# Perceptrón (Y Winnow)

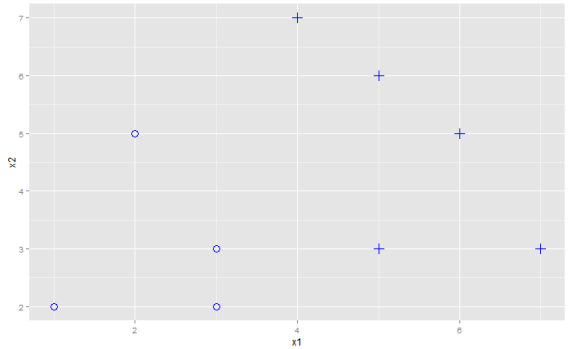

## 75.06 Organización de Datos

# Perceptrón

- Algoritmo creado por F. Rosenblatt a fines de la década del 50.
- Se lo asoció erroneamente con la forma en la que trabaja el cerebro humano
- El algoritmo fue dejado de lado y resurgió

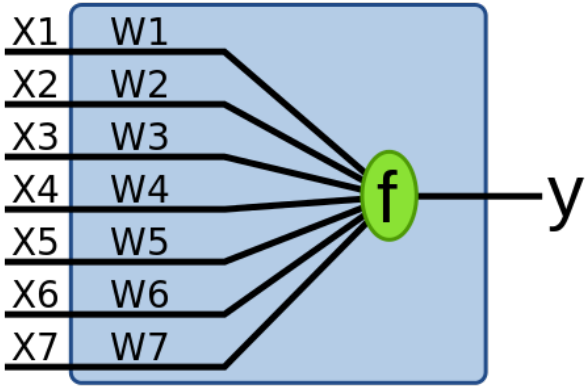
# Perceptrón

- Es un clasificador lineal binario (-1,1)
- Encuentra la frontera que permite separar datos de dos clases diferentes.



# Perceptrón

- Se combinan linealmente los inputs usando pesos w
- El signo de $w*x_i$ determina su clase (-1 o 1)



# Algoritmo

Agregar una dimension con 1 a X

Inicializar w al azar (da lo mismo)

repetir hasta convergencia

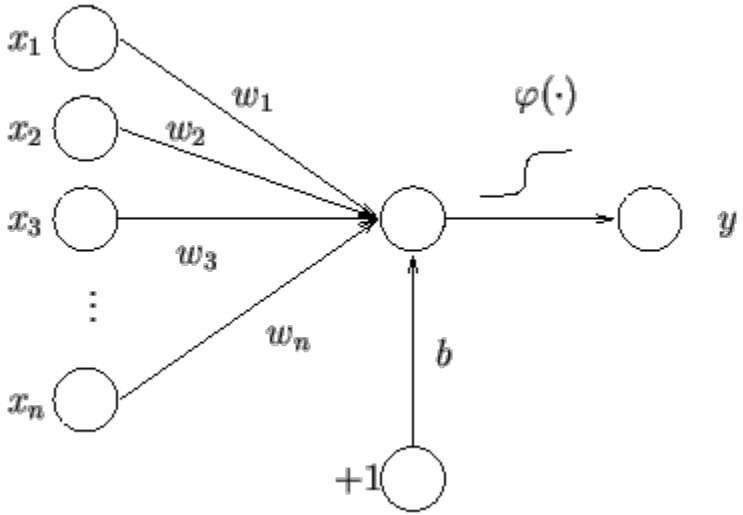
    tomar un punto de X

    calcular $w * x_i$

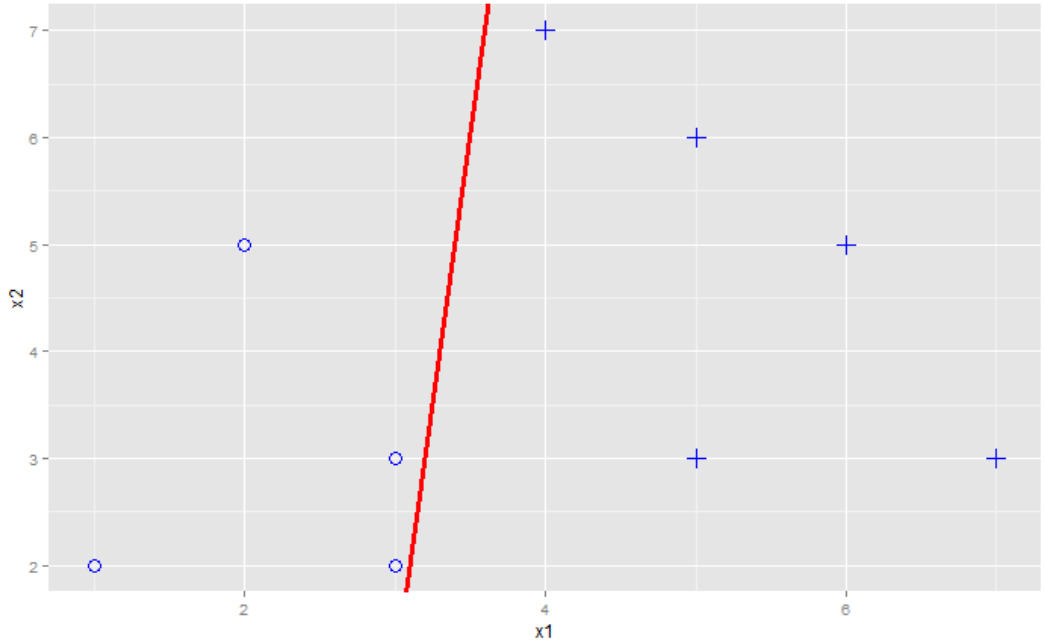
    si el signo de $w*x_i$ es <> de $y_i$

        $w = w + y_i * x_i$

# Graficamente



# Resultado



# Teorema de Convergencia

[Block, Novikoff]

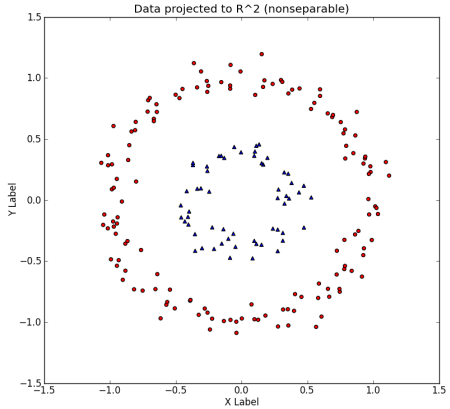
*"Si los datos son linealmente separables perceptrón converge en una cantidad acotada de pasos"*

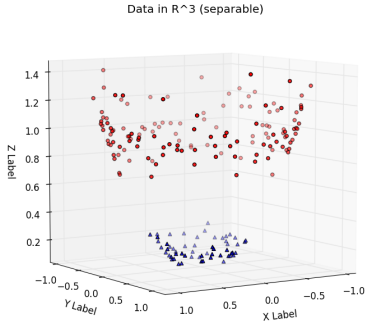
# Propiedades

- Orden sub-lineal (!)
- Converge o Falla (no aproxima)
- Funciona Online
- Escala perfectamente
- Paralelizable

# Datos no lineales

Idea: Mapear de n dimensiones a m (m>>n) de forma tal que en m los datos sean linealmente separables por un hiperplano.



# Función Kernel

Es una función que mapea de n a m dimensiones con m>>n

Ej: $f(x1,x2) = (x1,x2,x1^2+x2^2)$

# Kernel Trick

Existen funciones kernel que nos permiten calcular el producto interno de f(x1)*f(x2) sin tener que expandir x1 y x2 a m dimensiones.

$$K(x1,x2) = f(x1) * f(x2)$$

Siendo f(x) la función que lleva de n a m dims.

# Kernels conocidos

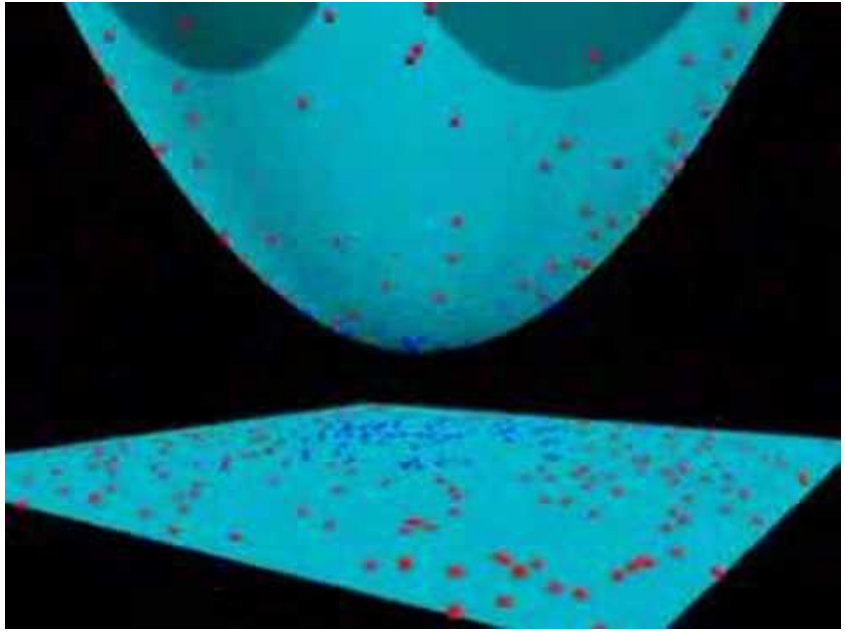
$K(x1,x2) = (x1 * x2)^d$

$K(x1,x2) = (x1*x2 + r)^d$

$K(x1,x2) = \exp(-|x1*x2|^2/2\ s^2)$ [RBF]

$K(x1,x2) = \tanh(nx1*x2+v)$

# Kernel RBF en acción



# Perceptrón con Kernel

Initialize $\boldsymbol{\alpha}$ to an all-zeros vector of length $n$, the number of training samples.

For some fixed number of iterations, or until the model makes no mistakes on the training set:

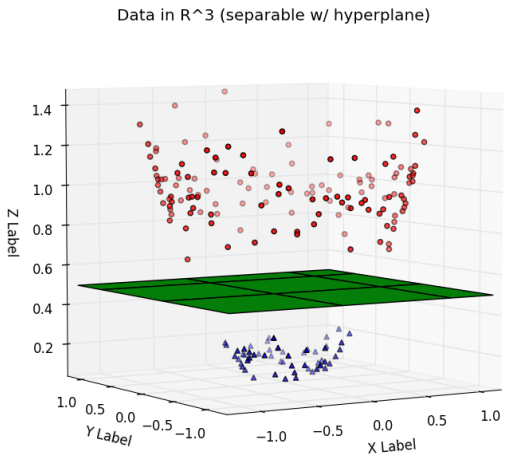
For each training example $\mathbf{x}, y$:

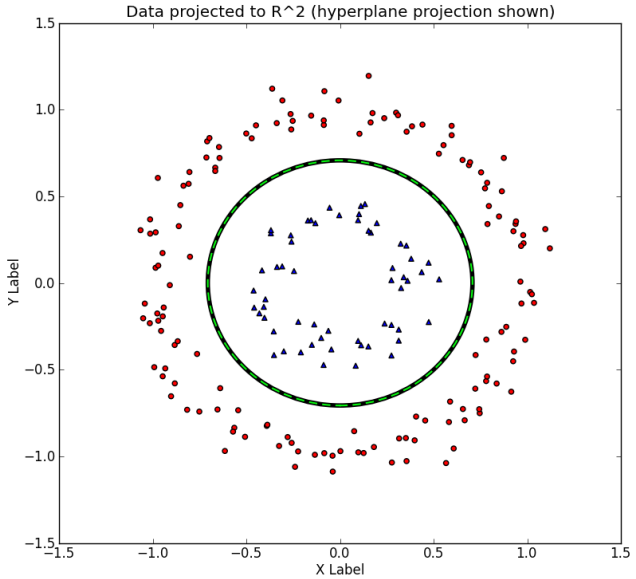
Let $\hat{y} = \operatorname{sgn} \sum_{i}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x})$

If $\hat{y} \neq y$, perform an update:

$\alpha_i := \alpha_i + 1$

# Perceptrón con Kernel



# Repaso

- Algoritmo online
- Sub-lineal
- Convergencia
- Kernel-Trick

# Winnow

- X = vectores booleanos (1,0,1,0,1,0,0,1)
- Y = 0 o 1 (clases)
- w = vector a aprender tal que

  w*x >= t para y==1

  w*x <= t para y==0
- t es un umbral (n/2 funciona bien en gral)

# Winnow

- El algoritmo es muy similar a perceptrón.
- Empezamos con w = 1,1,1,1,1,1
- En cada paso calculamos $w*x_i$
- Si clasifica mal entonces

    si y era 0 a cada elemento de w al cual le corresponde x==1 lo dividimos por alfa

    si y era 1 a cada elemento de w al cual le corresponde x==1 lo multiplicamos por alfa