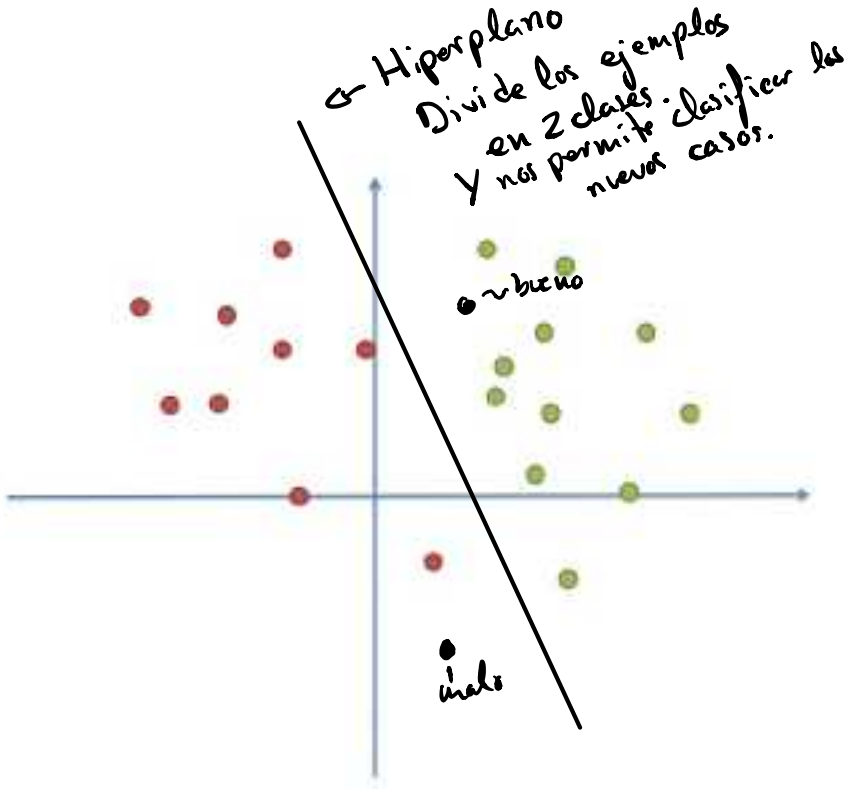
Perceptrón (Rosenblatt, 1959)

- Forma más simple de red de neuronas, formada por una sola neurona
- Estuvo inspirada en el modelo de célula de McCulloch-Pitts y en estudios en los ’60 sobre la visión de las ranas
- Adaptación supervisada



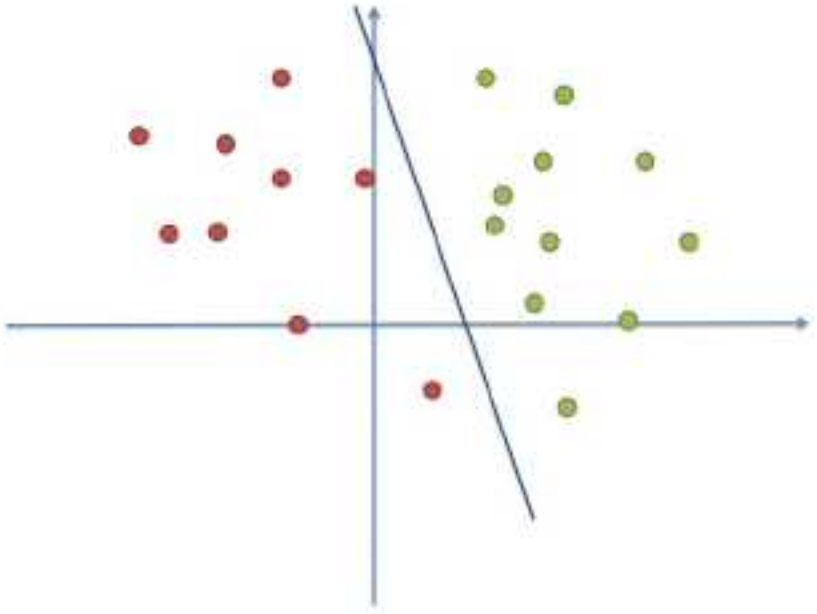
- Se usa la **función escalón** como función de activación
- Tareas de clasificación lineal: dado un conjunto de ejemplos o patrones, determinar el hiperplano capaz de discriminar los patrones en dos clases

Objetivo



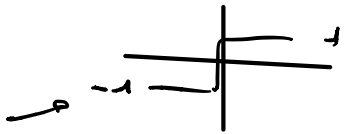
Ejemplos: vector de valor de las entradas (atributos) $\bar{x} = (x_1, x_2, \ldots, x_n)$

Objetivo



Hiperplano (2 dimensiones/atributos): $w_1 x_1 + w_2 x_2 + U = 0$

Perceptrón simple (PS)



- Se dispone de un vector de valores de entrada, $\bar{x}$, y un valor de salida, $f(\bar{x})$, dado por las fórmulas

$$RED(\bar{x}) = U + \sum_{i=1}^{n} w_i x_i = \sum_{i=0}^{n} w_i x_i$$

$$f(\bar{x}) = \begin{cases} 1 & \text{si } RED(\bar{x}) > 0 \\ -1 & \text{en caso contrario} \end{cases}$$

- Si $f(\bar{x}) = 1$, pertenece a la clase +
- Si $f(\bar{x}) = -1$, pertenece a la clase -

Regla de aprendizaje del PS

Algoritmo iterativo.

- Se inicializan aleatoriamente los pesos w_i y U
- En cada iteración se modifican los w_i de forma que el hiperplano separe *completamente* cada ejemplo → se van actualizando
  - se elige un ejemplo $\{\bar{x}, c(\bar{x})\} = \{(x_1, x_2, \ldots, x_n), c(\bar{x})\}$ donde $c(\bar{x})$ es la clase (1 ó -1)
  - se calcula la salida de la red $y(\bar{x}) = f(U + \sum_{i=1}^{n} w_i x_i)$ → se sigue el procedimiento y la func. f. → Salida dada por el ejemplo.
  - si $y(\bar{x}) = c(\bar{x})$ se contínua con la siguiente iteración → lo ha hecho correctamente
  - si no, se actualizan los pesos y el umbral → se ha equivocado y se actualiza para recoger ese error.

Regla de aprendizaje del PS

- Si $U = w_0$ y $\bar{w} = (w_0, w_1, \ldots, w_n)$
- En cada iteración, los pesos se actualizan de acuerdo a:

cuando falla → de menos ⇒ sumo

$$w_i = w_i + \begin{cases} x_i & \text{si } \bar{x} \text{ mal clasificado como } - \\ -x_i & \text{si } \bar{x} \text{ mal clasificado como } + \end{cases}$$

↳ de más ⇒ resto (sobreestima)

$$w_i = w_i + c(\bar{x})x_i$$

anterior · clase verdadera · valor entrada · learning rate

Es el peso actual ± valor de la entrada

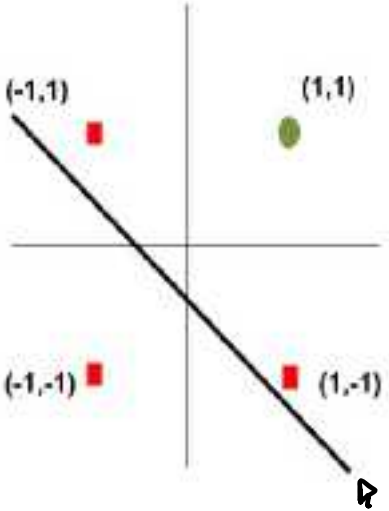
\+ subestima · − sobreestima.

Para el umbral el ± es siempre de 1.

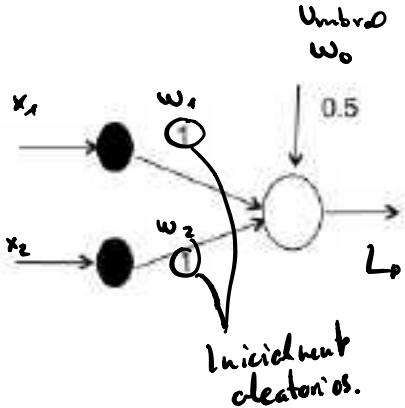
Para el umbral se suma o resta 1 desde el valor aleatorio.

Ejemplo del AND

x_1	x_2	AND
-1	-1	-1
1	-1	-1
-1	1	-1
1	1	1



Todavía no está bien ajustada. Se irá recolocando.



$y = w_0 + x_1 w_1 + x_2 w_2$

Ejemplo del AND

Verdadero — Hallado con Eximi y luego func. activ.

X=(-1,-1), d(x)=-1 → Y=f(-1.5)=-1 → Bien clasificado

X=(1,-1), d(x)=-1 → Y=f(0.5)=1 → Mal clasificado

} veredicto

Subestima → Actualizamos.

Nuevos parámetros

$w_1(1) = 1 - 1 = \mathbf{0}$

$w_2(1) = 1 - (-1) = \mathbf{2}$

$u(1) = 0.5 - 1 = \mathbf{-0.5}$

X=(-1,1), d(x)=-1 → Y=f(1.5)=1 → Mal clasificado

Subestima → ↓

Nuevos parámetros

$w_1(1) = 0 - (-1) = \mathbf{1}$

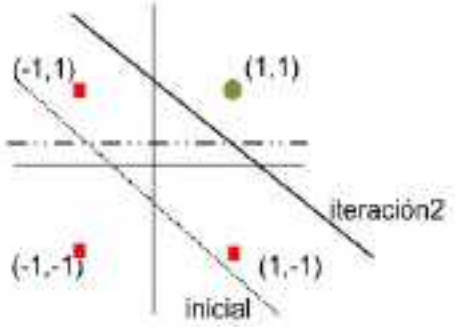
$w_2(1) = 2 - 1 = \mathbf{1}$

$u(1) = -0.5 - 1 = \mathbf{-1.5}$

Y=f(-2.5)=-1 → Bien clasificado

Y=f(-1.5)=-1 → Bien clasificado

X=(1,1), d(x)=1 → Y=1 → Bien clasificado



Un hiperplano solución es:

$\mathbf{x_1 \cdot w_1 + x_2 \cdot w_2 - 1.5 = 0}$

lo que es lo mismo:

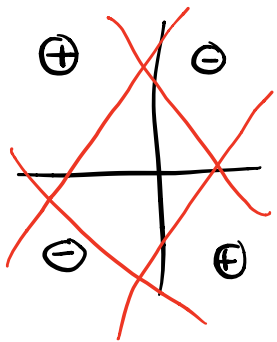
$\mathbf{x_1 + x_2 - 1.5 = 0}$

Termina cuando se estabiliza superando los ejemplos

¿Qué puede aprender el Perceptrón Simple?

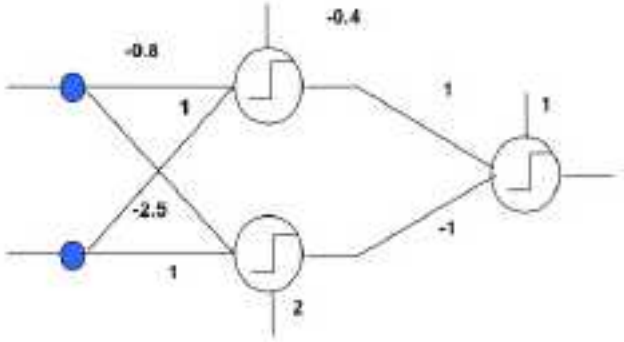
- Las cosas que puede clasificar son hiperplanos en espacios de cualquier dimensión
- No puede obtener clasificaciones basadas en curvas, ni clasificaciones multiplanares
- Ejemplo más conocido y simple de por qué no funciona es el XOR

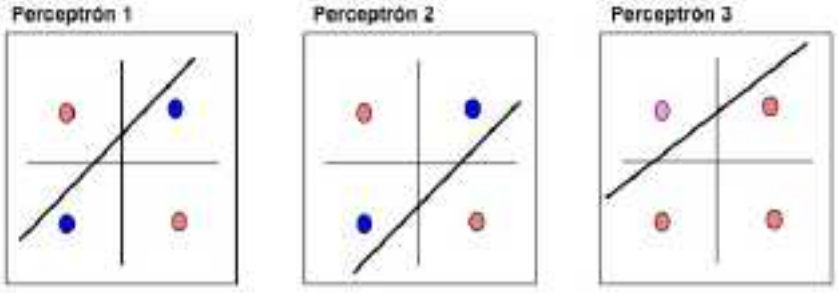
x_1	x_2	Clase
+1	+1	-
+1	-1	+
-1	-1	-
-1	+1	+



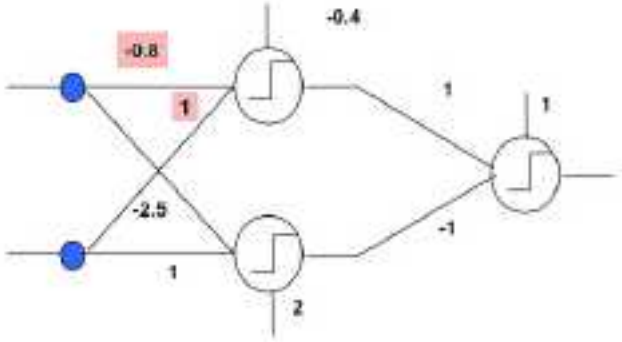
No se puede separar con un solo hiperplano

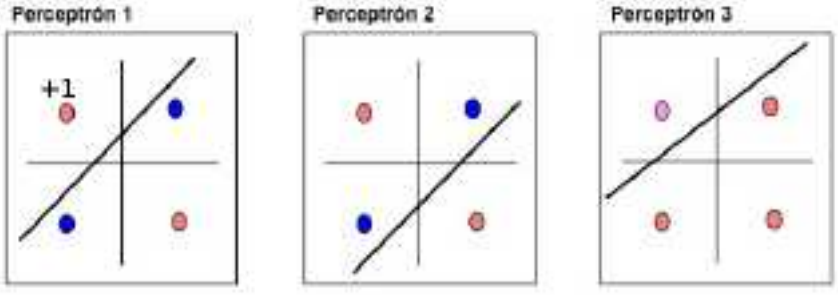
Posible solución: combinación de varios perceptrones



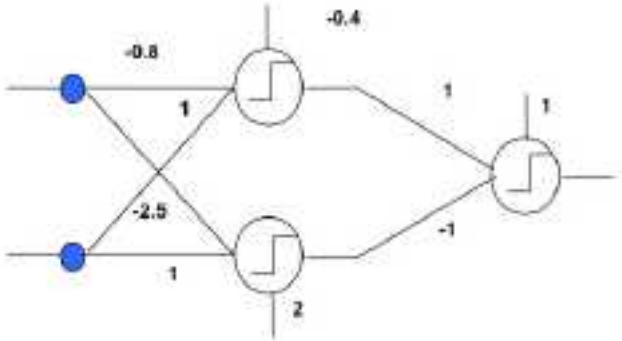


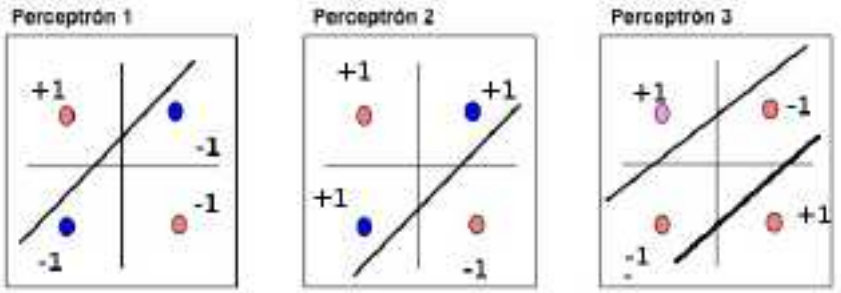
Posible solución: combinación de varios perceptrones



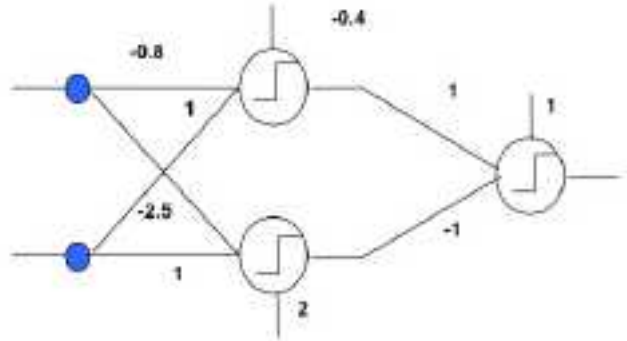


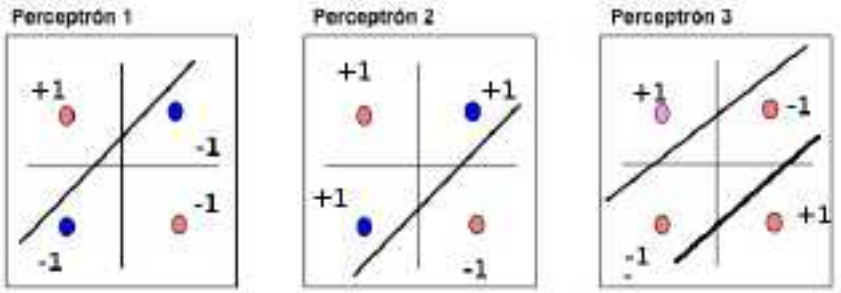
Posible solución: combinación de varios perceptrones





Posible solución: combinación de varios perceptrones





Pero en este caso, ¿cuál debe ser el algoritmo de aprendizaje?

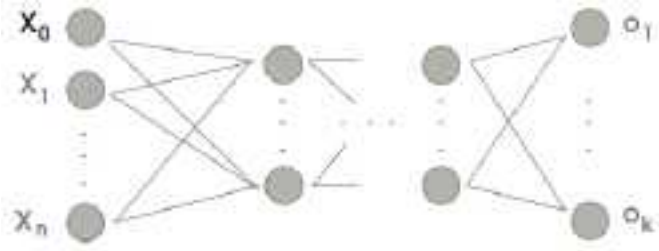
En este tema

Técnicas bioinspiradas

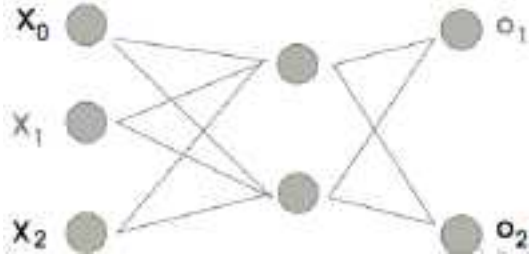
Perceptrón Multicapa

- Si se encadenan varios perceptrones, entonces se puede representar cualquier función de las entradas, incrementando el número de neuronas o de capas.
- El algoritmo de aprendizaje se denomina **retropropagación**.

La estructura general es:



Un ejemplo podría ser:



Representación en redes de neuronas

- Supongamos que queremos representar un atributo en la entrada que tiene cinco posibles valores. Podemos elegir entre:

  1. Tener una entrada con cinco posibles valores
  2. Tener una entrada para cada valor
  3. Tener tres entradas en las que se codifican en binario los valores

Representación en redes de neuronas

- Supongamos que queremos representar un atributo en la entrada que tiene cinco posibles valores. Podemos elegir entre:
  1. Tener una entrada con cinco posibles valores
  2. Tener una entrada para cada valor
  3. Tener tres entradas en las que se codifican en binario los valores
- Lo mismo ocurre en la salida
- ¿Cuál es la mejor representación?
  - Se tienen varios grados de libertad:
    - número y tipo de neuronas
    - conectividad
  - Se tiene que elegir el valor para los parámetros del algoritmo de entrenamiento

En este tema

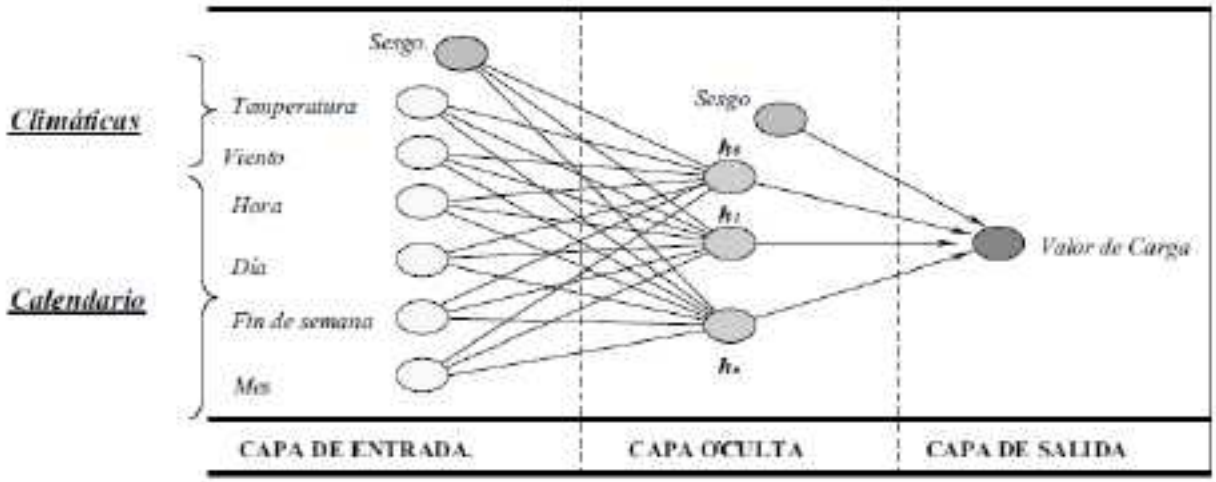
Técnicas bioinspiradas

Predicción

- Múltiples campos: bolsa, meteorología, energía (predicción del consumo), etc.
- **De series temporales**: predecir los valores futuros a partir de pasados
  - Suele definirse una ventana N: cuántos valores pasados se consideran
  - En general: N entradas y una salida
- **Otras**: predicción en función de otras variables

Ejemplo: Predicción de demanda eléctrica

- Depende de factores meteorológicos, estacionales y diarios (laborable/festivo)

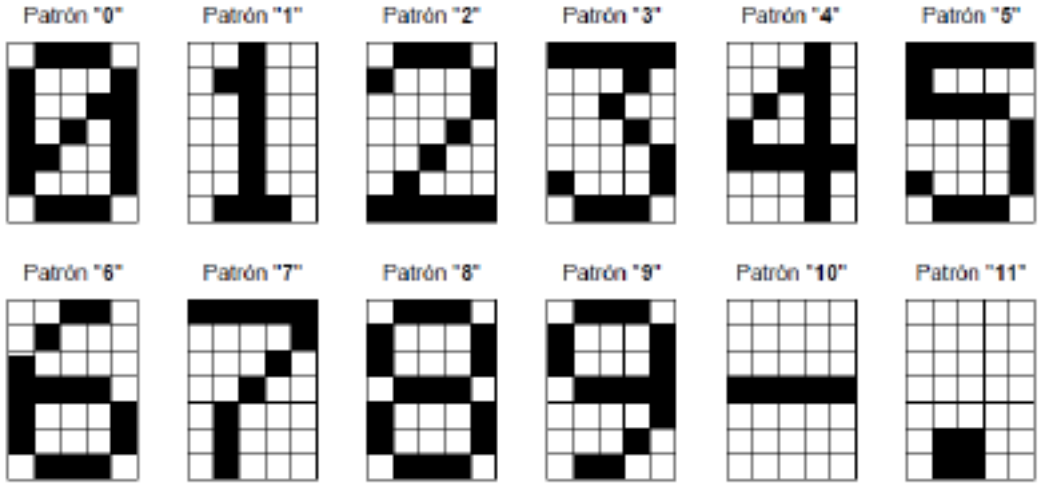


Clasificación

- Entradas: atributos
- Salida: clase

Ejemplo: Reconocimento de caracteres

- Entradas: Una matriz de puntos que pueden ser blancos o negros (35 entradas)
- Salidas: Una salida por caracter (12 salidas)



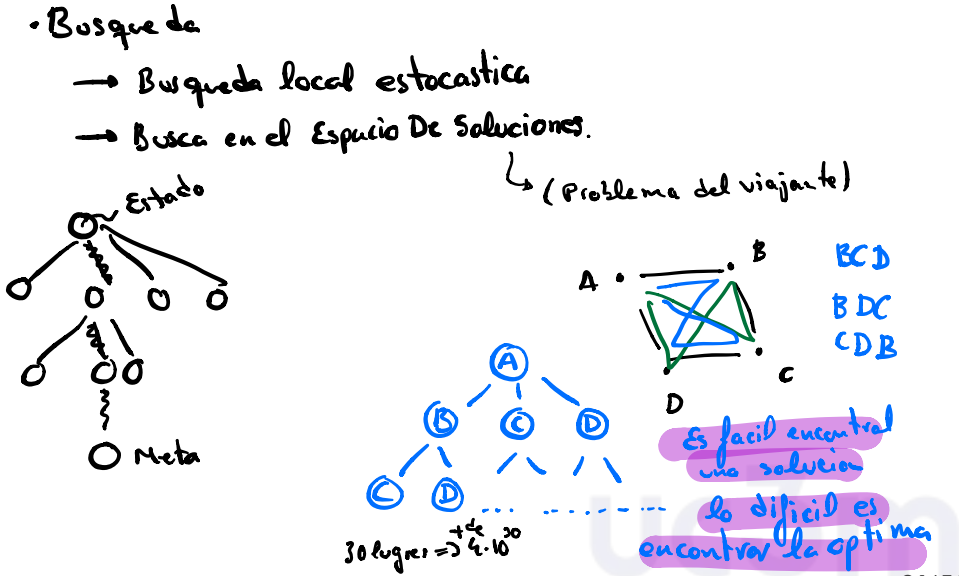
- `www.jcee.upc.es/JCEE2002/Aldabas.pdf`

Demostración

- Reconocimiento de caracteres
  https://www.youtube.com/watch?v=ocB8uDYXtt0
- Conducir un coche
  https://www.youtube.com/watch?v=0Str0Rdkxxo

En este tema

Técnicas bioinspiradas

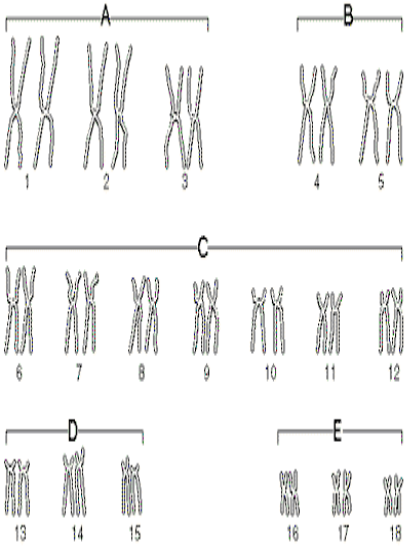


En este tema

Computación Evolutiva

- Búsqueda basada en evolución simulada:
  - se mantiene una colección de *soluciones potenciales* para un problema
  - se evoluciona a través de generaciones por recombinación de las más adecuadas, y eliminación de las menos adecuadas
  - se termina cuando se obtiene una solución *suficientemente* adecuada
- Son algoritmos intensivos en búsqueda local estocástica
- Se pueden ver como técnicas de optimización numérica o combinatoria
- Los factores clave son la representación, los operadores y la función de adecuación (*fitness*)

Inspiración biológica



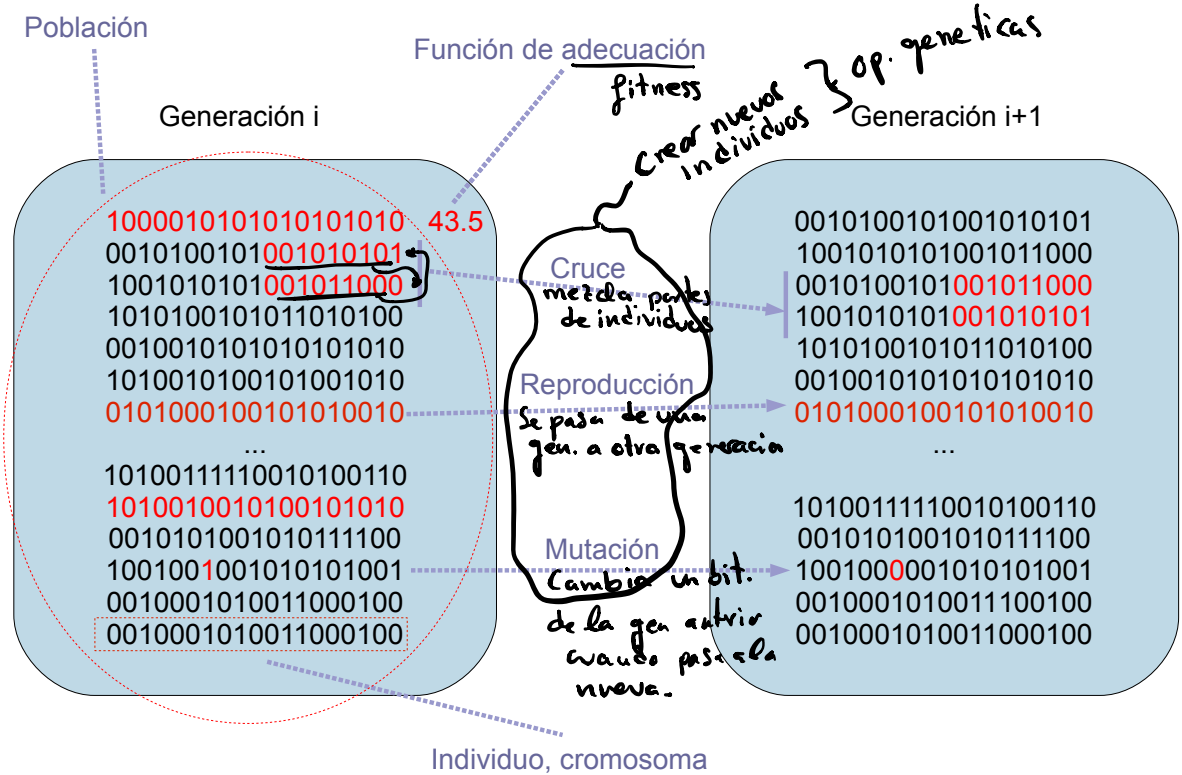
1000010101010101010
0010100101001010101
1001010101001011000
1010100101011010100
0010010101010101010
1010010100101001010
0101000100101010010
1010011111001010011 0
1010010010100101010
0010101001010111100
1001001001010101001
0010001010011000100

Funcionamiento

- Los algoritmos funcionan por **generaciones** (ciclos)
- En cada generación se tiene una **población**
- Una población está compuesta por un conjunto de p **individuos**
- Cada individuo representa una **solución** al problema
- Se dispone de un conjunto de **operadores** que transforman los individuos
- Se utiliza una **función de adecuación** (*fitness*) para evaluar los individuos

Funcionamiento

Cada elemento es una solución y hay que buscar entre ellos la mejor secuencia.



En este tema

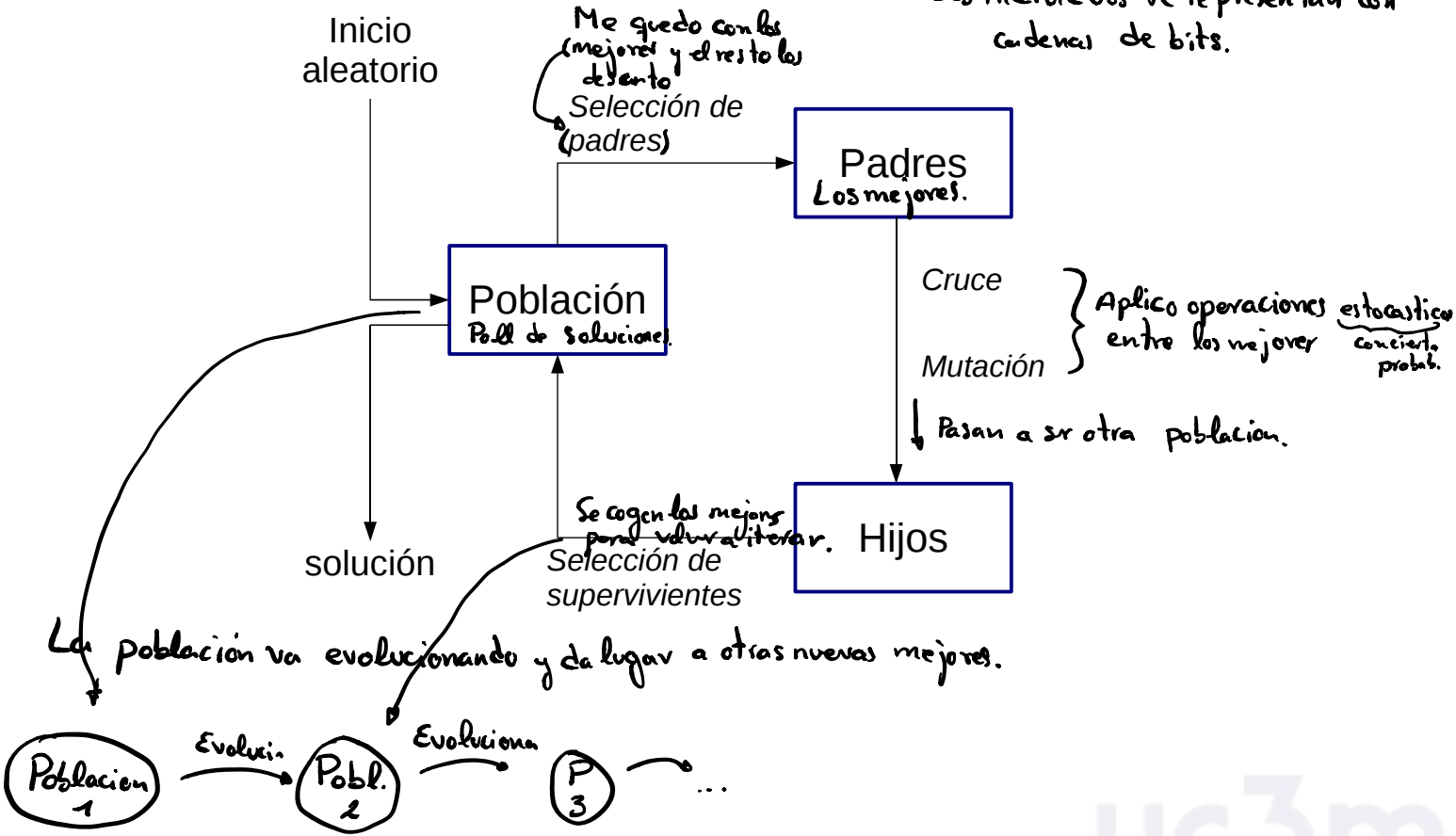
Técnicas bioinspiradas

Esquema Algoritmo Evolutivo

Entender esto ¡Importante!

~~La función de adecuación del Prob. del viajante, que cogela de menor kms se llama~~ ⇒ Función de Fitness

Los individuos se representan con cadenas de bits.



Algoritmo Prototipo

Algoritmo ComputaciónEvolutiva (F,p,r,m)

F: Función de Adecuación/Evaluación, (*fitness*)
p: número de individuos/solución en la población
r: fracción de población que se reemplaza en cada paso
m: tasa de mutación

$P \leftarrow$ generar p soluciones iniciales, calculando su $F(h)$
Mientras que NOT final
seleccionar $P_s \leftarrow (1-r) \cdot p$ soluciones con mayor F
cruzar: seleccionar aleatoriamente $r \cdot p$ pares de solución de P_s
favorecer aquellos con mayor F
$\forall h_1, h_2$, recombinar h_1 y h_2
añadir los resultados a P_s
mutar: cambiar aleatoriamente un m% de los individuos de P_s
actualizar $P \leftarrow P_s$
evaluar $\forall h \in P$, calcular $F(h)$
Devolver el h con mayor $F(h)$

HASTA AQUÍ

Condición de Terminación

- Variantes:
  - Se encuentra una solución con un valor de la función de adecuación por encima de un umbral
  - Se realiza un número determinado de iteraciones (generaciones)
  - Se alcanza un límite de tiempo de cómputo
  - El valor de la función de adecuación no cambia significativamente de una generación a otra
  - Por decisión manual
- Se pueden combinar varias

Representación: Distintas aproximaciones

- Algoritmos Genéticos (GA,Holland): cadenas de bits
- Programas Evolutivos (EP, Michalewicz): estructuras de datos
- Estrategias Evolutivas (EE, Rechenberg y Schwefel): números reales
- Programación Evolutiva (EP, Fogel): Máquinas de Estados Finitos
- Programación Genética (GP, Koza): árboles
- Sistemas clasificadores (LCS, Holland): reglas

En este tema

Función de Adecuación *(Fitness)*

- Genotipo: cromosoma (por ejemplo, cadena de bits en algoritmos genéticos)
- Fenotipo: lo que significa la cadena de bits
- La función de adecuación debe primero transformar el genotipo en el fenotipo
- Ejemplos:
  - TSP de N ciudades, $log_2 N$ bits/ciudad, $N \times log_2 N$ para una solución
  - Encontrar el máximo de una función $f(x)$ donde x está entre 0 y 100, $log_2 101$ bits

Operadores. Reproducción

- Selección: de $k < p$ individuos que pasarán a la siguiente generación (mediante ruleta, torneo, sorteo, . . .)
  - Ruleta: se asigna a cada individuo un valor proporcional a $\frac{f_i}{\sum_i f_i}$ y se itera k veces seleccionando un individuo con reemplazo mediante un número aleatorio
  - Torneo: se itera k veces, seleccionando dos individuos (con reemplazo) y eligiendo el mejor. Los peores individuos no tienen posibilidad de ser elegidos

Operadores

- Cruce (*crossover*): permite mezclar el material genético de los progenitores en la idea de obtener “mejores” individuos

$$\left.\begin{matrix}1010100\\0011110\end{matrix}\right\} 1010110$$

Operadores

- Cruce (*crossover*): permite mezclar el material genético de los progenitores en la idea de obtener “mejores” individuos

$$\left.\begin{array}{l} 1010100 \\ 0011110 \end{array}\right\} 1010110$$

- Mutación (*mutation*): permite luchar contra la “pérdida de alelos”. Introduce diversidad genética

$$1010100 \longrightarrow 1011100$$

Operadores

- Cruce (*crossover*): permite mezclar el material genético de los progenitores en la idea de obtener “mejores” individuos

$$\left.\begin{matrix}1010100\\0011110\end{matrix}\right\} 1010110$$

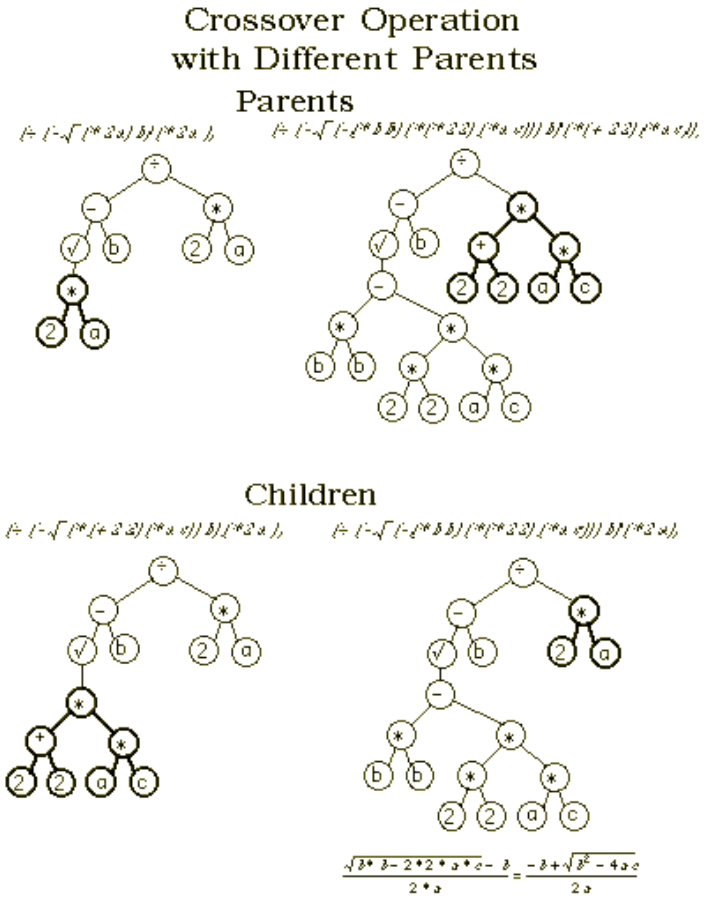
- Mutación (*mutation*): permite luchar contra la “pérdida de alelos”. Introduce diversidad genética

$$1010100 \longrightarrow 1011100$$

- Inversión (mutación estructural): permite reordenar alelos en un gen. No se usa tanto como anteriores

Otros ejemplos: GP

- Cada algoritmo evolutivo define sus operadores en función de la representación
- En Programación Genética (GP) la representación son programas en forma de árbol



Demostración

- TSP (Traveling Salesman Problem o problema del viajante)
  `https://www.youtube.com/watch?v=94p5NUogClM`
- Evolución de vehículos
  `http://rednuht.org/genetic_cars_2/`

Resumen Computación Evolutiva

- Los algoritmos evolutivos intentan reproducir el proceso de selección natural en el que los mejores individuos (soluciones) tienen mayor oportunidad de sobrevivir y reproducirse para la siguiente generación (iteración).
- La eficiencia de los algoritmos depende de la buena elección de la representación de los individuos, la función de adecuación y los operadores
- Al igual que otros enfoques de búsqueda local pueden encontrar mínimos/máximos locales
- En general no garantizan encontrar la mejor solución, pero en muchas ocasiones basta encontrar una solución suficientemente buena
- Son capaces de obtener varias soluciones simultáneamente