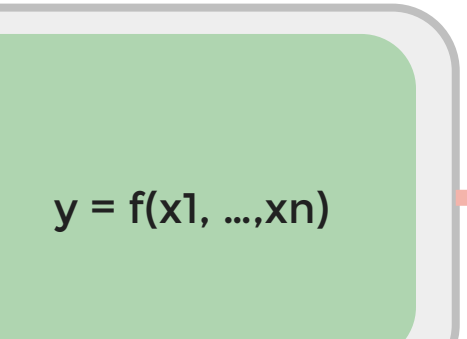
Otros modelos de clasificación

Fernanda Mora
12 octubre, 2016

Diplomado de Minería de Datos
para el soporte a la toma de decisiones

Contenido

- Máquinas de soporte vectorial
  - Ejercicio 1
- Redes neuronales
  - Ejercicio 2



Máquinas de soporte vectorial

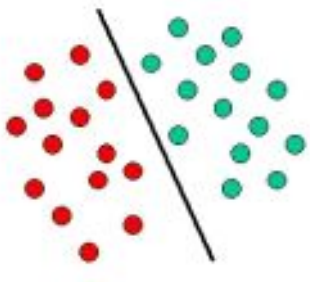
Preliminares

- Las máquinas de soporte vectorial fueron introducidas por Boser, Guyon y Vapnik en 1992 y se volvieron populares
- Algoritmo teóricamente muy bien sustentado (Teoría de Aprendizaje Estadístico de los 60's)
- Buen desempeño empírico: se ha aplicado exitosamente en muchas y diversas áreas: bioinformática (cadenas de ADN, estructura de proteínas), computer vision (reconocimiento de imágenes y texto), etc.
- Mucha gente trabaja con ellas: machine learning, optimización, estadística, redes neuronales, análisis funcional

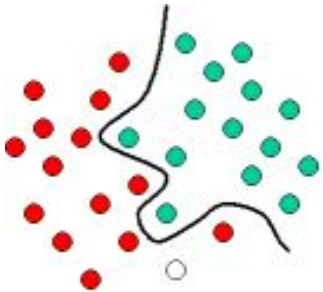
Preliminares

- Las máquinas de soporte vectorial se usan en problemas de clasificación y regresión
- Las veremos para clasificación

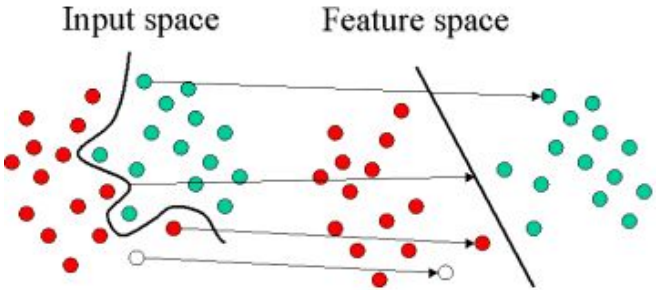
Clasificador lineal, sencillo



Clasificador no lineal, complicado



Máquina de soporte vectorial



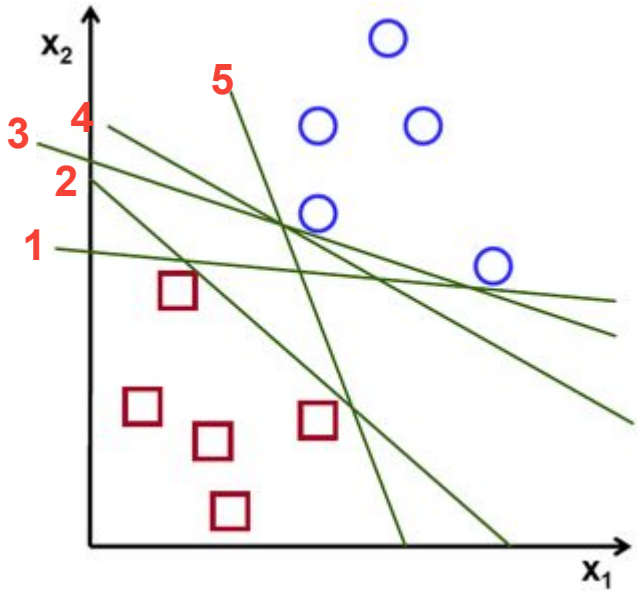
Funciones kernel *mapean de un espacio a otro*

Preliminares

- Las variables de soporte $X_1,...,X_k$ pueden ser tanto cuantitativas como categóricas
- SVM construye un hiperplano o conjunto de hiperplanos en un espacio de dimensionalidad muy alta buscando separar lo más posible las dos clases
- Esta será la separación “óptima”: buscar el hiperplano que tenga la máxima distancia a los datos
- Por eso también se les conoce como clasificadores de margen máximo

Preliminares

- Ejemplo en 2 dimensiones: ¿Cuál es la recta separadora óptima?

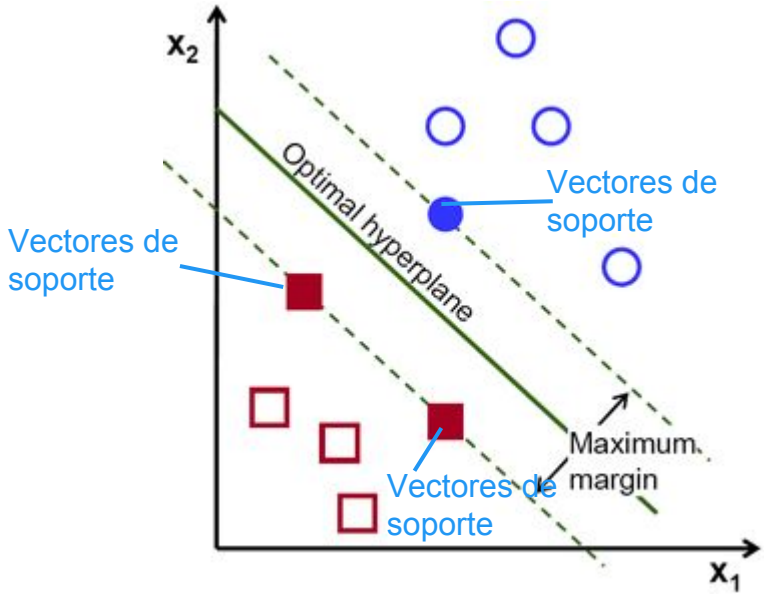


La recta no debe pasar cerca de los datos porque será pobre al clasificar a nuevas observaciones

La recta óptima debe tener la máxima distancia a los datos

Preliminares

- Ejemplo en 2 dimensiones: ¿Cuál es la recta separadora óptima?



El algoritmo de MSV busca la recta/plano que tenga la máxima distancia a los datos que están más cercanos

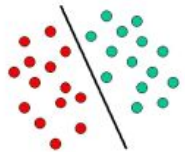
A estos datos se les llama vectores de soporte

Son los más difíciles de clasificar y son los que definen cómo será el plano de MSV

Entre más grande el margen, menor el error

Máquinas de soporte vectorial

Cuando los x's son linealmente separables



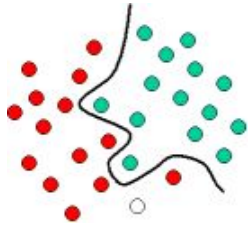
- Supongamos que tenemos un hiperplano: $f(x) = \beta_0 + \beta^T x,$
- Queremos que el plano sea único más cercano a los vectores de soporte sea único : $|\beta_0 + \beta^T x| = 1$
- La distancia de un vector x=(x0, x1,...x_n) a un plano se puede calcular como: $\text{distance} = \frac{|\beta_0 + \beta^T x|}{\|\beta\|}.$
- Entonces: $\text{distance}_{\text{support vectors}} = \frac{|\beta_0 + \beta^T x|}{\|\beta\|} = \frac{1}{\|\beta\|}.$ y el margen $M = \frac{2}{\|\beta\|}$
- El problema de Maximizar M equivale a:

$$\min_{\beta,\beta_0} L(\beta) = \frac{1}{2}\|\beta\|^2 \text{ subject to } y_i(\beta^T x_i + \beta_0) \geq 1 \ \forall i,$$

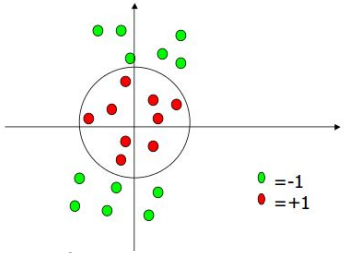
- Se resuelve usando multiplicadores de Lagrange para encontrar las betas (¡la solución es única!)

Máquinas de soporte vectorial

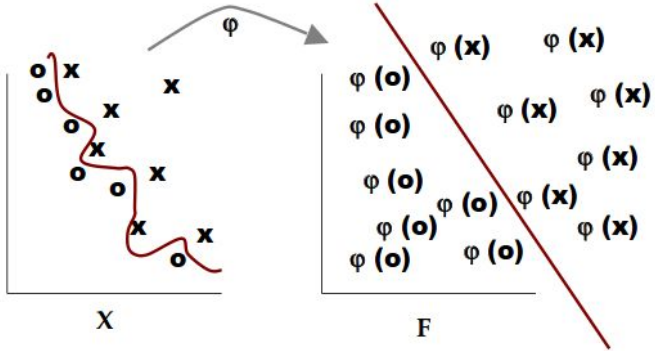
Cuando los x's NO son linealmente separables



- Si las x's no son linealmente separables, entonces hay que usar primero las funciones kernel



- Idea: mandar los vectores X a una dimensión mucho más alta usando una función ϕ en donde sí sean linealmente separables



Máquinas de soporte vectorial

Cuando los x's NO son linealmente separables

- La formulación dual del problema (créanlo), es así:

Dual problem:

$$\max L_D(a_i) = \sum_{i=1}^{l} a_i - \frac{1}{2}\sum_{i=1}^{l} a_i a_j y_i y_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right)$$

$$\text{s.t.} \sum_{i=1}^{l} a_i y_i = 0 \;\&\; a_i \geq 0$$

Lo queremos definir así para tener productos puntos

- Conociendo las a_i podemos obtener los coeficientes buscados $\mathbf{w} = \sum_{i=1}^{l} a_i y_i \mathbf{x}_i$ y la predicción queda así: $f(x) = \mathbf{w} \bullet \mathbf{u} + b = (\sum_{i=1}^{l} a_i y_i \mathbf{x}_i \bullet \mathbf{u}) + b$

Máquinas de soporte vectorial

Cuando los x's NO son linealmente separables

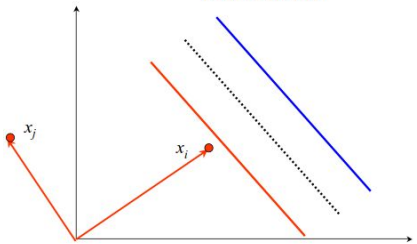
- Sólo los vectores de soporte tendrán $\left(\mathbf{x}_i \cdot \mathbf{x}_j\right) \neq 0$

Dual problem:

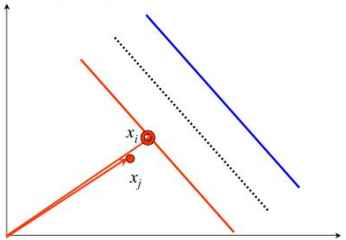
$$\max L_D(a_i) = \sum_{i=1}^{l} a_i - \frac{1}{2} \sum_{i=1}^{l} a_i a_j y_i y_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right)$$

$$\text{s.t.} \sum_{i=1}^{l} a_i y_i = 0 \;\&\; a_i \geq 0$$

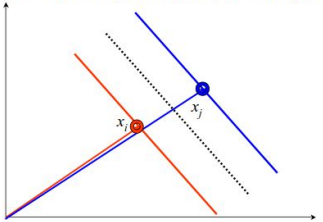
2 dissimilar (orthogonal) vectors don't count at all



2 vectors that are similar but predict the same class are redundant



Insight into inner products, graphically: 2 very very similar x_i, x_j vectors that predict difft classes tend to maximize the margin width



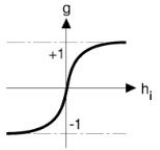
Máquinas de soporte vectorial

Funciones Kernel

- ϕ será la función que mapee de un espacio a otro
- $\phi(X_i)\phi(X_j)$ puede ser computacionalmente muy costoso (no lo sabemos)
- Funciones Kernel: Reemplazar a todos los productos punto por la función Kernel $K(\mathbf{X}_i, \mathbf{X}_j) = \phi(\mathbf{X}_i) \bullet \phi(\mathbf{X}_j)$

$$K(\mathbf{X}_i, \mathbf{X}_j) = \begin{cases} \mathbf{X}_i \cdot \mathbf{X}_j & \text{Linear} \\ (\gamma \mathbf{X}_i \cdot \mathbf{X}_j + C)^d & \text{Polynomial} \\ \exp(-\gamma \mid \mathbf{X}_i - \mathbf{X}_j \mid^2) & \text{RBF} \\ \tanh(\gamma \mathbf{X}_i \cdot \mathbf{X}_j + C) & \text{Sigmoid} \end{cases}$$

Se usa en redes neuronales

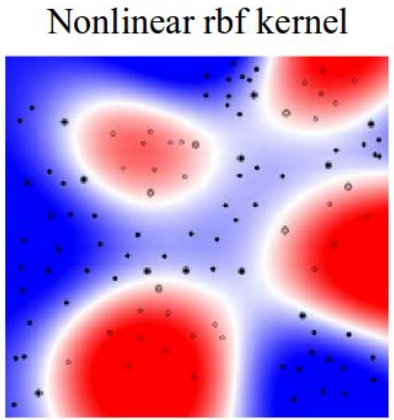


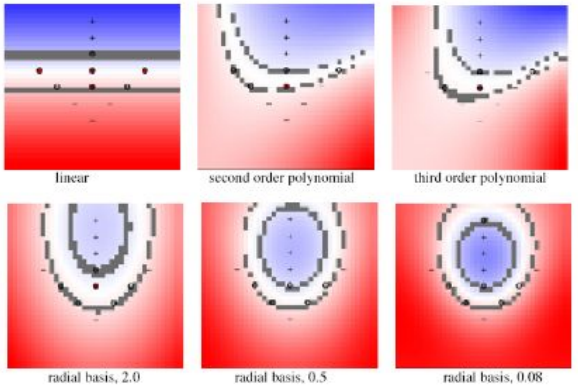
- La función kernel K permite calcular rápidamente los productos punto en el espacio transformado

Máquinas de soporte vectorial

Funciones Kernel

- Los Kernels generalizan la noción de similaridad del producto punto
-

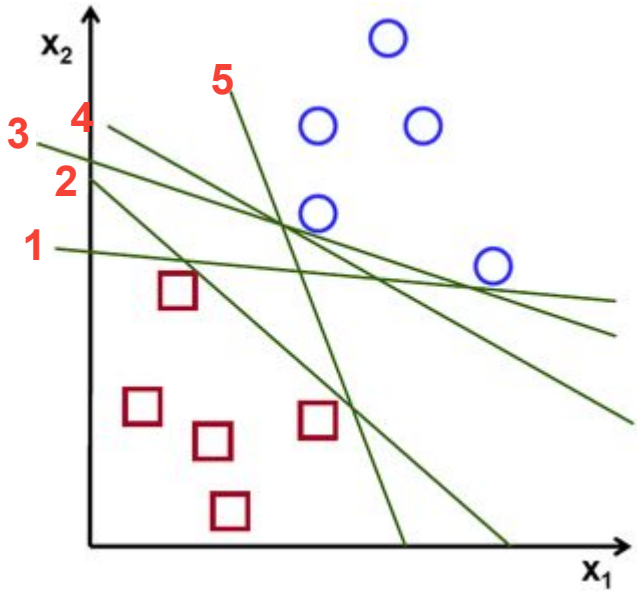




Redes neuronales

Preliminares

- Vimos que las máquinas de soporte vectorial generan un único plano:

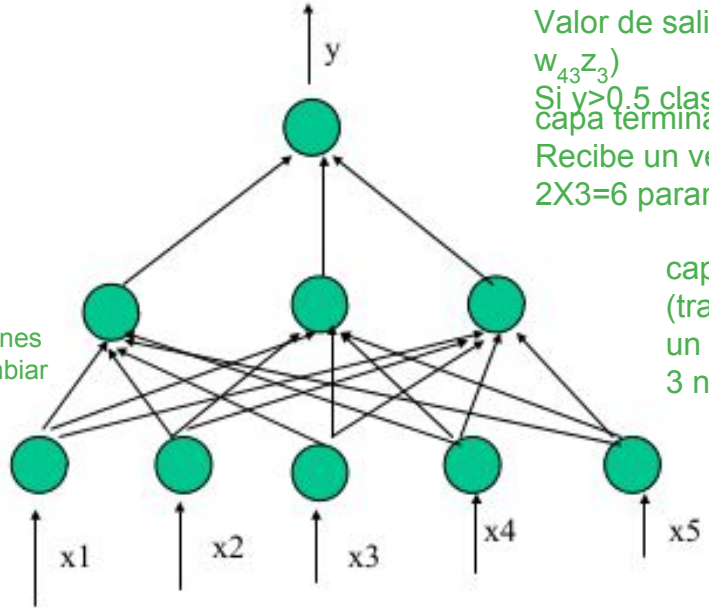


¡Redes neuronales pueden generar mulítiples planos!

Preliminares

- Las redes neuronales datan de antes de los 50's
- Hacen un uso intensivo de la computadora: el problema siempre ha sido entrenarlas
- Su justificación proviene de que, bajo algunos supuestos, pueden aproximar razonablemente bien una función continua (Cybenko)
- El problema es el aprendizaje de los parámetros (no siempre es posible)
- Se dejaron un poco en el olvido y resurgieron con deep learning cuando las computadoras incrementaron su poder

Red neuronal de 1 capa intermedia



Las conexiones pueden cambiar

Valor de salida $y=g(w_{40}+w_{41}z_1+w_{42}z_2+w_{43}z_3)$

Si y>0.5 clasifico como 1, e.o.c. como 0

capa terminal (1 neurona)

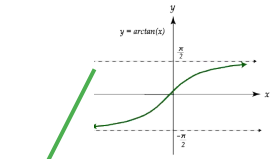
Recibe un vector $z=(z_1,z_2,z_3)$

2X3=6 parametros

capa intermedia: z=g(w'x)
(transforma los datos y añade un sesgo)
3 neuronas, 6X3 parámetros

inputs: 5 neuronas

g es no lineal y suave:



f(t)= arctan(t)

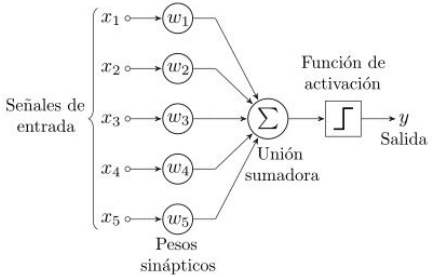
$$P(t) = \frac{1}{1+e^{-t}}$$

¡La cantidad de parámetros crece considerablemente!

¡Necesitamos muchos datos para entrenarlas!

Regresión logística: Perceptrón de 1 capa intermedia con una neurona

El perceptrón puede usar también el kernel-trick



Redes neuronales

- ¿Todo esto para qué? ¡Queremos minimizar el error!

$$E = \sum_{i=1}^{n} (c_i - y_i)^2 = \sum_{i=1}^{n} (c_i - f(x, w))^2$$

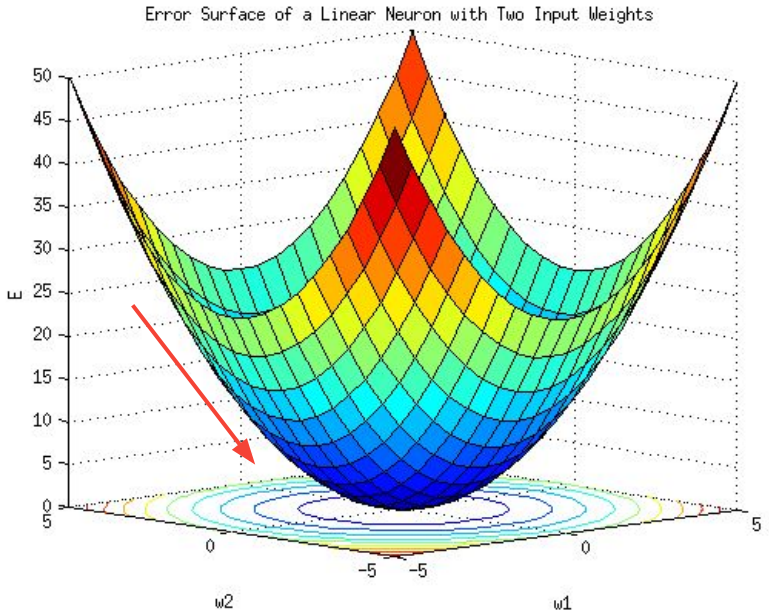
c_i la etiqueta correcta

y_i la etiqueta que predecimos

Los parámetros son los pesos *w*

- Dificultad: es una función no lineal (no como en regresión!)
- Mencionamos que el problema de las redes es entrenarlas
- Generalmente se minimiza usando algún algoritmo de descenso en gradiente $w_{ij} \rightarrow w_{ij} + \eta \frac{\partial E}{w_{ij}}$

Descenso en gradiente

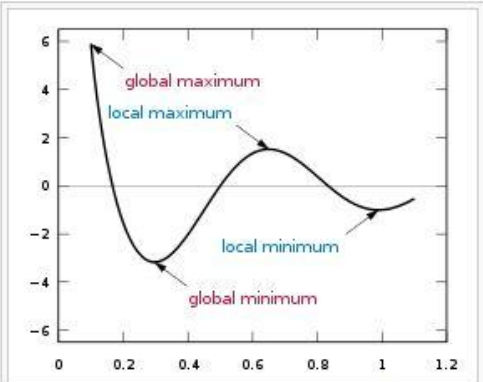


$$w_{ij} \rightarrow w_{ij} + \eta \frac{\partial E}{w_{ij}}$$

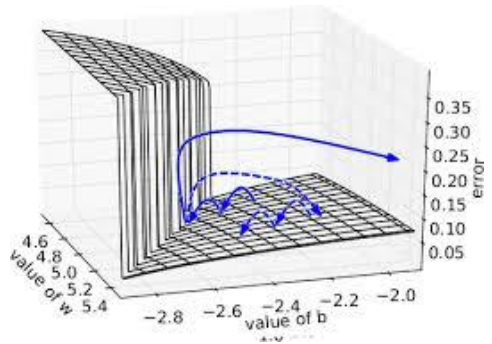
- Es un método de optimización
- No es trivial obtener el gradiente del error
- Para obtener el gradiente del error, se usa un método llamado **propagación hacia atrás de los errores (backpropagation)**
- **Es muy difícil**

Redes neuronales

- Además no es trivial diseñar bien la estructura de la red neuronal: ¿cuántas neuronas?, ¿cuántas capas?
- Problemas de convergencia de los algoritmos de estimación (mínimos locales o vanishing gradient)



Local and global maxima and minima for cos(3π*x*)/*x*, 0.1≤*x*≤1.1



Gracias