Redes neuronales artificiales

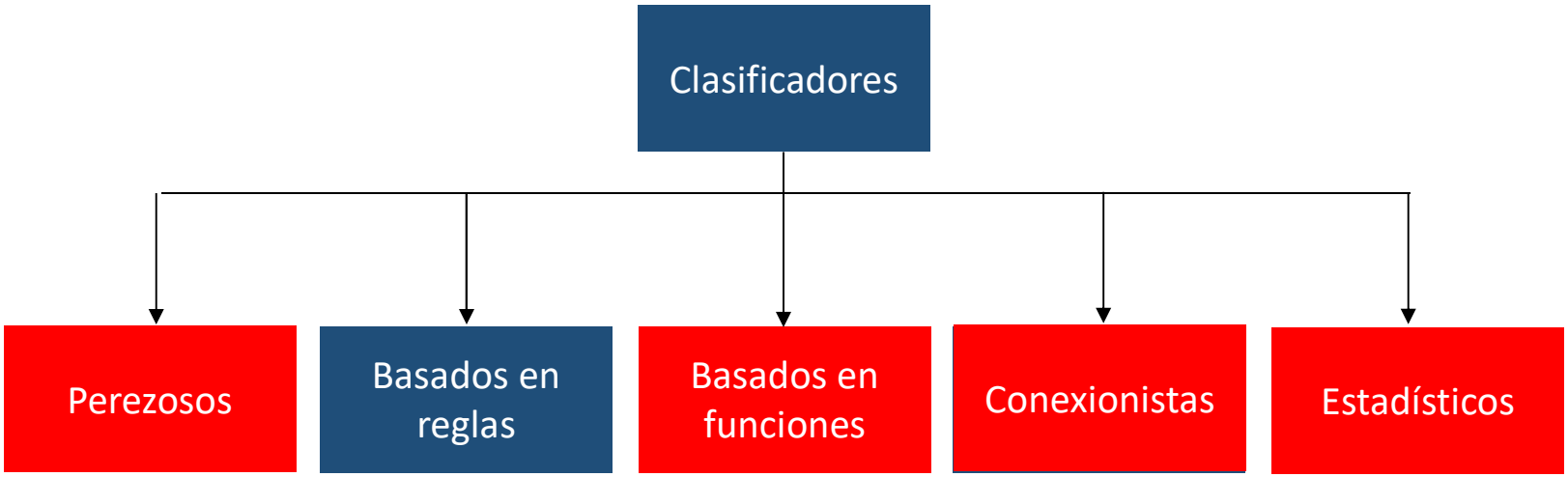
Aprendizaje maquinal

Dr. Rubén Acevedo
ruben.acevedo@uner.edu.ar

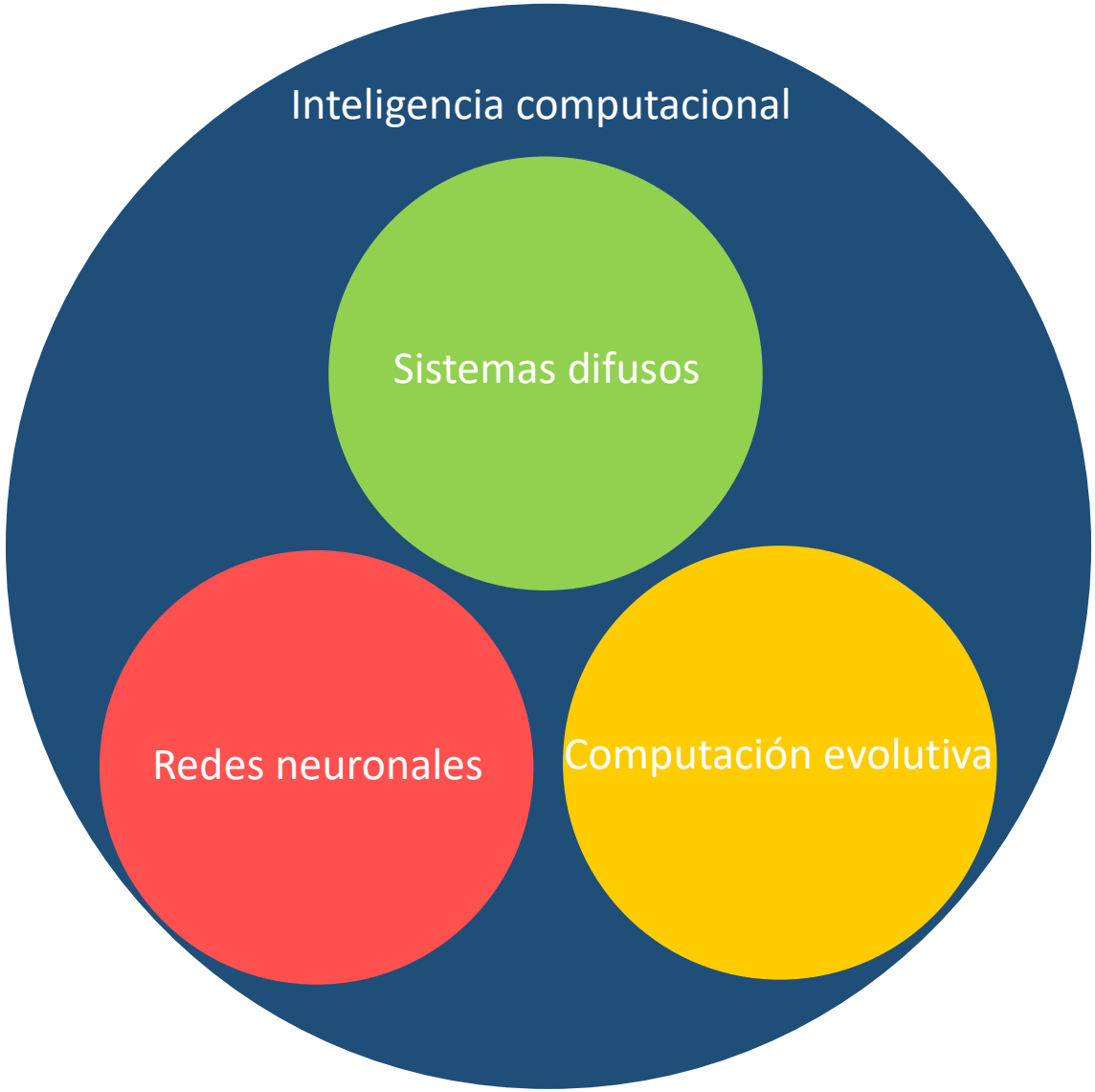
Tecnicatura Universitaria
Procesamiento y Exploración de Datos

Redes neuronales artificiales

Clasificadores



Redes neuronales artificiales



Redes neuronales artificiales

Definiciones

Una forma de computación *inspirada en modelos biológicos.*

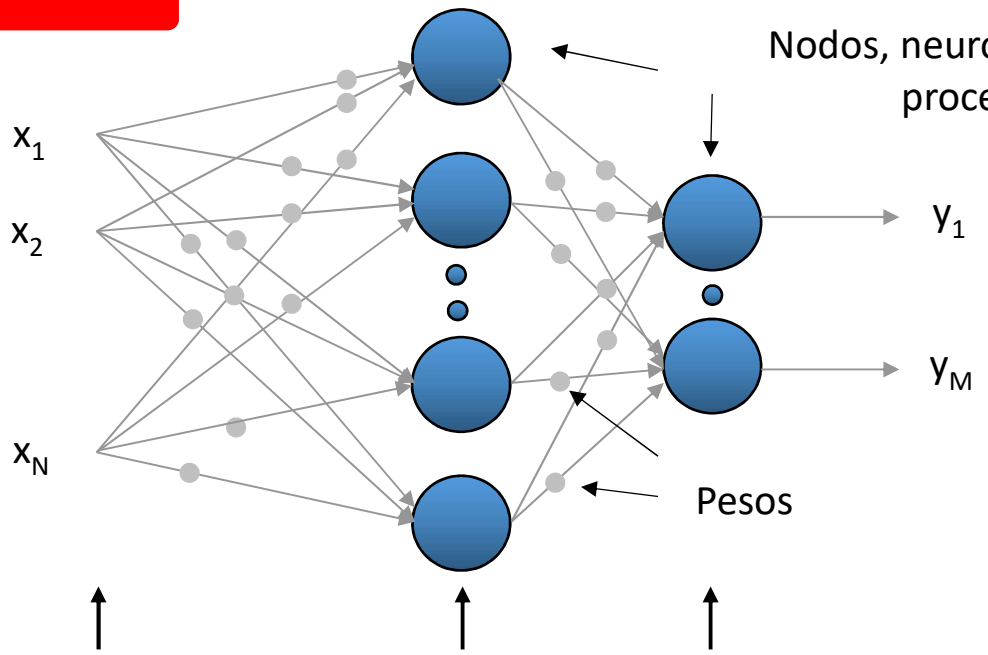
Un modelo matemático compuesto por un gran número de elementos procesales organizados en niveles.

Un sistema de computación compuesto por un gran número de elementos de procesamiento simples y muy interconectados, los cuales procesan información por medio de su estado dinámico como respuesta a entradas externas.

Son sistemas compuestos por elementos de procesamiento simples (usualmente adaptativos) interconectadas masivamente en paralelo y con organización jerárquica, las cuales intentan interactuar con los objetos del mundo real del mismo modo que lo hace el sistema nervioso biológico.

Redes neuronales artificiales

Definiciones



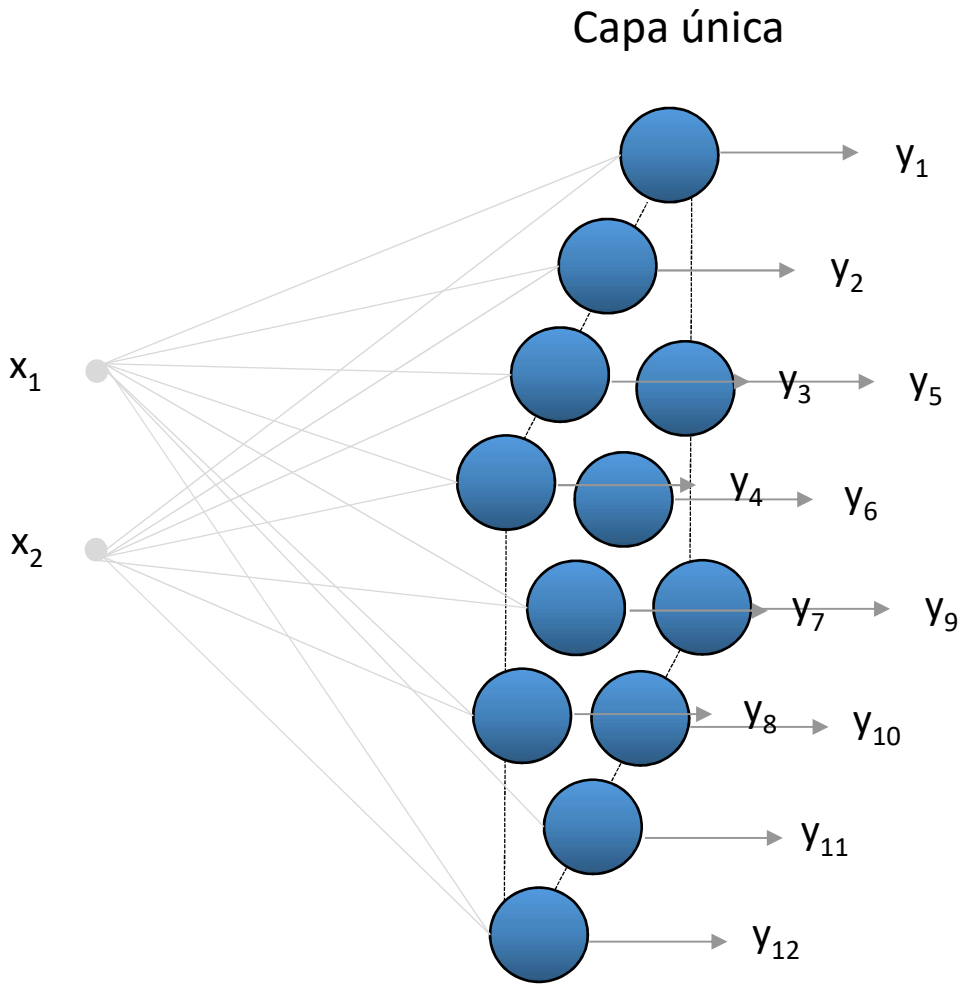
$$X = \begin{pmatrix} x_1 \\ x_2 \\ : \\ : \\ x_N \end{pmatrix}$$, x_i se denominan *características*. $$Y = \begin{pmatrix} y_1 \\ y_2 \\ : \\ : \\ y_M \end{pmatrix}$$

Instancia o patrón de entrada

Salida

Redes neuronales artificiales

Definiciones



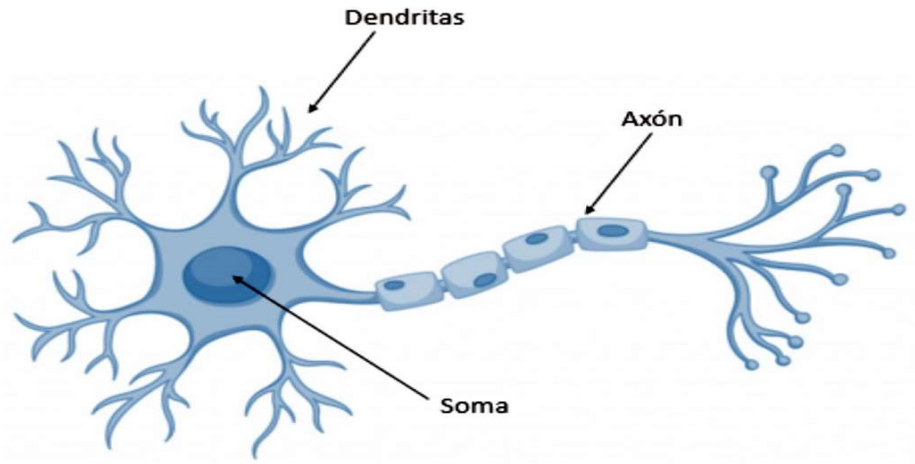
Redes neuronales artificiales

Definiciones

Descripción de nodos

El cerebro humano contiene más de cien mil millones de neuronas (nodos).

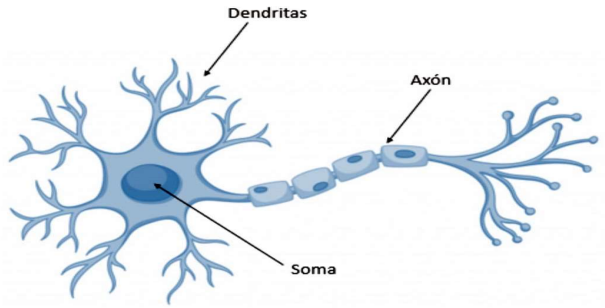
La clave para el procesamiento de la información son las conexiones entre nodos llamadas *sinápsis.*

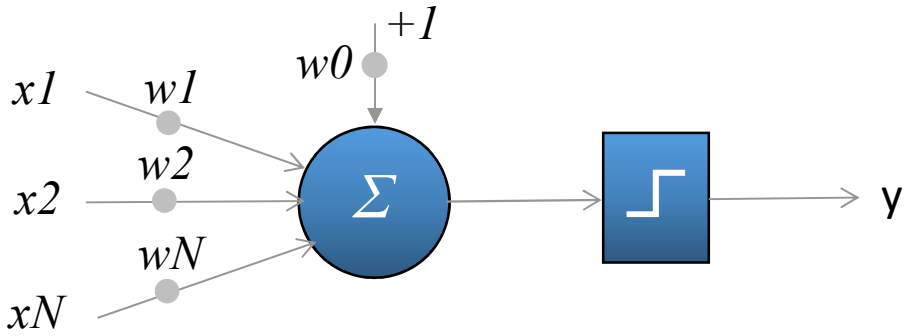


Redes neuronales artificiales

Definiciones

Nodo o neurona artificial





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

Nodos binarios: y = +1, -1

y = +1, 0

Nodos reales: y → [0, 1]

Redes neuronales artificiales

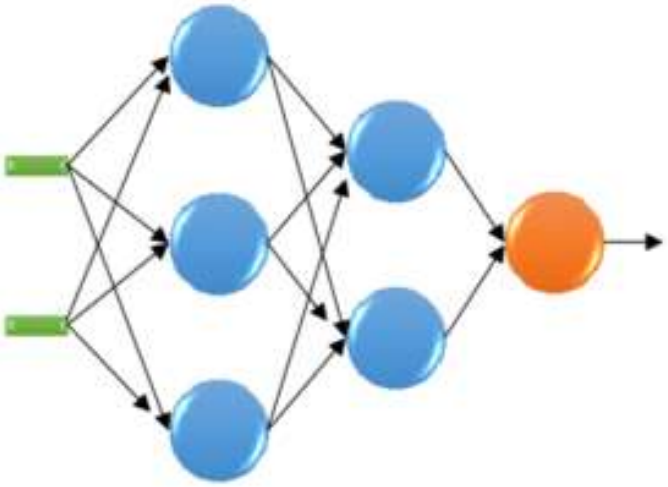
Clasificación

Redes anteroalimentadas (*feedforward*): se caracterizan por que las interconexiones de sus elementos o nodos son siempre unidireccionales. También se denominan *estáticas,* porque con una entrada determinada producen una única salida,

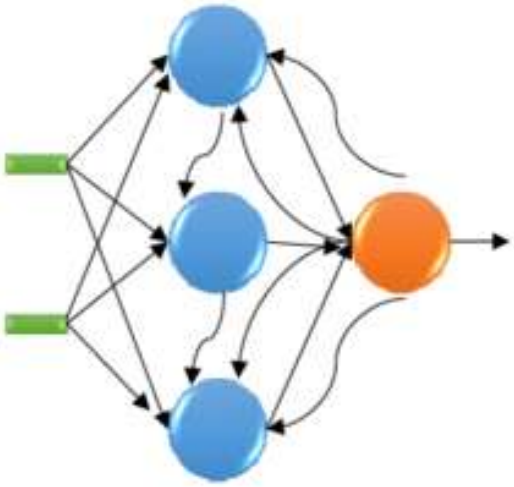
Redes recurrentes (*feedback*): al contrario que en las redes anteroalimentadas, en este caso hay conexiones de capas posteriores hacia capas anteriores. En consecuencia, se generan bucles y la salida resulta de una evolución a través de una serie de estados tras la presentación de la entrada.

Redes neuronales artificiales

Clasificación



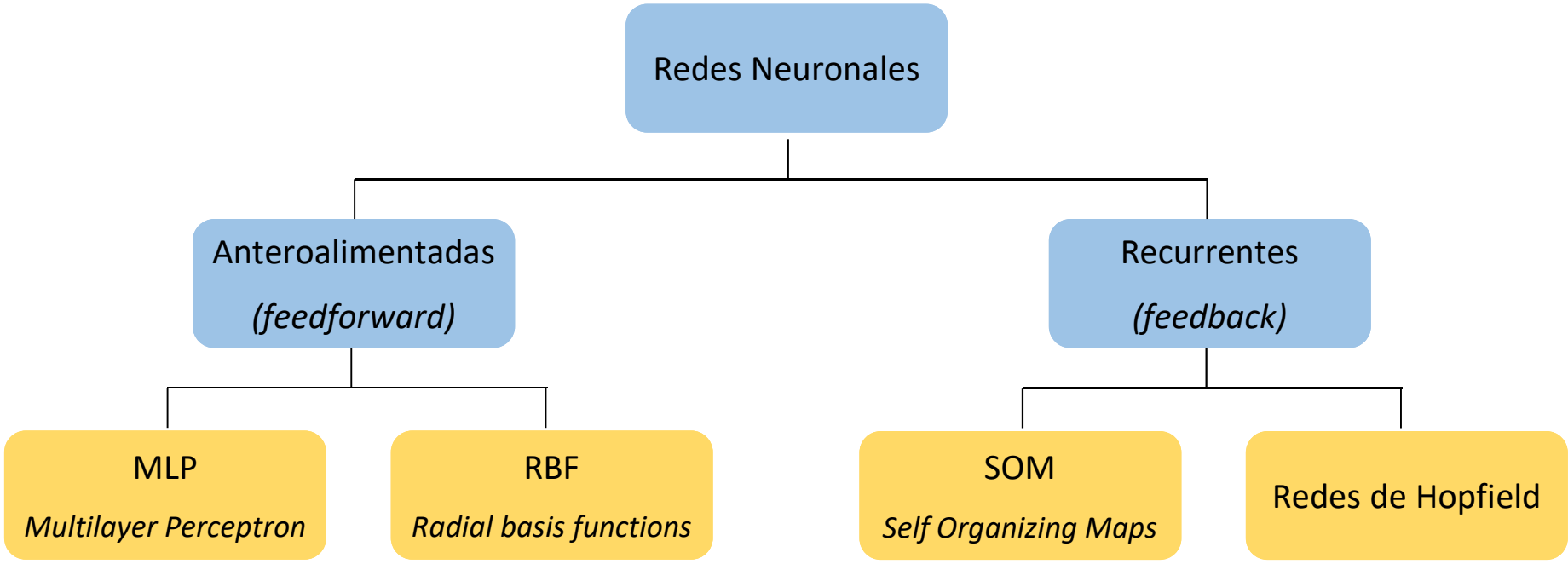
Redes anteroalimentadas



Redes recurrentes

Redes neuronales artificiales

Clasificación



Redes neuronales artificiales

Aprendizaje

La red es estimulada por el ambiente.

Se generan cambios en su estructura en función de los estímulos recibidos.

Con la nueva estructura, la red responde al ambiente de una manera diferente.

Un conjunto de reglas bien definidas, que logran la solución del problema de aprendizaje de una red, se denomina *algoritmo de aprendizaje*.

Redes neuronales artificiales

Aprendizaje

Las RNA tienen la capacidad de almacenar conocimientos experimentales, para su posterior utilización en la resolución de problemas reales. Este proceso tiene dos características:

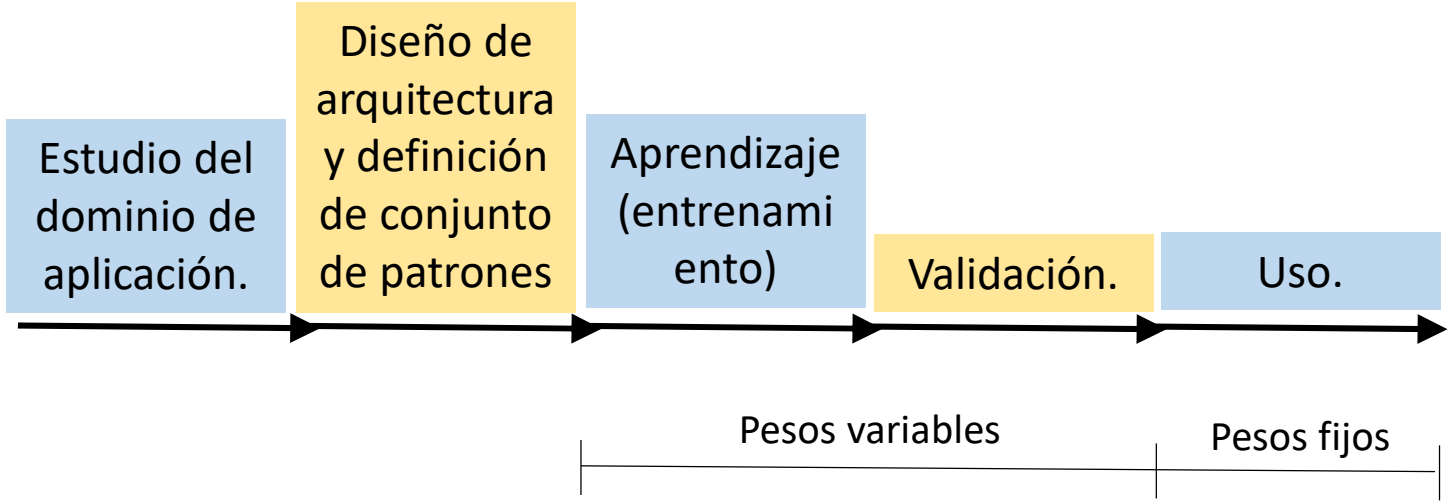
1) El conocimiento es adquirido por la red mediante un proceso de aprendizaje.

2) El conocimiento se almacena en la red en forma distribuida a través de los pesos que caracterizan cada conexión entre neuronas (*sinapsis*).

Aprendizaje

On line: en este caso la red neuronal puede aprender durante su funcionamiento habitual.

Off line: una fase de *aprendizaje* (entrenamiento) y una fase de *uso* (operación) o funcionamiento, existiendo un conjunto de datos de entrenamiento y un conjunto de datos de test o prueba, que serán utilizados en la correspondiente fase.



Redes neuronales artificiales

Aprendizaje

Paradigmas

Aprendizaje supervisado: Tal vez el esquema más utilizado a lo largo de la historia, este paradigma de aprendizaje está en función de *salida deseada* vs *salida obtenida*. Ejemplo: perceptron simple y multicapa (multilayer perceptron).

Aprendizaje no supervisado: Se presentan a la red grandes cantidades de datos, en los que estos algoritmos buscan regularidades estadísticas, definiendo en base a ellas *categorías* o *clusters*. Ejemplo: mapas de auto-organización (self organizing maps).

Aprendizaje híbrido: Utiliza una combinación de los dos anteriores, bien en un orden o en el otro. Ejemplo. Redes de funciones de base radial (radia basis functions)

Redes neuronales artificiales

Aprendizaje

Biológicamente se acepta que la información memorizada en el cerebro se relaciona con la *fuerza sináptica* entre neuronas.

En las RNA se considera que el conocimiento se encuentra representado en los *pesos de las conexiones*.

El proceso de aprendizaje se basa en *cambios en los pesos*.

Redes neuronales artificiales

Aprendizaje

Peso Nuevo = Peso Viejo + Cambio de Peso

$$w_{ij}(n+1) = w_{ij}(n) + \Delta w_{ij}(n)$$

- Aprendizaje supervisado
  - Corrección de error.
  - Por refuerzo.
  - Estocástico.
- Aprendizaje no supervisado
  - Hebbiano
  - Competitivo y comparativo

Validación

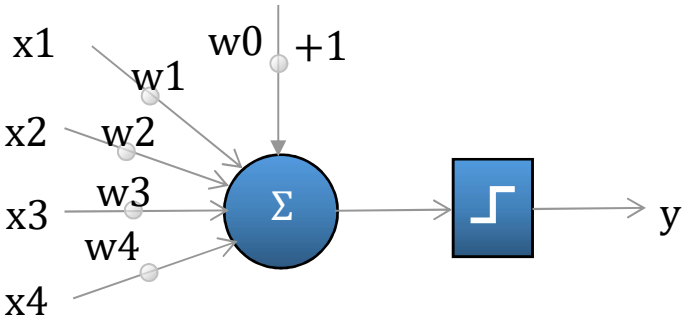
El paso posterior al entrenamiento es comprobar si la red neuronal puede resolver nuevos ejemplos (patrones) del problema para el que ha sido entrenada.

Para esto se requiere de otro conjunto de datos, denominado *conjunto de validación o prueba*, en el cual para cada ejemplo de este conjunto se conoce la clase a la que pertenece (salida deseada).

Esto permite comparar la salida de la red neuronal con la salida deseada y evaluar distintos índices de desempeño, tales como *tasa de aciertos*, *sensibilidad* y *especificidad* entre otros.

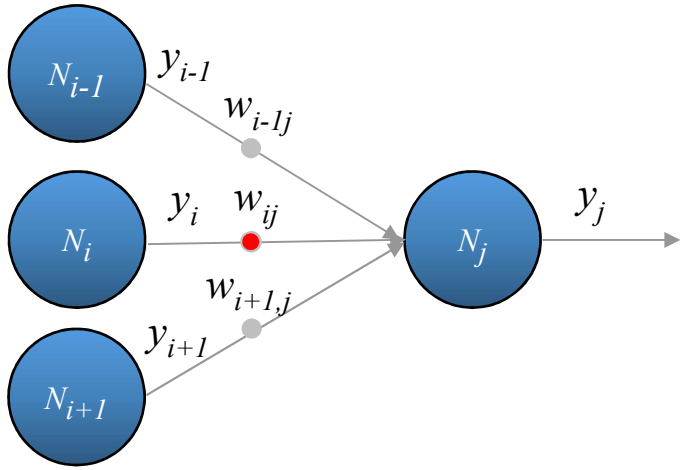
Redes neuronales artificiales

Perceptron simple



$$y = sign(\sum_{i=1}^{N} x_i . w_i - w_0)$$

Consiste en ajustar los pesos de las conexiones de la red en función de la diferencia entre los valores deseados y los obtenidos a la salida de la red, es decir, en función del error cometido en la salida.



$$w_{ij}(n+1) = w_{ij}(n) + \underbrace{\mu.e.y_i}_{\Delta w} = w_{ij}(n) + \mu.\left(d_j - y_j\right).y_i$$

Inicialización de pesos: lo habitual es inicializar los pesos con valores aleatorios dentro de un determinado rango.

Orden de presentación de patrones: si bien ni hay uno establecido en general es beneficioso que sea de forma aleatoria.

Detención del proceso:

- Cuando el error cuadrático alcanza un valor mínimo.
- Cuando la variación del error cuadrático varía dentro de un determinado umbral.
- Cuando se alcanza una cantidad prefijada de iteraciones.

Perceptron simple

Algoritmo entrenamiento

Regla del perceptron

a) Inicializar los pesos aleatoriamente **w** entre (-0.5 , 05).

b) Escoger un patrón $\boldsymbol{x}$ del conjunto de entrenamiento, calcular la salida y el error con la salida deseada y_d.

c) Adaptar los pesos: empezando con la capa de salida, y *"modificando hacia atrás"*, según la ecuación

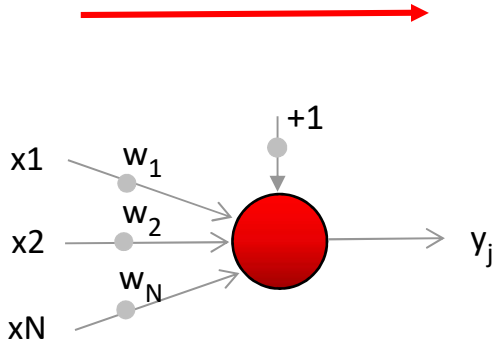
$$w_i(n+1) = w_i(n) + \mu.(y_d - y).\mathrm{x}_i$$

d) Volver al paso b) y repetir el proceso hasta que los pesos converjan.

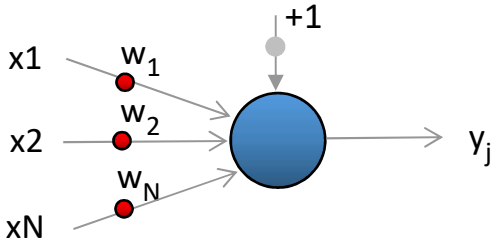
Redes neuronales artificiales

Perceptron simple

Algoritmo entrenamiento



$$y = sign(\sum_{i=1}^{N} x_i . w_i - w_0)$$

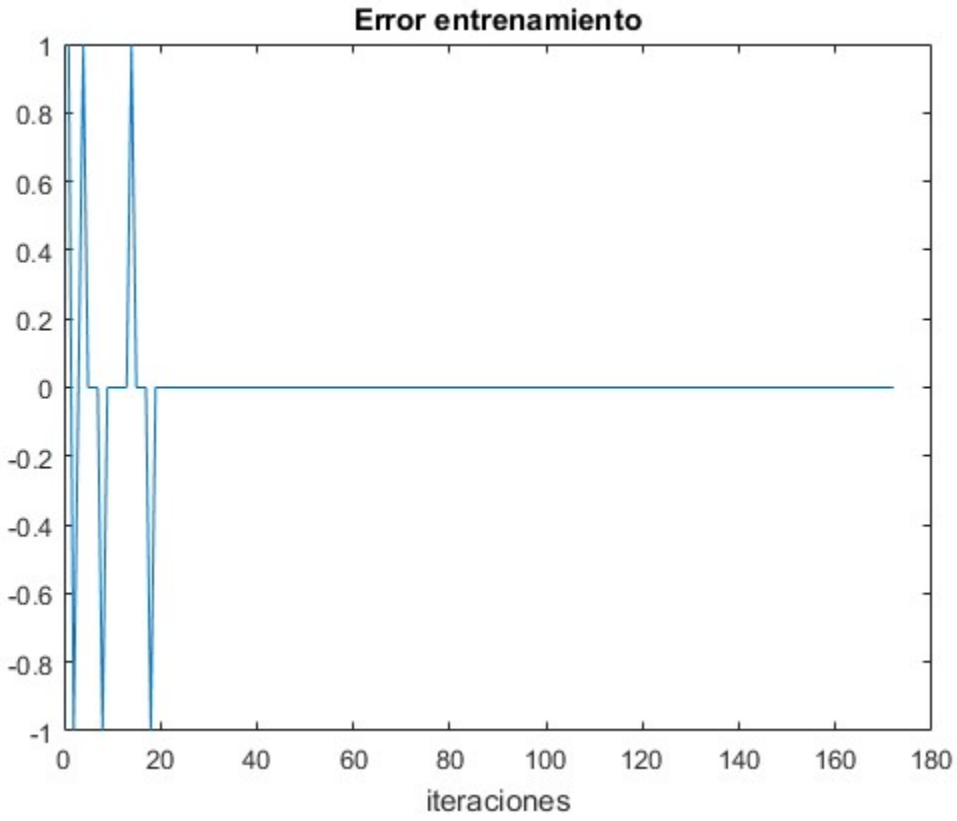


$$w_i(n+1) = w_i(n) + \mu.(y_d - y).\mathrm{x}_i$$

Redes neuronales artificiales

Perceptron simple

Algoritmo entrenamiento



$$w_i(n+1) = w_i(n) + \mu.(y_d - y).\mathrm{x}_i = w_i(n) + \Delta w$$

Redes neuronales artificiales

Perceptron simple

Separabilidad lineal

$$y = sign(\sum_{i=1}^{N} x_i . w_i - w_0)$$

$$y = sign\left(\sum_{i=1}^{2} x_i . w_i - w_0\right) = sign(x_1 . w_1 + x_2 . w_2 + w_0)$$

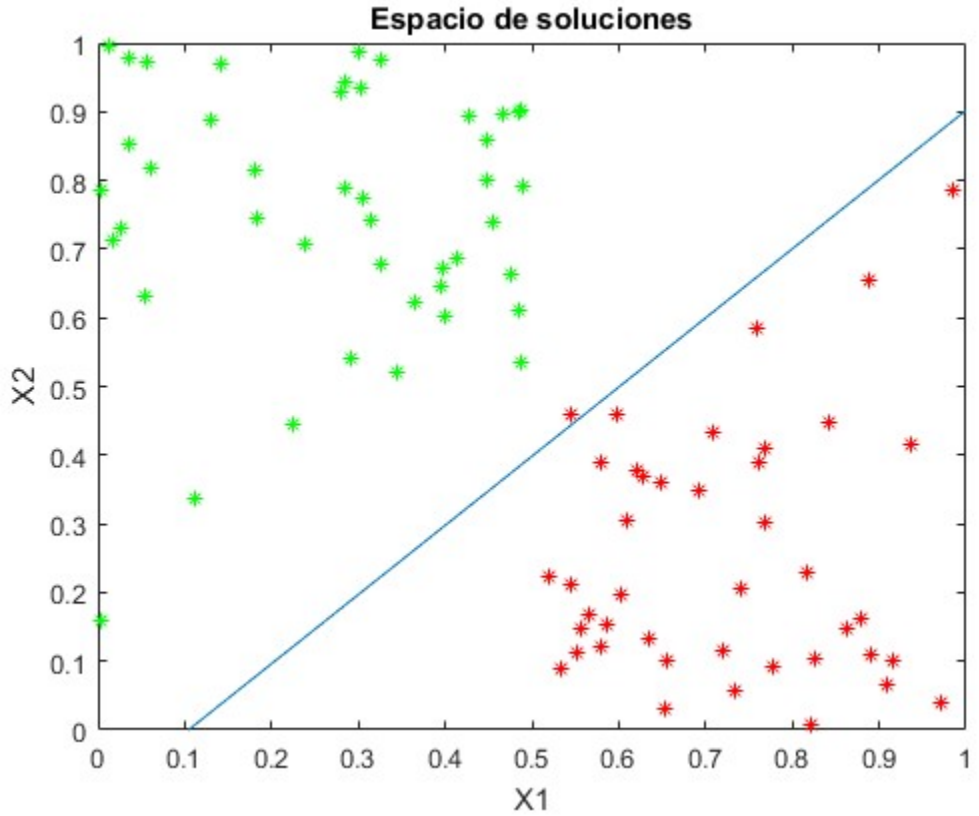
Frontera de decisión: $y = x_1 . w_1 + x_2 . w_2 + w_0 = 0$

$$x_1 . w_1 + x_2 . w_2 + w_0 = 0 \rightarrow x_2 = x_1 . \frac{w_1}{w_2} + \frac{w_0}{w_2}$$

Redes neuronales artificiales

Perceptron simple

Separabilidad lineal

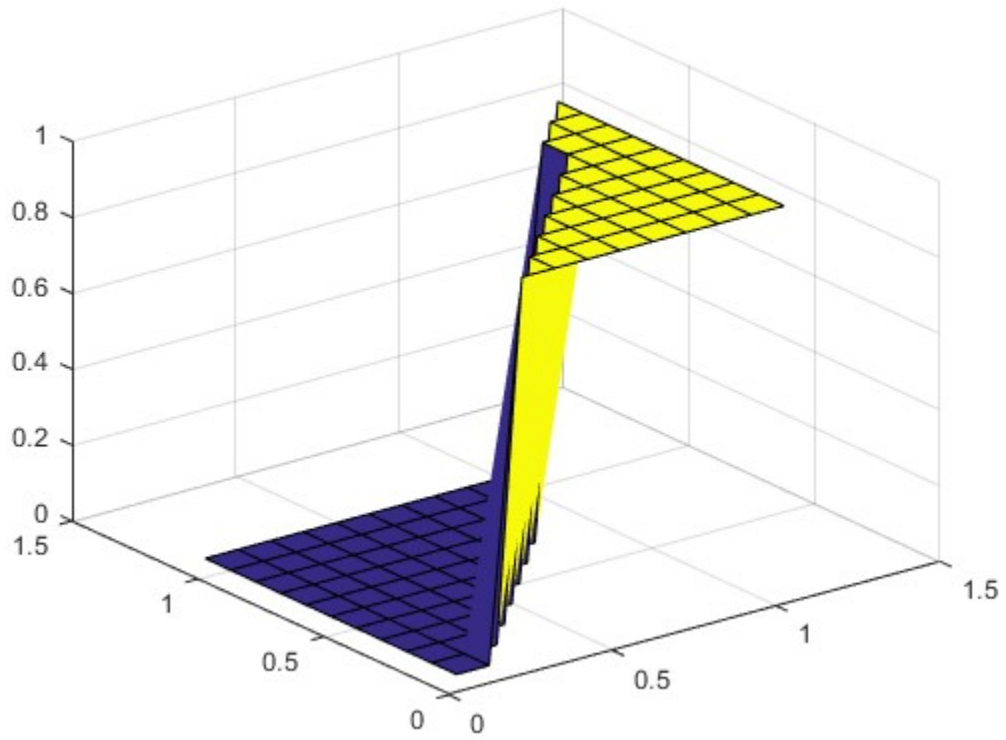


$$x_2 = x_1 . \frac{w_1}{w_2} + \frac{w_0}{w_2}$$

Redes neuronales artificiales

Perceptron simple

Separabilidad lineal

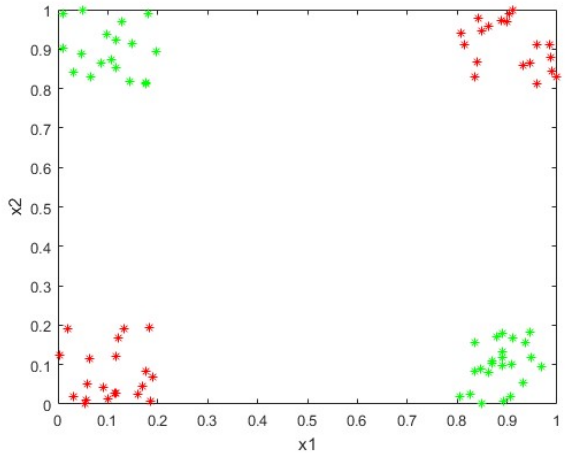


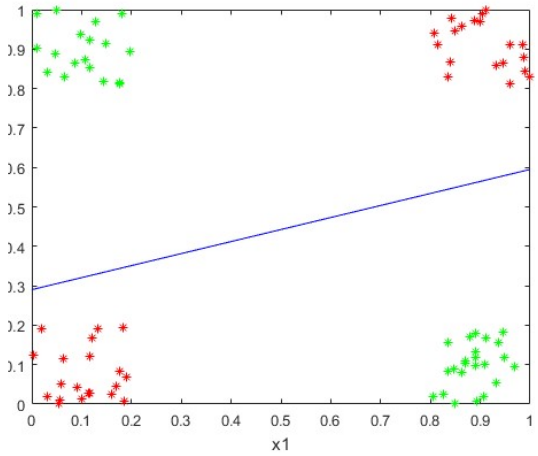
Redes neuronales artificiales

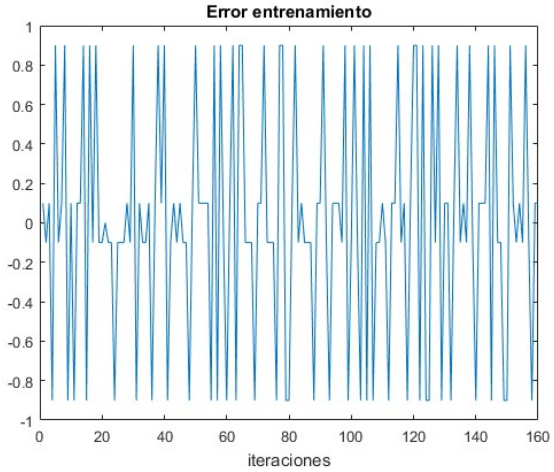
Perceptron simple

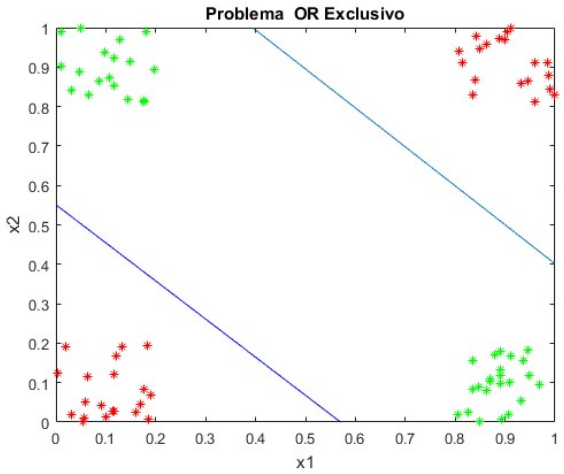
Separabilidad lineal

Problema del Or-Exclusivo (XOR)





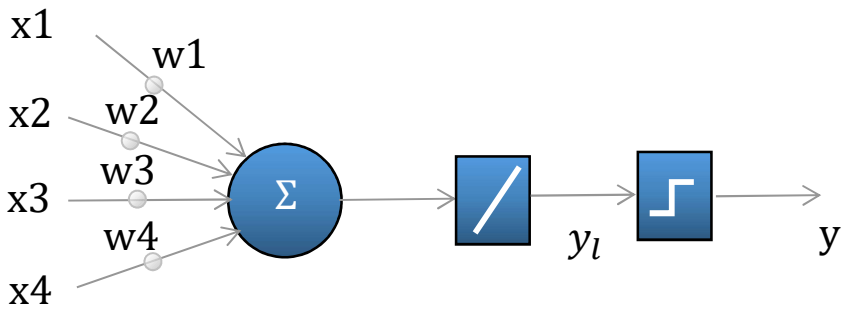




→ Perceptron MultiCapas

AdaLinE

Adaptive Linear Element



$$y_l = \sum_{i=1}^{N} x_i . w_i$$

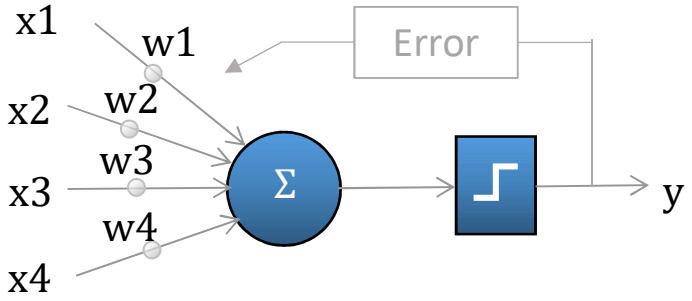
$$y = sign(y_l)$$

Redes neuronales artificiales

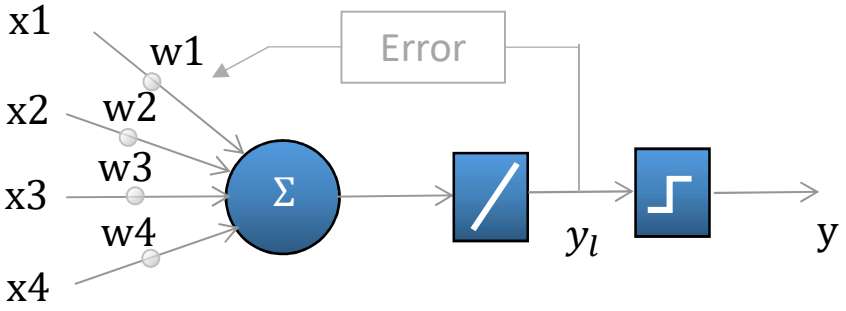
AdaLinE

Algoritmo entrenamiento

Perceptron simple



Adaline



Redes neuronales artificiales

AdaLinE

Algoritmo entrenamiento

$$e_n = y_d - y_l = y_d - \sum_{i=1}^{N} x_i . w_i$$

$$ECM = \sum_{n=1}^{M} {e_n}^2 = \sum_{n=1}^{M} \left(y_d - \sum_{i=1}^{N} x_i . w_i \right)^2$$

Pesos que minimizan el ECM $\rightarrow \frac{\partial}{\partial w_i} ECM = 0$

$$w_i(n+1) = w_i(n) + \mu . (y_d - y_l) . \mathrm{x}_i$$

AdaLinE

Algoritmo entrenamiento

Algoritmo Regla Delta

a) Inicializar los pesos aleatoriamente **w** entre (-0.5 , 05).

b) Escoger un patrón $\boldsymbol{x}$ del conjunto de entrenamiento, calcular la salida y el error entre la salida lineal y_l y la salida deseada y_d.

c) Adaptar los pesos: empezando con la capa de salida, y *"modificando hacia atrás"*, según la ecuación

$$w_i(n+1) = w_i(n) + \mu.(y_d - y_l).\mathrm{x}_i$$

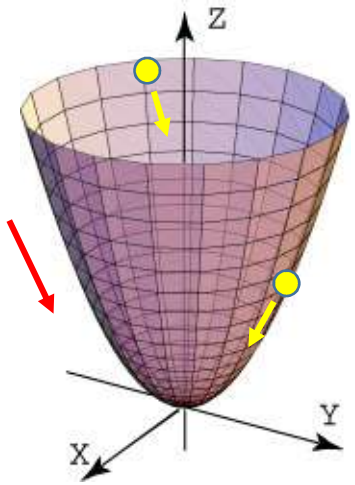
d) Volver al paso b) y repetir el proceso hasta que los pesos converjan.

Redes neuronales artificiales

AdaLinE

Algoritmo entrenamiento

Gradiente descendiente del error



$$ECM = \frac{1}{N}.\sum_{n=1}^{N} e(n)^2 = \sum_{n=1}^{N} (y(n) - y_l(n))^2$$

Redes neuronales artificiales

Fin de la clase