INTRODUCCIÓN A LA COMPUTACIÓN NEURONAL

Sancho Salcedo Sanz

Departamento de Teoría de la Señal y Comunicaciones

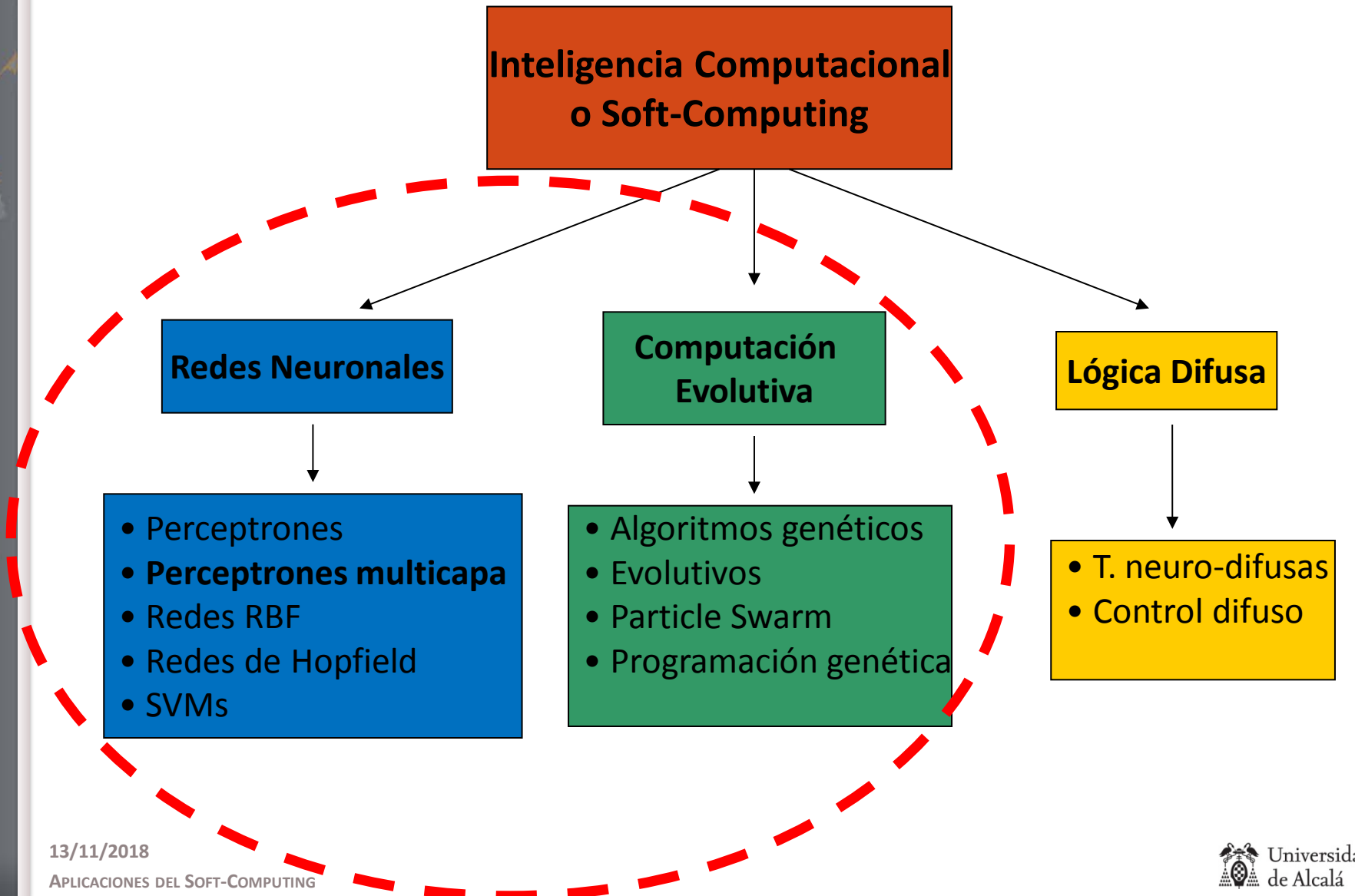
Universidad de Alcalá

ÍNDICE

- **Parte II: Computación neuronal**
- **Teoría**
  - Redes neuronales artificiales.
    - Perceptrones multicapa.
    - Extreme Learning Machines.
  - Máquinas de vectores soporte.
    - Problemas de clasificación
    - Problemas de regresión
  - Otros algoritmos de reconocimiento de patrones.
    - K-NN
    - K-medianas

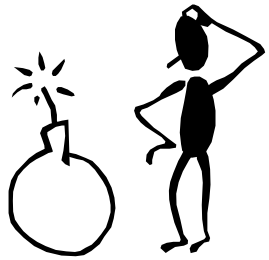
(BREVE)
INTRODUCCIÓN

INTRODUCCIÓN

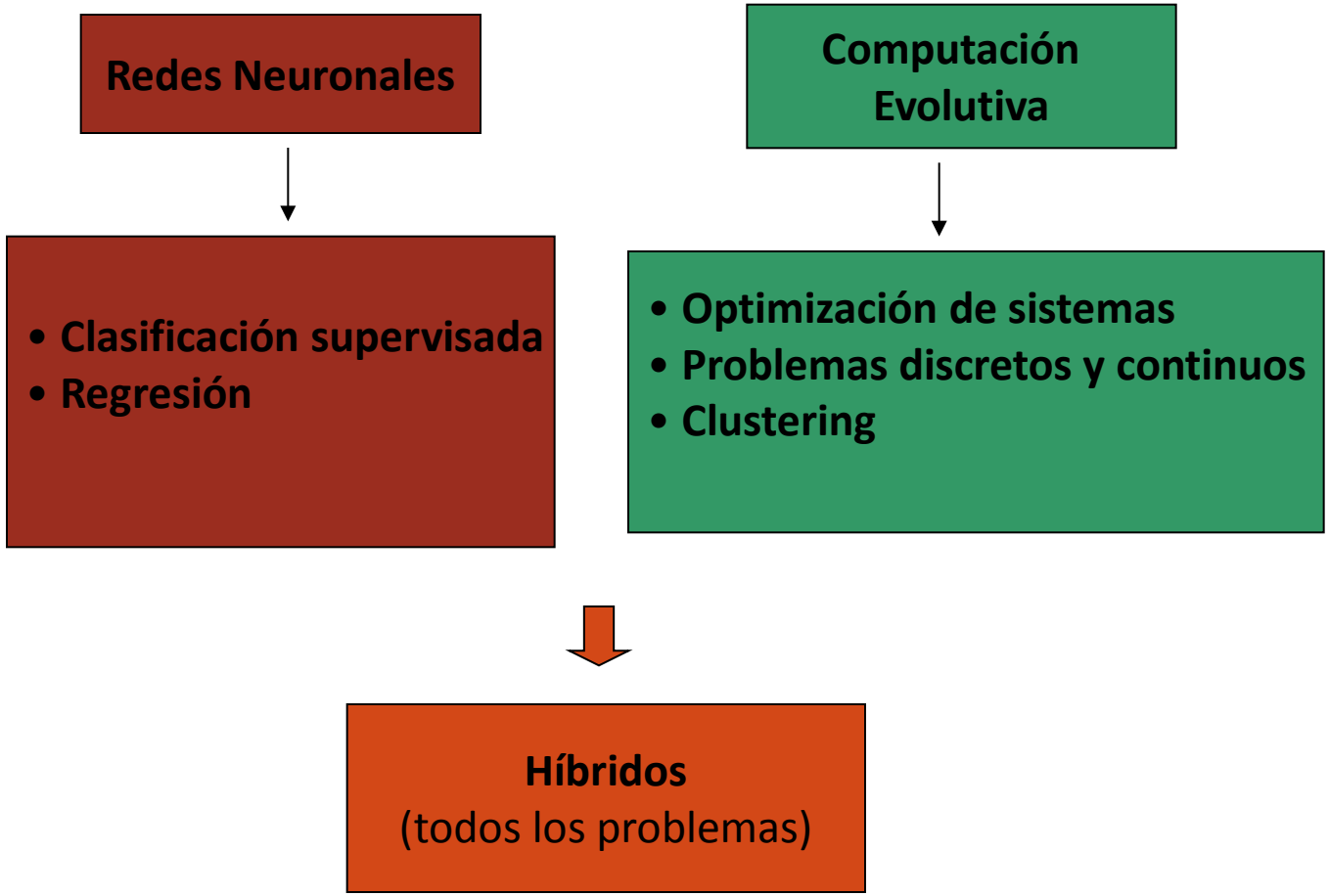


INTRODUCCIÓN

- **¿Cuándo y por qué usar técnicas de Soft-Computing?**
- Fundamentalmente cuando nada más funcione:
  - Problemas difíciles en que las técnicas tradicionales no puedan ser aplicadas o den malos resultados.
  - Usualmente son Problemas de tipo discreto, de optimización combinatoria, clasificación, regresión, control etc.
  - Amplia gama de posibilidades para abordar estos problemas, muchos algoritmos y sus correspondientes hibridaciones.
  - Complejidad de programación pequeña, conceptualmente sencillos y muy potentes en cuanto a los resultados obtenidos.

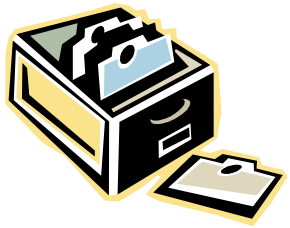


INTRODUCCIÓN



INTRODUCCIÓN

- **Problemas tipo**



- **Clasificación:**

  Un problema de clasificación consiste en asignar un conjunto de datos de entrada (usualmente vectores) a una serie de clases predeterminadas.

  – Supervisada: Hay una serie de ejemplos anteriores para guiar la clasificación. A partir de estos ejemplos se construye la máquina de clasificación, que traza fronteras entre las clases.

  – No supervisada: No existen ejemplos anteriores. La clasificación debe ser llevada a cabo sólo a partir de los datos existentes. Normalmente se habla de problemas de clustering en este caso.

INTRODUCCIÓN

- **Ejemplos de problemas de clasificación:**
- **Clasificación supervisada:**

  Ejemplo: Clasificación de días por producción de energía en un parque eólico. Cada día estaría definido por una serie de variables de tipo meteorológico. El problema sería clasificar el día en dos clases, si se va a producir más de X potencia o menos de X. Este problema sería también muy fácil de extender a N clases.

- **Clasificación no supervisada:**

  Ejemplo: Clasificación de días en diferentes situaciones sinópticas. De nuevo un día estaría definido por una serie de variables de tipo meteorológico, y el problema consiste en hacer un cluster de días semejantes que definan situaciones sinópticas similares.

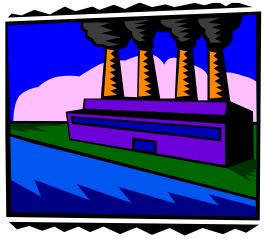
INTRODUCCIÓN

- **Problemas tipo**
- **Regresión:**

  Un problema de regresión consiste en completar una determinada función a partir de una serie de datos discretos de la misma.

- **Ejemplos:**

  1. Predicción de viento-potencia en parques eólicos.
  2. Predicción de temperatura-consumo eléctrico.
  3. Predicción de contaminantes en ciudades.



INTRODUCCIÓN

- **Problemas tipo**

- **Optimización de sistemas:**
  - Aplicación típica de algoritmos de tipo evolutivo.
  - Consiste en abordar el diseño óptimo de sistemas, **DIFICILES !!!!.**
  - Codificación del problema, función objetivo,
  - Maximización o minimización obtención de solutiones quasi-óptimas.

Ejemplos:

1. Diseño de parques eólicos.
2. Diseño de redes de comunicaciones móviles.

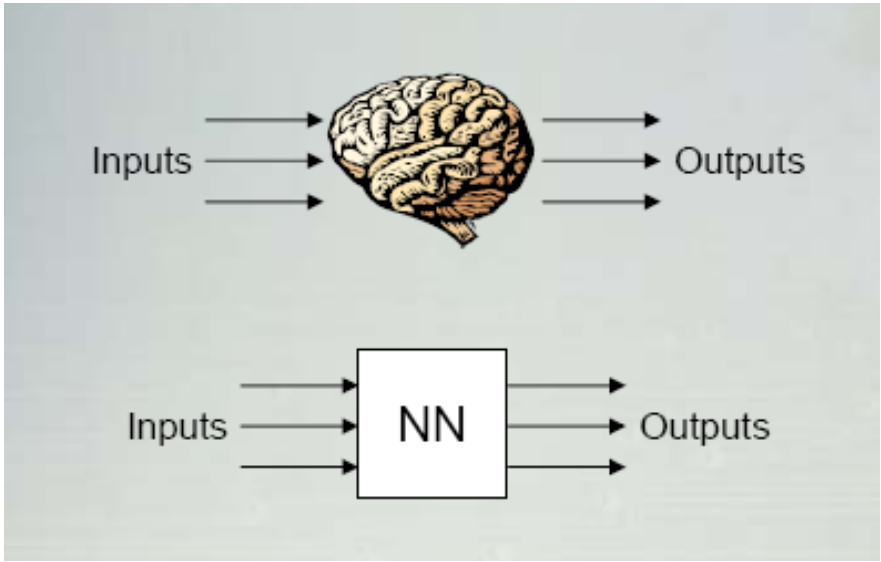
PROGRAMA DE TEORÍA

COMPUTACIÓN NEURONAL

INTRODUCCIÓN A LAS REDES NEURONALES ARTIFICIALES

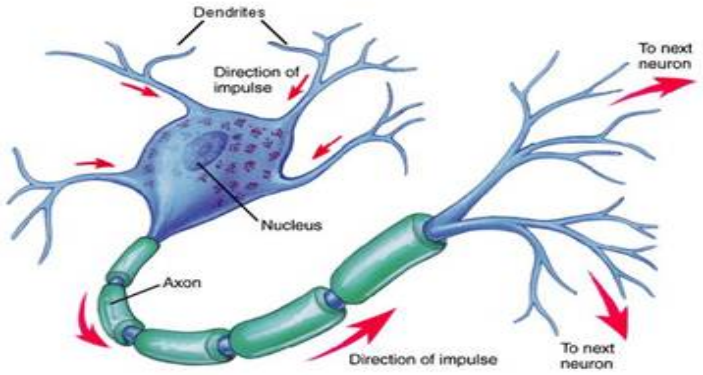
Redes Neuronales Artificiales

- **¿Qué son?**
- Sistemas de procesamiento de la información.
- Inspirados en el cerebro.
- **Aprenden!!** a partir de una serie de ejemplos (muestras).
- Perceptrón multicapa

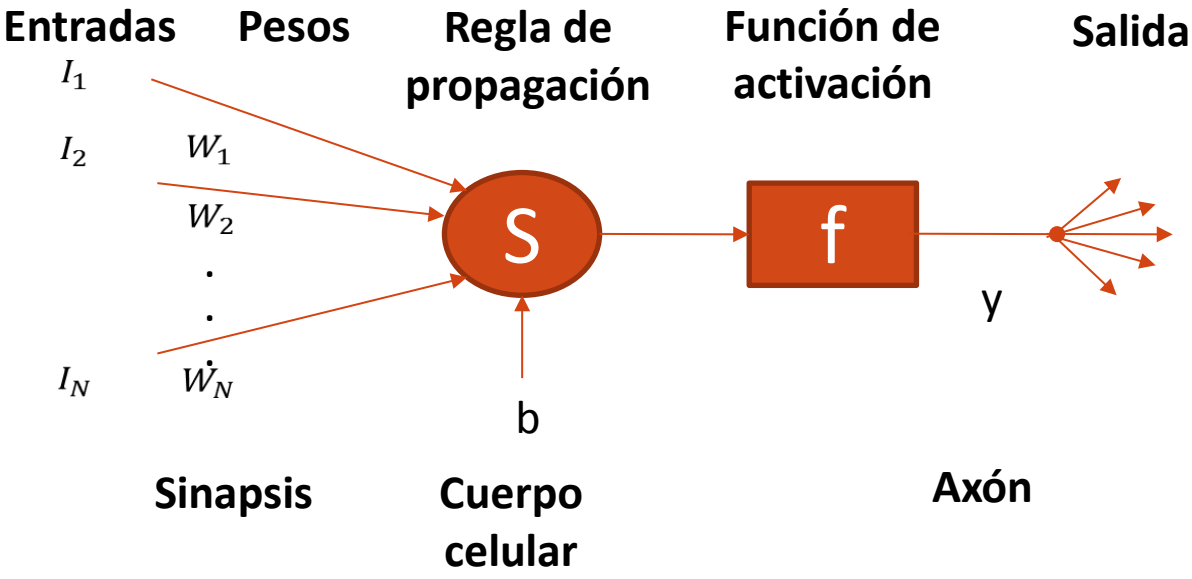


REDES NEURONALES ARTIFICIALES

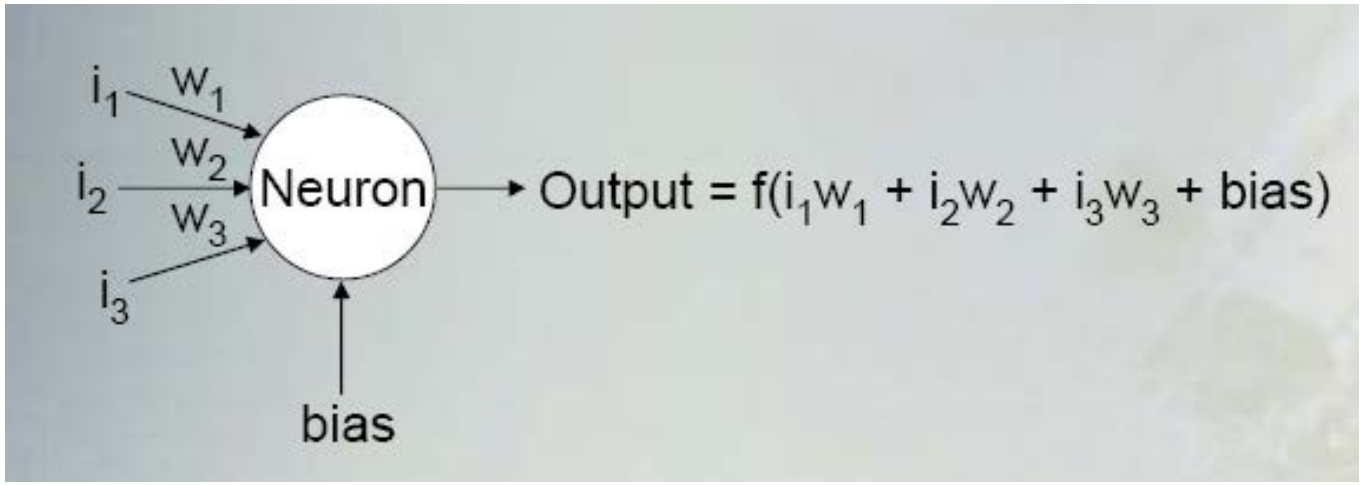
- **Un modelo de neurona real:**



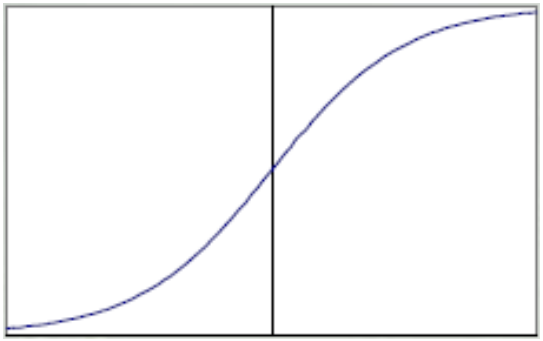
- **Un modelo de neurona artificial:**



REDES NEURONALES ARTIFICIALES

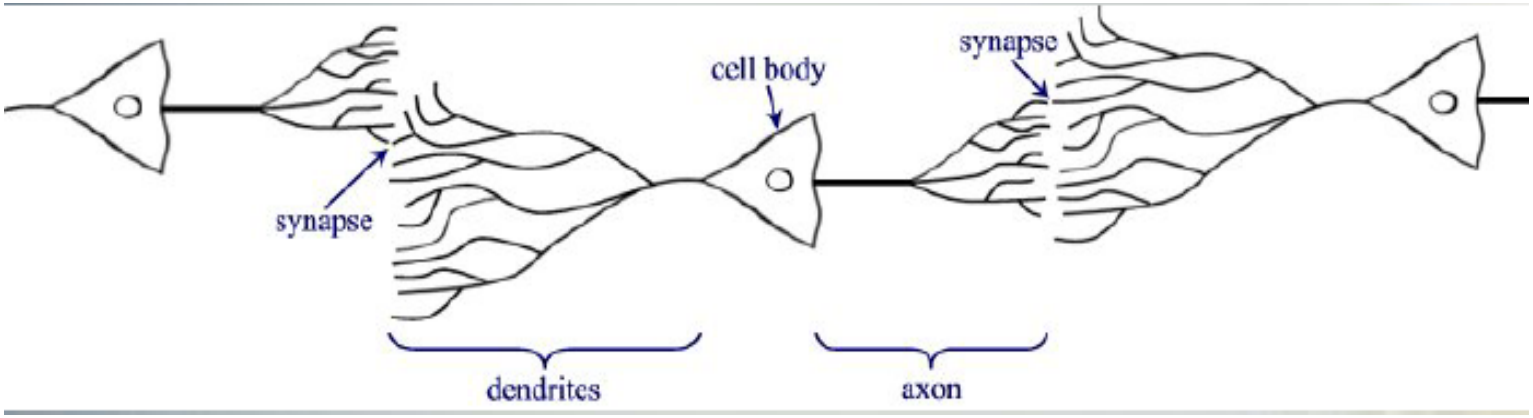


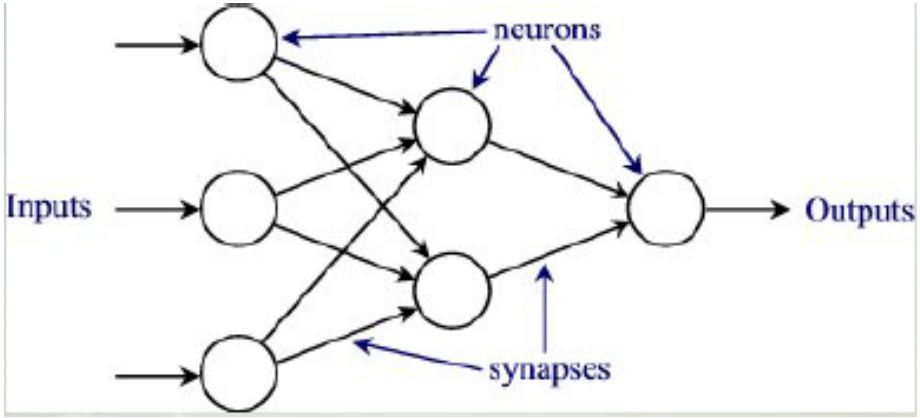
La función f(x) (función de activación) suele ser la función logística o similar:



Redes Neuronales Artificiales

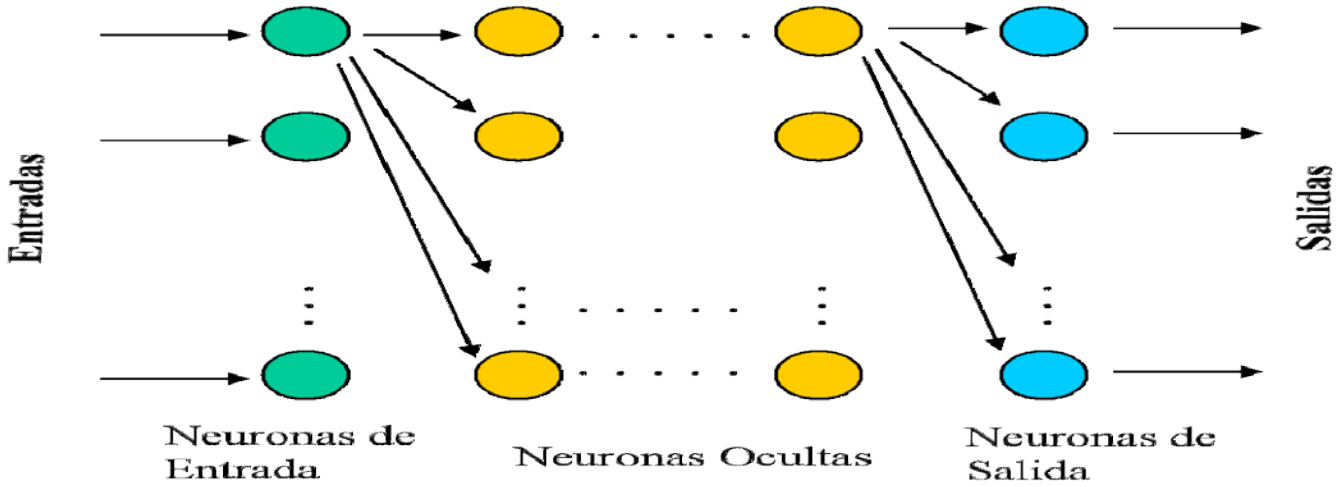
- Una NN está formada por asociaciones de neuronas, imitando las asociaciones que se dan en el cerebro.





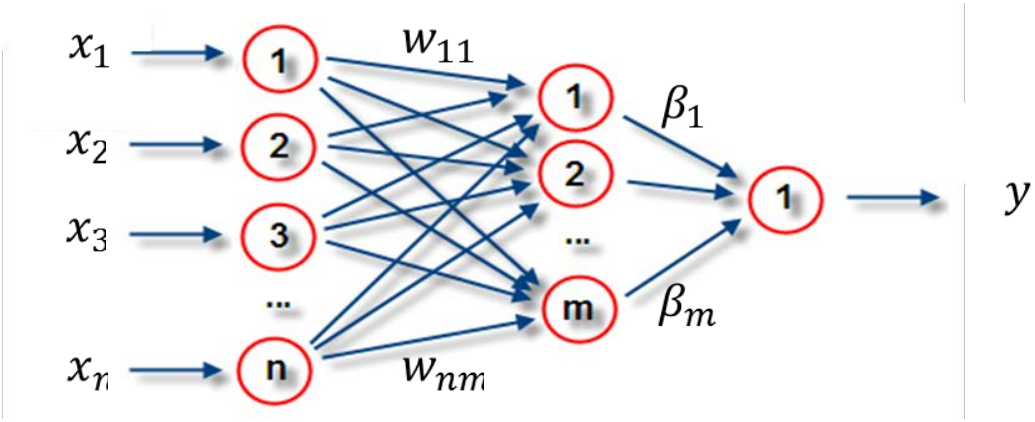
Clasificación Redes Neuronales.

- Redes Monocapa: Compuesta por una única capa de neuronas.
- Redes Multicapa: Son aquellas cuyas neuronas se organizan en varias capas.
- Redes Unidireccionales (feedforward): La información circula en un único sentido desde las neuronas de entrada a las de salida.
- Redes recurrentes o realimentadas: la información puede circular entre las capas en cualquier sentido.



Perceptrón Multicapa

- Formado por varias capas de unidades computacionales interconectadas.
- Normalmente se suele utilizar una sola capa oculta de 'm' neuronas.



- Pesos de entrada w_{ij}, Pesos de salida β_j
- Función de salida

$$y = \sum_{j=1}^{m} \beta_j f(w_j \cdot x + b_j)$$

Redes Neuronales Artificiales

- **¿Como se entrena un perceptrón multicapa?**.
  - Dado un conjunto de datos iniciales, parte de estos datos forman el llamado conjunto de entrenamiento.
  - El conjunto de entrenamiento será usado para ajustar los pesos de la red de forma óptima, para ajustarse lo mejor posible a los datos, teniendo en cuenta la generalización de las soluciones.
  - Existen diversos algoritmos de entrenamiento para perceptrones. El más popular es el back-propagation, que tiene en cuenta un error de tipo cuadrático entre la salida del perceptrón y la salida correcta.
  - Existen otros tipos de algoritmos como el Levenberg-Marquart para realizar este proceso.
  - Parte de los datos se reservan para obtener un conjunto de test, en que probar la bondad de la máquina obtenida.
  - A veces se usan conjuntos de validación para parar el entrenamiento sin overfitting.

Regla de aprendizaje

- La mas utilizada es la regla Widrow-Hoff también conocida como regla LMS (Least Mean Squares).
- Esta reglas consiste en utilizar una función de error que mida el rendimiento de la red.

$$E[w_i] = \frac{1}{2}\sum_{j=1}^{p}\left(t_{ji} - y_{ji}\right)^2$$

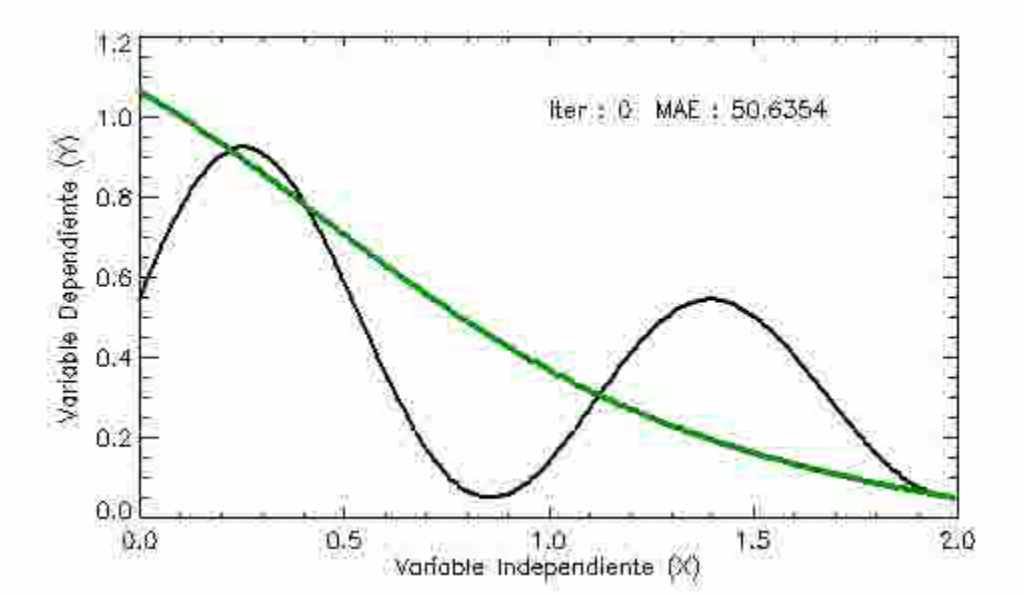
- Una vez elegida esta función, se produce un proceso de optimización utilizando el descenso por gradiente.

$$w_i(t+1) = w_i(t) - \varepsilon \nabla E(w_i) \text{ donde } \varepsilon \text{ es la tasa de aprendizaje}$$

- Este proceso de optimización se realiza de forma iterativa acercándose de forma asintótica a la solución, pues el tamaño de los incrementes cada vez es menor.

Redes Neuronales Artificiales

- **Ejemplo entrenamiento perceptrón en problema de regresión:**



Vamos a ver el video

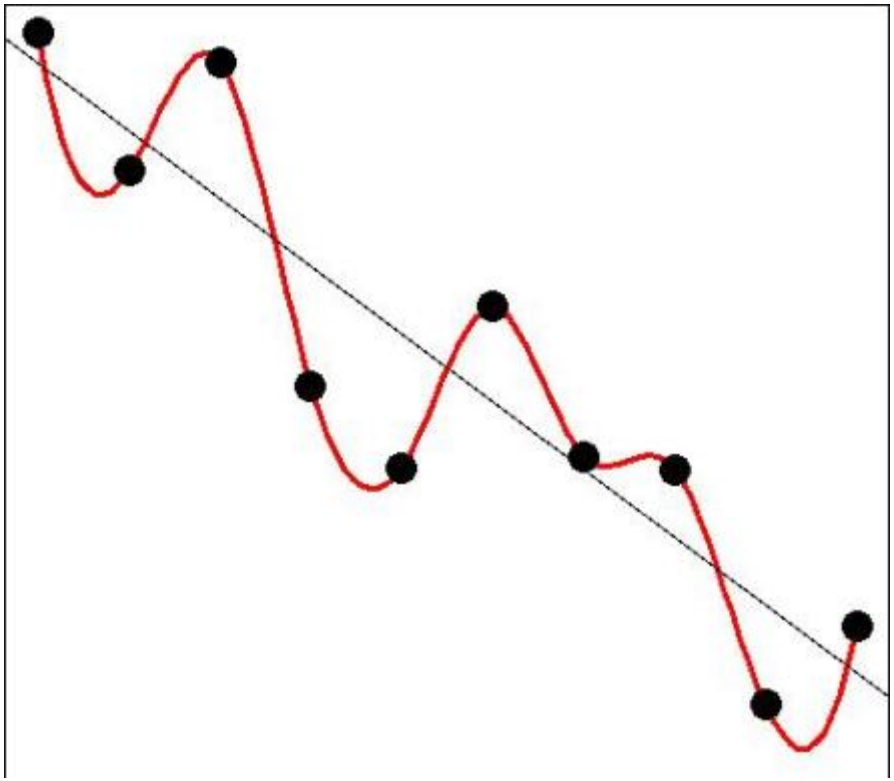
Redes Neuronales Artificiales

- **Ejemplo entrenamiento perceptrón en problema de clasificación:**

Vamos a ver el video

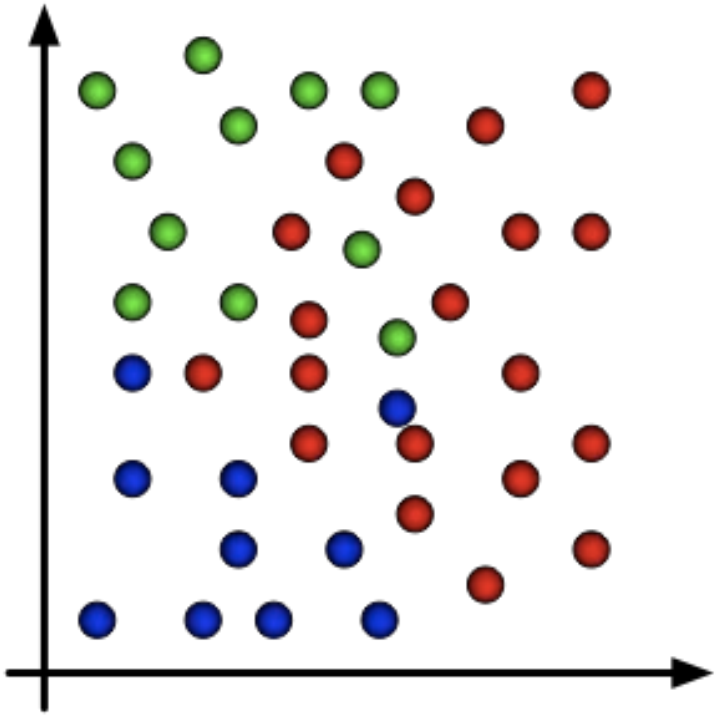
- **El problema del overfitting (regresión)**.

- Suponed un modelo generado a partir de una recta (lineal), con cierta aleatoriedad (línea negra). Si sobreentrenamos un modelo de regresión puede ocurrir el efecto de la imagen (línea roja).



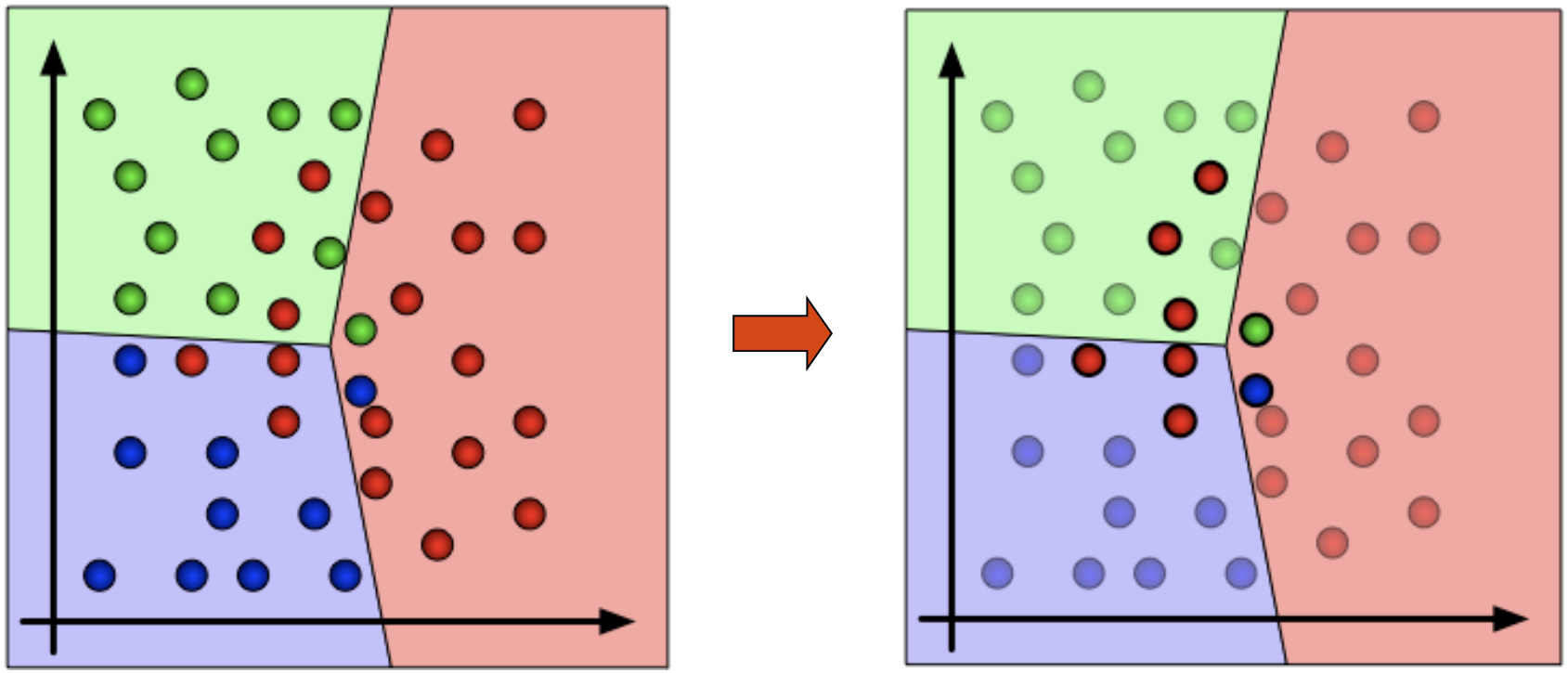
Redes Neuronales Artificiales

- **El problema del overfitting (clasificación)**. Problema de clasificación supervisada: Buscar la frontera óptima de clasificación, para cuando aparezca otra muestra, clasificarla.



Redes Neuronales Artificiales

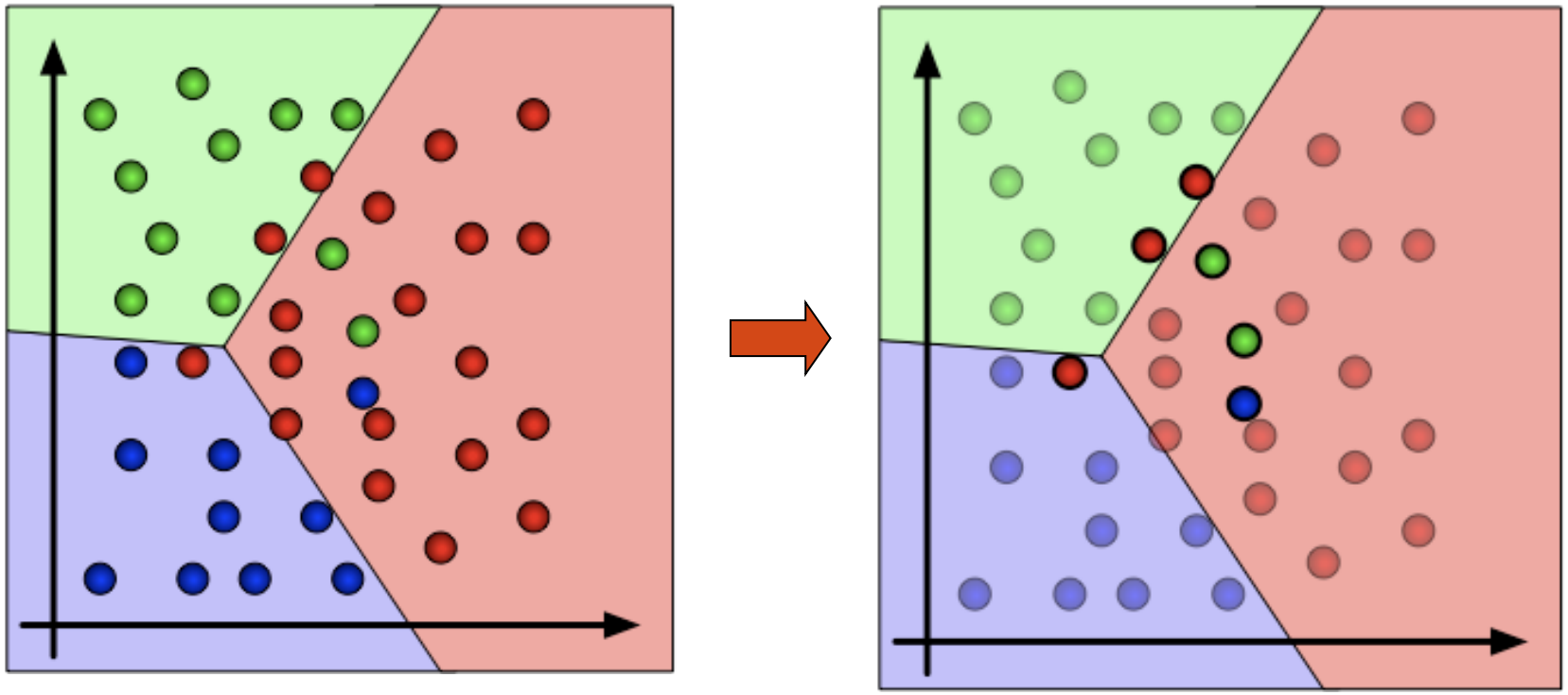
- **Primera frontera de decisión**



Ocho Errores!!

Redes Neuronales Artificiales

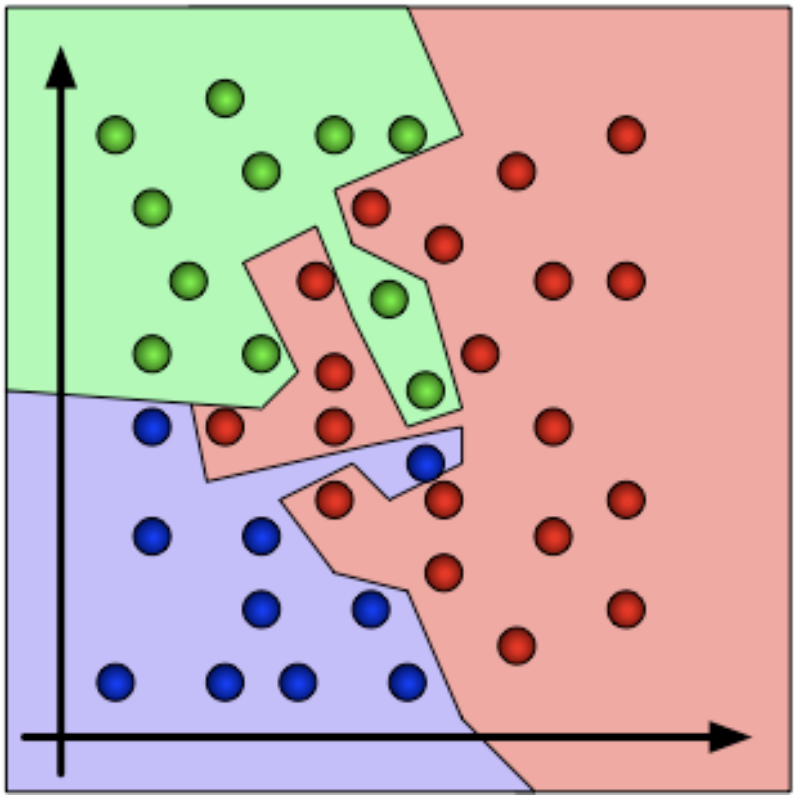
- **Nueva frontera de decisión**.



Seis Errores!!

Redes Neuronales Artificiales

- **Tercera frontera de decisión**.



Cero Errores!!

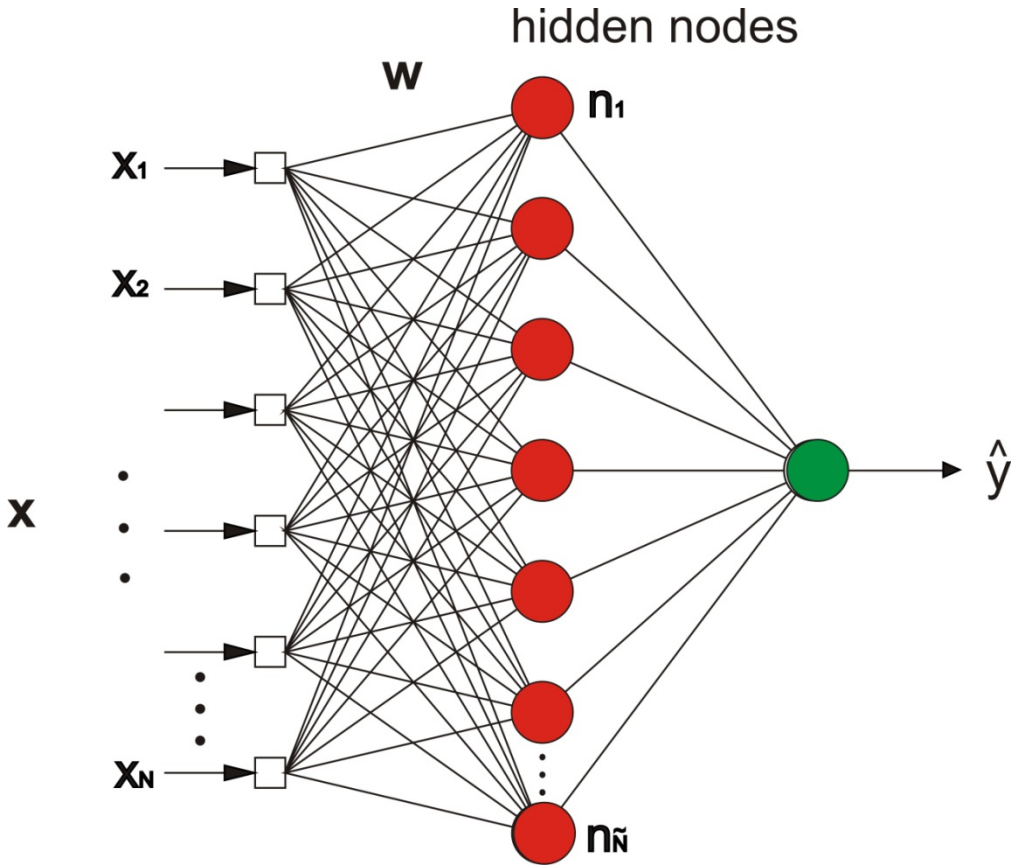
Redes Neuronales Artificiales

Vamos a ver el video

Redes Neuronales Artificiales

- **Implementación de una red neuronal: Antes de que cunda el pánico!!!**
  - Paquetes de simulación matemática como Matlab o Phyton tienen toolboxes con las redes y entrenamientos ya implementados.
  - El toolbox de Matlab por ejemplo es realmente muy sencillo de usar y además de perceptrones tiene implementados otros tipos de redes neuronales como RBFs y mapas autoorganizados.
  - Para la gran mayoría de las aplicaciones reales, los perceptrones implementados en Matlab dan resultados excelentes. También se usan muchas veces en investigación.
  - Ver **http://www.mathworks.es/products/neuralnet/**

Extreme Learning Machines (ELM)



Extreme Learning Machine (ELM)

- Para un conjunto arbritrario de distintas muestras (x_i, t_i), donde $x_i = [x_{i1}, \ldots\ldots, x_{in}]$ y $t_i = [t_{i1}, \ldots\ldots, t_{im}]$, con K nodos ocultos y una función de activación f(x).

- La salida de un sistema multicapa se modela como:

$$\sum_{i=1}^{K} \beta_i f_i(x_j) = \sum_{i=1}^{K} \beta_i f(w_i \cdot x_j + b_i) = o_j \qquad j = 1, \ldots\ldots, N,$$

- Considerando que este MLP puede aproximar esas N muestras con un error cero, lo que significa:

$$\sum_{i=1}^{K} \beta_i f(w_i \cdot x_j + b_i) = t_j$$

- Esta serie de ecuaciones puede escribirse de la siguiente forma:

$$H\beta = T$$

REDES NEURONALES ARTIFICIALES

$$H = \begin{bmatrix} g(w_1 \cdot x_1 + b_1) & \cdots & g(w_1 \cdot x_1 + b_K) \\ \vdots & \ddots & \vdots \\ g(w_1 \cdot x_N + b_1) & \cdots & g(w_{\tilde{N}} \cdot x_N + b_K) \end{bmatrix} \quad \beta = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_K \end{bmatrix} \quad T = \begin{bmatrix} t_1 \\ \vdots \\ t_N \end{bmatrix}$$

- La afirmación anterior ($||H\beta - T|| = 0$) es correcta cuando N=K.
- Pero normalmente el numero de nodos de la capa oculta será mucho menor que el número de muestras. En ese caso lo que se intenta es $\min||H\beta - T||$.
- En el caso de $K \leq N$, la matriz H no es cuadrada y no existe solución tal que $H\beta = T$.
- En este caso los pesos óptimos de salida se obtendrán como solución de un problema de mínimos cuadrados.

$$\hat{\beta} = H^{\dagger}T$$

Donde $H^{\dagger}$ es la pseudo inversa de Moore-Penrose

Algoritmo ELM

- Dado un conjunto de entrenamiento $\aleph = \{(x_i, t_i) | x_i \in R^n, t_i \in R^m, i = 1, \ldots, N\}$
- Función de activación g(x).
- Con un número de nodos en la capa oculta de Ñ.

I. Asignamos de forma aleatoria los pesos w_i y bias b_i, $i = 1, \ldots, Ñ$.

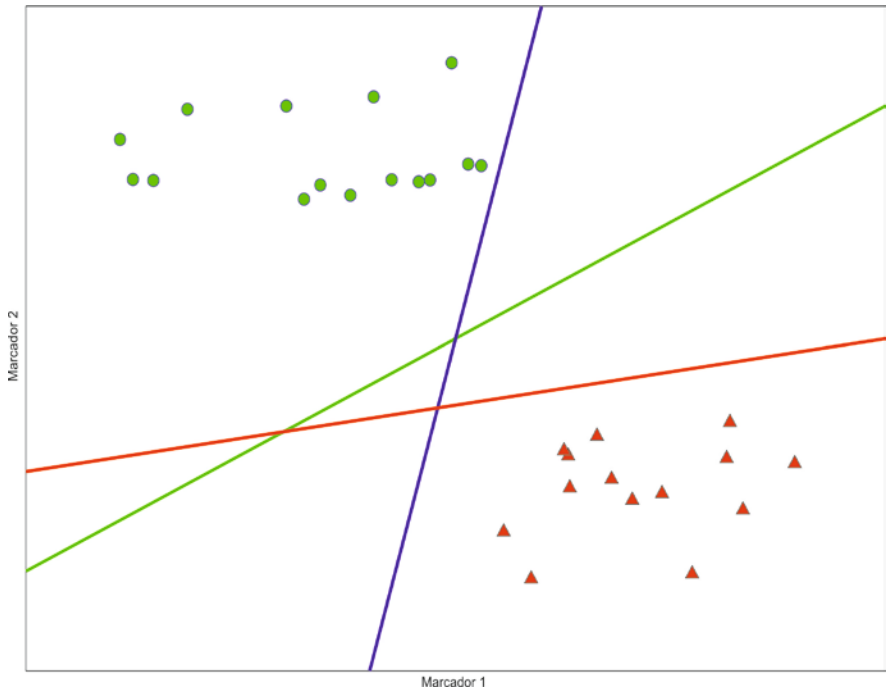
II. Calculamos la matriz de la capa oculta H.

III. Calculamos la pseudo-inversa de Moore-Penrose de dicha matriz.

IV. Calculamos los pesos de salida β.

$$\beta = H^{\dagger}T$$

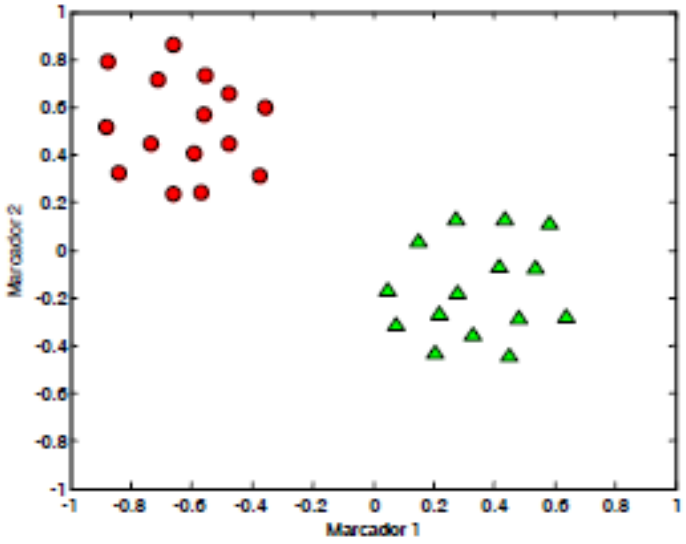
MÁQUINAS DE VECTORES SOPORTE



MÁQUINAS DE VECTORES SOPORTE

Idea Básica

Supongamos un problema de clasificación de tipos de fruta, en el que se intenta determinar a partir de unos atributos (A1, A2) si es una cereza (puntos rojos) o una uva (verdes) .



Separación de sujetos mediante una plano:

- Π: <w,x> + b =0
- W es el vector normal.
- b es el desplazamiento.

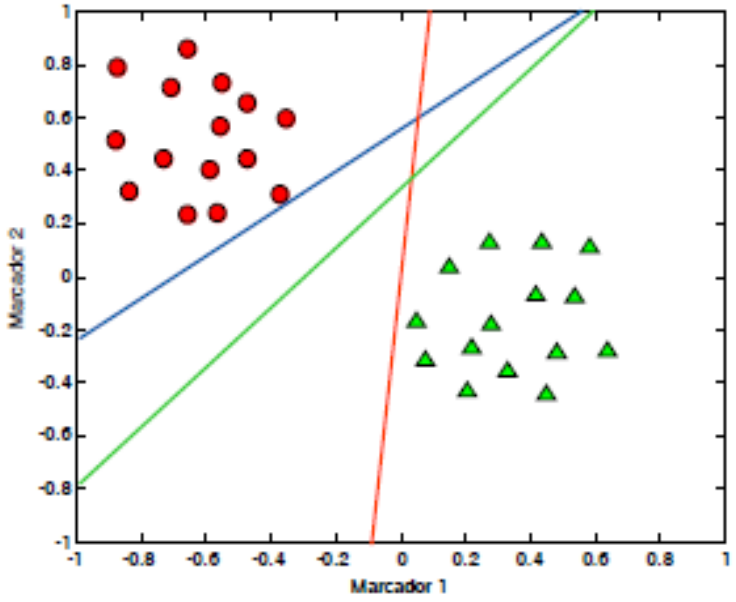
Función de decisión:

$f(x)=sign(<w,x>+b)=sign(\sum_{i=1}^{d} w_i x_i+b)$

Máquinas de Vectores Soporte

¿Qué plano elegir?

- Existen infinitos planos que separan perfectamente el conjunto de muestras.
- Tenemos que elegir aquel plano cuya distancia con respecto a las muestras de cada una de las clases sea máxima (en este caso el plano verde).



- Esto permite distinguir de forma mas clara las regiones de cada una de las clases.
- El objetivo es :

$$\text{Minimizar } \tfrac{1}{2}\|w\|^2$$

$$\text{s.a} \quad y_i(\langle w, x_i \rangle + b) \geq 1$$

MÁQUINAS DE VECTORES SOPORTE

Solución Problema Optimización

Para resolver el problema de optimización anterior utilizamos multiplicadores de Lagrange:

$$Lp(w, b, \alpha_i) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha_i (y_i(\langle x_i, w\rangle + b) - 1)$$

$$\text{s.a} \quad y_i(\langle w, x_i\rangle + \text{b} \geq 1$$
$$\alpha_i \geq 0$$

Condiciones Kurash-kuhn-Tucker:

$$\alpha_i (y_i(\langle x_i, w\rangle + b) - 1) = 0$$

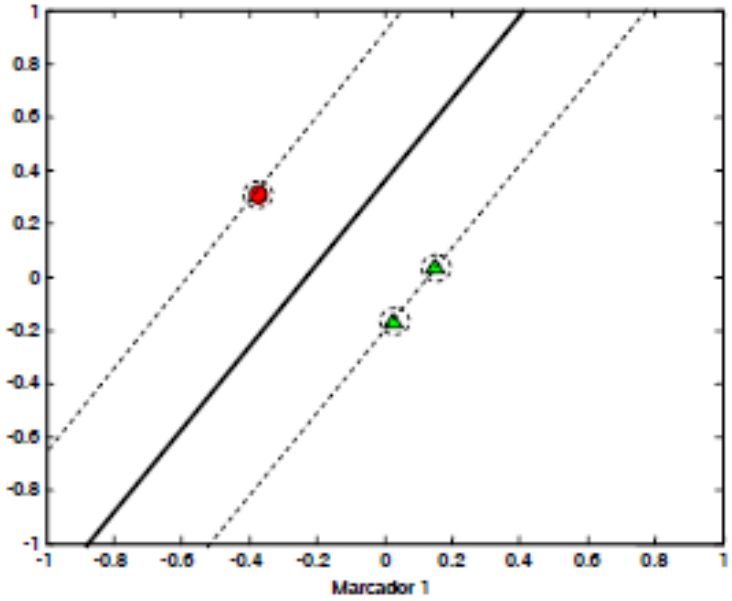
Resolviendo el problema obtenemos:

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i \qquad \sum_{i=1}^{n} \alpha_i y_i = 0 \qquad f(x) = \sum_{i=1}^{n} \alpha_i y_i \langle x_i, x\rangle + b$$

MÁQUINAS DE VECTORES SOPORTE

Vectores Soporte

- Al maximizar la distancia de separación, solo se toman en consideración las muestras que se encuentran en los límites.
- Los patrones que determinan el plano son denominados **'Vectores Soporte'.**

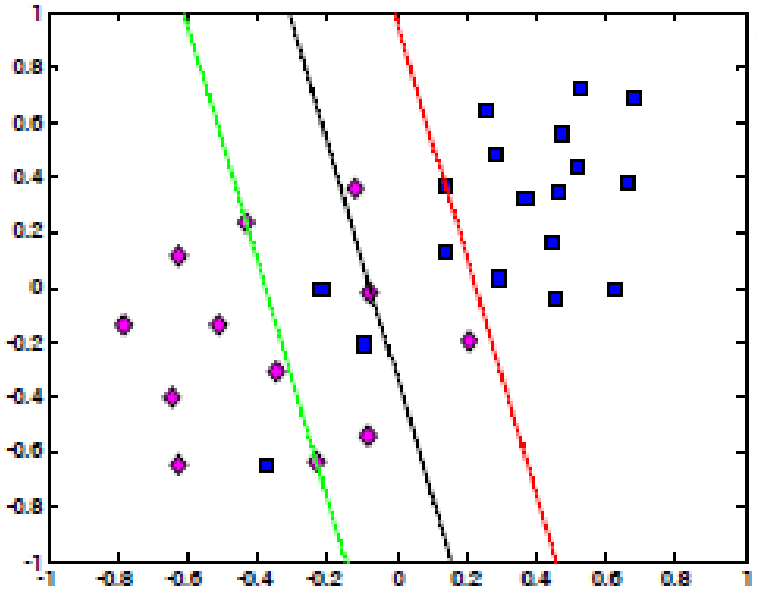


- Estos Vectores Soporte son aquellos vectores del conjunto de entrenamiento que proporcionan un multiplicador $\alpha_i > 0$.
- Estos vectores soporte son los elementos críticos, ya que ellos son los que proporcionan las solución del problema. Si quitáramos el resto de elementos y realizáramos de nuevo el entrenamiento el resultado sería el mismo, es decir, los vectores soporte son los únicos que aportan información.

MÁQUINAS DE VECTORES SOPORTE

Variables de Pérdidas

- No siempre es posible separar por completo dos clases de muestras mediante un plano.
- Introducir variable de error ξ.

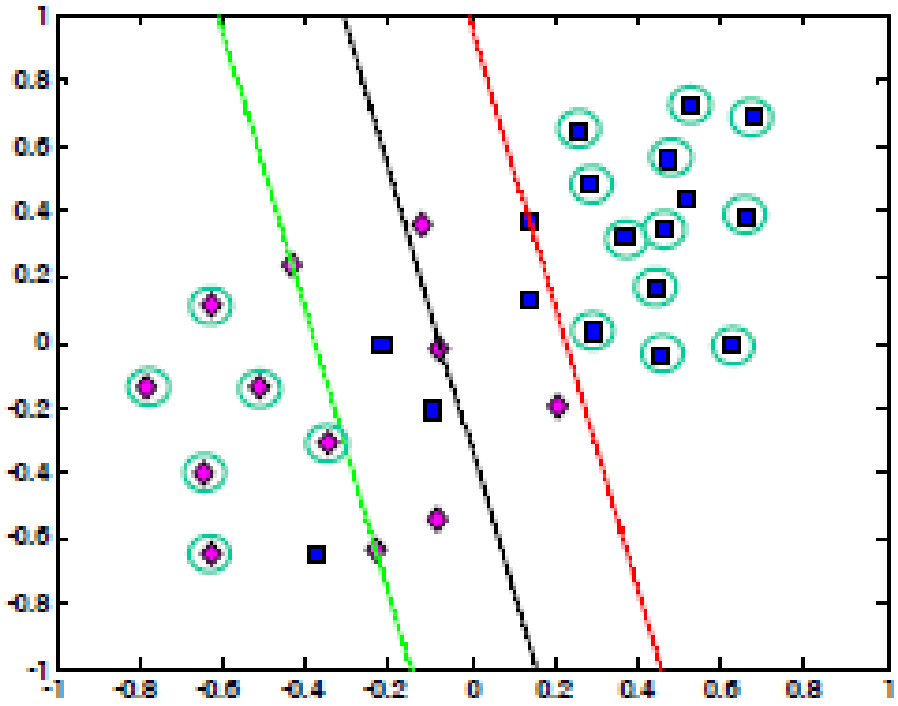


- Puesto que en el caso no separable se van a cometer errores, sería lógico asignar un coste a estos errores. El nuevo problema a optimizar sería:

$$\min \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i$$

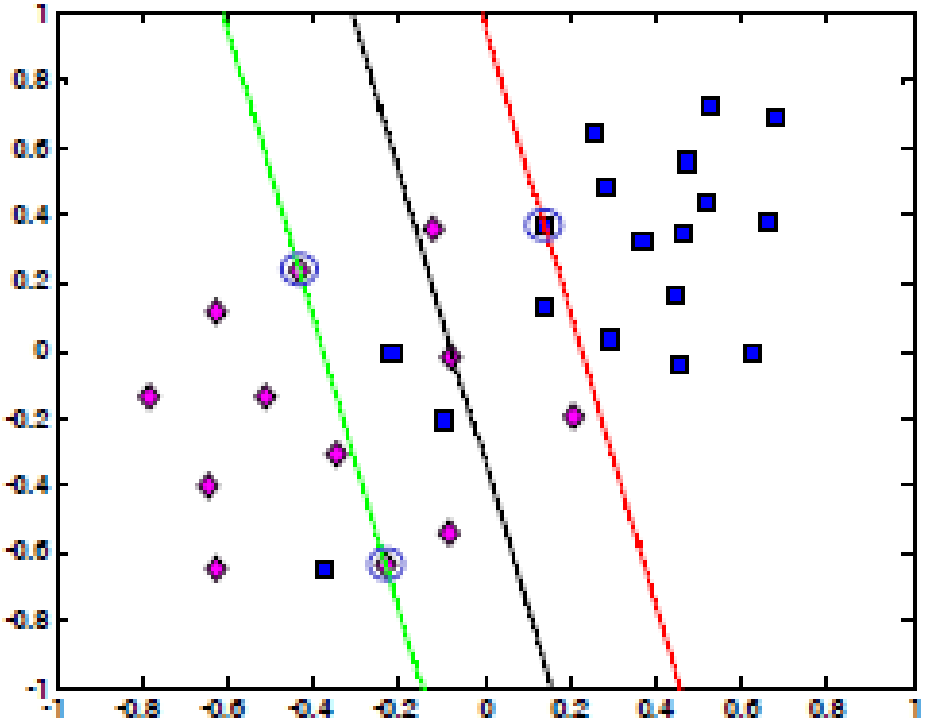
$$s.a\begin{cases} y_i(\langle x_i, w\rangle + b) - 1 + \xi_i \geq 0, & \forall i \\ \xi_i \geq 0, & \forall i \end{cases}$$

MÁQUINAS DE VECTORES SOPORTE



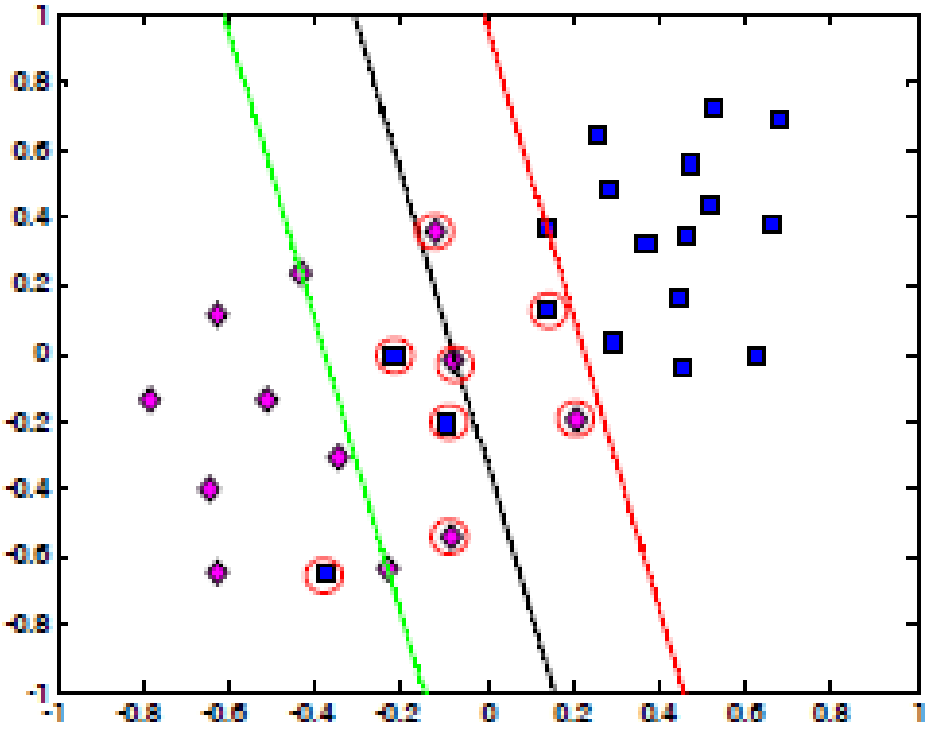
$\alpha_i = 0$ No son vectores soporte

MÁQUINAS DE VECTORES SOPORTE



$0 < \alpha_i < C$ Vectores soporte

MÁQUINAS DE VECTORES SOPORTE

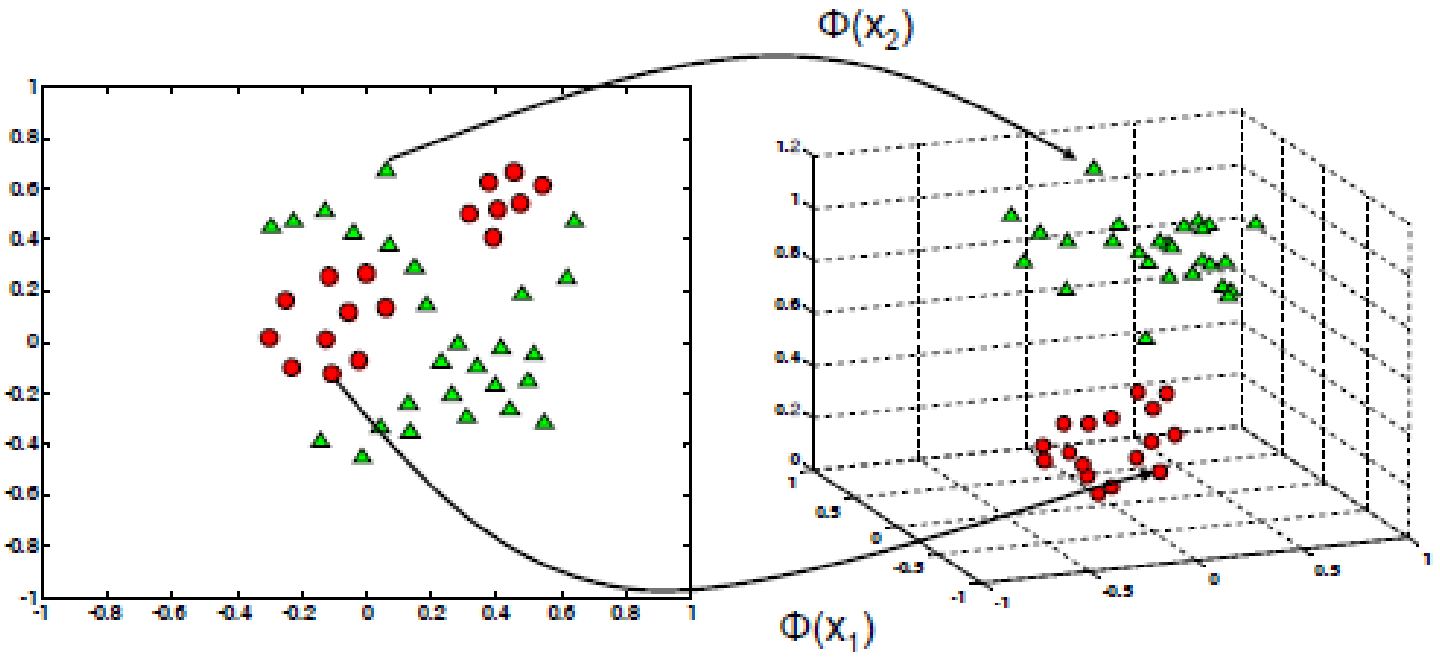


$\alpha_i = C$ Vectores soporte tipo 2

MÁQUINAS DE VECTORES SOPORTE

Problemas No Lineales

- La mayoría de los problemas que existen no son lineales.
- Con la teoría anterior la tasa de error obtenida sería muy elevada.
- ***Solución***: Transformamos los datos a un espacio de características superior donde estos sean linealmente separables. Pudiendo trabajar de la misma forma que lo hemos hecho hasta ahora.



MÁQUINAS DE VECTORES SOPORTE

Problema No Lineal

- Ahora, en lugar de considerar el conjunto de vectores $\{x_1, \ldots\ldots, x_n\}$ se consideran los vectores del espacio transformado $\{\Phi(x_1), \ldots\ldots, \Phi(x_n)\}$.

- La solución del problema queda definida a través del producto escalar de estos nuevos vectores.

$$\mathrm{K}(x_i, x) = \langle \Phi(x_i), \Phi(x) \rangle$$

- Este producto escalar es realizado mediante una función Kernel. Esto permite obtener una SVM que consume la misma cantidad de recursos que una lineal.

- Kernel mas utilizado:

Gaussiano: $K(x_i, x) = \langle \Phi(x_i), \Phi(x) \rangle = e^{-\gamma ||x_i - x||^2}$

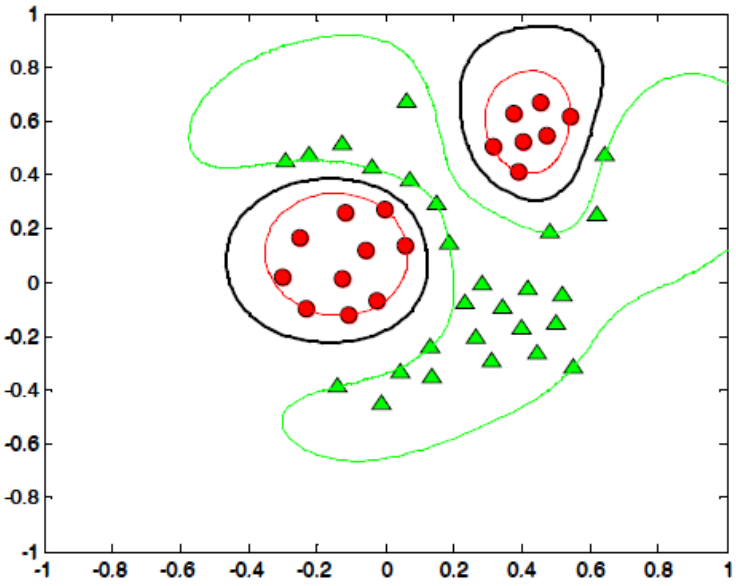
- Función de decisión queda expresada como:

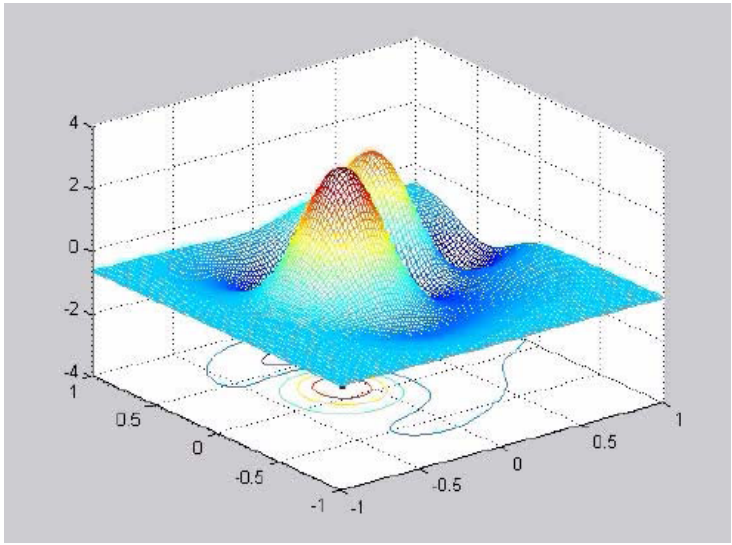
$$f(x) = \sum_{i=1}^{Nsv} \alpha_i y_i k(x_i, x) + b$$

MÁQUINAS DE VECTORES SOPORTE

Problema No Lineal

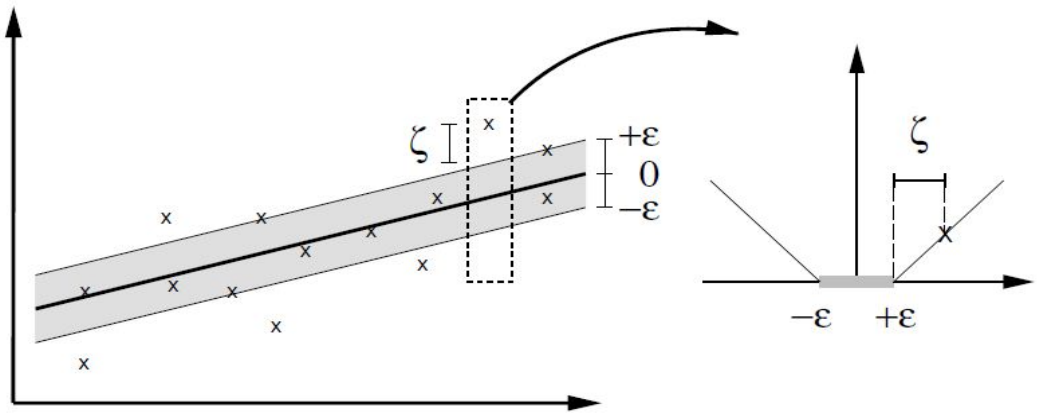
Ejemplo de SVM con Kernel Gaussiano:





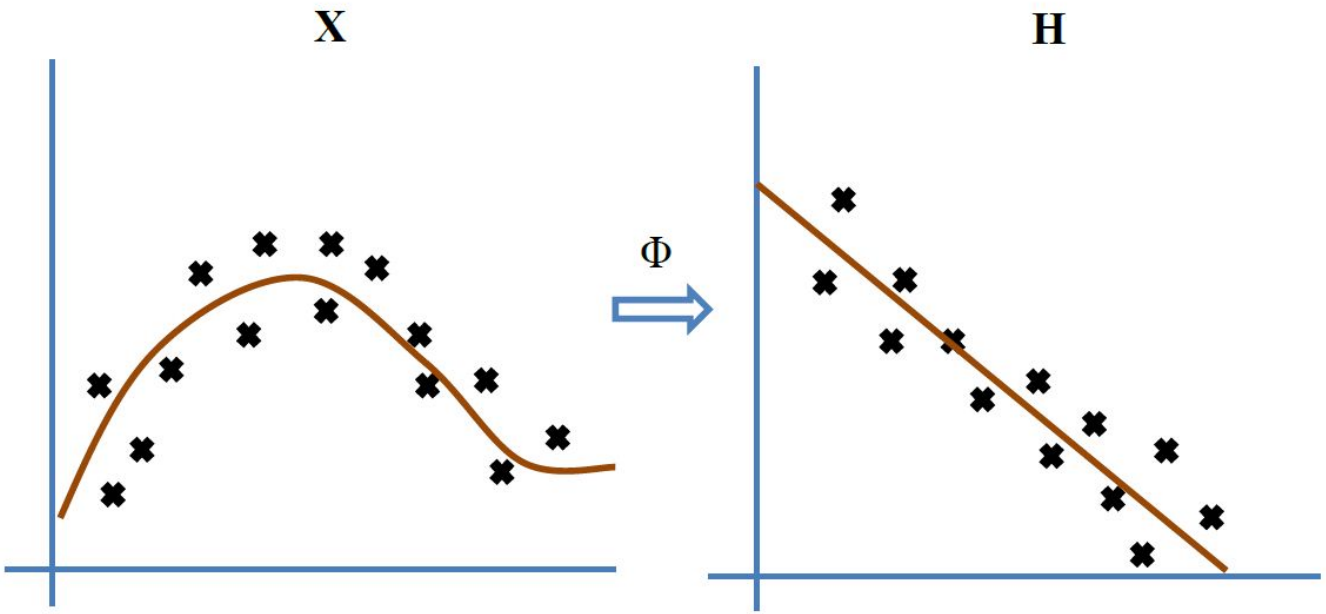
MÁQUINAS DE VECTORES DE SOPORTE

- Máquinas de vectores de soporte para regresión:
  - Minimizar:
    - $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\xi_i + \xi_i^*$
  - Sujeto a:
    - $y_i - w^T\phi(x_i) - b \leq \varepsilon + \xi_i$
    - $-y_i + w^T\phi(x_i) + b \leq \varepsilon + \xi_i^*$
    - $\xi_i, \xi_i^* \geq 0$



Máquinas de Vectores de Soporte

- Problemas no lineales
- Solución: transformar a un espacio de dimensión mayor



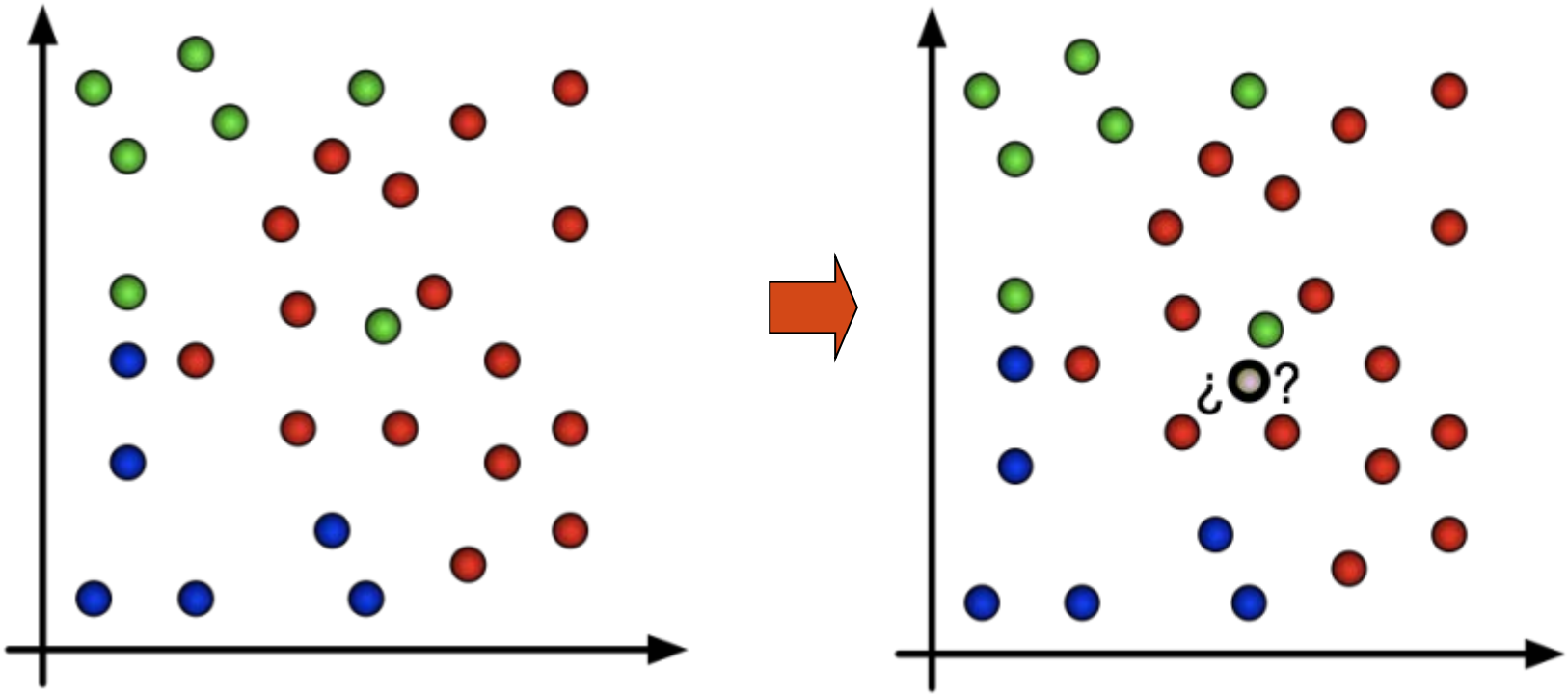
OTROS ALGORITMOS DE RECONOCIMIENTO DE PATRONES

Otros algoritmos de reconocimiento de patrones

- **El algoritmo de k vecinos más próximos (K-NN)**
- Es un algoritmo para problemas de clasificación supervisada.
- Se buscan los k patrones con menor distancia a una muestra dada.
- La selección de la clase de dicha muestra se realiza por votación mayoritaria de las clases de los vecinos.
- Número de operaciones: proporcional al tamaño del conjunto de entrenamiento.

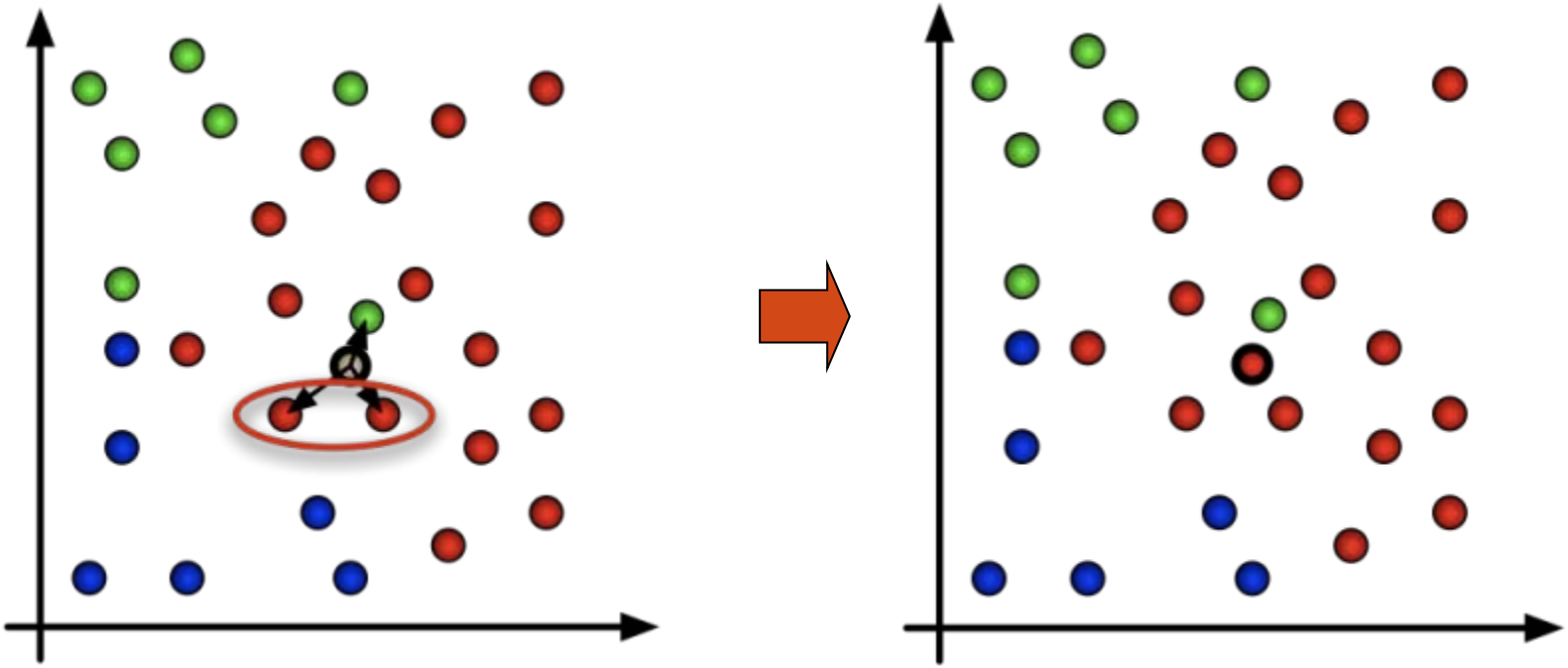
Otros algoritmos de reconocimiento de patrones

- **Ejemplo de implementación de un algoritmo de 3 vecinos más próximos (3-NN):**



Otros algoritmos de reconocimiento de patrones

- **Ejemplo de implementación de un algoritmo de 3 vecinos más próximos (3-NN):**

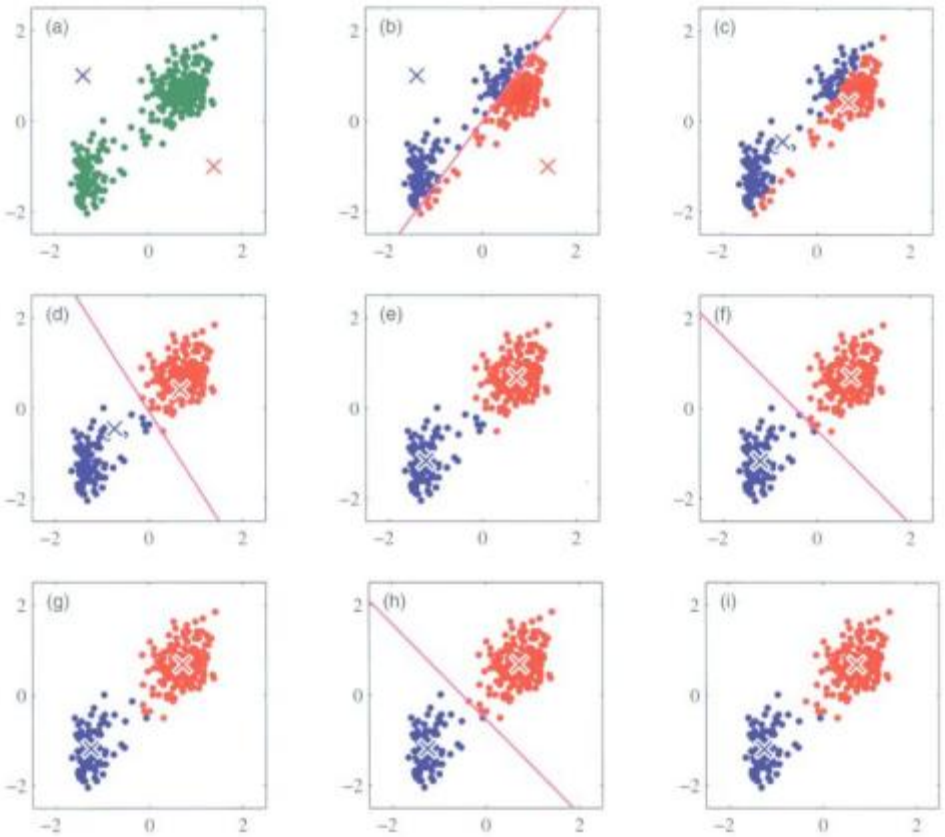


Otros algoritmos de reconocimiento de patrones

- **El algoritmo de las K-medianas (K-Means)**
  - Es un algoritmo para problemas de clasificación NO supervisada.
  - Se trata de hacer clustering a partir de un conjunto de datos iniciales.
  - Este algoritmo exige fijar de antemano el número de clusters (K).
  - Se situan los K clusters en el espacio de características aleatoriamente.
  - Se asignan los datos a su cluster más cercano.
  - Se recalculan la posición de los clusters mediante el baricentro de los datos asignados a cada cluster.
  - Se procede en bucle, hasta que la posición de los clusters no varíe.

Otros algoritmos de reconocimiento de patrones

- **Ejemplo de implementación de un algoritmo de K-Means:**



INTRODUCCIÓN A LA COMPUTACIÓN NEURONAL

Sancho Salcedo Sanz

Departamento de Teoría de la Señal y Comunicaciones

Universidad de Alcalá