APRENDIZAJE MEDIANTE TÉCNICAS BASADAS EN BIOLOGÍA

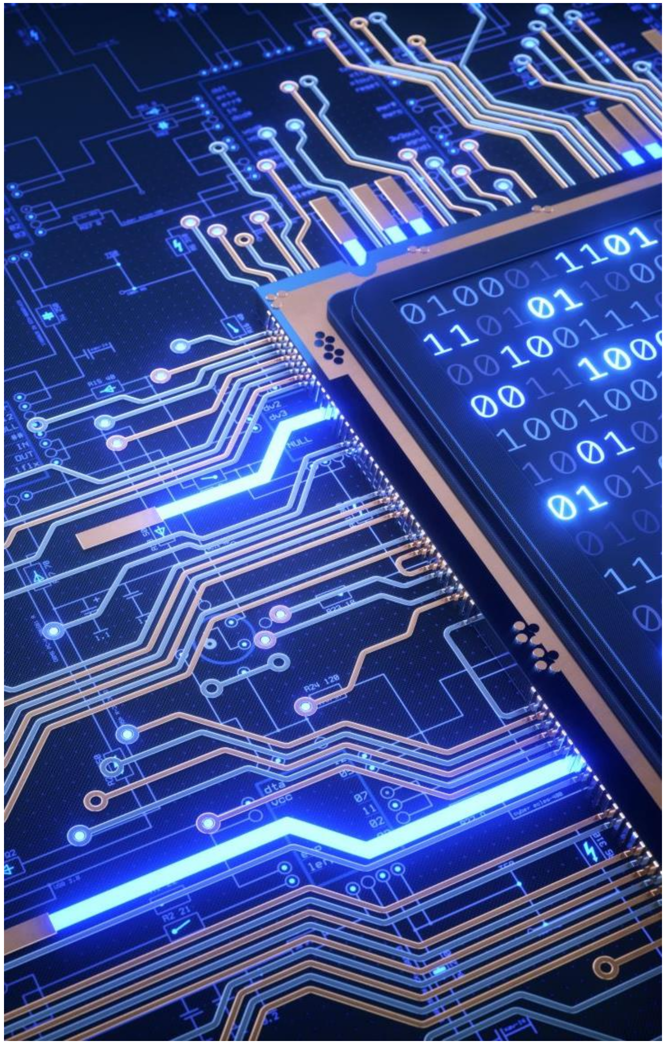
José Manuel Molina López
Grupo de Inteligencia Artificial Aplicada

REDES NEURONALES

Sistemas de Aprendizaje basados en sistemas biológicos

TEMAS

INTRODUCCIÓN
PERCEPTRÓN SIMPLE. LIMITACIONES
PERCEPTRÓN MULTICAPA: ALGORITMO DE RETROPROPAGACIÓN
APLICACIONES DE LAS REDES DE NEURONAS ARTIFICIALES



Redes de Neuronas Artificiales

INTRODUCCIÓN

REDES DE NEURONAS ARTIFICIALES. INTRODUCCIÓN

- Durante varias décadas los científicos han perseguido la construcción de algoritmos capaces de procesar información al igual que el **cerebro humano**

- En la actualidad existe un gran número de estructuras diferentes de redes de neuronas artificiales.

- Se caracterizan porque gozan de propiedades como

    la capacidad de aprendizaje

    la capacidad de aproximación a partir de ejemplos

    la tolerancia a fallos

- Las redes de neuronas se utilizan para afrontar una gran variedad de problemas:

  Aproximación, Predicción, Clasificación

    reconocimiento de patrones (imagen, voz, caracteres)

    compresión y análisis de datos

    robótica

Redes de Neuronas Artificiales. Introducción

Breve historia

- Los primeros estudios fueron realizados por McCulloch y Pitts en 1949.

  Mostraron la habilidad de un grupo de neuronas conectadas para llevar a cabo la implementación de ciertas funciones lógicas.

- Durante la década de los 50 hubo un considerable crecimiento en este campo.

  Rosenblatt en 1958 introdujo la primera arquitectura de red de neuronas artificial con capacidad de aprendizaje: **El perceptrón simple**.

- En 1969 y debido a que se demostraron las serias limitaciones de dicha red, 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 volvieron a renacer

  En 1986, Rumelhart, Mc Celland and Willians propusieron el **percpetrón multicapa** y el algoritmo de retropropagación

Redes de Neuronas Artificiales. Introducción

Neurona artificial

Unidad elemental de una red de neuronas artificiales

- Recibe un conjunto de **señales de entradas** procedentes del mundo exterior o de otras neuronas.

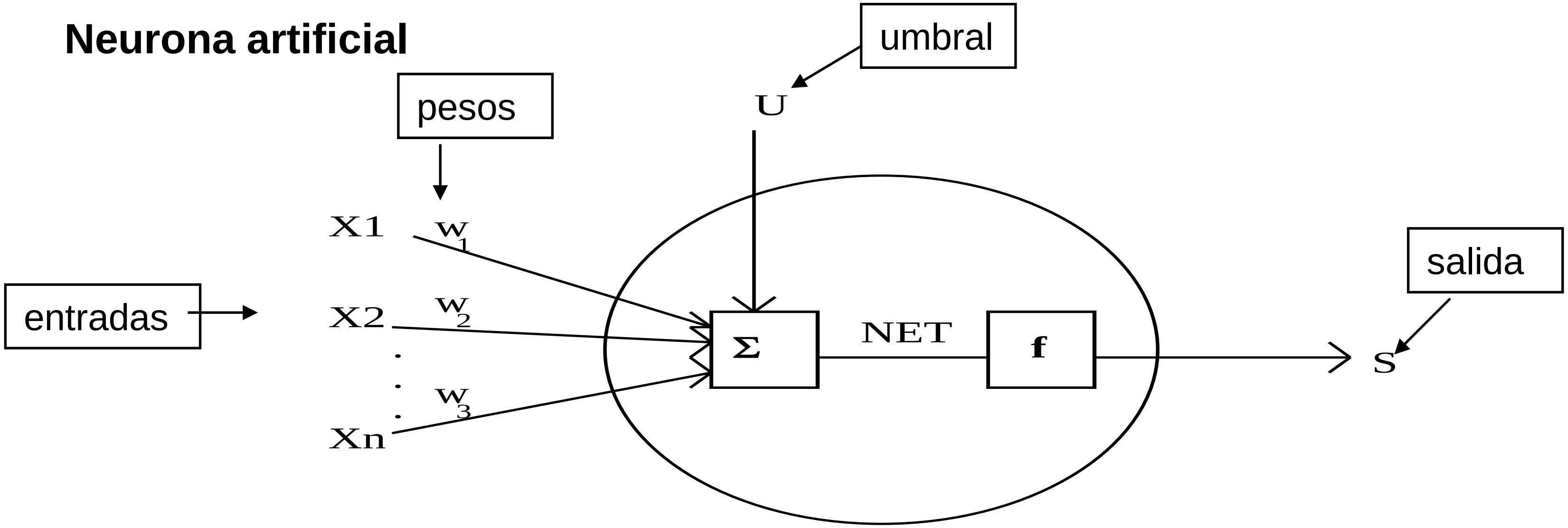
- Las señales de entrada se reciben a través de unas conexiones, las cuales tienen un número real asociado llamado **peso**

- **Procesa la información** recibida, mediante una serie de operaciones simples

- Emite una **señal de salida** como respuesta a las señales de entrada

Redes de Neuronas Artificiales. Introducción

Neurona artificial



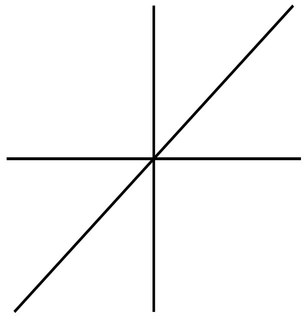
La salida de la neurona es: S = f(NET)

f es la **función de activación**

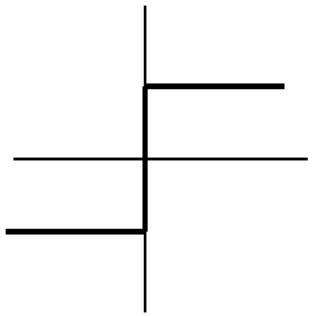
$$NET = X_1 * w_1 + X_2 * w_2 + \ldots + X_n * w_n + U = \sum_{i=1}^{n} X_{\substack{1\\i}} * w_i + U$$

Redes de Neuronas Artificiales. Introducción

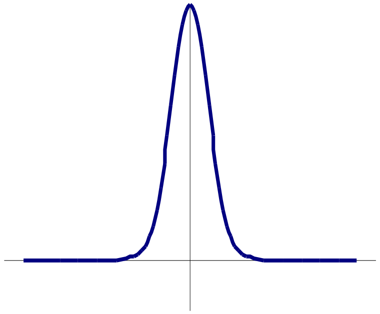
Funciones de activación



Función lineal



Función umbral

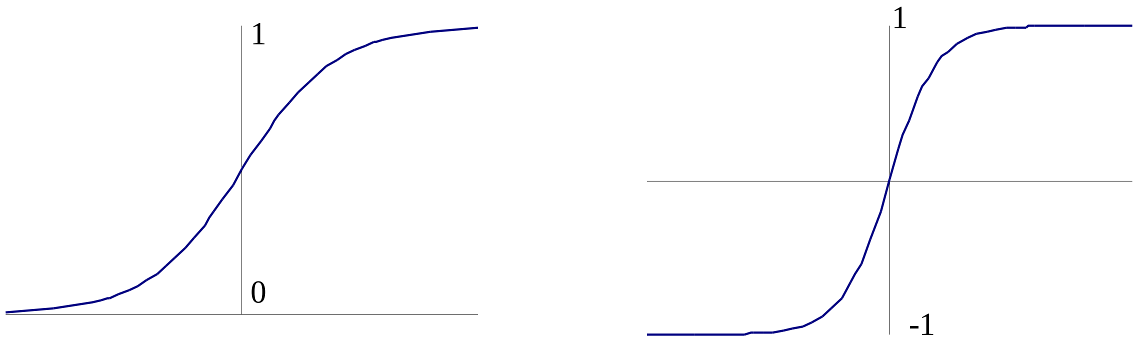


Función gausiana

f(x)=x

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

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



Funciones sigmoidales

Función en (0,1)

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

Función en (-1,1)

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

Redes de Neuronas Artificiales. Introducción

Otros conceptos

- **Redes de neuronas**: es un conjunto de neuronas artificiales conectadas entre sí mediante una serie de arcos llamados ***conexiones***. Estas conexiones tienen números reales asociados, llamados ***peso de la conexión***.

- Las neuronas generalmente se distribuyen en **capas** de distintos niveles, con conexiones que unen las neuronas de distintas capas y/o neuronas de una 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.
  La modificación de los pesos se realiza en base a un criterio establecido y que permite que la red "aprenda" a dar las respuestas adecuadas.
  Supervisado
  No Supervisado

Redes de Neuronas Artificiales. Introducción

Otros conceptos

- **Aprendizaje supervisado**: Para cada patrón (ejemplo) presentado a la red existe una respuesta deseada. La respuesta de la red se compara con su salida deseada y en base a esa comparación se ajustan los pesos de la red.

- **Aprendizaje no supervisado**: no se especifica a la red cual es la respuesta correcta. A través de unas reglas de aprendizaje, la red descubre las relaciones presentes en los ejemplos.

- **Patrones de entrenamiento**: Conjunto de muestras o ejemplos para realizar el aprendizaje (determinación de pesos y umbrales).

- **Patrones de test o validación**: Conjunto de ejemplos utilizados para evaluar la capacidad de **generalización** de la red.

- **Ciclo de aprendizaje**: Presentación del conjunto completo de patrones de entrenamiento una única vez.

Redes de Neuronas Artificiales. Introducción

Clasificación de redes de neuronas

En base a las conexiones:

- Redes feedforward
  Conexiones en un sólo sentido
  Perceptron simple y multicapa, Redes de Base Radial

- Redes recurrentes
  Conexiones en todas las direcciones

- Redes parcialmente recurrentes
  Unas pocas conexiones recurrentes
  Red de Jordan, Red de Elman

En base al aprendizaje:

- Redes supervisadas
  Redes feedforward
  Redes recurrentes

- Redes no supervisadas
  Kohonen, ART

Redes de Neuronas Artificiales

PERCEPTRÓN SIMPLE. LIMITACIONES

Redes de Neuronas Artificiales. Perceptrón simple

Introducción

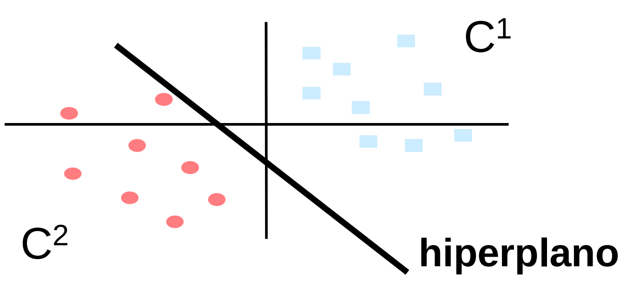
- Forma más simple de red de neuronas
- Estuvo inspirada en el modelo de célula de McCulloch-Pitts
- Modelo propuesto por Rosenblatt en 1959
- Adaptación supervisada
- Tareas de clasificación lineal: Dado un conjunto de **ejemplos o patrones**, determinar el **hiperplano** capaz de discriminar los patrones en dos clases

Ejemplos

Puntos de $\Re^n$: $(x_1, ..., x_n)$

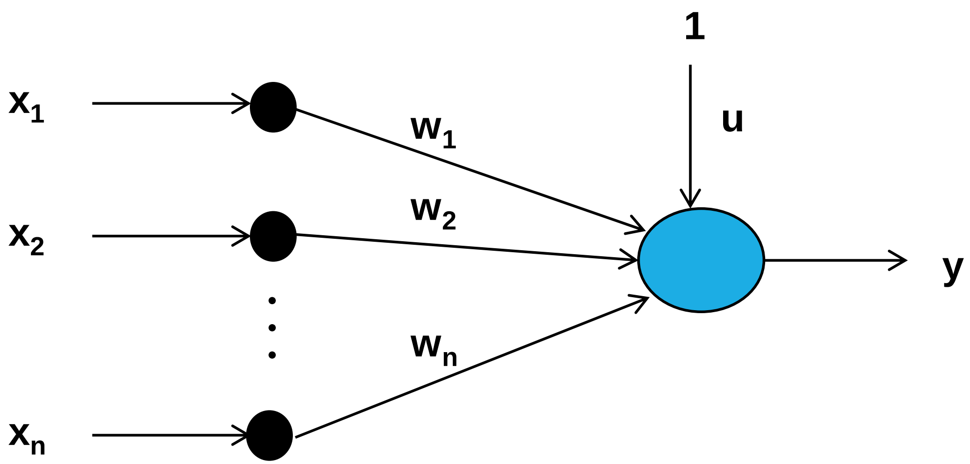
Hiperplano

$$w_1 \cdot x_1 + ... + w_n \cdot x_n + w_0 = 0$$



Redes de Neuronas Artificiales. Perceptrón simple

Arquitectura



$$\boxed{y = f\,(x_1 \cdot w_1 + x_2 \cdot w_2 + \dots + x_n \cdot w_n + u)}$$

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

La red puede utilizarse para clasificar los patrones de entrada

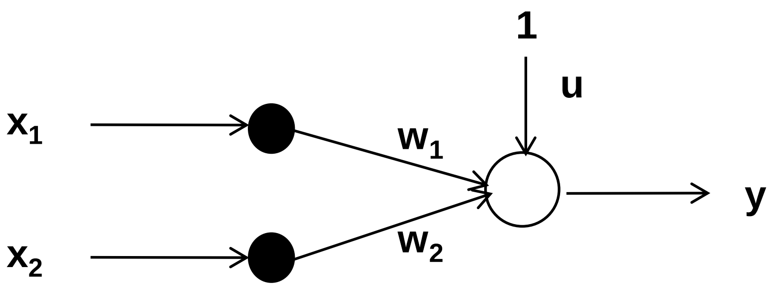
$$\text{Si } x_1 \cdot w_1 + x_2 \cdot w_2 + \dots + x_n \cdot w_n + u > 0 \;\Rightarrow\; y = 1 \;\Rightarrow\; (x_1, \dots, x_n) \in C^1$$

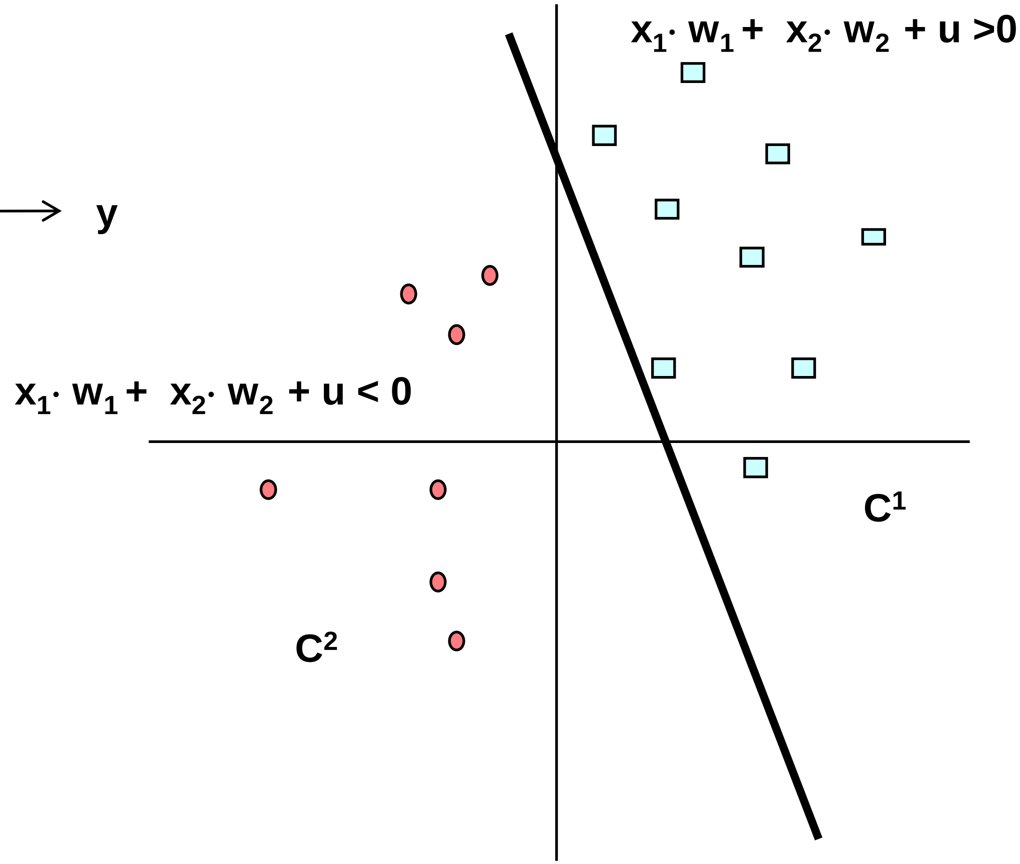
$$\text{Si } x_1 \cdot w_1 + x_2 \cdot w_2 + \dots + x_n \cdot w_n + u \leq 0 \;\Rightarrow\; y = -1 \;\Rightarrow\; (x_1, \dots, x_n) \in C^2$$

$$\boxed{\textbf{Hiperplano: } x_1 \cdot w_1 + x_2 \cdot w_2 + \dots + x_n \cdot w_n + u = 0}$$

Redes de Neuronas Artificiales. Perceptrón simple

Arquitectura. Ejemplo para patrones bidimensionales





Hiperplano

$$\boxed{x_1 \cdot w_1 + x_2 \cdot w_2 + u = 0}$$

Redes de Neuronas Artificiales. Perceptrón simple

Definición del proceso de aprendizaje

Proceso iterativo supervisado: modificación de los parámetros de la red (pesos y umbral) hasta encontrar el hiperplano discriminante

Número finito de iteraciones

Dado

Conjunto de patrones
Vector de entrada: $x = (x_1, x_2, ..., x_n)$ Salida deseada: $d(x)$ $d(x) = 1$ si $x \in C^1$ $d(x) = -1$ si $x \in C^2$

Encontrar

Hiperplano discriminante
$(w_1, w_2, ..., w_n, u)$ tales que $x_1 \cdot w_1 + x_2 \cdot w_2 + ... + x_n \cdot w_n + u = 0$ separe las clases C^1 y C^2

Redes de Neuronas Artificiales. Perceptrón simple

Pasos del proceso de aprendizaje

Paso 1: Inicialización aleatoria de los pesos y el umbral de la red

$$\{w_i(0)\}_{i=0,\ldots,n} \quad u(0)$$

Paso 2: Se toma un patrón entrada-salida $[x=(x_1,x_2, \ldots, x_n), d(x)]$

Paso 3: Se calcula la salida de la red: $y= f(x_1\cdot w_1 + x_2\cdot w_2 + \ldots + x_n\cdot w_n +u)$

Si $y = d(x)$ (clasificación correcta) se vuelve al paso 2

Si $y \neq d(x)$ (clasificación incorrecta) se modifican los parámetros y se vuelve al paso 2

$$\boxed{w_i (t+1) = w_i (t) + d(x) \cdot x_i \qquad u(t+1) = u(t) + d(x)}$$

Ley de aprendizaje

Si $x \in C^1$, $d(x) = 1 \Rightarrow w_i (t+1) = w_i (t) + x_i \quad u(t+1) = u(t) + 1$

Si $x \in C^2$, $d(x) = -1 \Rightarrow w_i (t+1) = w_i (t) - x_i \quad u(t+1) = u(t) - 1$

Redes de Neuronas Artificiales. Perceptrón simple

Regla de aprendizaje de Windrow-Hoff (1960)

$$w_i(t+1) = w_i(t) + (d(x) - y(x)) \cdot x_i \qquad u(t+1) = u(t) + (d(x) - y(x))$$

- Para casos en los que la función de activación sea entre 0 y 1
- Comportamiento similar a la ley anterior
- La idea es utilizar el error cometido por la red para adaptar los pesos

Esta idea hizo que surgiera la arquitectura conocida como: **ADALINE**

La única diferencia con el perceptrón simple es que el valor y(x) que aparece en la ley de aprendizaje es la respuesta de la red, antes de aplicarle la función de activación umbral.

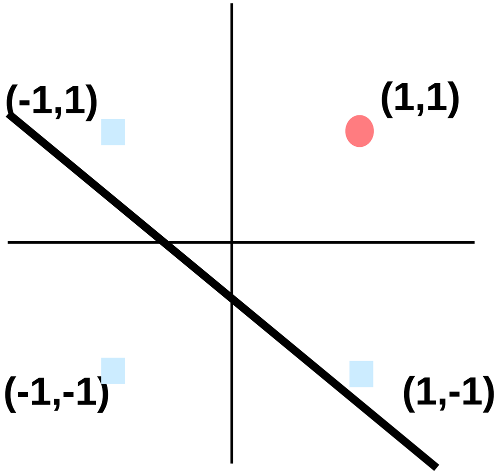
$$y(x)= x_1 \cdot w_1 + x_2 \cdot w_2 + ... + x_n \cdot w_n +u$$

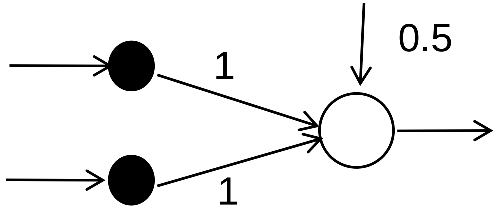
La salida de la red $v(x) = f(y(x))$

Redes de Neuronas Artificiales. Perceptrón simple

Ejemplo: Función lógica AND

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





Redes de Neuronas Artificiales. Perceptrón simple

Ejemplo: Función lógica AND

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

Nuevos parámetros

$w_1(1) = 1 - 1 = \mathbf{0}$
$w_2(1) = 1 - (-1) = \mathbf{2}$ → Y=f(-2.5)=-1 → Bien clasificado
$u\ (1) = 0.5 - 1 = \mathbf{-0.5}$

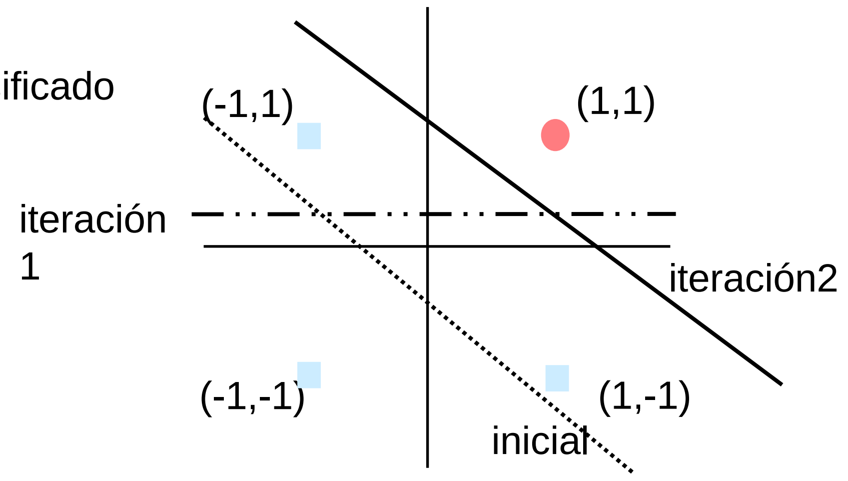
X=(-1,1), d(x)=-1 → Y=f(1.5)=1 → Mal clasificado

Nuevos parámetros

$w_1(1) = 0 - (-1) = \mathbf{1}$
$w_2(1) = 2 - 1 = \mathbf{1}$ → Y=f(-1.5)=-1 → Bien clasificado
$u\ (1) = -0.5 - 1 = \mathbf{-1.5}$

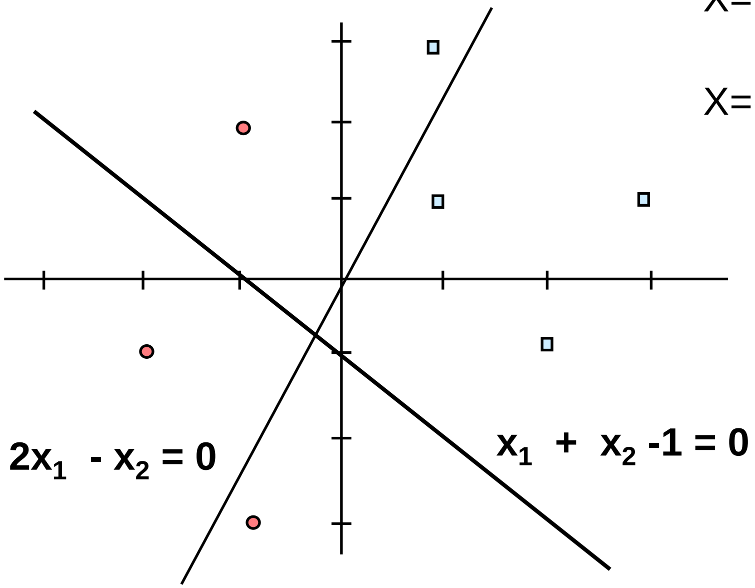
X=(1,1), d(x)=1 → Y=1 → Bien clasificado

Un hiperplano solución es: $\mathbf{x_1 \cdot + \ x_2 \cdot w_2 - 1.5 = 0}$



Redes de Neuronas Artificiales. Perceptrón simple

Razón de aprendizaje



Sea $w_1=w_2=U=1$

$X=(-2,-1)$, $d(x)=-1$ ⇨ $Y=-1$ ⇨ Bien clasificado

$X=(-1,2)$, $d(x)=-1$ ⇨ $Y=1$ ⇨ Mal clasificado

$w_1(1) = 1 + 1 = \mathbf{2}$ $w_2(1) = 1 - 2 = \mathbf{-1}$ $u(1) = 1 - 1 = \mathbf{0}$

Los nuevos parámetros no clasifican correctamente a patrones que anteriormente estaban bien clasificados: (1,3) y (-1,-3)

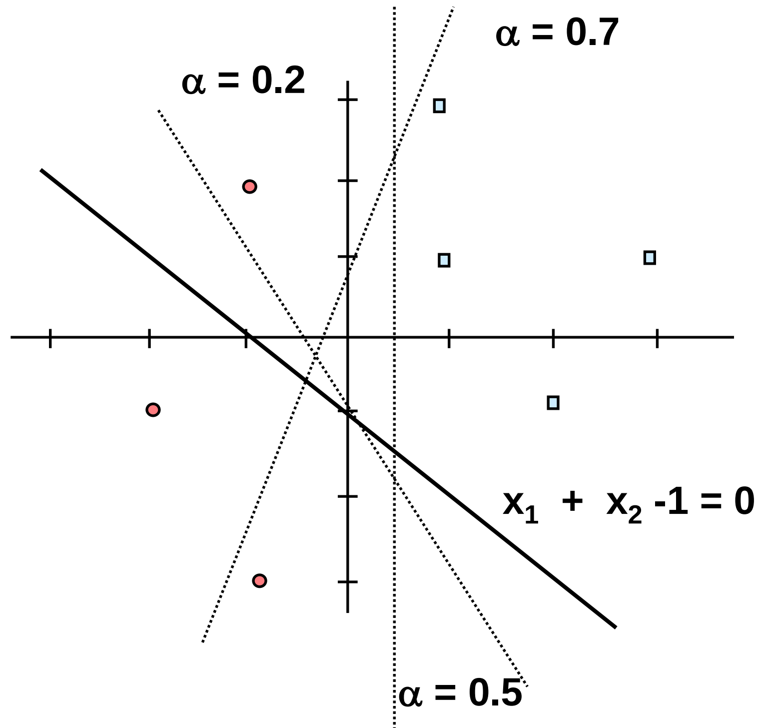
Estas situaciones pueden evitarse introduciendo la razón de aprendizaje

$$\boxed{w_i(t+1) = w_i(t) + \alpha \cdot d(x) \cdot x_i \qquad u(t+1) = u(t) + \alpha \cdot d(x)}$$

α: controla la brusquedad de las modificaciones de los parámetros $\mathbf{0 < \alpha < 1}$

Redes de Neuronas Artificiales. Perceptrón simple

Razón de aprendizaje



$\alpha = 0.7$
$w_1(1) = 1 <- \alpha \cdot (-1) = 1.7$
$w_2(1) = 1 - \alpha \cdot 2 = -0.4$
$u(1) = 1 - \alpha \cdot 1 = 0.3$

$\alpha = 0.5$
$w_1(1) = 1 - \alpha \cdot (-1) = 1.5$
$w_2(1) = 1 - \alpha \cdot 2 = 0$
$u(1) = 1 - \alpha \cdot 1 = 0.5$

$\alpha = 0.2$
$w_1(1) = 1 - \alpha \cdot (-1) = 1.2$
$w_2(1) = 1 - \alpha \cdot 2 = 0.6$
$u(1) = 1 - \alpha \cdot 1 = 0.8$

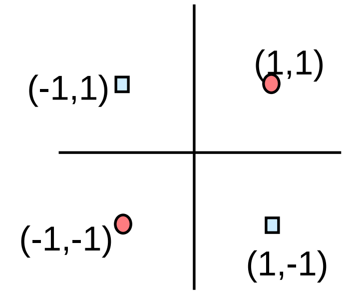
Redes de Neuronas Artificiales. Perceptrón simple

Limitaciones del perceptrón

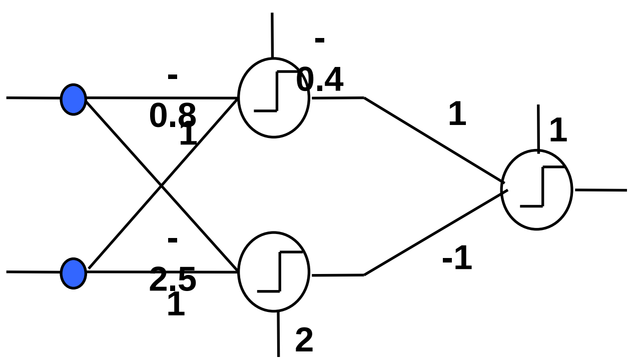
Si no existe un hiperplano ⇒ La ley de aprendizaje no encuentra la solución

Función XOR (OR exclusivo)

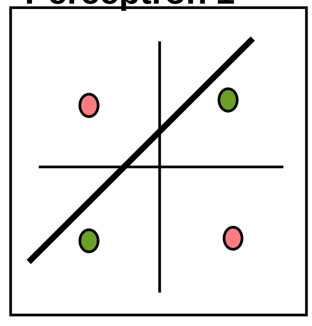
x_1	x_2	d(x)
-1	-1	-1
-1	1	1
1	-1	1
1	1	-1



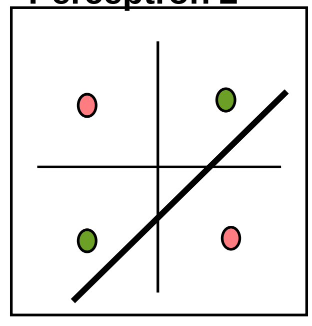
Una solución: **combinando varios perceptrones**



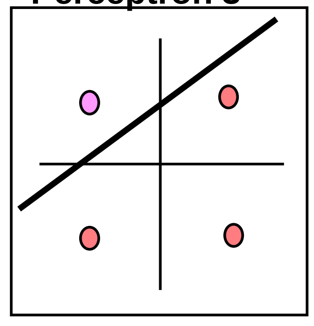
Perceptrón 1



Perceptrón 2



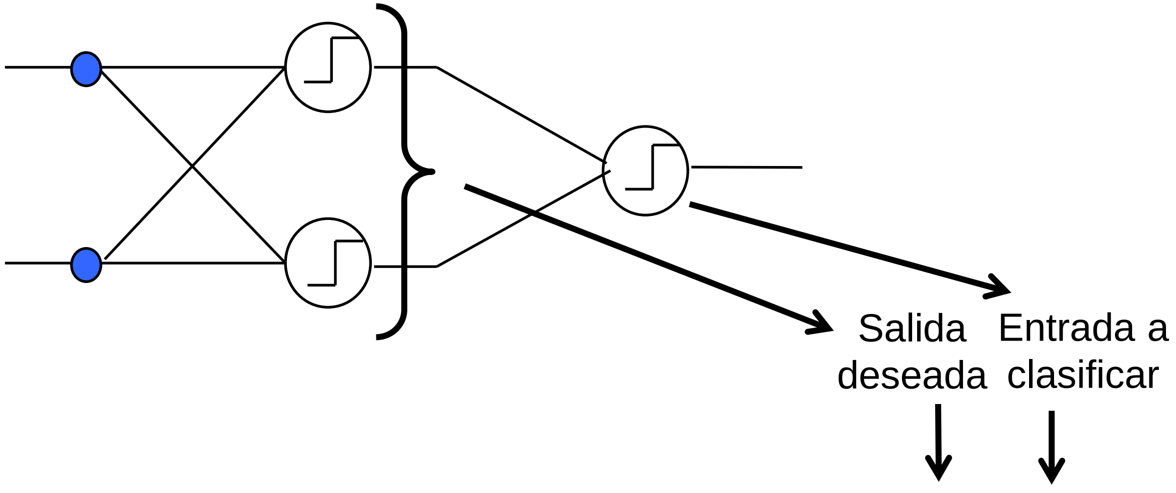
Perceptrón 3



Redes de Neuronas Artificiales. Perceptrón simple

Limitaciones del perceptrón

Esta aproximación puede ser complicada de llevar a cabo en al práctica, pues la ley de aprendizaje no es aplicabe y los pesos tendrían que ser determinados mediante un proceso manual



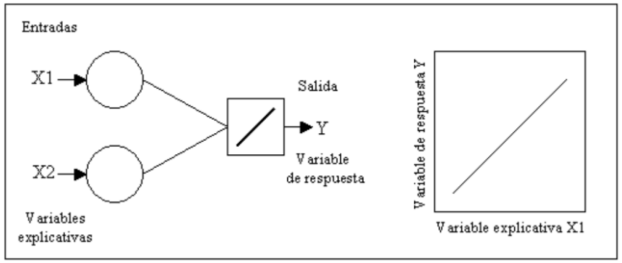
La ley de aprendizaje no es aplicable

$$\boxed{w_i(t+1) = w_i(t) + d(x) \cdot x_i}$$

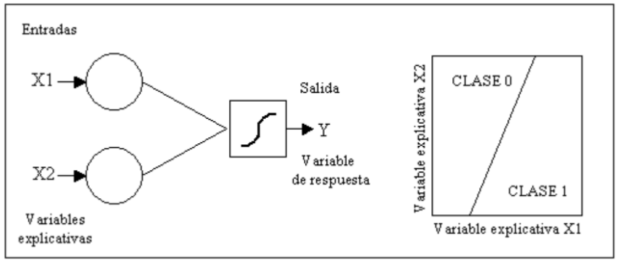
Redes de Neuronas Artificiales

PERCEPTRÓN MULTICAPA: ALGORITMO DE RETROPROPAGACIÓN

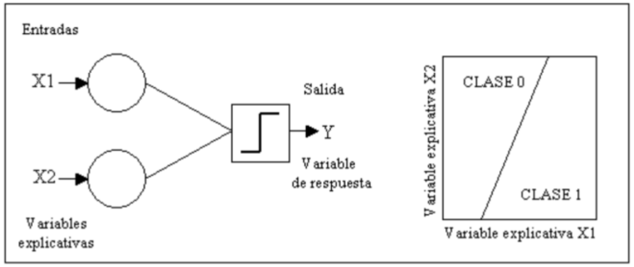
Redes de Neuronas Artificiales. Perceptrón multicapa



Perceptrón simple con función lineal = Modelo de regresión lineal.

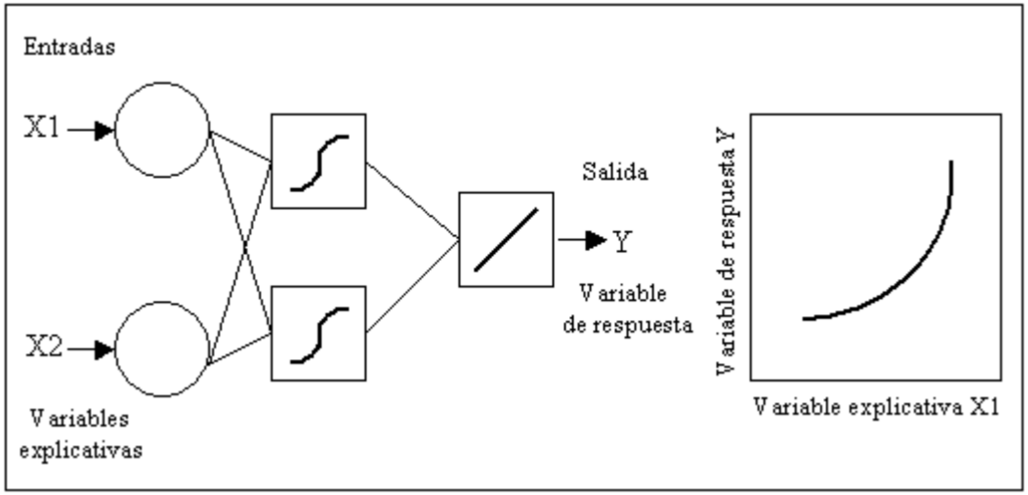


Perceptrón simple con función logística = Modelo de regresión logística.



Perceptrón simple con función umbral = Análisis discriminante lineal.

Propuesto por Rumelhart, Hinton y Williams en 1986 para solventar las limitaciones del perceptrón simple: no linealidad

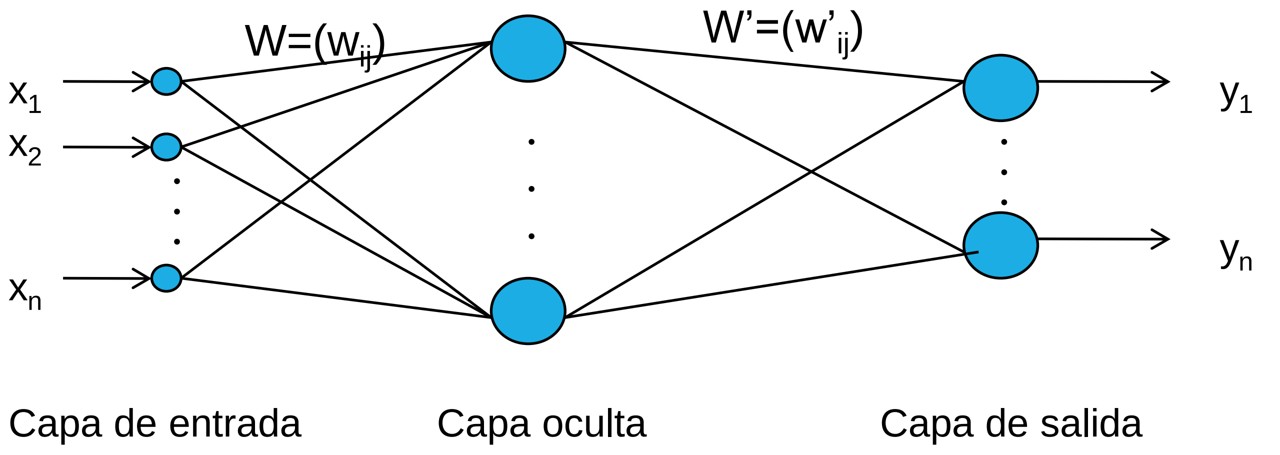


. *Perceptrón multicapa con función lineal en la salida = Modelo de regresión no lineal.*

Redes de Neuronas Artificiales. Perceptrón multicapa

Arquitectura

- El perceptrón multicapa tiene sus neuronas agrupadas en capas.
- Cada neurona en cada capa está conectada a todas las neuronas de la siguiente capa.
- Cada neurona procesa la información recibida y la respuesta se propaga a través de la conexión correspondiente y actúa como entrada para todas las neuronas de la siguiente capa.



Redes de Neuronas Artificiales. Perceptrón multicapa

Arquitectura

Dado un perceptrón multicapa con 3 capas, n neuronas en la capa de entrada, m neuronas en la capa de salida y r neuronas ocultas, las activaciones de las neuronas se calculan del siguiente modo:

Capa de entrada: recibe los patrones del exterior

$$a_i = x_i \qquad i=1,2,...,n$$

Capa oculta:

$$b_j = f\left(\sum_{i=1}^{n} w_{ij} \cdot a_i + u_j\right) \qquad j=1,2,...,r \qquad \text{f: función sigmoidal}$$

Capa de salida: proporciona la respuesta de la red para cada patrón de entrada

$$y_j = f\left(\sum_{i=1}^{r} w'_{ij} \cdot b_i + v_j\right) \qquad j=1,2,...,m$$

Extensión a varias capas ocultas

Redes de Neuronas Artificiales. Perceptrón multicapa

Arquitectura

Número de neuronas en la red

- Número de neuronas de entrada y salida viene dado por el problema a resolver, dependiendo de la codificación de la información
- Número de neuronas ocultas y/o número de capas ocultas: por prueba y error

Teorema (Aproximadores universales)

Toda función continúa puede aproximarse utilizando un perceptrón multicapa con una única capa oculta

El resultado no dice nada acerca del número de neuronas ocultas

Redes de Neuronas Artificiales. Perceptrón multicapa

Aprendizaje

Proceso iterativo supervisado: modificación paulatina de los parámetros de la red (pesos y umbrales) hasta que la salida de la red sea lo más próxima posible a la salida deseada o esperada para cada patrón de entrenamiento

<u>Dado</u>

Conjunto de patrones o ejemplos
Vector de entrada: $x=(x_1, x_2 , ..., x_n)$ Vector de salida deseada: $t(x)=(t_1, t_2 , ..., t_m)$

<u>Encontrar</u>

Pesos $W=(w_{ij})$, $W'=(w'_{ij})$ y umbrales $U=(u_i)$, $V=(v_j)$ tales que

$t(x) \approx y(x) \quad \forall$ patrón x $\quad \Leftrightarrow \|t(x)-y(x)\| \approx 0 \ \forall x$

$\Leftrightarrow$ Minimizar $E=\Sigma \|t(x)-y(x)\|$

Aprendizaje $\Leftrightarrow$ **Minimizar Error** $\Rightarrow$

- Problema de optimización no lineal (funcion sigmoidal)
- Método de descenso del gradiente

Redes de Neuronas Artificiales. Perceptrón multicapa

Aprendizaje

Sea un perceptrón multicapa, el error para el patrón [$x=(x_1, x_2, ..., x_n)$, $t(x)$] se expresa como:

m salidas:

$$e(x) = \|t(x) - y(x)\| = \frac{1}{2}\sum_{i=1}^{m}(t_i(x) - y_i(x))^2$$

1 salida:

$$e(x) = \frac{1}{2}(t(x) - y(x))^2$$

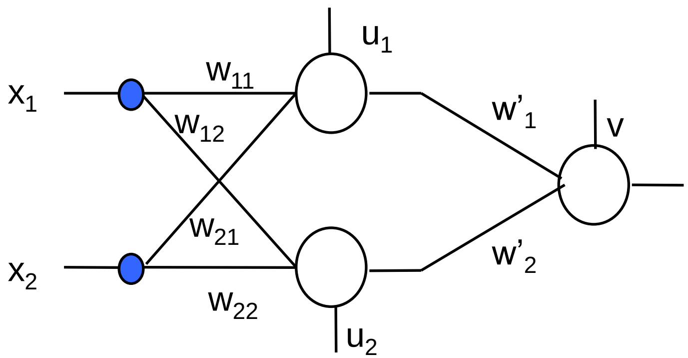
El método de descenso del gradiente consiste en modificar los parámetros de la red siguiente la dirección negativa del gradiente del error:

$$w^{nuevo} = w^{anterior} + \alpha \cdot \left(-\frac{\partial e}{\partial w}\right) = w^{anterior} - \alpha \cdot \frac{\partial e}{\partial w} \qquad \forall \text{ parámetro } w$$

El algoritmo de retropropagación es el resultado de aplicar dicho método al perceptrón multicapa

Redes de Neuronas Artificiales. Perceptrón multicapa

Aprendizaje: Algoritmo de retropropagación



$$b_1 = f(w_{11} \cdot x_1 + w_{21} \cdot x_2 + u_1)$$

$$b_2 = f(w_{12} \cdot x_1 + w_{22} \cdot x_2 + u_2)$$

$$y = f(w'_1 \cdot b_1 + w'_2 \cdot b_2 + v)$$

Pesos de la capa oculta a la de salida: w'$_1$, w'$_2$

$$w'^{\,nuevo}_1 = w'^{\,anterior}_1 - \alpha \cdot \frac{\partial e}{\partial w'_1}$$

Como: $e(x) = \frac{1}{2}(t(x) - y(x))^2 \quad \Longrightarrow \quad \frac{\partial e}{\partial w'_1} = -(t(x) - y(x)) \cdot \frac{\partial y}{\partial w'_1}$

Teniendo en cuenta la expresión de la salida y que f'(x)=f(x)(1-f(x)):

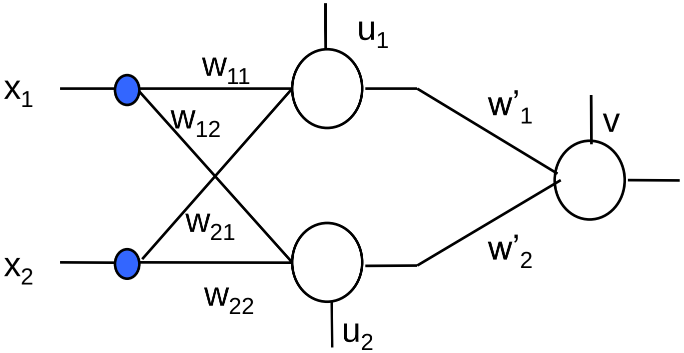
$$\frac{\partial y}{\partial w'_1} = f'(w'_1 \cdot b_1 + w'_2 \cdot b_2 + v) \cdot b_1 = y \cdot (1 - y) \cdot b_1$$

Por tanto:

$$w'^{\,nuevo}_1 = w'^{\,anterior}_1 + \alpha \cdot (t(x) - y(x)) \cdot y \cdot (1 - y) \cdot b_1 = w'^{\,anterior}_1 + \alpha \cdot d(x) \cdot b_1$$

Redes de Neuronas Artificiales. Perceptrón multicapa

Aprendizaje: Algoritmo de retropropagación



Para **w'_1:** $$w'^{\,nuevo}_1 = w'^{\,anterior}_1 + \alpha \cdot d(x) \cdot b_1$$

con $$d(x) = (t(x) - y(x)) \cdot y \cdot (1 - y)$$

entonces $$w'^{\,nuevo}_2 = w'^{\,anterior}_2 + \alpha \cdot d(x) \cdot b_2$$

$$v^{nuevo} = v^{anterior} + \alpha \cdot d(x)$$

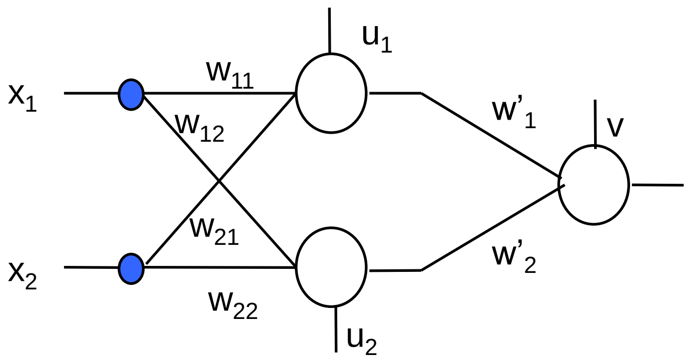
Para **una red con r neuronas ocultas y una neurona de salida, los pesos de la capa oculta a la neurona de salida (w'_i)** se modifican siguiendo las ecuaciones:

$$w'^{\,nuevo}_i = w'^{\,anterior}_i + \alpha \cdot d(x) \cdot b_i \qquad i=1,2, \ldots, r$$

con $$d(x) = (t(x) - y(x)) \cdot y \cdot (1 - y)$$

Redes de Neuronas Artificiales. Perceptrón multicapa

Aprendizaje: Algoritmo de retropropagación



$$b_1 = f(w_{11} \cdot x_1 + w_{21} \cdot x_2 + u_1)$$

$$b_2 = f(w_{12} \cdot x_1 + w_{22} \cdot x_2 + u_2)$$

$$y = f(w'_1 \cdot b_1 + w'_2 \cdot b_2 + v)$$

➡ **Pesos de la capa de entrada a la oculta: w_{ij}** $\quad w_{12}^{\text{nuevo}} = w_{12}^{\text{anterior}} - \alpha \cdot \dfrac{\partial e}{\partial w_{12}}$

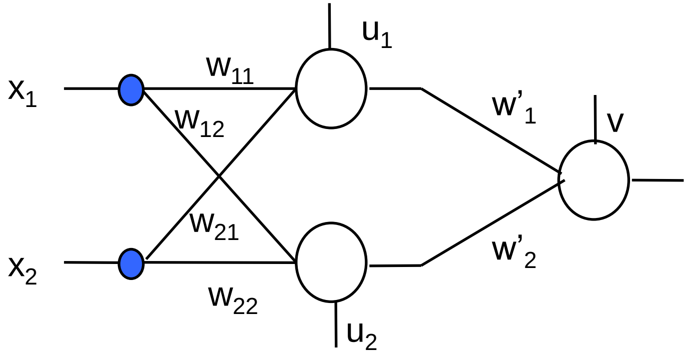
$$\frac{\partial e}{\partial w_{12}} = -(t(x) - y(x)) \cdot \frac{\partial y}{\partial w_{12}} = -(t(x) - y(x)) \cdot y \cdot (1 - y) \cdot \frac{\partial b_2}{\partial w_{12}} = d(x) \cdot \frac{\partial b_2}{\partial w_{12}}$$

$$\frac{\partial b_2}{\partial w_{12}} = f'(w_{12} \cdot x_1 + w_{22} \cdot x_2 + u_1) \cdot x_1 = b_2 \cdot (1 - b_2) \cdot x_1$$

Por tanto: $\quad \dfrac{\partial e}{\partial w_{12}} = d(x) \cdot b_2 \cdot (1 - b_2) \cdot x_1 \qquad w_{12}^{\text{nuevo}} = w_{12}^{\text{anterior}} + \alpha \cdot a_2(x) \cdot x_1$

Redes de Neuronas Artificiales. Perceptrón multicapa

Aprendizaje: Algoritmo de retropropagación



Para $\mathbf{w_{12}}$:
$$w_{12}^{nuevo} = w_{12}^{anterior} + \alpha \cdot a_2(x) \cdot x_1$$

con $a_2(x) = d(x) \cdot b_2 \cdot (1 - b_2)$

entonces
$$w_{11}^{nuevo} = w_{11}^{anterior} + \alpha \cdot a_1(x) \cdot x_1$$
$$w_{21}^{nuevo} = w_{21}^{anterior} + \alpha \cdot a_1(x) \cdot x_2$$
$$w_{22}^{nuevo} = w_{22}^{anterior} + \alpha \cdot a_2(x) \cdot x_2$$
$$u_1^{nuevo} = u_1^{anterior} + \alpha \cdot a_1(x)$$
$$u_2^{nuevo} = u_2^{anterior} + \alpha \cdot a_2(x)$$

Para **una red con r neuronas ocultas y una neurona de salida, los pesos de la capa de entrada a la capa oculta ($\mathbf{w_{ij}}$)** se modifican siguiendo las ecuaciones:

$$w_{ij}^{nuevo} = w_{ij}^{anterior} + \alpha \cdot a_j(x) \cdot x_i \qquad u_j^{nuevo} = u_j^{anterior} + \alpha \cdot a_j(x)$$

i=1,2, ..., n; j=1,2,...,r

$$a_j(x) = d(x) \cdot b_j \cdot (1 - b_j)$$

Redes de Neuronas Artificiales. Perceptrón multicapa

Aprendizaje: Algoritmo de retropropagación

Resumen de las ecuaciones

Dado un perceptrón multicapa con 3 capas, n neuronas en la capa de entrada, 1 neurona en la capa de salida y r neuronas ocultas, los parámetros se modifican siguiendo las siguientes ecuaciones:

➡ **Pesos de la capa oculta a la neurona de salida (w'_i) y umbral de la salida**

$$w'^{\,nuevo}_i = w'^{\,anterior}_i + \alpha \cdot d(x) \cdot b_i \qquad v^{nuevo} = v^{anterior} + \alpha \cdot d(x)$$

i=1,2, ..., r

$$d(x) = (t(x) - y(x)) \cdot y \cdot (1 - y)$$

➡ **Pesos de la capa de entrada a la capa oculta (w_{ij}) y umbrales de las ocultas**

$$w_{ij}^{nuevo} = w_{ij}^{anterior} + \alpha \cdot a_j(x) \cdot x_i \qquad u_j^{nuevo} = u_j^{anterior} + \alpha \cdot a_j(x)$$

i=1,2, ..., n; j=1,2,...,r

$$a_j(x) = d(x) \cdot b_j \cdot (1 - b_j)$$

Los errores cometidos en la capa de salida se propagan (hacia atrás) a las neuronas de la capa oculta

Redes de Neuronas Artificiales. Perceptrón multicapa

Proceso de aprendizaje o entrenamiento

Paso 1: Inicialización aleatoria de los pesos y umbrales

Paso 2: Dado un patrón del conjunto de entrenamiento (x, t(x)), se presenta el vector x a la red y se calcula la salida de la red para dicho patrón, y(x)

Paso 3: Se evalúa el error e(x) cometido por la red

Paso 4: Se modifican todos los parámetros de la red utilizando las ecuaciones anteriormente descritas

Paso 5: Se repiten los pasos 2, 3, y 4 para todos los patrones de entrenamiento, completando así un ciclo de aprendizaje

Paso 6: Se realizan n ciclos de aprendizaje (pasos 2,3,4 y 5) hasta que se verifique el criterio de parada establecido

Redes de Neuronas Artificiales. Perceptrón multicapa

Proceso de aprendizaje o entrenamiento

Criterio de parada del proceso: Combinación de los siguientes factores

Evaluación del error de entrenamiento: Una vez realizado un ciclo de aprendizaje, se evalúa el error cometido por la red para todos los patrones de entrenamiento.

$$E = \sum_{\forall x} e(x)$$

Si E es constante de un ciclo a otro, parar el aprendizaje.

E es constante de un ciclo a otro $\Leftrightarrow$ Los parámetros dejan de sufrir modificaciones $\Leftrightarrow \frac{\partial E}{\partial w} = 0 \Leftrightarrow \mathrm{Min}\,E$

Evaluación del error de validación o de test: Cada cierto número de ciclos de entrenamiento, se presenta a la red todos los patrones de test, calculando la salida de la red para dichos patrones y evaluando el error cometido por la red.

$$E_{test} = \sum_{\forall x de test} e_{test}(x)$$

Si E_{test} evoluciona favorablemente, continuar.

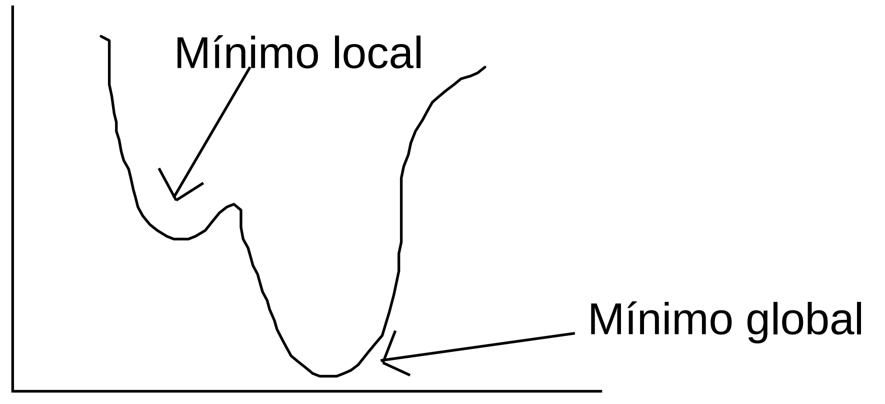
Si E_{test} no evoluciona favorablemente, detener el aprendizaje. No éxito

Si E_{test} ha alcanzado una cota deseada, detener el aprendizaje. Éxito

Redes de Neuronas Artificiales. Perceptrón multicapa

Proceso de aprendizaje o entrenamiento

Mínimo global y local



Desventaja del método: El proceso puede caer en un mínimo local del error

El aprendizaje finaliza cuando $\dfrac{\partial E}{\partial w} \approx 0$,

hecho que también sucede en los mínimos locales.

⇨ El método detecta que cualquier pequeño cambio en los pesos, positivo o negativo, incrementa el error.
No es capaz de determinar en qué dirección deben moverse los pesos para que el error vuelva a decrecer.

Redes de Neuronas Artificiales. Perceptrón multicapa

Razón de aprendizaje

- El valor α es el encargado de controlar **cuánto** se desplazan los pesos en la dirección negativa del gradiente.

- Influye en la **velocidad de convergencia del algoritmo**, puesto que determina la magnitud del desplazamiento.

Valores grandes de α	•favorecen a una convergencia más rápida •en las proximidades del mínimo existe el riesgo de saltar y oscilar alrededor de él
Valores pequeños de α	•la convergencia es más lenta •evita el problema de saltar el mínimo

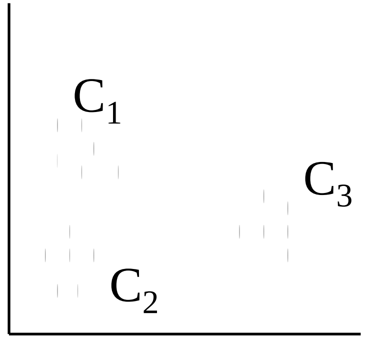
Redes de Neuronas Artificiales

APLICACIONES DE LAS REDES DE NEURONAS ARTIFICIALES

Redes de Neuronas Artificiales. Aplicaciones

Clasificación. Formulación del problema

El problema de clasificación consiste en establecer una correspondencia entre un conjunto de elementos dados y un conjunto clases.



Tres clases totalmente separadas

Dado un conjunto de ejemplos, no siempre es posible conocer cuántas clases tienen que ser consideradas.

Para poder afrontar un problema de clasificación con redes de neuronas supervisadas es condición necesaria y suficiente conocer el número de clases
Si el número de clases es desconocido es necesario utilizar técnicas no supervisadas

Redes de Neuronas Artificiales. Aplicaciones

Clasificación. Resolución utilizando el perceptrón multicapa

Sean $X_i=(X_{i1},...,X_{in})$ i=1,...K, un conjunto de patrones y C_1, C_2,..., C_m, m clases diferentes

La red tendrá n neuronas de entrada que reciben los patrones $X_i=(X_{i1},...,X_{in})$

m neuronas de salida que representan las m clases C_1, C_2,..., C_m,

Aprendizaje

La salida deseada para cada patrón de entrada X_i es una m-tupla $(a_i,....,a_m)$ donde

$a_j=1$ si X_i pertenece a la clase C_j

$a_k=0$ para todo k distinto de j

Codificación de los patrones de entrada

Los ejemplos a clasificar pueden ser de cualquier naturaleza, imágenes, juegos, letras, etc. y tienen que ser codificados en un vector de R^n. La codificación de los patrones juega un papel muy importante en el problema de clasificación hasta el punto que una codificación no adecuada puede producir resultados no satisfactorios.

Redes de Neuronas Artificiales. Aplicaciones

Predicción de series temporales. Formulación

Una serie temporal se puede definir como una función $x : \begin{cases} \Re^{+} \longrightarrow \Re \\ t \longrightarrow x(t) \end{cases}$

El comportamiento temporal viene dado por ec. diferenciales o ec. en diferencias

Ejemplo: x(t+1) = a x(t) (1 - x(t)) Serie temporal logística

El problema de predicción surge cuando la relación entre x(t) y sus valores anteriores es desconocida.

Determinar **f tal que x(t+1) = f(x(t-d), ..., x(t)) para todo t=d, d+1, d+2, ...**

Necesidad de definir el **valor d**, el cual dependerá de cada serie temporal. Se pueden utilizar técnicas para su determinación, como el análisis de los espectros de frecuencia, análisis de Fourrier, prueba y error.

Redes de Neuronas Artificiales. Aplicaciones

Predicción de series temporales. Resolución utilizando el perceptrón multicapa

Dado x(0), x(1), ...,x(N), y dado el valor d, la red tendrá:

- d neuronas de entrada que reciben los valores (x(t-d), x(t-d-1), ..., x(t-1), x(t))
- Una neurona de salida. La salida deseada para el patrón de entrada (x(t-d),..., , x(t)) es el valor x(t+1).

Una vez entrenada la red, se utiliza con el propósito de predicción:

- **Un paso**: Se presenta a la red el patrón (x(t-d), x(t-d-1), ..., x(t-1), x(t))
  y se calcula su salida, x'(t+1).

- **Múltiples pasos**: Con el patrón (x(t-d), x(t-d-1), ..., x(t-1), x(t)) se predice **x'(t+1)**
  Con el patrón (x(t-d-1), x(t-d-2), ..., x(t), **x'(t+1)**) se predice **x'(t+2)**
  Con el patrón (x(t-d-2), x(t-d-3), ..., **x'(t+1)**, **x'(t+2)**) se predice **x'(t+3)**
  Y así sucesivamente hasta conseguir la predicción deseada

ALGORITMOS GENÉTICOS

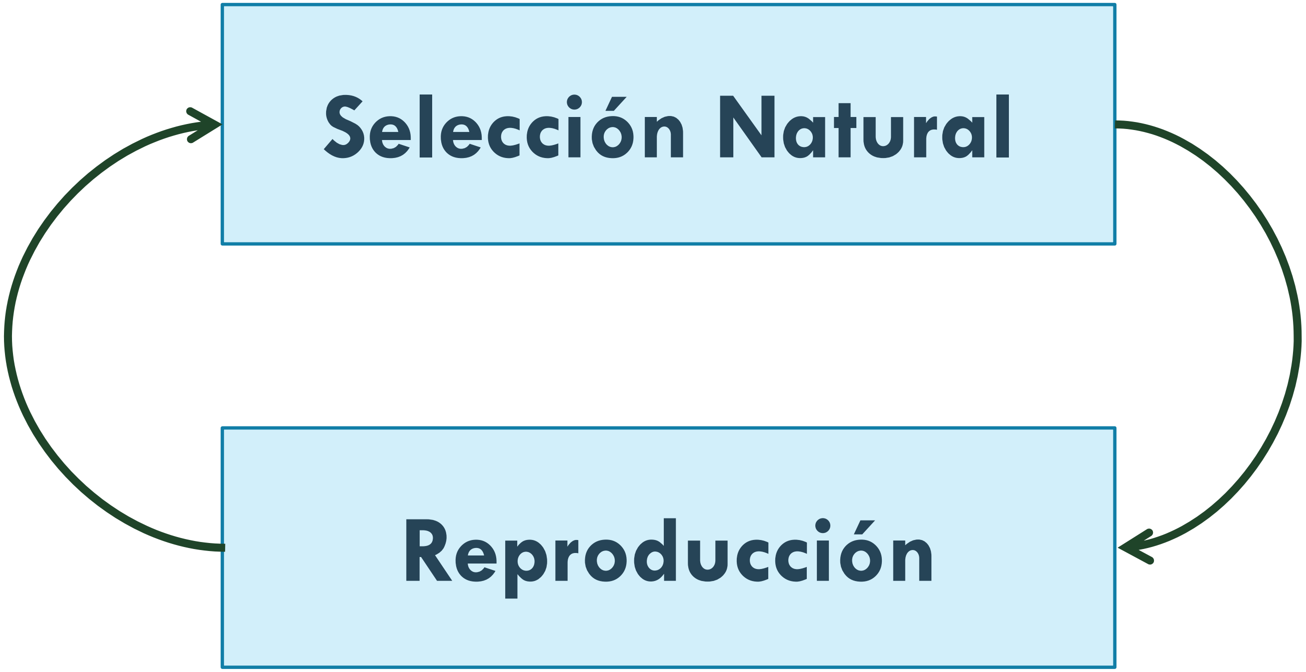
Sistemas de Aprendizaje basados en sistemas biológicos

ANTECEDENTES

Un investigador llamado **John Holland** de la Universidad de Michigan era consciente de la importancia de la selección natural, y a fines de los 60s desarrolló una técnica que permitió incorporarla a un programa.

Su objetivo era lograr que las computadoras aprendieran por sí mismas. A la técnica que inventó Holland se le llamó originalmente "planes reproductivos", pero se hizo popular bajo el nombre "algoritmo genético" tras la publicación de su libro en 1975.

EVOLUCIÓN NATURAL



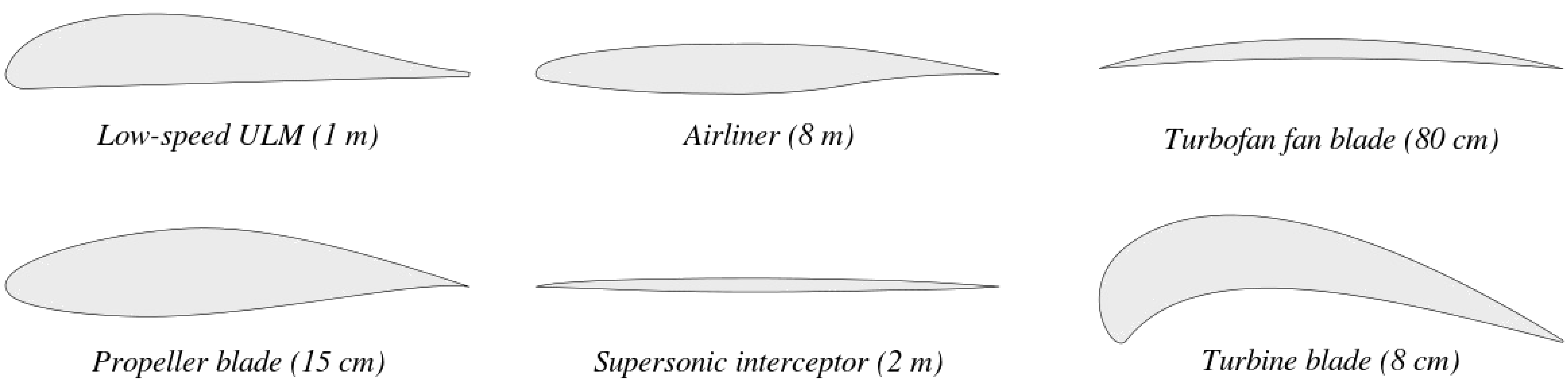
ALGORITMOS GENÉTICOS

Objetivo: optimizar algo

Individuo: posible solución

Ejemplo:

- Objetivo: optimizar resistencia aerodinámica

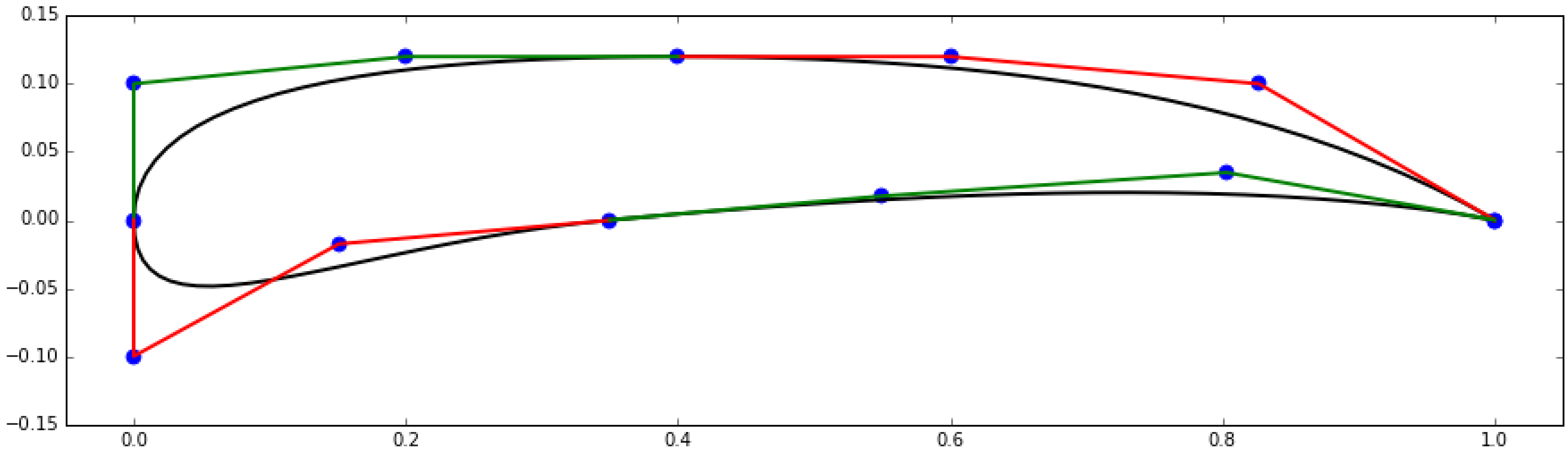


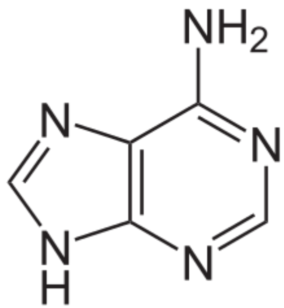
INDIVIDUOS

Conjunto de individuos = población

Definidos por parámetros.

Ejemplo

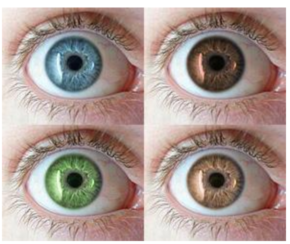




Base Nitrogenada

ATA GTC CCA TGG ATT GTA ACG GCG

Gen



Traits / Características

Performances / Desempeños

Fitness / Aptitud

1

Bit → 100110100110101100

Gen

Transcripción

`x_point_1 = 0.3456` —Problema→ `aero_lift = f(x,y,z)`

Traits /Características

Performances / Desempeños

Función de Fitness

`fitness = f(results)`

Fitness /Aptitud

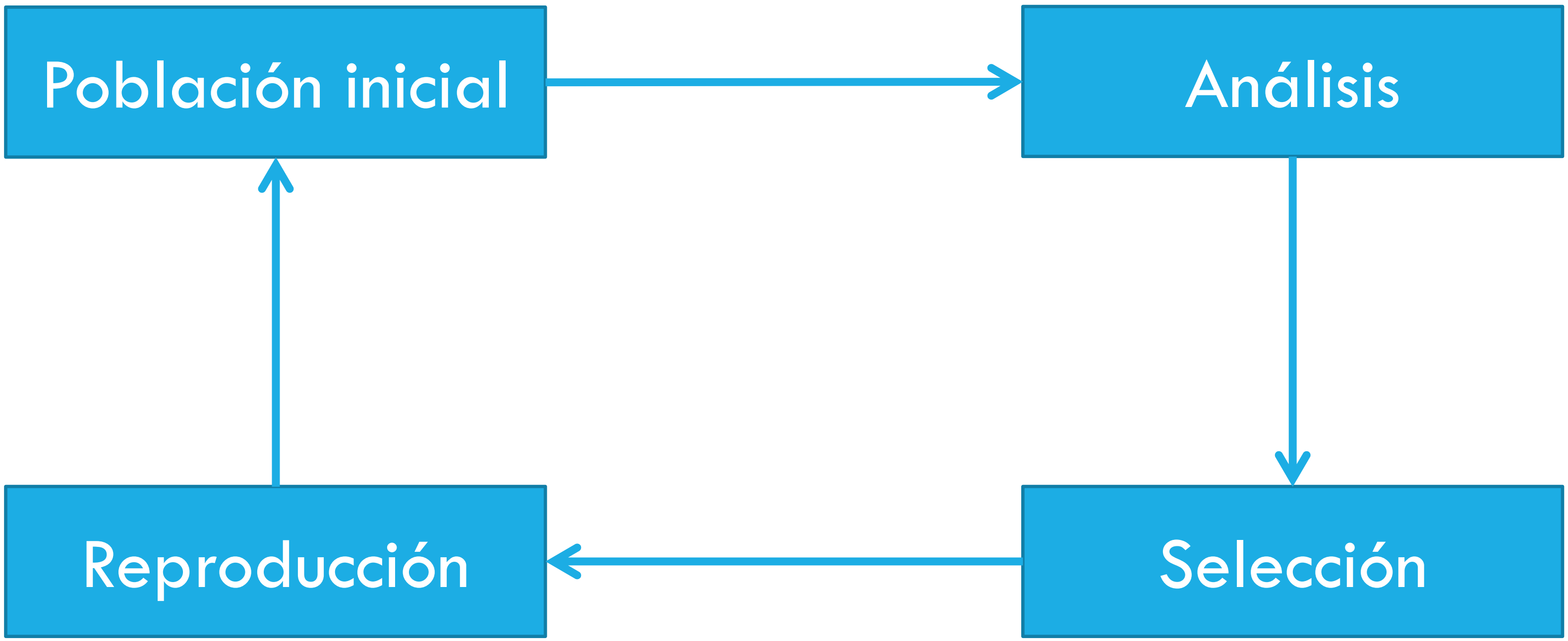
FUNCIÓN DE FITNESS

La función más importante del algoritmo

Condensa en un solo valor la calidad de una solución.

Suele contener condicionales para desechar zonas no interesantes.

Bucle principal: Generación



SELECCIÓN

Mortalidad diferencial, aleatoria o semi-aleatoria

Elegir qué individuos mueren y cuáles se reproducen

Equilibrio:

- Suficientes plazas para la siguiente generación
- Pérdida de información

REPRODUCCIÓN

Reproducción diferencial, aleatoria o semi-aleatoria.

Genera una población nueva para la siguiente generación.

2 fases:

- Cruzamiento
- Mutación

Si se conservan pocos de la generación anterior: Elite Clones

CRUZAMIENTO

Padre 1 : 10010100101010010101

Padre 2 : 10101001100101001100

Hijo: 10110100101101001101

Parámetros:

- Número de puntos de corte
- Posición de los puntos de corte

MUTACIÓN

Añade variedad al acervo genético (gene pool)

Permite explorar soluciones nuevas

Antes de la mutación: 1001010010101

Después de la mutación: 1011010010001

APERTURA Y CIERRE DEL ALGORITMO

Población inicial aleatoria

Criterio de parada:

- Número de generaciones
- Estabilidad

MODULAR EL ALGORITMO: EL DILEMA EXPLORACIÓN-EXPLOTACIÓN

Exploración:

Buscar soluciones nuevas

Escapar de máximos locales

Añade ruido

Explotación

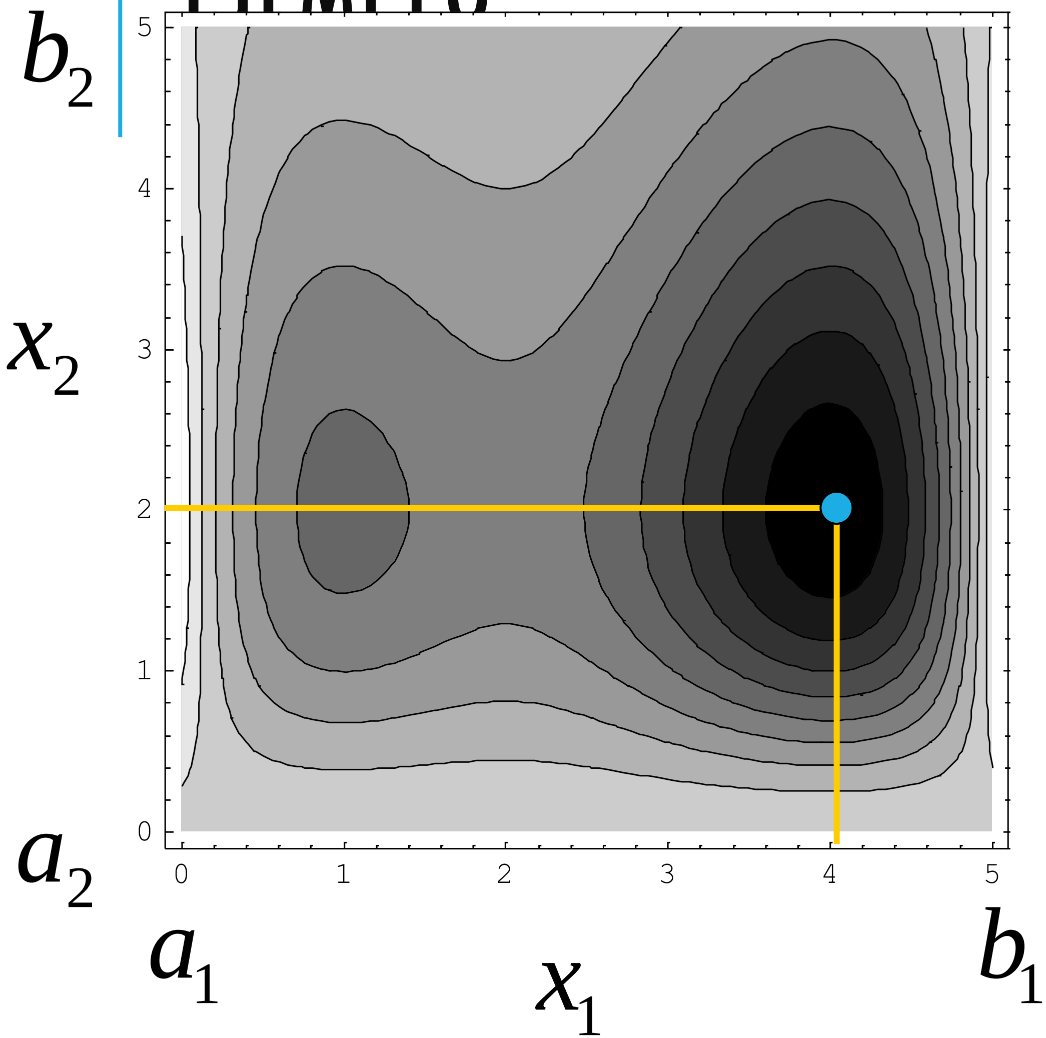
Afinar los máximos encontrados

Conservarlos

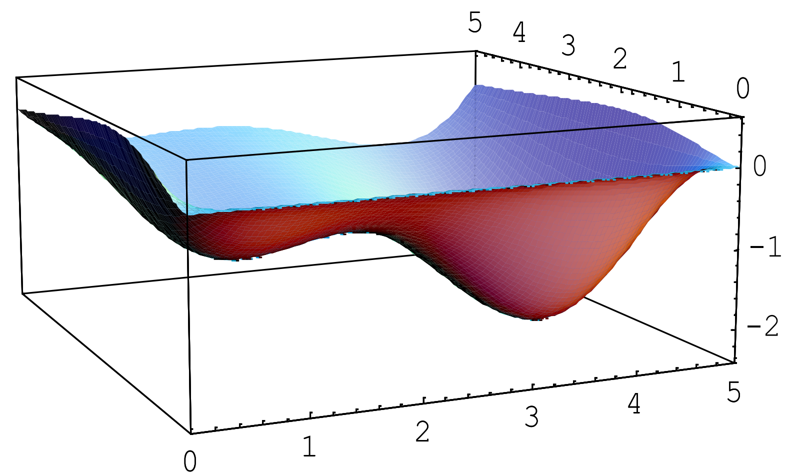
Atasca en máximos locales

Objetivo: maximizar *J* para una cierta combinación de valores de $\left\{x_1, x_2, x_3, \ldots, x_n\right\}$

EJEMPLO



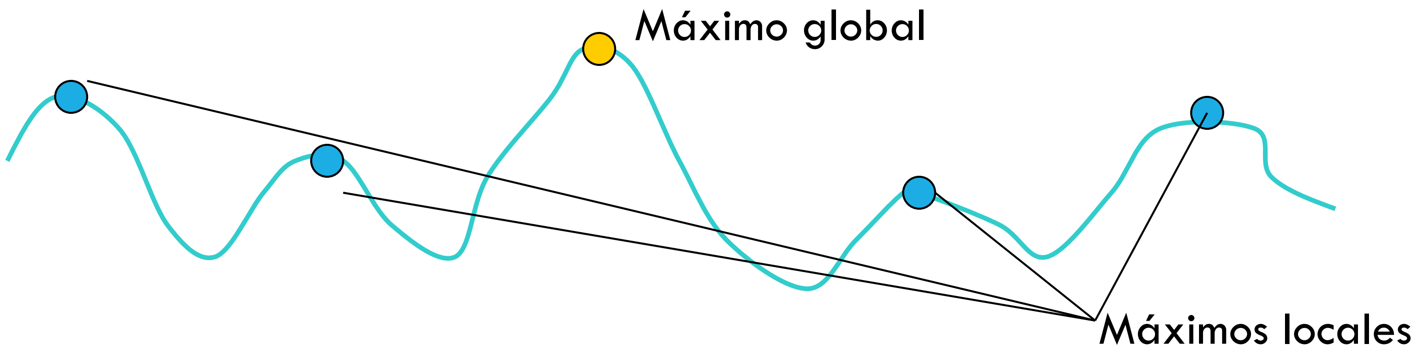
En 2D



Definir limites inferior y superior para cada parámetro: a y b

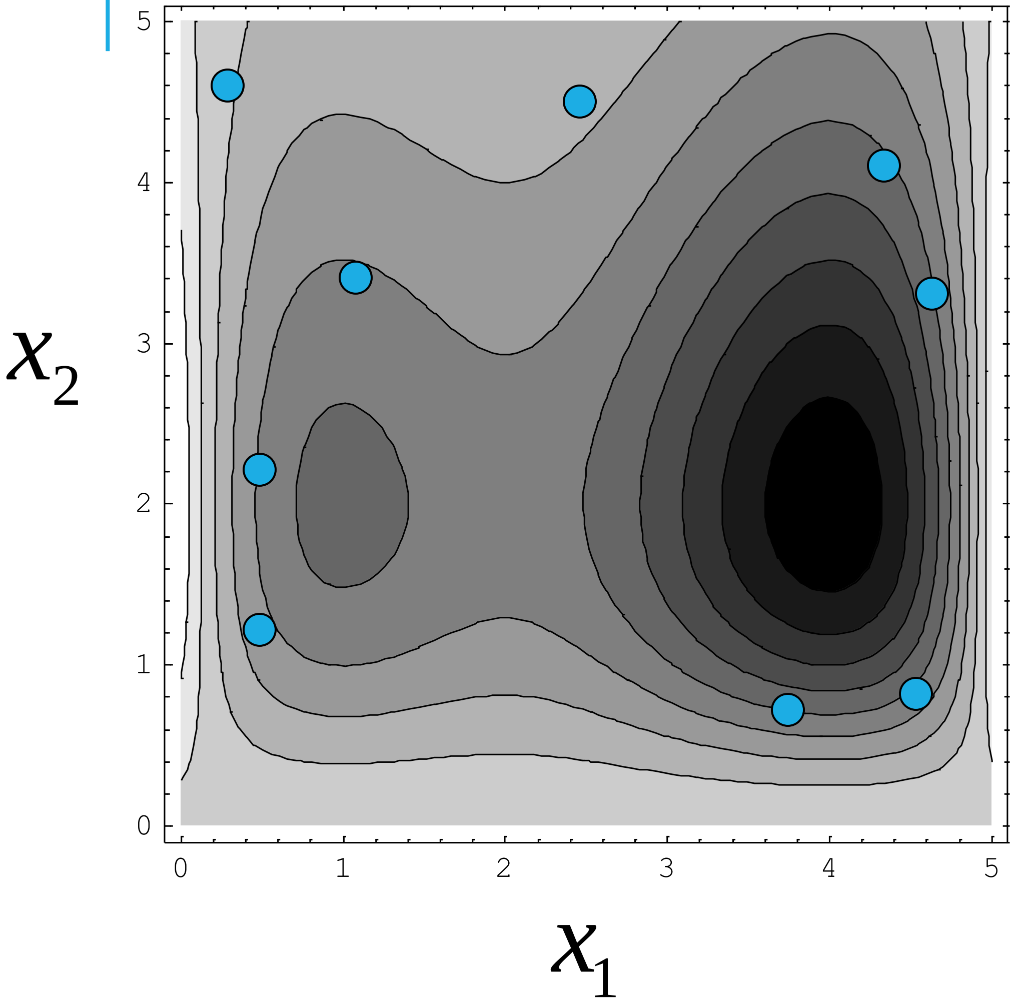
EJEMPLO

Objetivo: minimizar/maximizar J para una cierta combinación de valores de **x**



La mayoría de las veces para N>3 la cantidad de máximos/mínimos es enorme (en modelos no lineales)

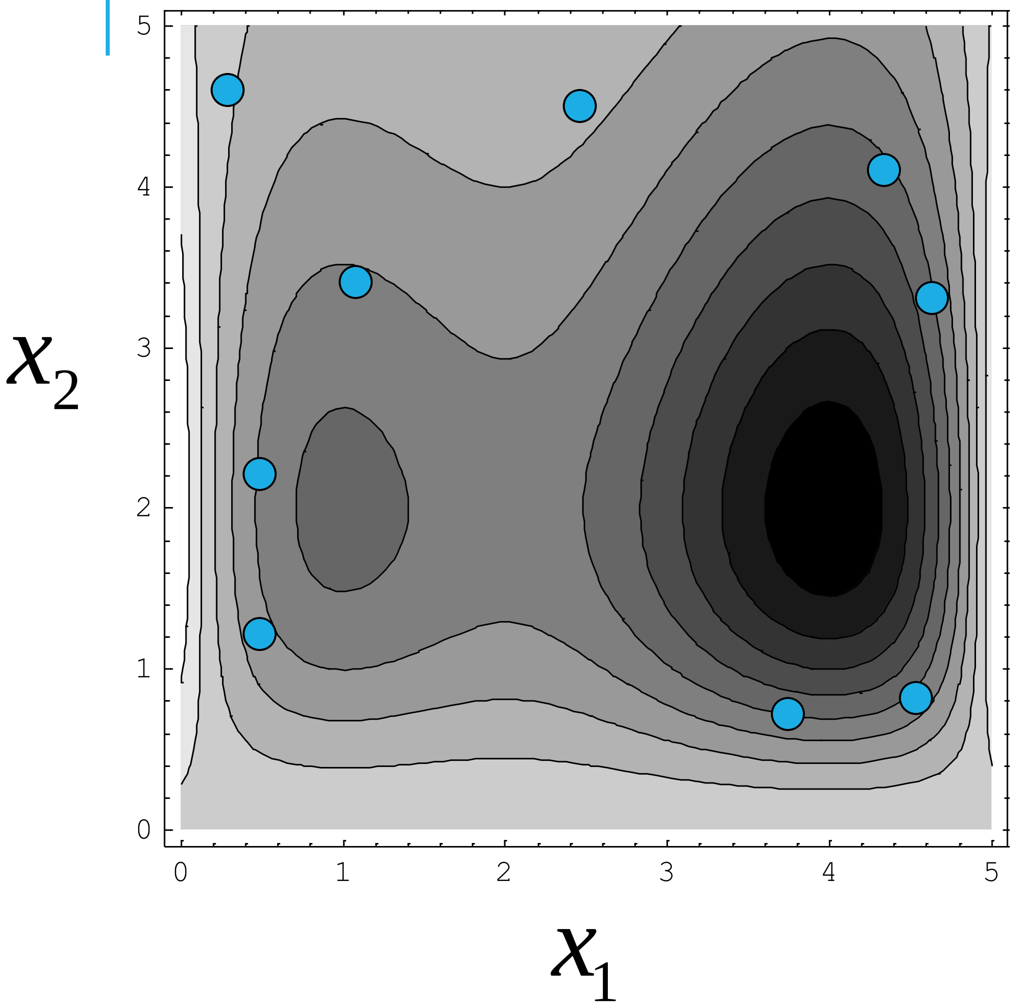
EJEMPLO



Genero N puntos aleatoriamente

$$\boldsymbol{x}^1, \boldsymbol{x}^2, \boldsymbol{x}^3, \ldots, \boldsymbol{x}^N$$

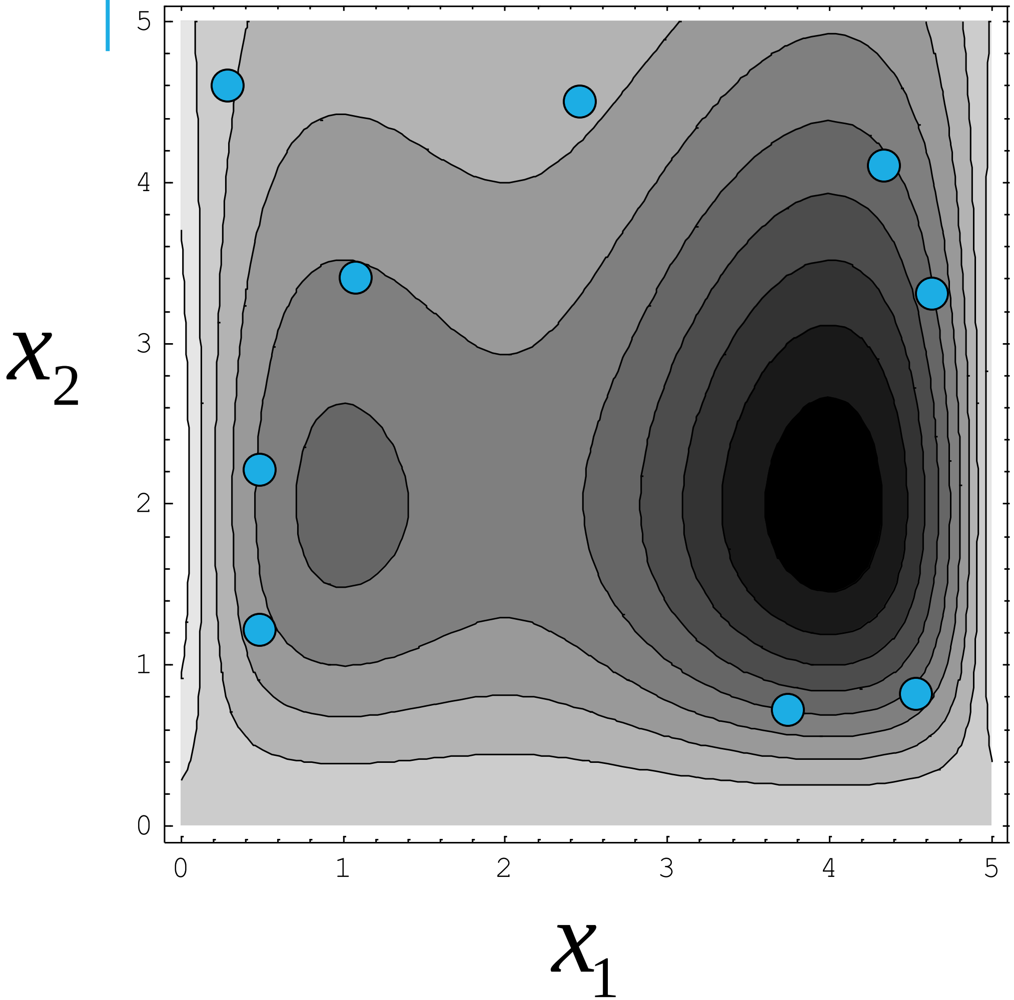
EJEMPLO



Obtengo el valor de la función objetivo J=f en esos N puntos

$$f(\boldsymbol{x}^1), f(\boldsymbol{x}^2), f(\boldsymbol{x}^3), \ldots, f(\boldsymbol{x}^N)$$

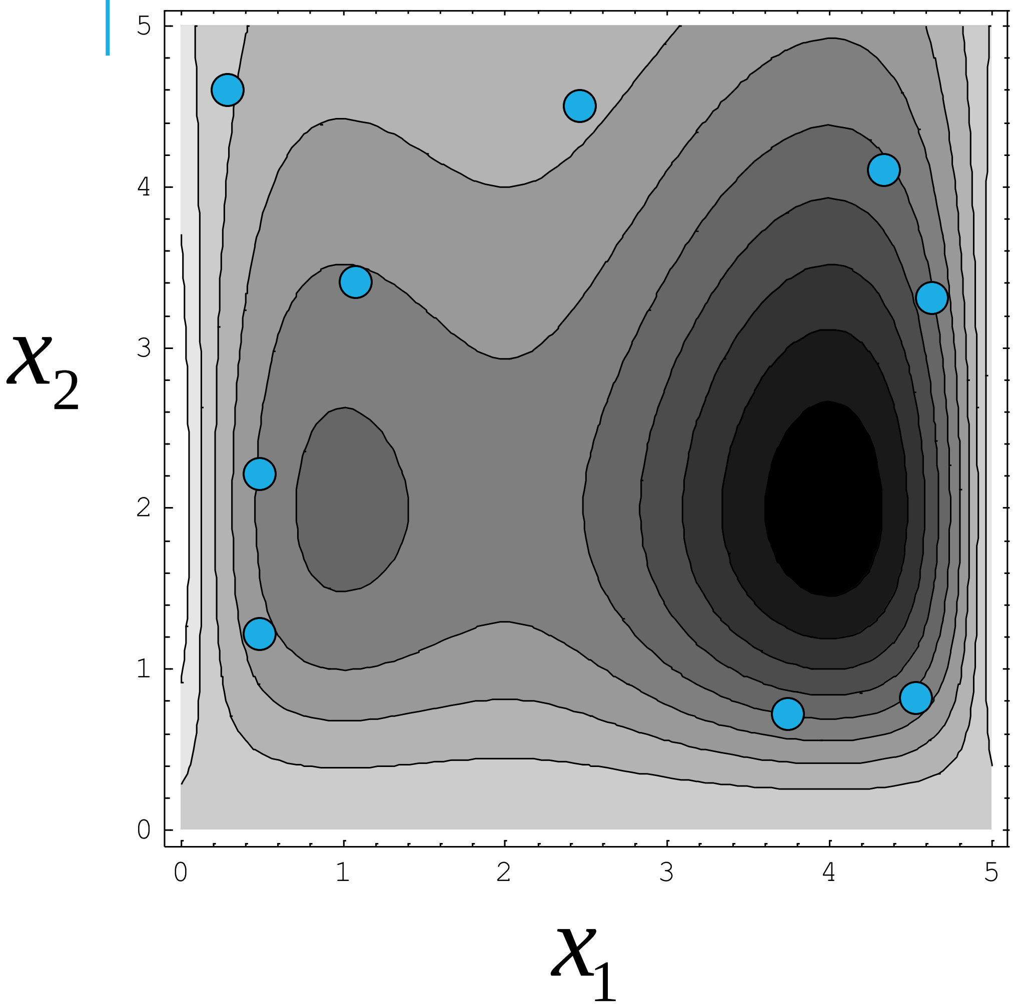
EJEMPLO



Ordeno de menor a mayor los N valores de la función objetivo

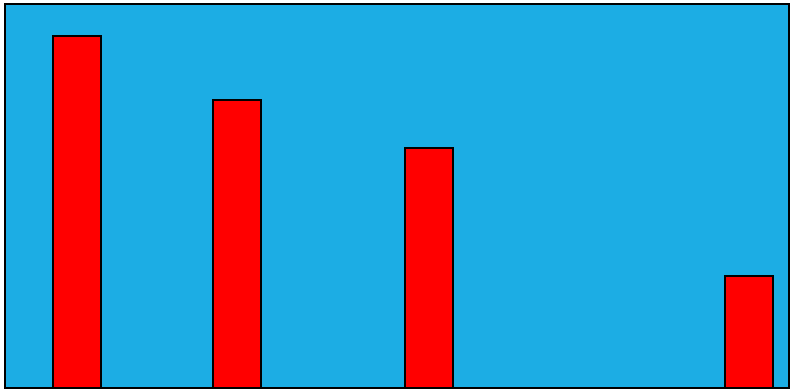
$$f^*(\boldsymbol{x}^1), f^*(\boldsymbol{x}^2), f^*(\boldsymbol{x}^3), \ldots, f^*(\boldsymbol{x}^N)$$

EJEMPLO



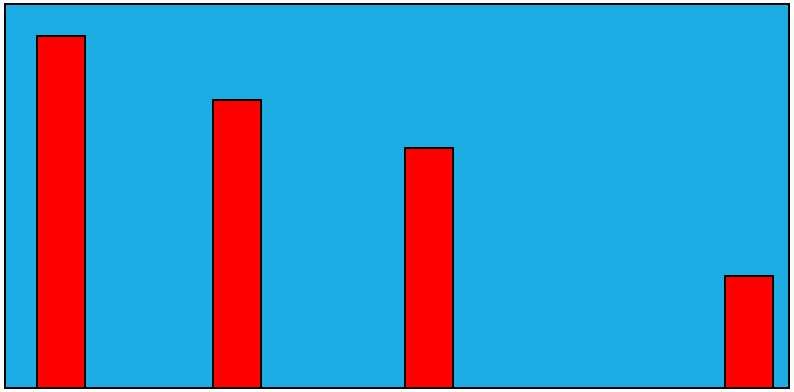
Asigno una ley de probabilidades
Trapezoidal a cada punto

$$f^*(\boldsymbol{x}^1), f^*(\boldsymbol{x}^2), f^*(\boldsymbol{x}^3), \ldots, f^*(\boldsymbol{x}^N)$$



EJEMPLO

$$f^*(\boldsymbol{x}^1),\ f^*(\boldsymbol{x}^2),\ f^*(\boldsymbol{x}^3),\ldots,\ f^*(\boldsymbol{x}^N)$$



Selecciono dos padres de acuerdo a la ley de probabilidades $\boldsymbol{x}^a, \boldsymbol{x}^b$

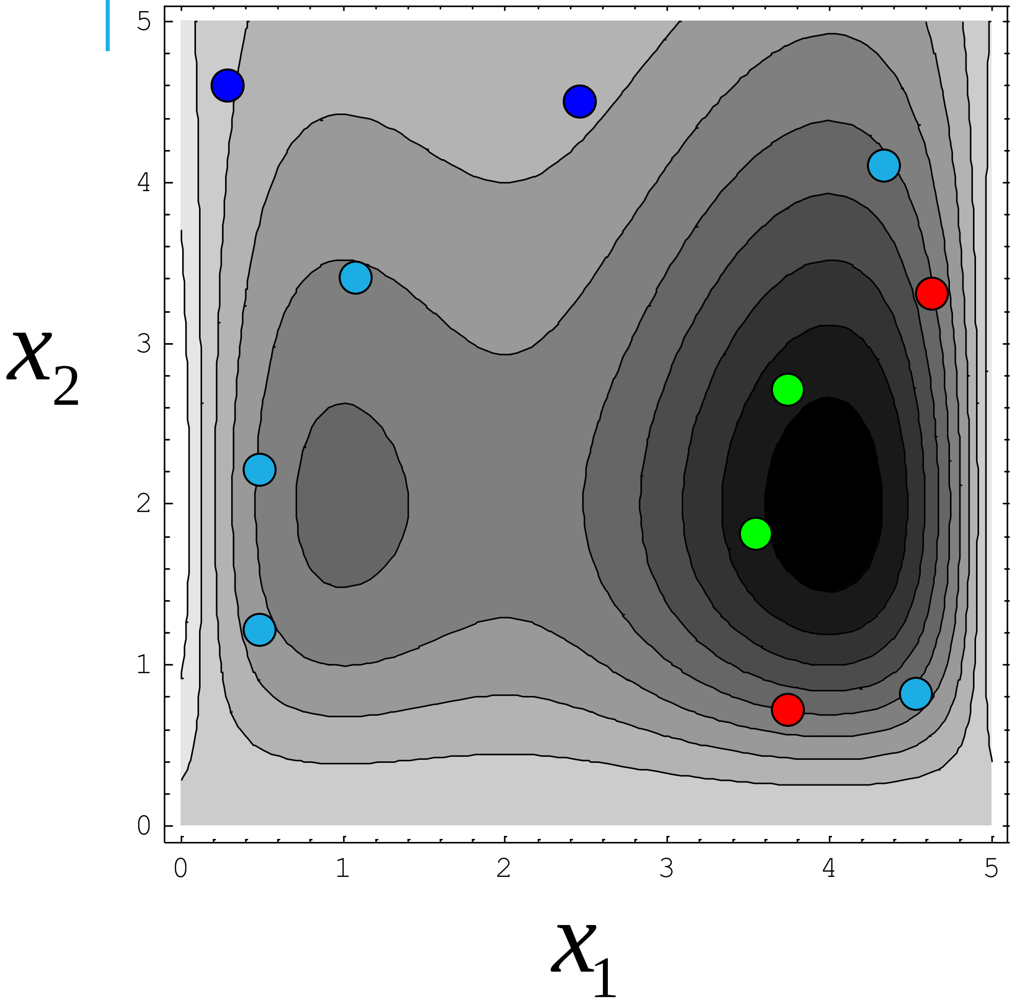
$\downarrow$

Genero dos hijos-descendientes (“offspring”) $\boldsymbol{x}^{a*}, \boldsymbol{x}^{b*}$

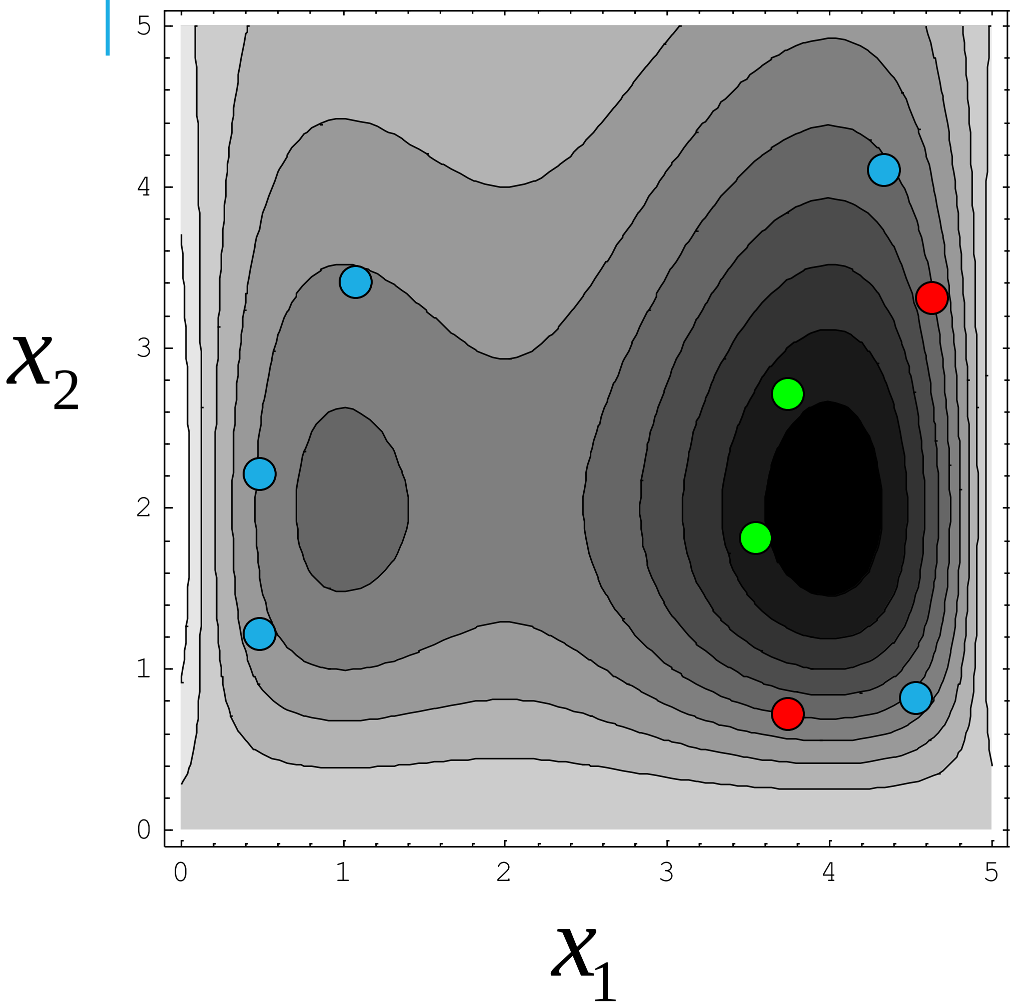
$\downarrow$

Calculo la función objetivo para los dos descendientes $f(\boldsymbol{x}^{a*}),\ f(\boldsymbol{x}^{b*})$

EJEMPLO



EJEMPLO



EJEMPLO

¿ Cómo generar un descendiente ?
Utilizando operadores de cruce y mutación

$$x^a = (x_1^a, x_2^a, x_3^a, \ldots, x_n^a)$$

$$x^b = (x_1^b, x_2^b, x_3^b, \ldots, x_n^b)$$

Cruce - crossover

Mutación - mutation

$$x^{a*} = (\boxed{x_1^a, x_2^a, \ldots, x_l^a}, x_{l+1}^b \ldots, x_n^b)$$

$$x^{b*} = (x_1^b, x_2^b, \ldots, x_l^b, \boxed{x_{l+1}^a, \ldots, x_n^a})$$

$$x^{a*} = (x_1^a, x_2^a, \ldots, x_i^*, \ldots, x_n^a)$$

Número aleatorio generado entre (a_i, b_i)