Redes Neuronales (2)

Carlos Hurtado L.

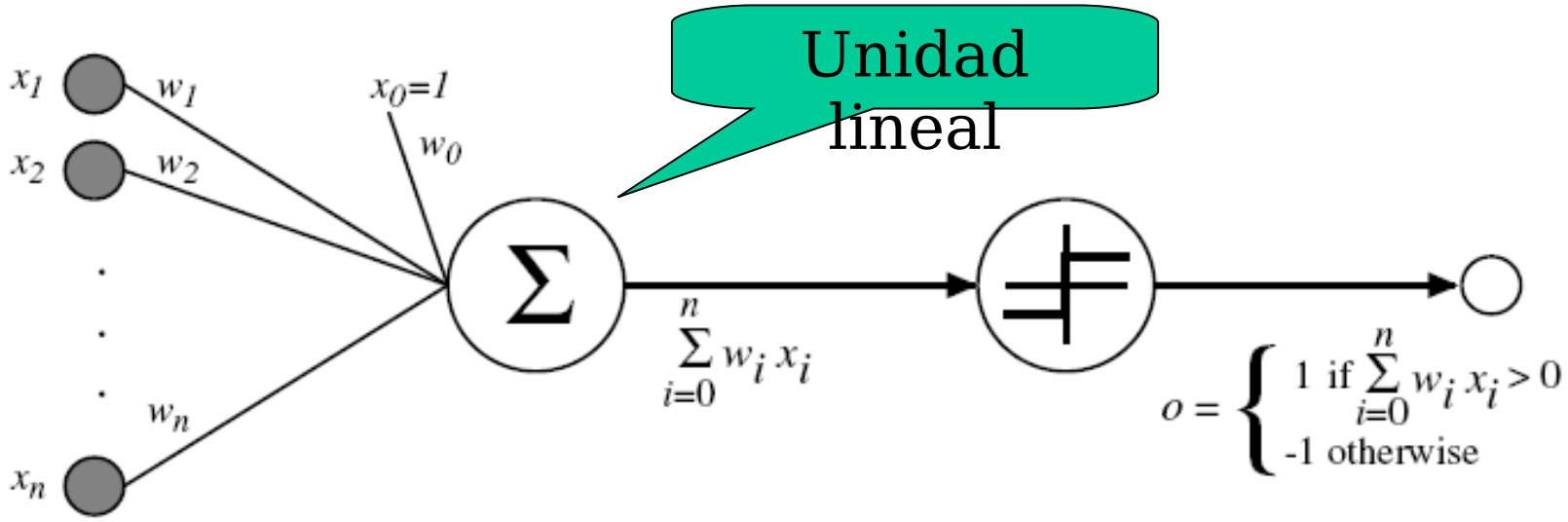
Depto. de Ciencias de la Computación, Universidad de Chile

Referencias

- Machine Learning, Tom Mitchell, Capítulo 4
- Apuntes del capítulo 4 de este libro:

  http://www.cs.cmu.edu/~tom/mlbook-chapter-slides.html

Perceptron



$$o(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 & \text{otherwise.} \end{cases}$$

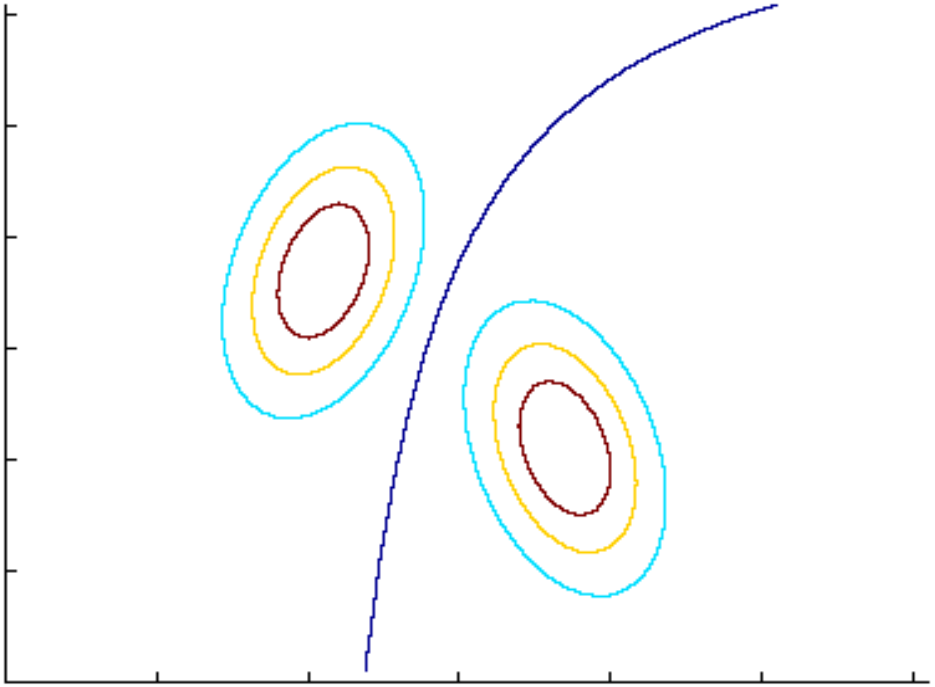
Perceptrón

- Hemos visto que es un modelo simple y efectivo para construir un clasificador binario para clases linealmente separables (o casi)
- Problema: generalizar el Perceptron a
  - Más de dos clases
  - Clases no son separables linealmente

Redes Neuronales y Funciones Discriminantes

- Para entender el fundamento de las redes neuronales necesitamos entender la noción de “funciones discriminantes”.

Funciones Discriminantes



Funciones Discriminantes

- Todo modelo de clasificación se puede ver como un conjunto de funciones discriminantes
- Para cada clase C_i, tenemos una función discriminante $g_i(\boldsymbol{x})$.
- Dado un dato nuevo x lo clasificamos en la clase C_i sii

$$g_i(x) = \max_j g_j(x)$$

Ejercicio

- Represente un árbol de decisión como una función discriminante.

Ejercicio

- Represente un árbol de decisión como una función discriminante.
- Solución:
  - Cada hoja del árbol define un región en el espacio de variables.
  - Para los puntos de cada región, las funciones discriminantes entregan las frecuencias de cada clase.

Ejemplo Funciones Discriminantes

- En un modelo Bayesiano tenemos

$$g_i(\boldsymbol{x}) = \log P(\mathrm{C}_i|\boldsymbol{x})$$

- Aplicando el teorema de Bayes una función discriminante equivalente:

$$g_i(\boldsymbol{x}) = \log P(\boldsymbol{x}|\mathrm{C}_i) + \log P(\mathrm{C}_i)$$

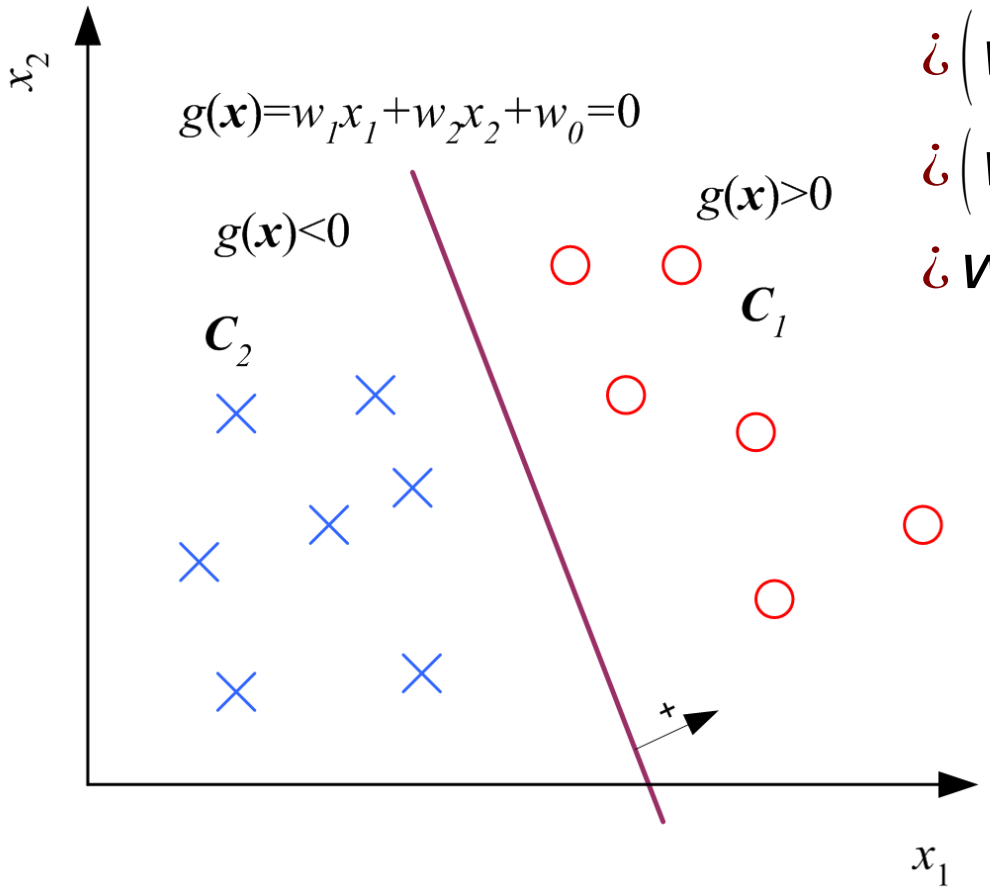
Funciones Discriminantes lineales

$$g_i(x) = w_i^T x + w_{i0} = \sum_{j=1}^{d} w_{ij} x_j + w_{i0}$$

Elegir C_i ssi

$$g_i(x) = \max_{j=1}^{K} g_j(x)$$

Perceptrón: Discriminantes Lineal para Dos Clases



$$g(x)=g_1(x)-g_2(x)$$

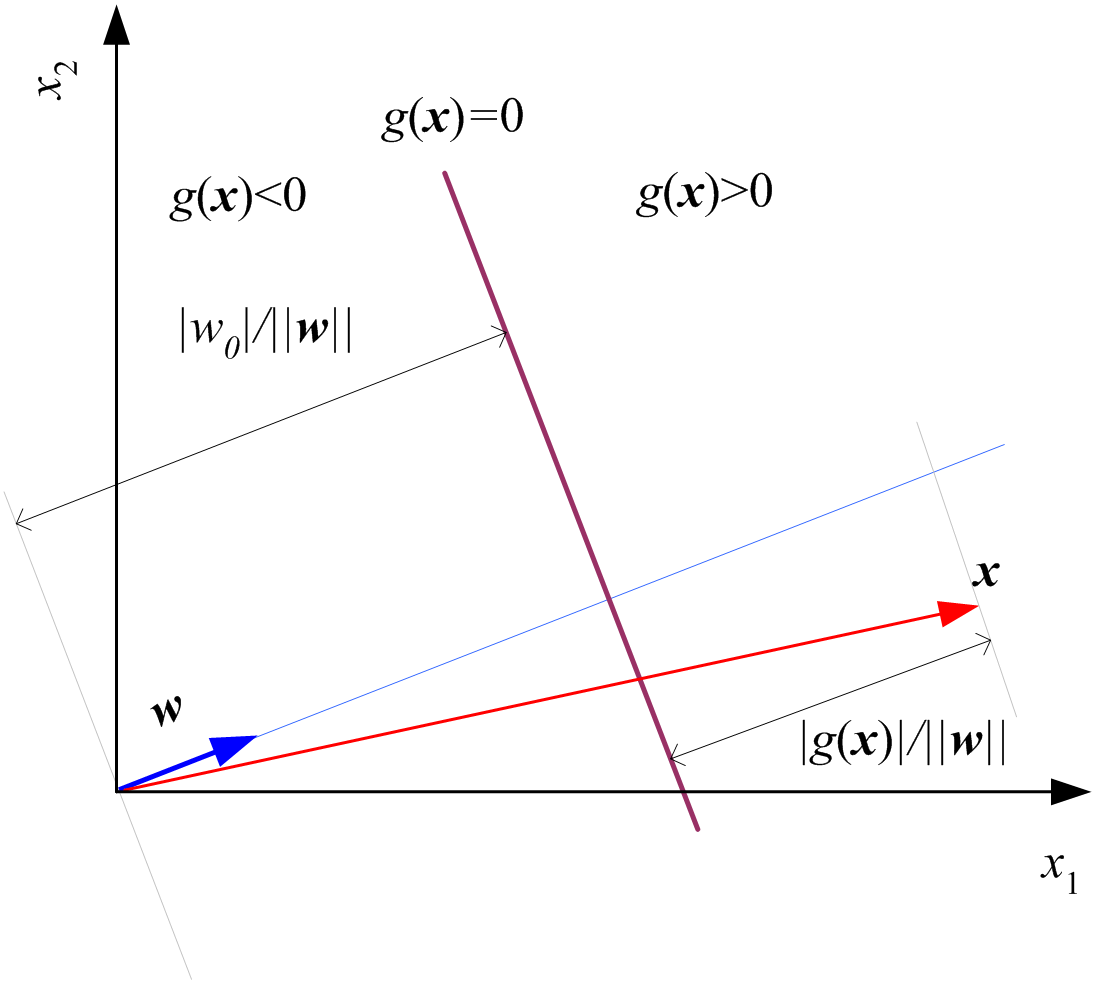
$$¿\left(w_1^T x+w_{10}\right)-\left(w_2^T x+w_{20}\right)$$

$$¿\left(w_1-w_2\right)^T x+\left(w_{10}-w_{20}\right)$$

$$¿\, w^T x+w_0$$

$$\text{choose} \begin{cases} C_1 & \text{if } g(x)>0 \\ C_2 & \text{otherwise} \end{cases}$$

Geometría


x_2
$g(\boldsymbol{x})=0$
$g(\boldsymbol{x})<0$
$g(\boldsymbol{x})>0$
$|w_0|/||\boldsymbol{w}||$
$\boldsymbol{x}$
$\boldsymbol{w}$
$|g(\boldsymbol{x})|/||\boldsymbol{w}||$
x_1

Geometría

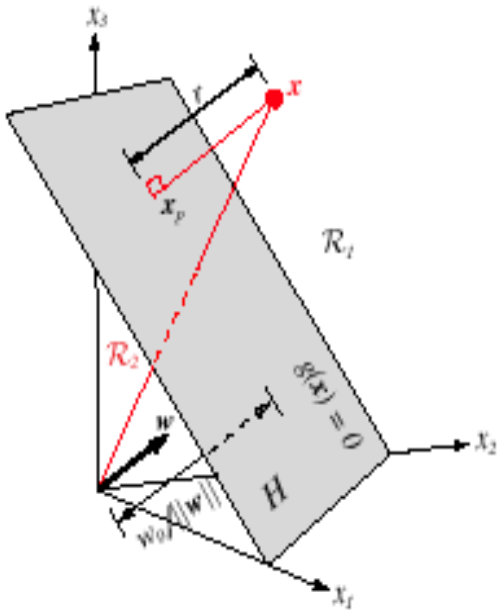
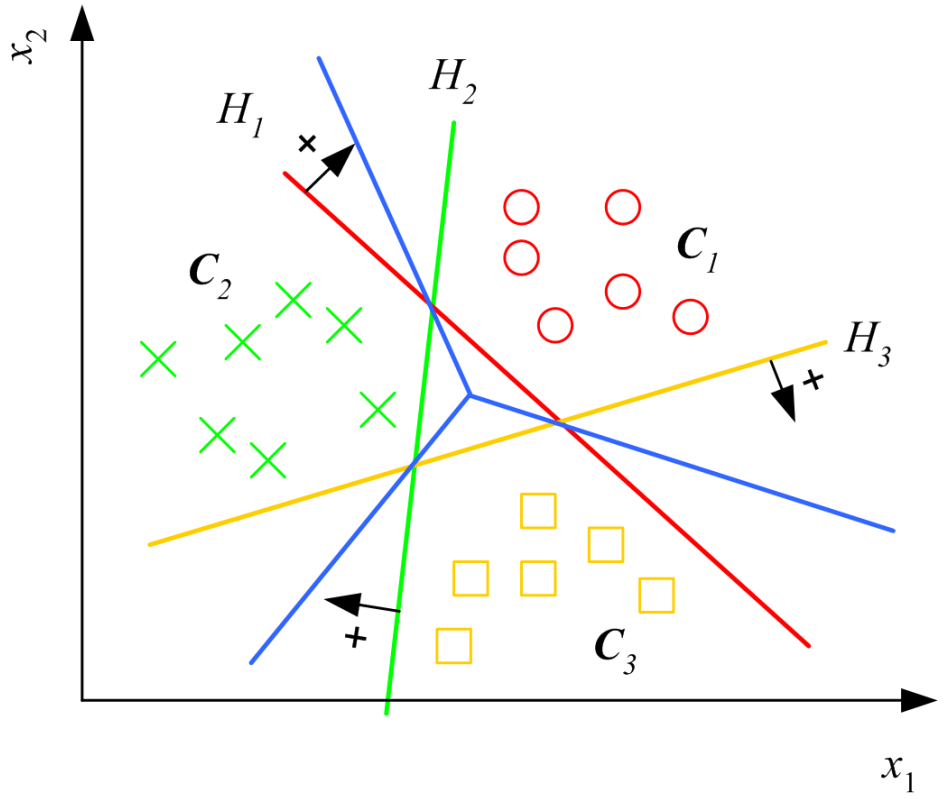


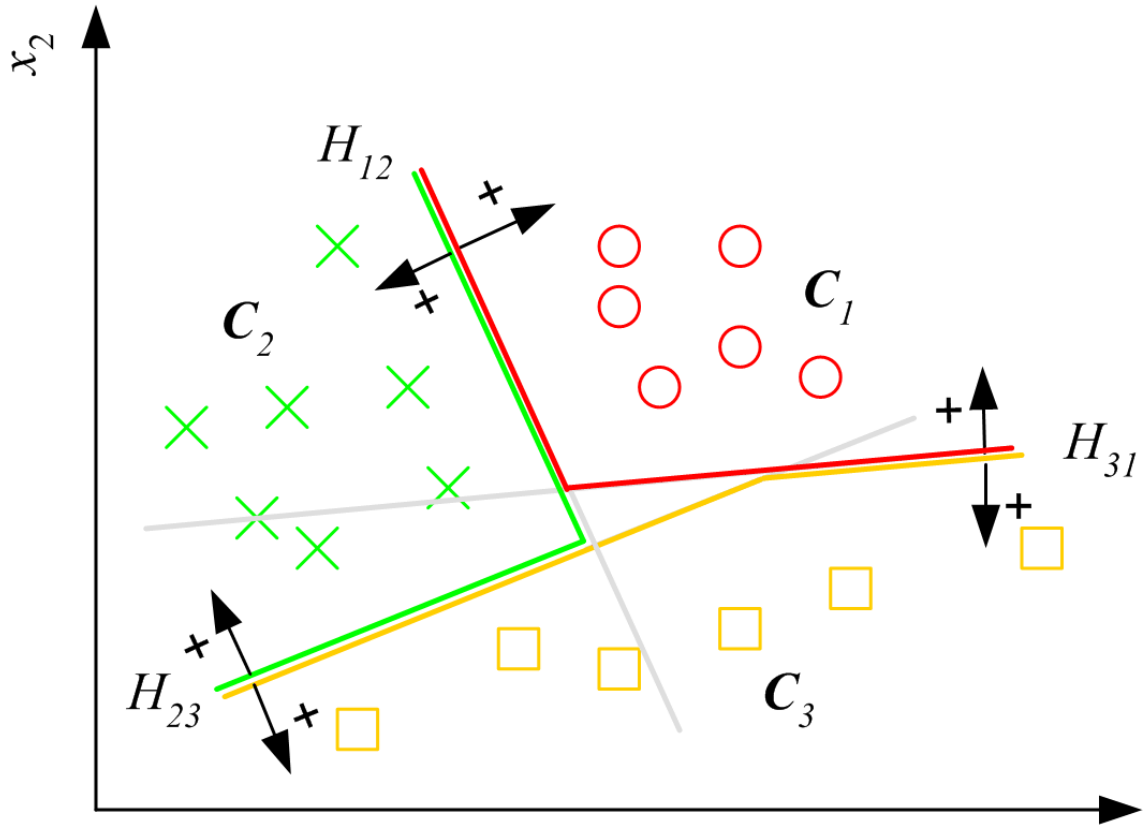
FIGURE 5.2. The linear decision boundary H, where $g(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_0 = 0$, separates the feature space into two half-spaces $\mathcal{R}_1$ (where $g(\mathbf{x}) > 0$) and $\mathcal{R}_2$ (where $g(\mathbf{x}) < 0$). From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Discriminantes Lineales para k > 1 Clases Separables Linealmente



Equivalente a entrenar k perceptrones

Discriminantes Lineales para k > 1 Clases Separables Linealmente de a Pares



Equivalente a entrenar k (k-1)/2 perceptrones

Otros ejemplos

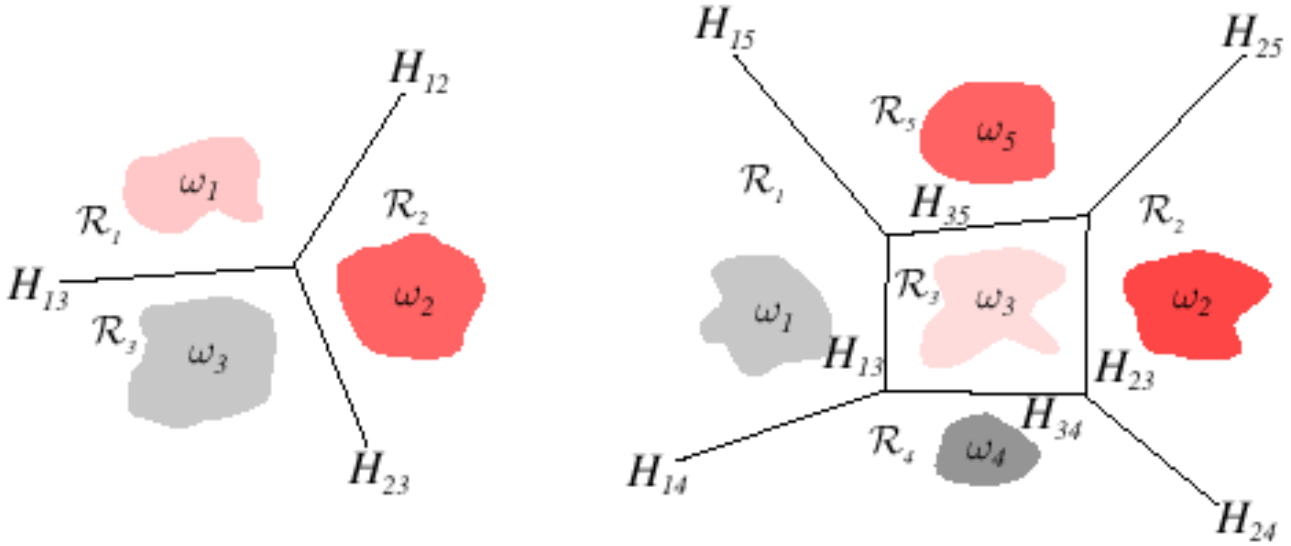
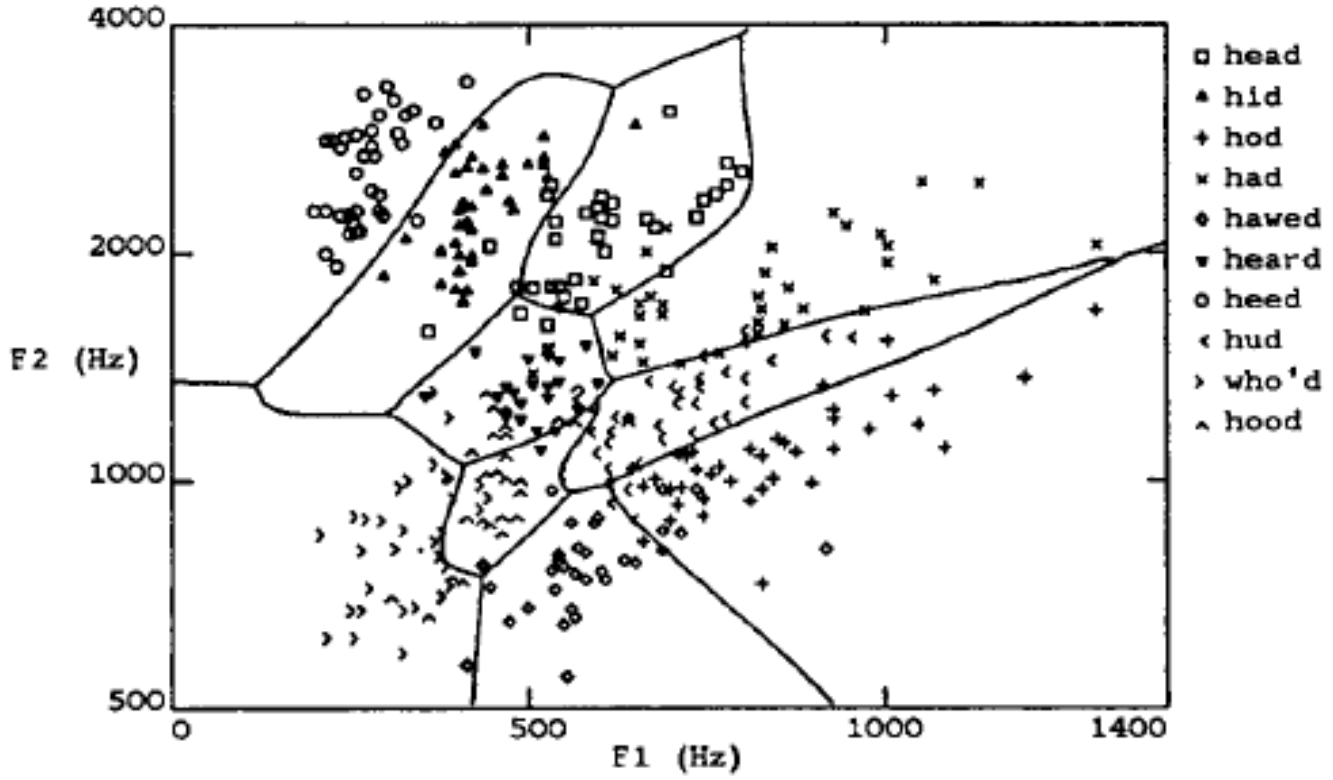


FIGURE 5.4. Decision boundaries produced by a linear machine for a three-class problem and a five-class problem. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

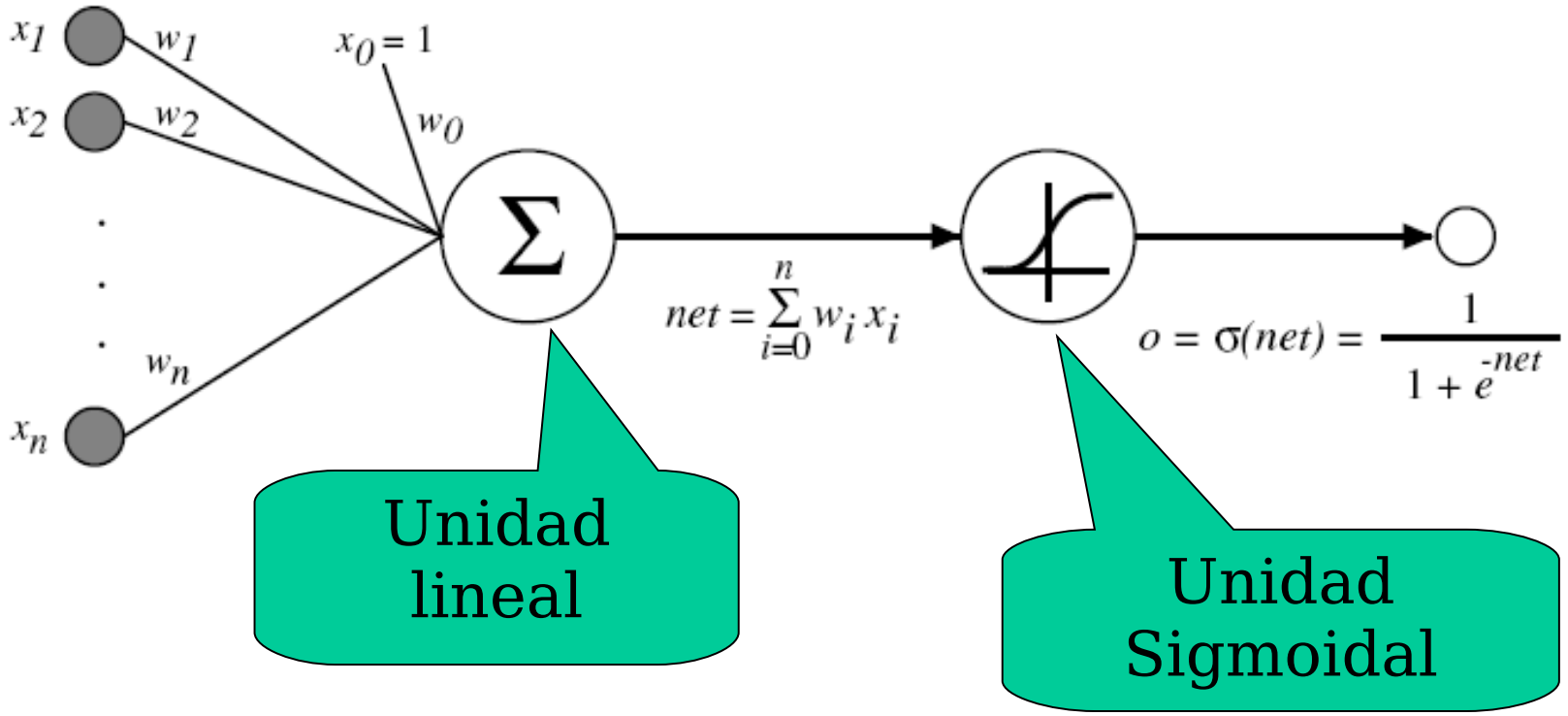
Limitaciones de los Discriminantes Lineales

- En muchos casos los límites de decisión no tienen forma poliédrica.
- Necesitamos modelar funciones discriminantes de formas más complejas.

Caso General: Reconocimiento de Fonemas



Unidad Sigmoidal



$\sigma(x)$ es la función sigmoidal $\dfrac{1}{1+e^{-x}}$

Otras Propiedades de Unidad Sigmoidal

- Permite modelar funciones de salida continua
- Permite modelar funciones de probabilidad (regresión logística)
- Derivable:

$$\frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x))$$

Redes de Unidades Sigmoidales y Límites de decisión

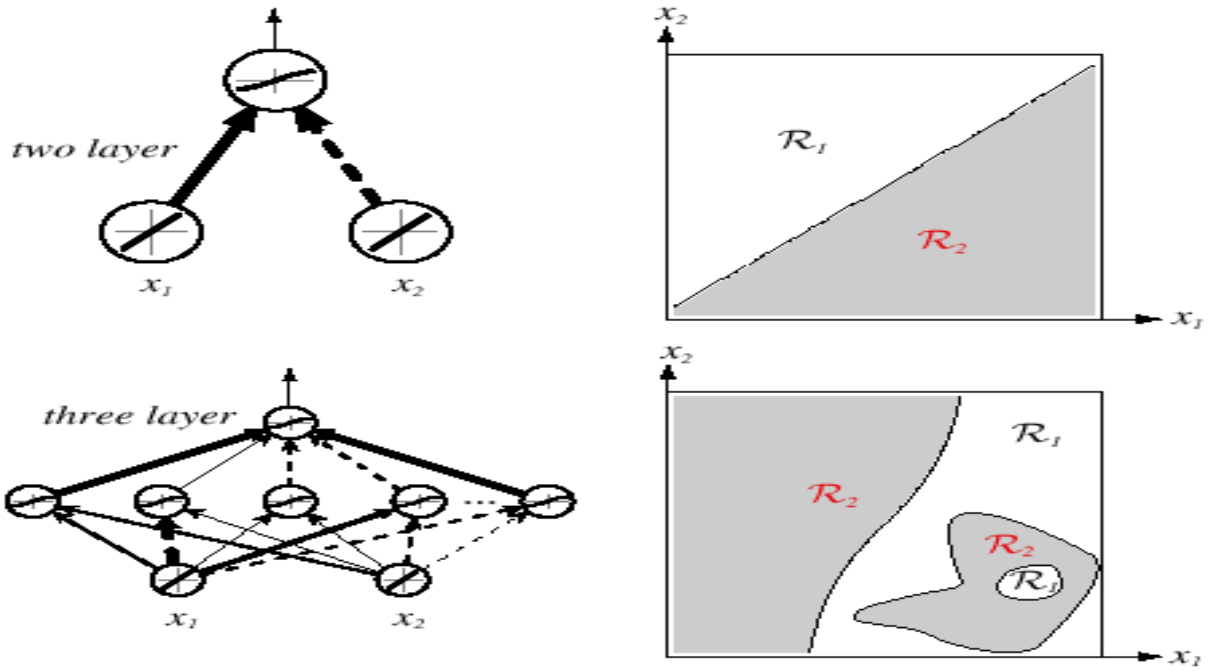
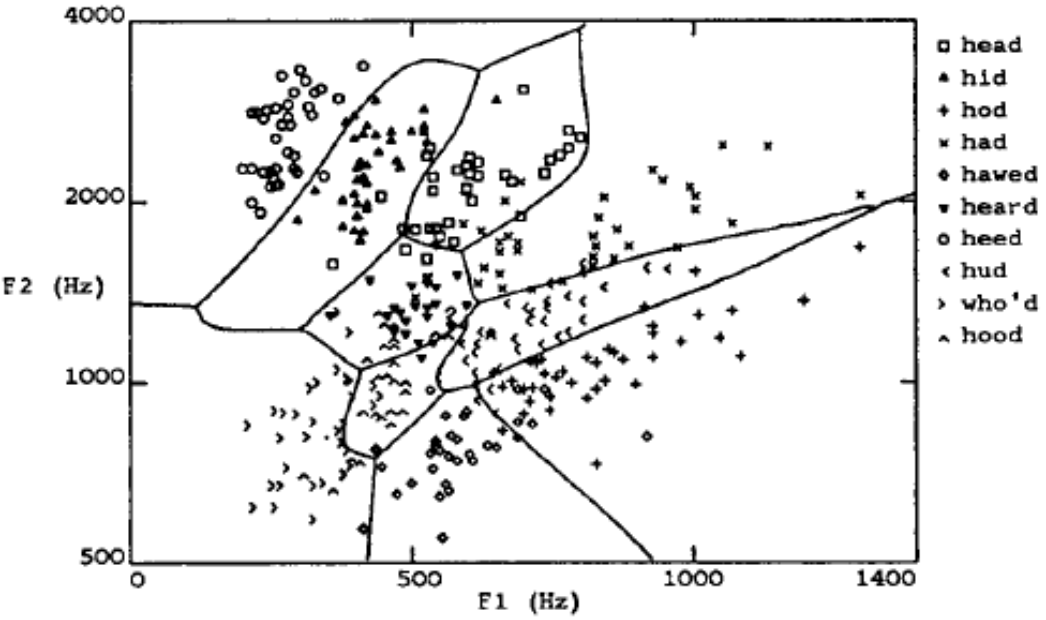
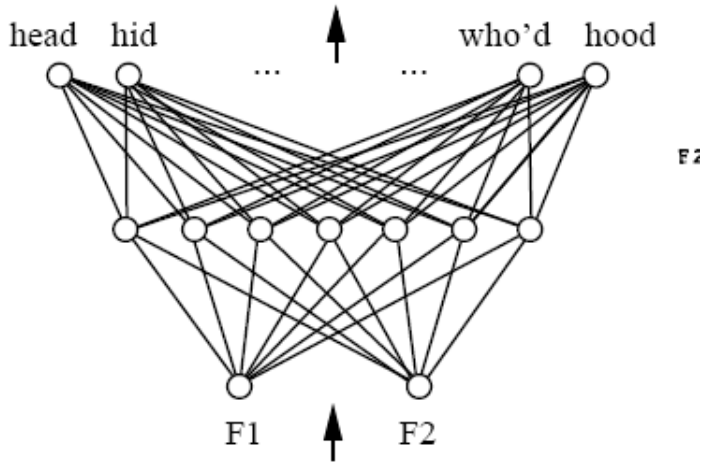


FIGURE 6.3. Whereas a two-layer network classifier can only implement a linear decision boundary, given an adequate number of hidden units, three-, four- and higher-layer networks can implement arbitrary decision boundaries. The decision regions need not be convex or simply connected. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

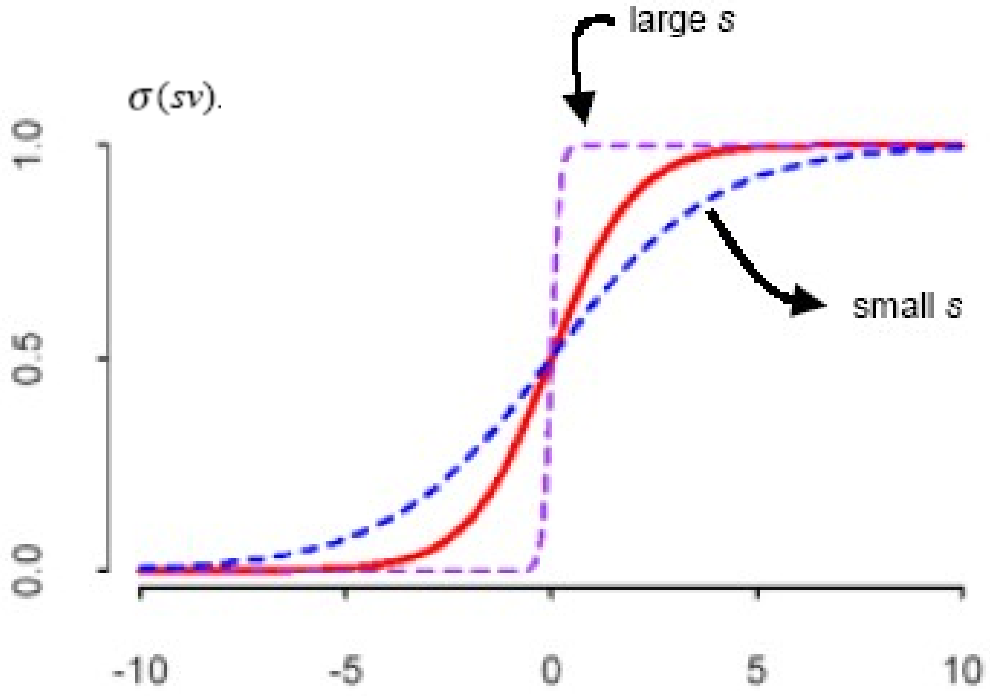
Redes de Múltiples Niveles basada en Unidades Sigmoidales





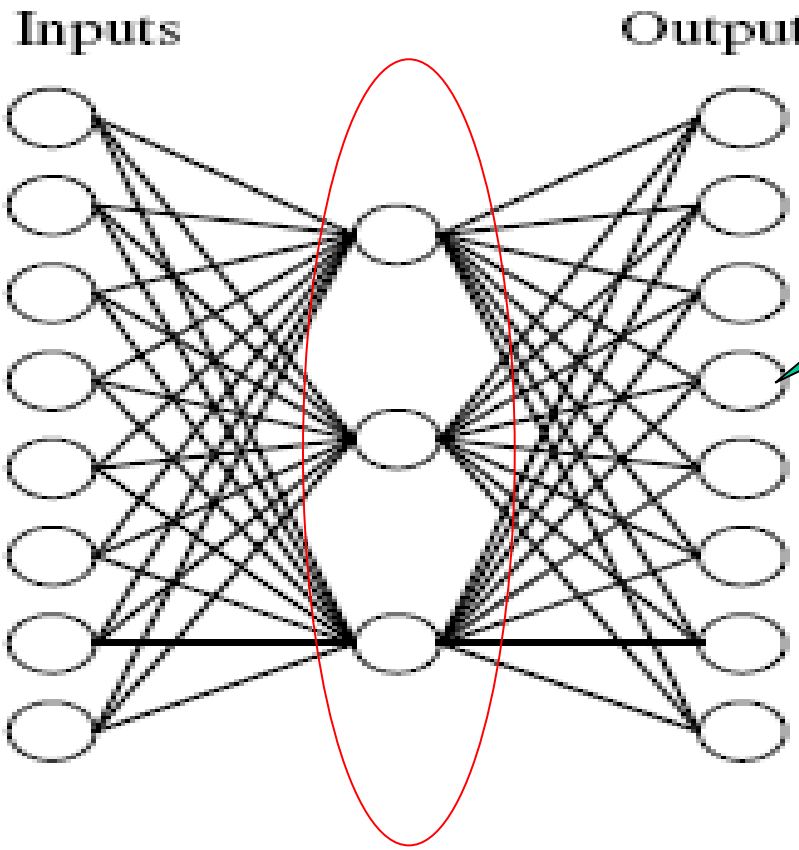
Ejemplo: sistema de reconocimiento de fonemas desde lenguaje hablado

Función Sigmoidal: tasa de activación



$$\sigma(x) \quad \frac{1}{1+e^{-x}}$$

Estructura Típica de Red de Unidades Sigmoidales



Output función discriminante para la clase i

Nivel Oculto

Poder expresivo de Redes de Unidades Sigmoidales

- Funciones Booleanas:
  - Pueden ser modeladas con redes con nivel oculto
  - El número de unidades del nivel oculto puede ser exponencial
- Funciones Continuas
  - Toda función continuas acotada puede ser aproximada con error fijo usando una red de unidades sigmoidales de un nivel oculto
  - Cualquier función puede ser aproximada con error fijo usando una red de unidades sigmoidales de dos niveles ocultos

Aproximación de una función acotada usando unidades sigmoidales

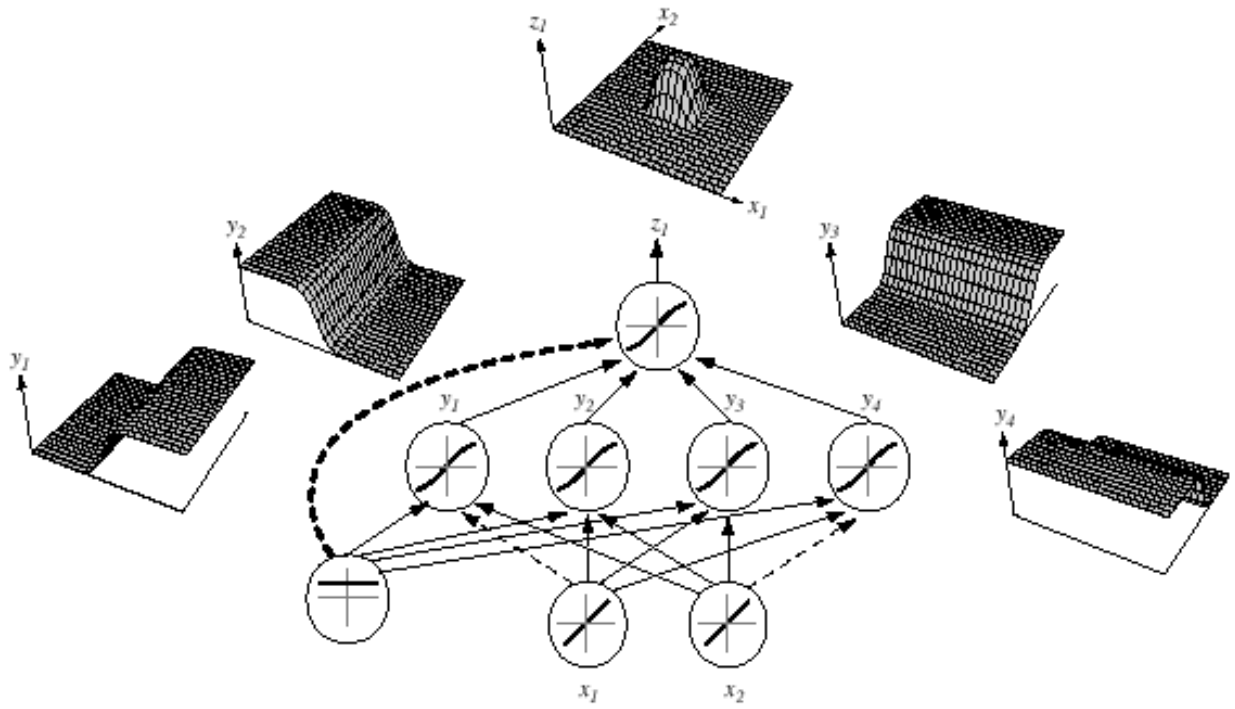
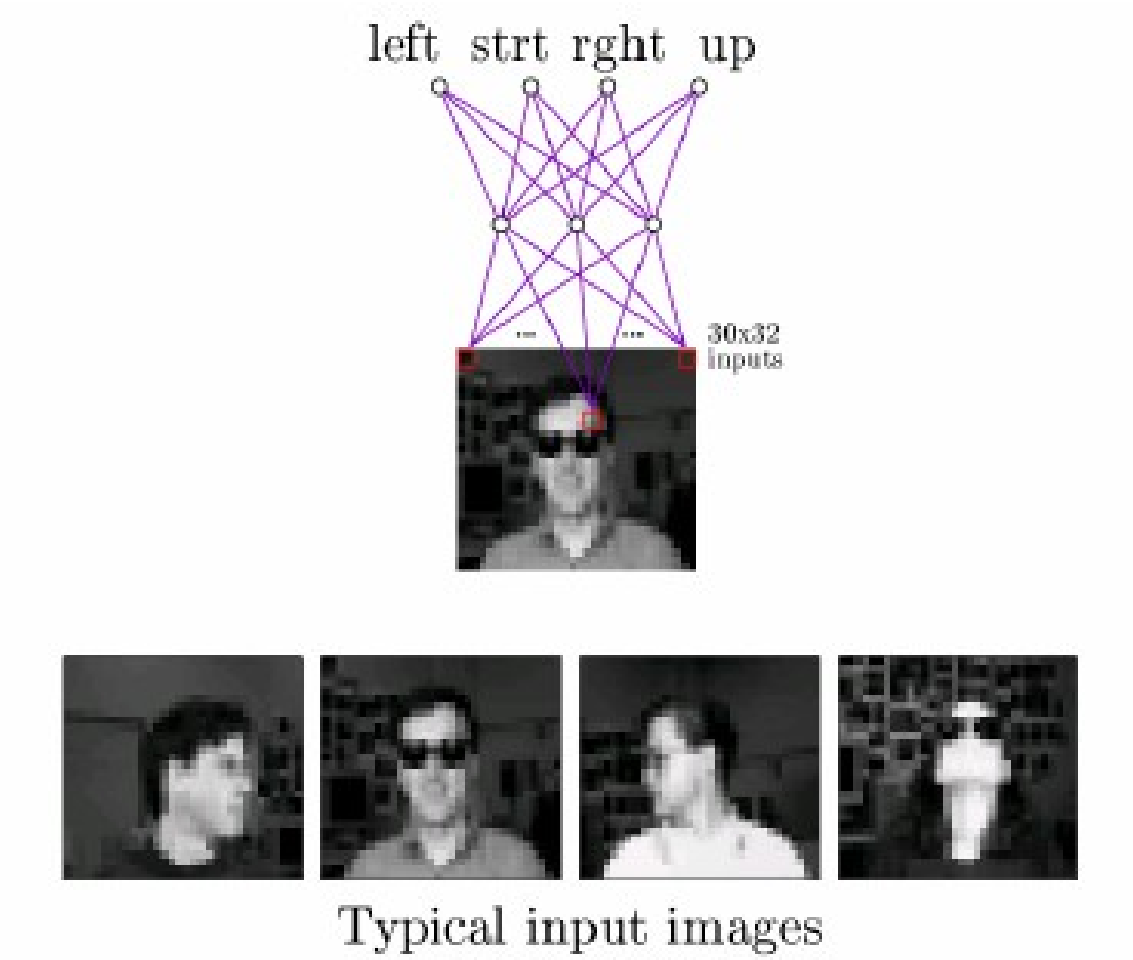


FIGURE 6.2. A 2-4-1 network (with bias) along with the response functions at different units; each hidden output unit has sigmoidal activation function $f(\cdot)$. In the case shown, the hidden unit outputs are paired in opposition thereby producing a "bump" at the output unit. Given a sufficiently large number of hidden units, any continuous function from input to output can be approximated arbitrarily well by such a network. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

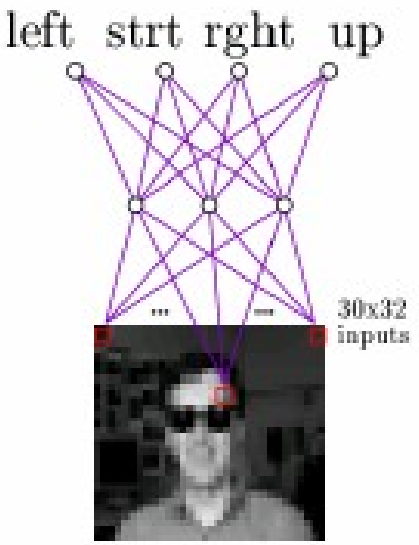
Ejemplo de Red con Nivel Oculto

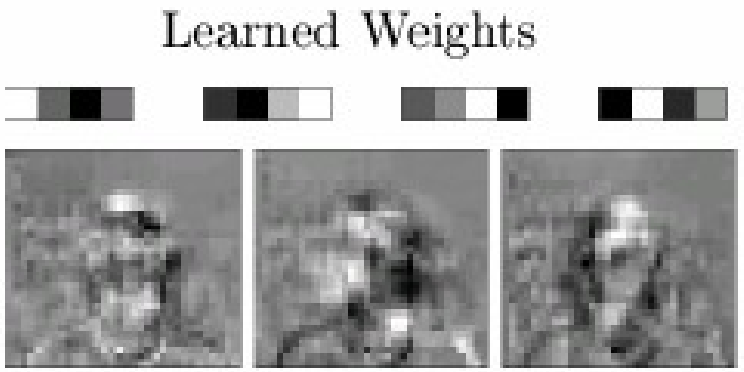


Typical input images

Reconoce la posición de la cabeza con un 90% de precisión.

Ejemplo de Red con Nivel Oculto

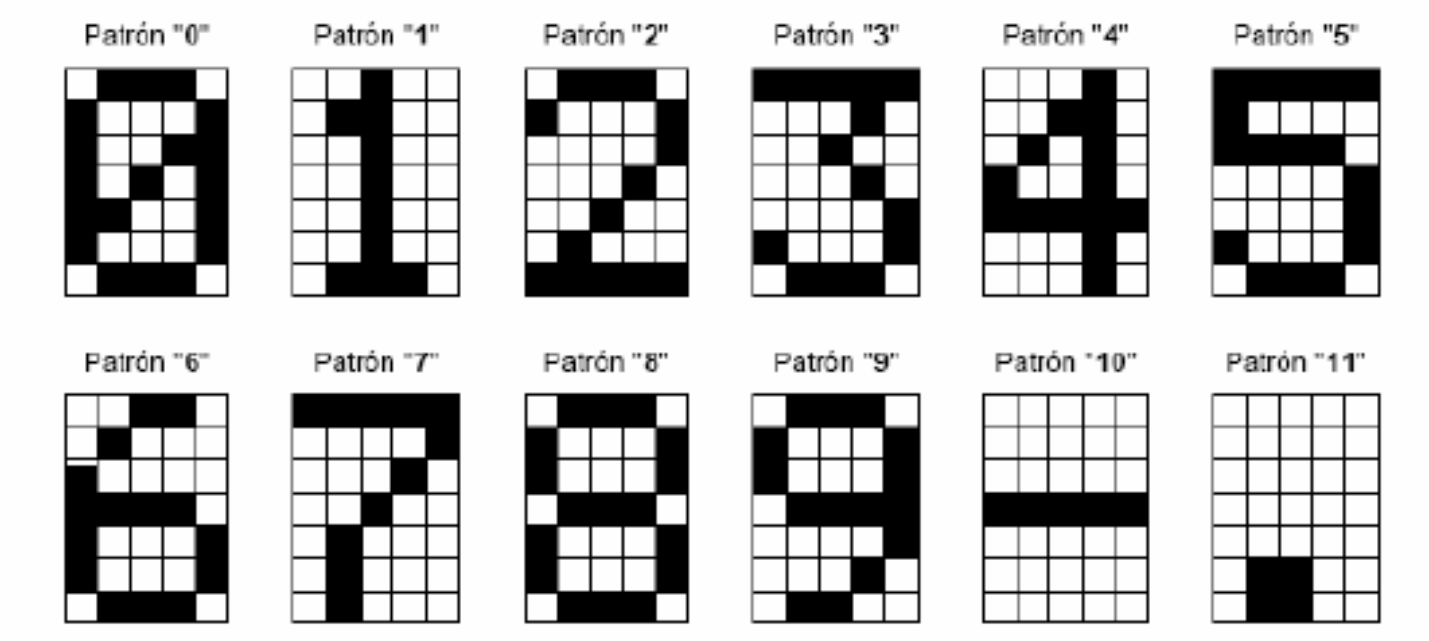




Typical input images

http://www.cs.cmu.edu/~tom/faces.html

Aplicación Típica: Reconocimiento de Patrones



Entrenamiento de una Red de Unidades Sigmoidales

Método de Retropropagación

Idea: aplicar descenso por el gradiente para la red completa.

Descenso por el Gradiente

Gradiente:

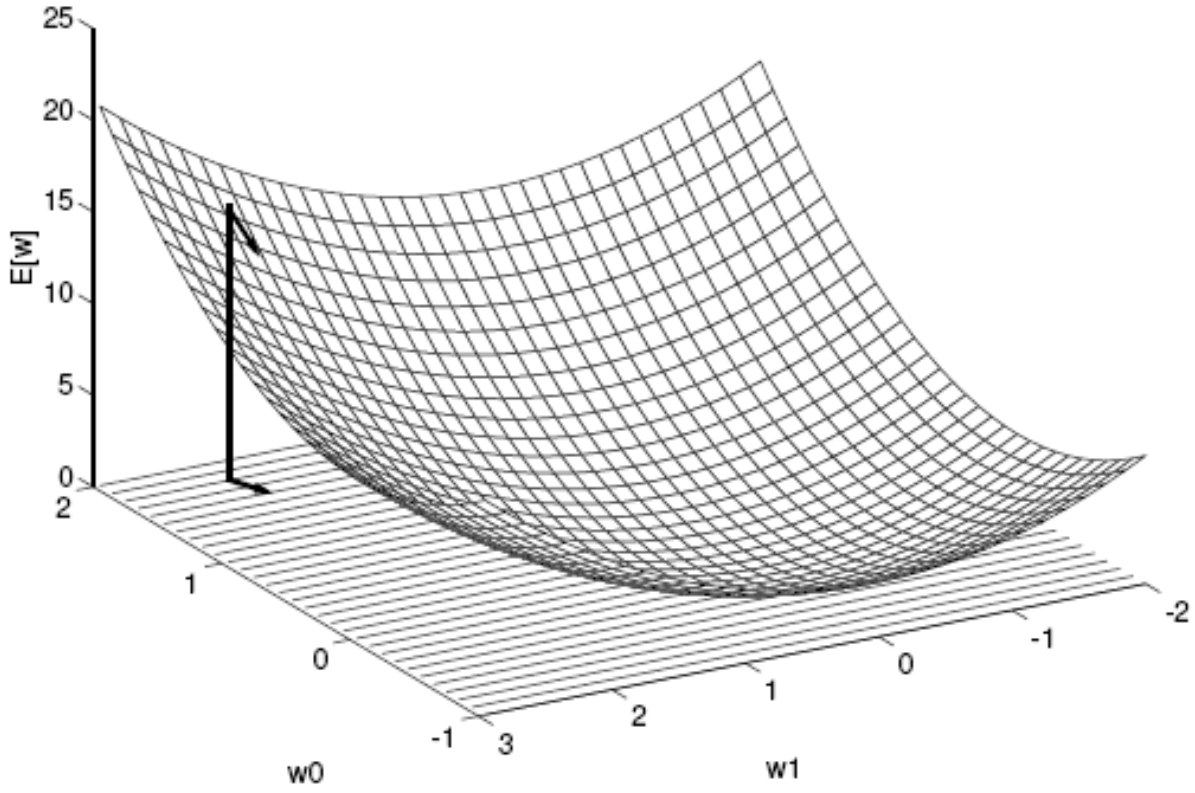
$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_n}\right]$$

Regla de Entrenamiento:

$$\Delta \vec{w} = -\eta \nabla E[\vec{w}]$$

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

Descenso por Gradiente (cont.)



Gradiente de Unidad Sigmoidal

$$\frac{\partial E}{\partial w_i} = -\sum_{d \in D} (t_d - o_d) o_d (1 - o_d) x_{i,d}$$

Usando descenso por gradiente podemos entrenar:

- Una unidad sigmoidal
- Red de unidades sigmoidales→ Retropropagación (Backpropagation)

Problema: no tiene forma parabólica, existen mínimo
Locales y globales.

Retropropagación (Backpropagation)

- For each training example, Do
  1. Input the training example to the network and compute the network outputs
  2. For each output unit k

$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$$

  3. For each hidden unit h

$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{h,k}\delta_k$$

  4. Update each network weight $w_{i,j}$

$$w_{i,j} \leftarrow w_{i,j} + \Delta w_{i,j}$$

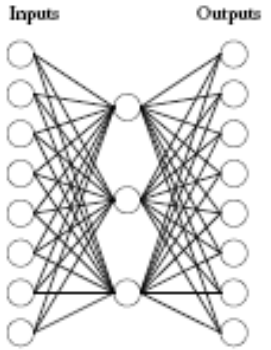
  where

$$\Delta w_{i,j} = \eta \delta_j x_{i,j}$$

Retropropagación

- Descenso por Gradiente sobre toda la red
- Fácil de generalizar a redes de grafos dirigidos arbitrarias
- Encuentra un mínimo local no necesariamente global
  - Funciona bien en la práctica (se pude corre múltiples veces para mejorar)
- Minimiza error sobre datos de entrenamiento
- Lento de entrenar (puede tomar cientos de iteraciones)
- Uso de la red después de entrenarse es rápido

Ejemplo: Función identidad

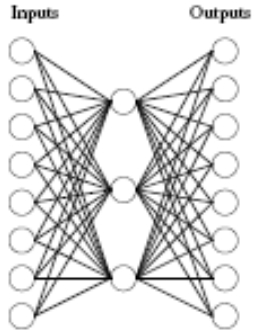


A target function:

Input		Output
10000000	→	10000000
01000000	→	01000000
00100000	→	00100000
00010000	→	00010000
00001000	→	00001000
00000100	→	00000100
00000010	→	00000010
00000001	→	00000001

Red después de aprendizaje

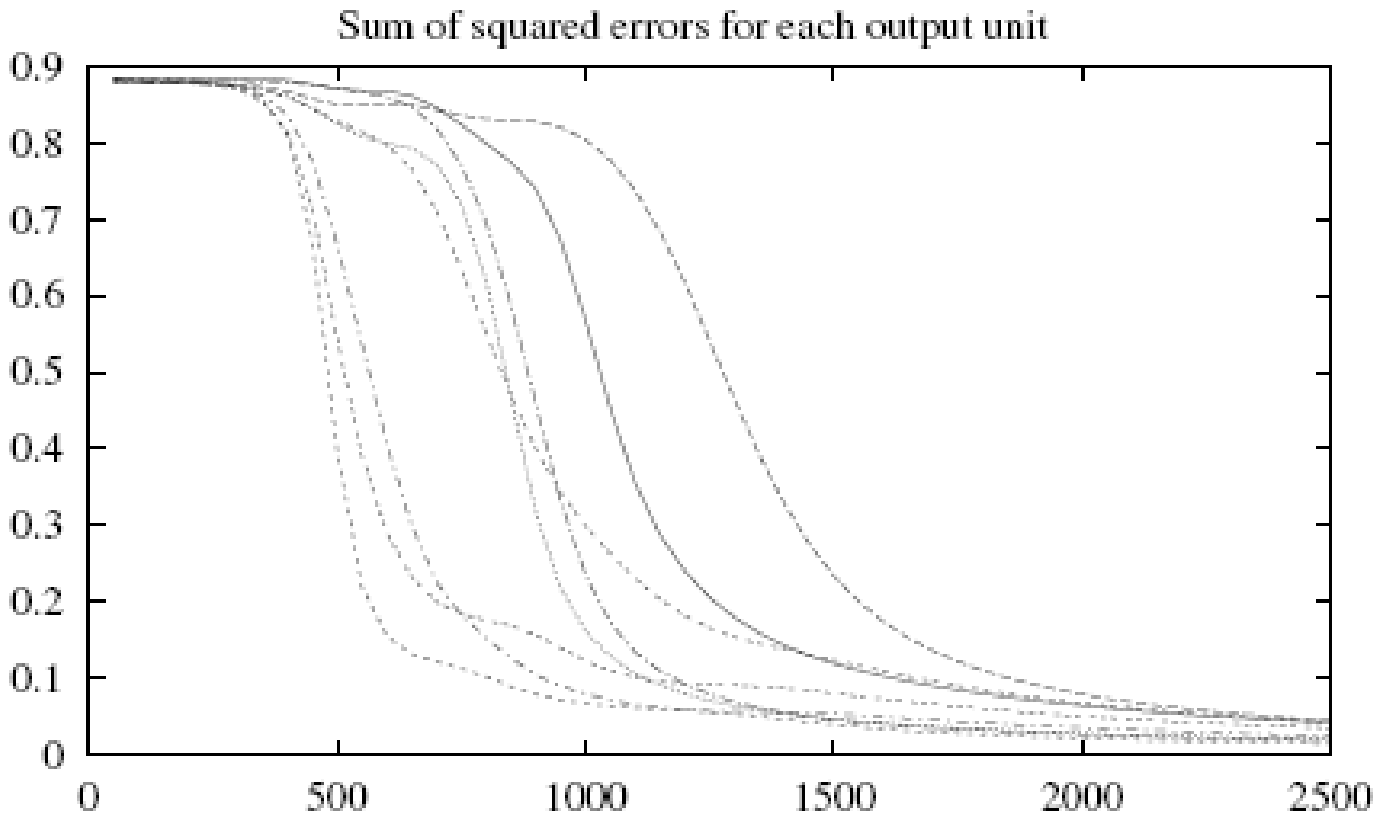
A network:



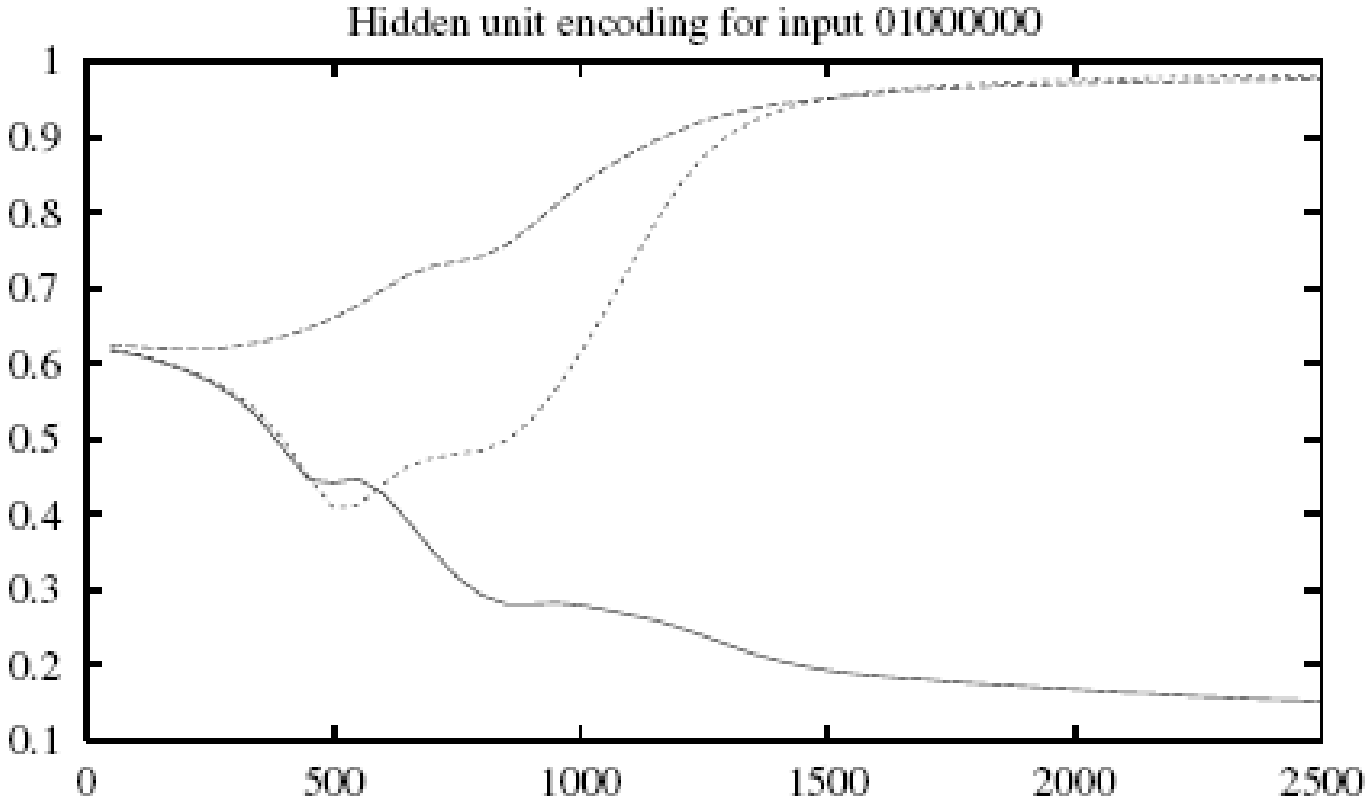
Learned hidden layer representation:

Input		Hidden Values				Output
10000000	→	.89	.04	.08	→	10000000
01000000	→	.01	.11	.88	→	01000000
00100000	→	.01	.97	.27	→	00100000
00010000	→	.99	.97	.71	→	00010000
00001000	→	.03	.05	.02	→	00001000
00000100	→	.22	.99	.99	→	00000100
00000010	→	.80	.01	.98	→	00000010
00000001	→	.60	.94	.01	→	00000001

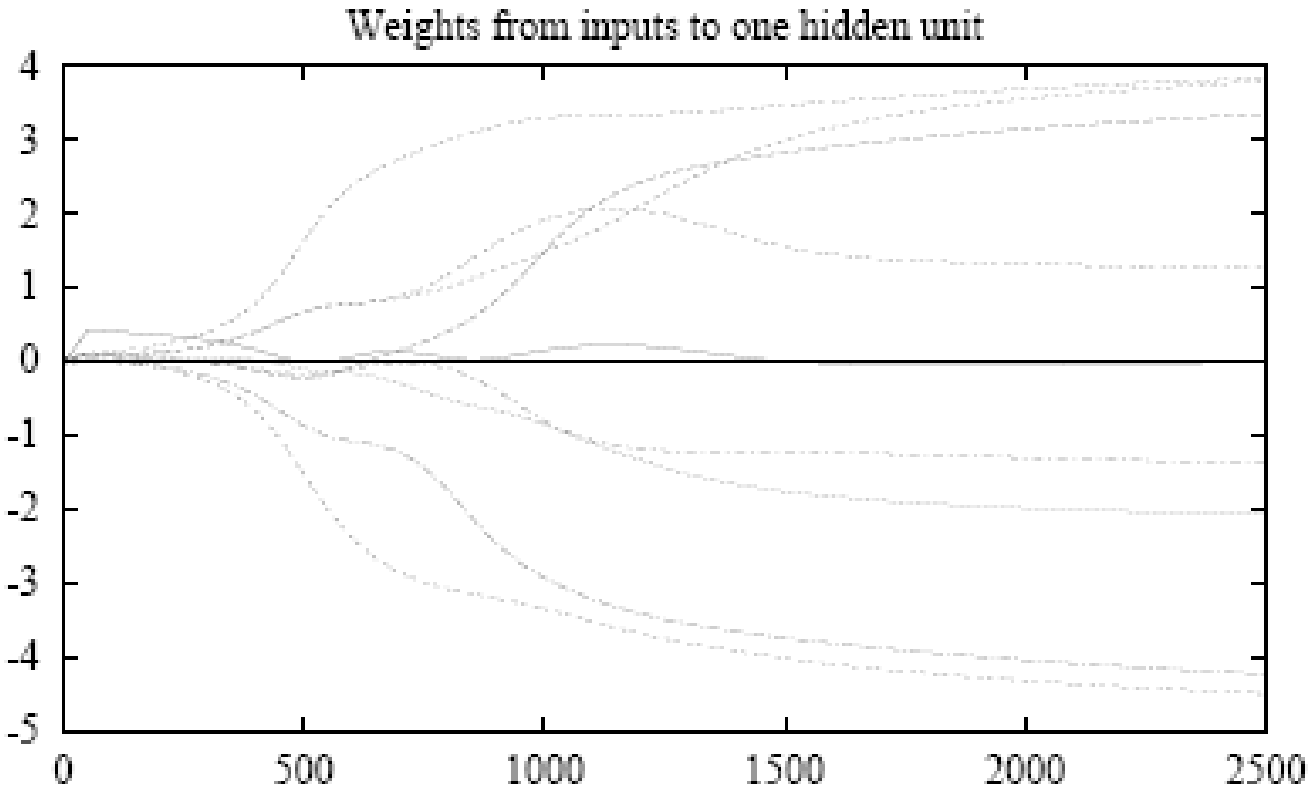
Entrenamiento



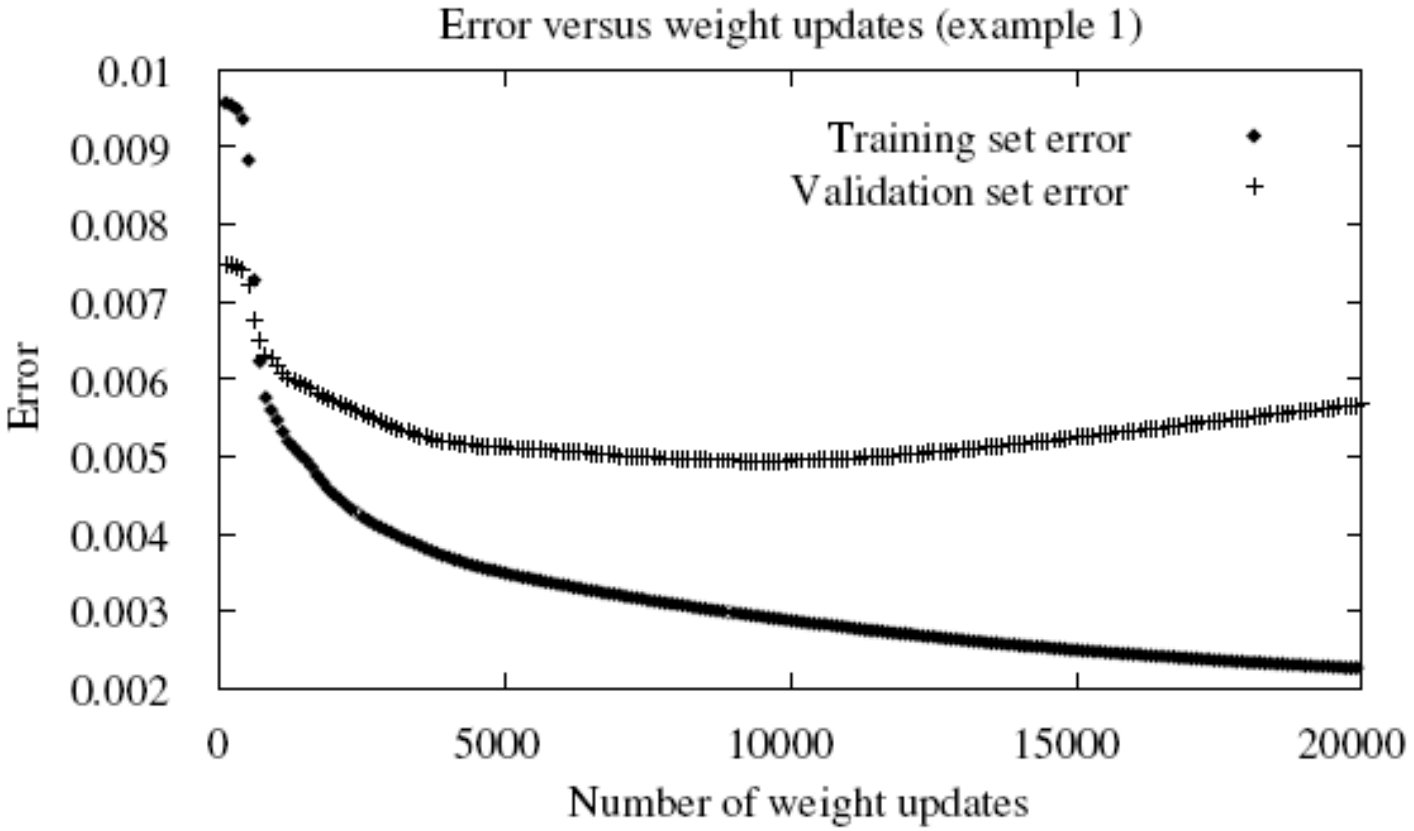
Entrenamiento



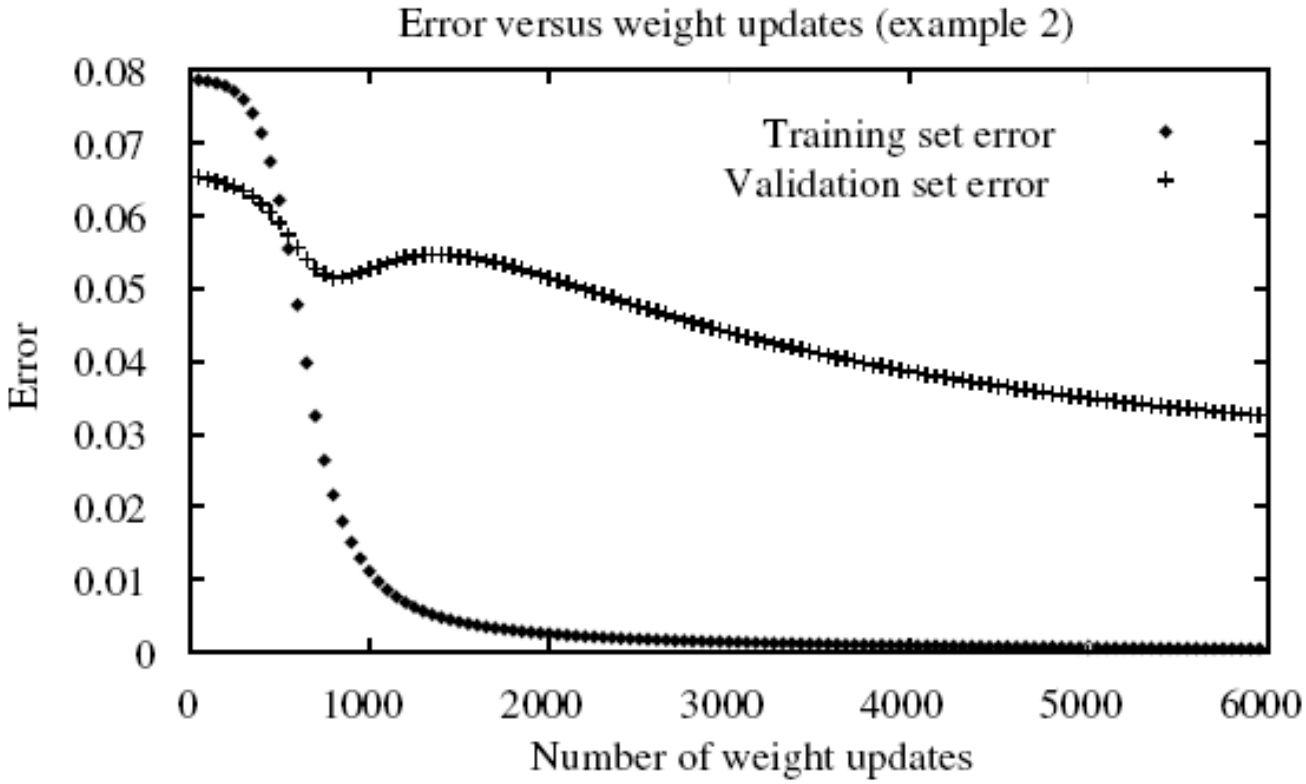
Entrenamiento



Sobreajuste

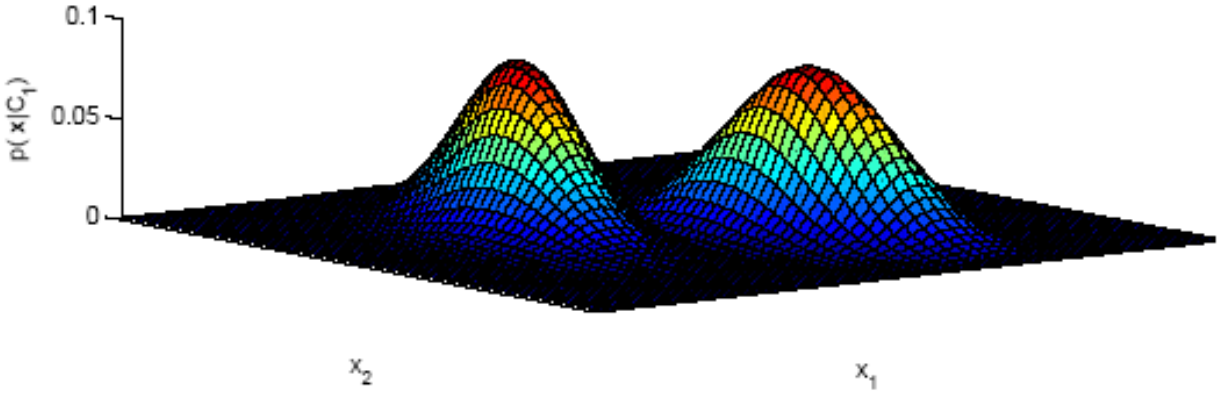


Sobreajuste (cont.)



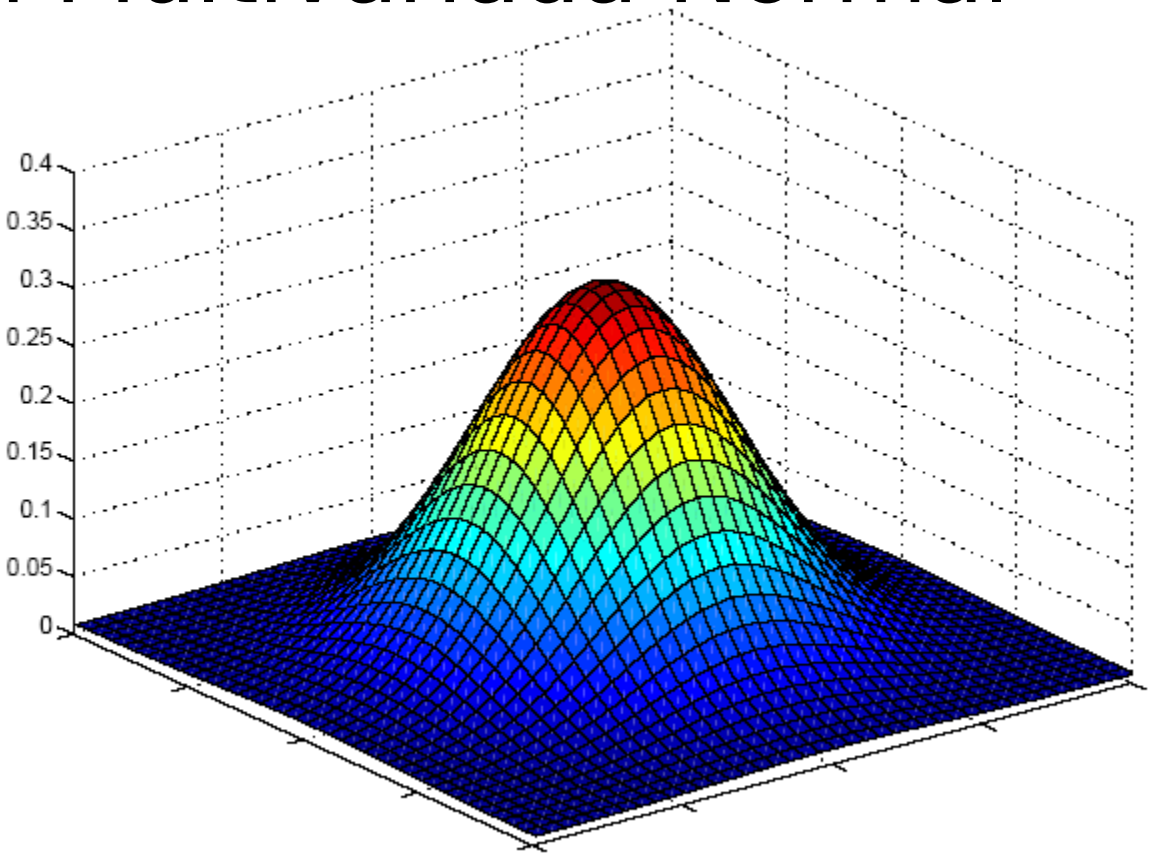
Las Unidades Sigmoidales Modelan el Sgte Caso

- Los datos de cada clase distribuyen de acuerdo a una distribución Normal Multivariada
- La matriz de covarianza es común a todas las clases
- La función discriminante para la clase C_i es $g_i(\boldsymbol{x}) = P(C_i|\boldsymbol{x})$ (posterior de x)
- En este coso puede demostrar que $g_i(\boldsymbol{x})$ corresponde a una unidad sigmoidal



post

Distribucion Multivariada Normal



$$x \sim N_d(\mu, \Sigma)$$

$$p(x) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right]$$

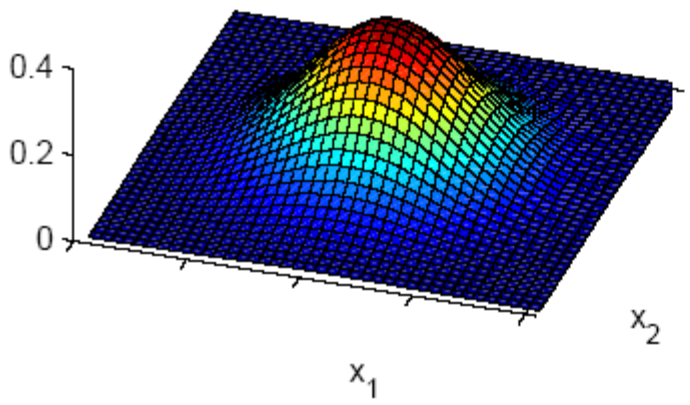
Distribucion Multivariada Normal (cont)

$$Media: \mu_i = \frac{\sum_{t=1}^{N} x_i^t}{N}, i = 1, \ldots, d$$

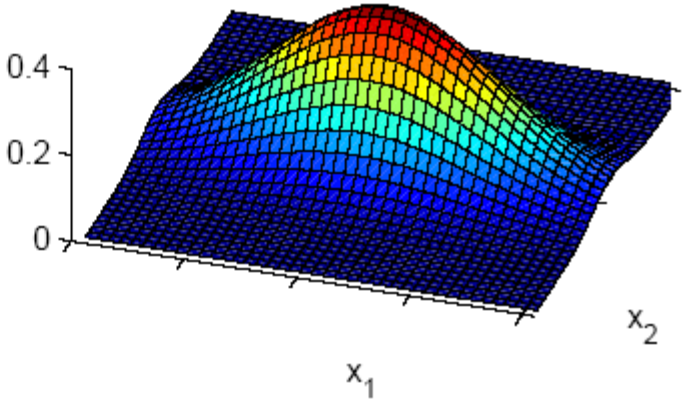
Matriz de Covarianza $$\sum : \sigma_{ij} = \frac{\sum_{t=1}^{N} \left(x_i^t - m_i \right) \left(x_j^t - m_j \right)}{N}$$

Matriz de Correlación $$R: r_{ij} = \frac{\sigma_{ij}}{\sigma_i \sigma_j}$$

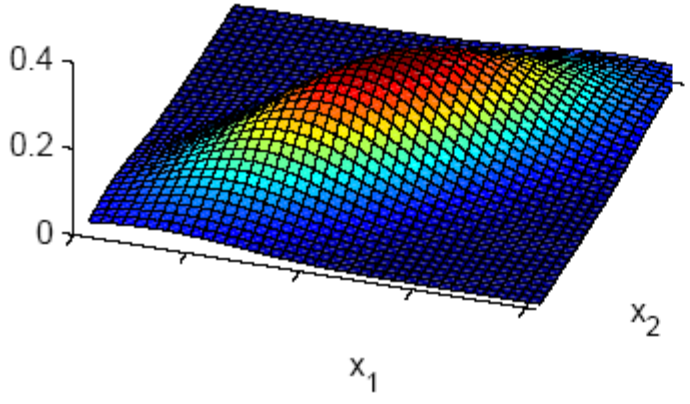
$Cov(x_1,x_2)=0$, $Var(x_1)=Var(x_2)$



$Cov(x_1,x_2)=0$, $Var(x_1)>Var(x_2)$



$Cov(x_1,x_2)>0$



$Cov(x_1,x_2)<0$