# SVM:
# Máquinas de Vectores Soporte

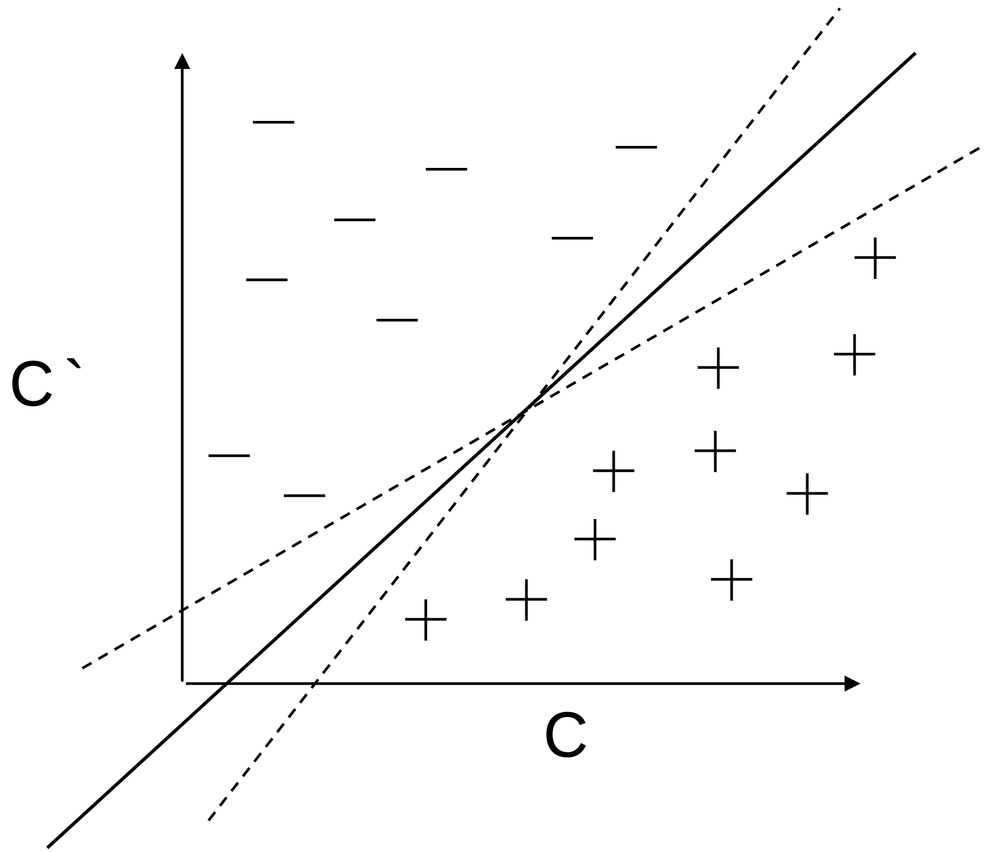
Carlos Alonso González
Grupo de Sistemas Inteligentes
Departamento de Informática
Universidad de Valladolid

# Contenido

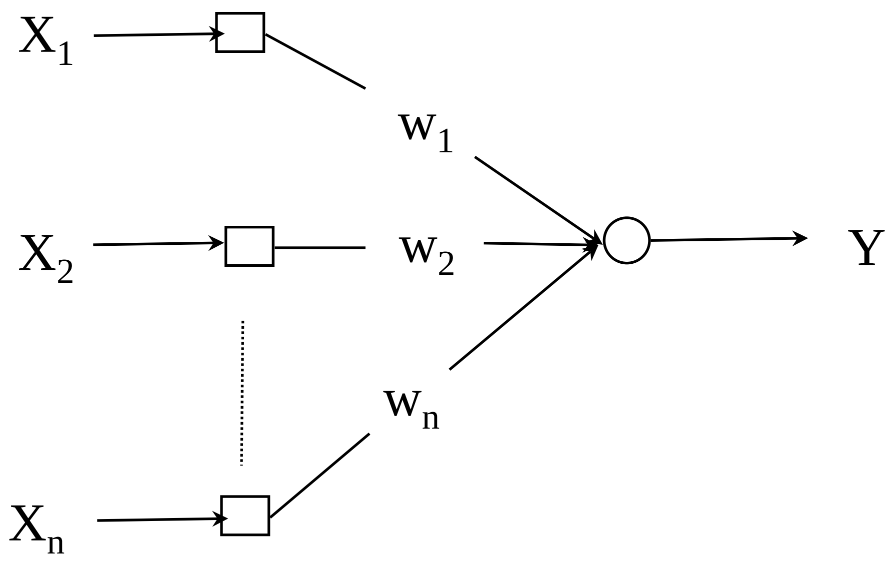
# Clasificación lineal con modelos lineales

- Si las clases son linealmente separables, basta con encontrar hiperplano que discrimine (por ejemplo perceptrón)
- También pueden utilizarse métodos de regresión lineal
- Adecuado para atributos numéricos

# Problema linealmente separable



# Perceptrón discreto




- $Y = sgn(\Sigma_{i=0}\ x_i\ .\ w_i) = \underline{X}.\underline{W}$

# Teorema de convergencia del perceptrón

- Rossemblat, 62, Nilson, 65

- Si el conjunto de patrones de entrenamiento es linealmente separable, la regla de aprendizaje del perceptrón converge a un vector de pesos que clasifica correctamente todos los ejemplos de entrenamiento

# 2. Regresión lineal

- Obtener la clase como una combinación lineal de atributos:

  $x = w_0 + w_1 a_1 + w_2 a_2 + \ldots + w_k a_k$

- Valor predicho para instancia i-esima

  $w_0 a_0^{(i)} + w_1 a_1^{(i)} + w_2 a_2^{(i)} + \ldots + w_k a_k^{(i)} = \Sigma_{j=0,k} w_j a_j^{(i)}$

  (añadiendo atributo $a_0^{(i)}=1$)

# Minimizar error cuadrático

- Ajustar coeficientes (*k+1* pesos) para minimizar error cuadrático sobre *n* ejemplos de entrenamiento
- Error cuadrático

$$\Sigma_{i=1,n} (x^{(i)} - \Sigma_{j=0,k} w_j a_j^{(i)})^2$$

$x^{(i)}$:clase ejemplo i-ésimo, $\Sigma_{j=0,k} w_j a_j^{(i)}$: clase predicha ejemplo i-esimo

- Mínimos cuadrados
  - Más ejemplos que atributos
  - Matriz no singular

# 3. Clasificación lineal: regresión lineal multirespuesta

- Cualquier técnica de regresión se puede utilizar para clasificación:
  - Entrenamiento: regresión para cada clase, con clase=1 para los ejemplos de la clase y clase=0 para los restantes
  - Prueba: calcular el valor de cada regresión y asignar la clase correspondiente a la regresión con máximo valor.
- Para regresión lineal: Regresión lineal multirespuesta

# Comportamiento Regresión lineal multirespuesta

- Interpretación:
  - **rlm** aproxima una función de pertenencia para cada clase (1 si la instancia pertenece a la clase, 0 en otro caso)
  - Para clasificar una instancia no vista, se calcula la función de pertenencia de cada clase y se selecciona la mayor
- Buen comportamiento si clases linealmente separables
- Interesante como elemento básico de otros métodos

# Problemas
# Regresión lineal multirespuesta

- Los valores de la función de pertenencia no se puede asimilar a probabilidades:
  - Fuera del rango [0,1]
- Mínimos cuadrado asume que los errores están normalmente distribuidos
  - Está suposición se viola en problemas de clasificación: los valores de la clases en los ejemplos de entrenamiento son 0 y 1.
- Solución: regresión logística

# 4. Regresión logística

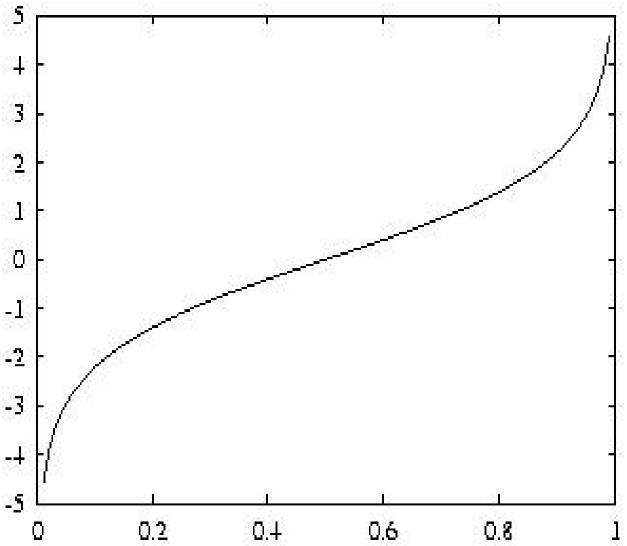
- A pesar del nombre, es un clasificador
- Realiza una transformación de variable y luego construye un clasificador lineal
- Caso binario
- Reemplaza:

  $$Pr[1/a_1, a_2, \ldots, a_n]$$

- Por:

  $$\log[Pr[1/a_1, a_2, \ldots, a_n]/(1-Pr[1/a_1, a_2, \ldots, a_n])]$$

# Transformación logística (*logit*)

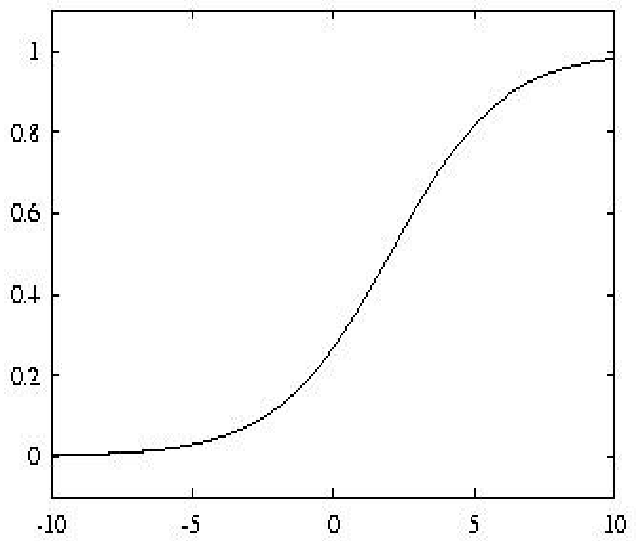



Aplica [0, 1] en $(-\infty, +\infty)$

- Modelo

$$\Pr[1/a_1, a_2, \ldots, a_n] = 1/(1 + e^{(-w_0 - w_1 a_1 - w_2 a_2 - \ldots - w_k a_k)})$$

# Ejemplo modelo regresión logística




Modelo con $w_0=0.5$, $w_1=1$

- Los parámetros se obtienen del conjunto de entrenamiento utilizando máxima verosimilitud (equivalente a mínimos cuadrados si distribución normal)

# Máxima verosimilitud (*Maximun likelihood*)

- Objetivo: maximizar la probabilidad de los datos de entrenamiento respecto a los parámetros
- Regresión logística maximiza *log-likelihood*

$$\Sigma_{i=1,n} \{(1-x^{(i)})\log(1-\Pr[1/a_1^{(i)}, a_2^{(i)},\ldots, a_n^{(i)}]) + x^{(i)}\log(\Pr[1/a_1^{(i)}, a_2^{(i)},\ldots, a_n^{(i)}])\}$$

Con $x^{(i)}$ la clase de los ejemplos de entrenamiento (0 ó 1)

- Obtener los $w_i$ que maximizan log-likelihood
  - Método sencillo: iteración mínimos cuadrados ponderados

# Regresión logística y problemas multiclase

- Regresión logísticas independiente para cada clase (similar a regresión lineal multirespuesta)
  - Problema: las estimaciones de probabilidades no suman 1
  - Solución: modelos acoplados maximizando *log-likelihood* para todas las clases
- Alternativa: clasificadores 2 a 2 (*pairwise classification*)

# Clasificadores 2 a 2

- Hay que construir un clasificador para cada par de clases (*pairwise classification*) usando sólo las instancias de las dos clases (clasificador binario)
- Si $k$ clases $k(k-1)/2$ clasificadores binarios (pero con menos datos)
- Clasificación final: voto
  - Se puede adaptar para obtener probabilidades

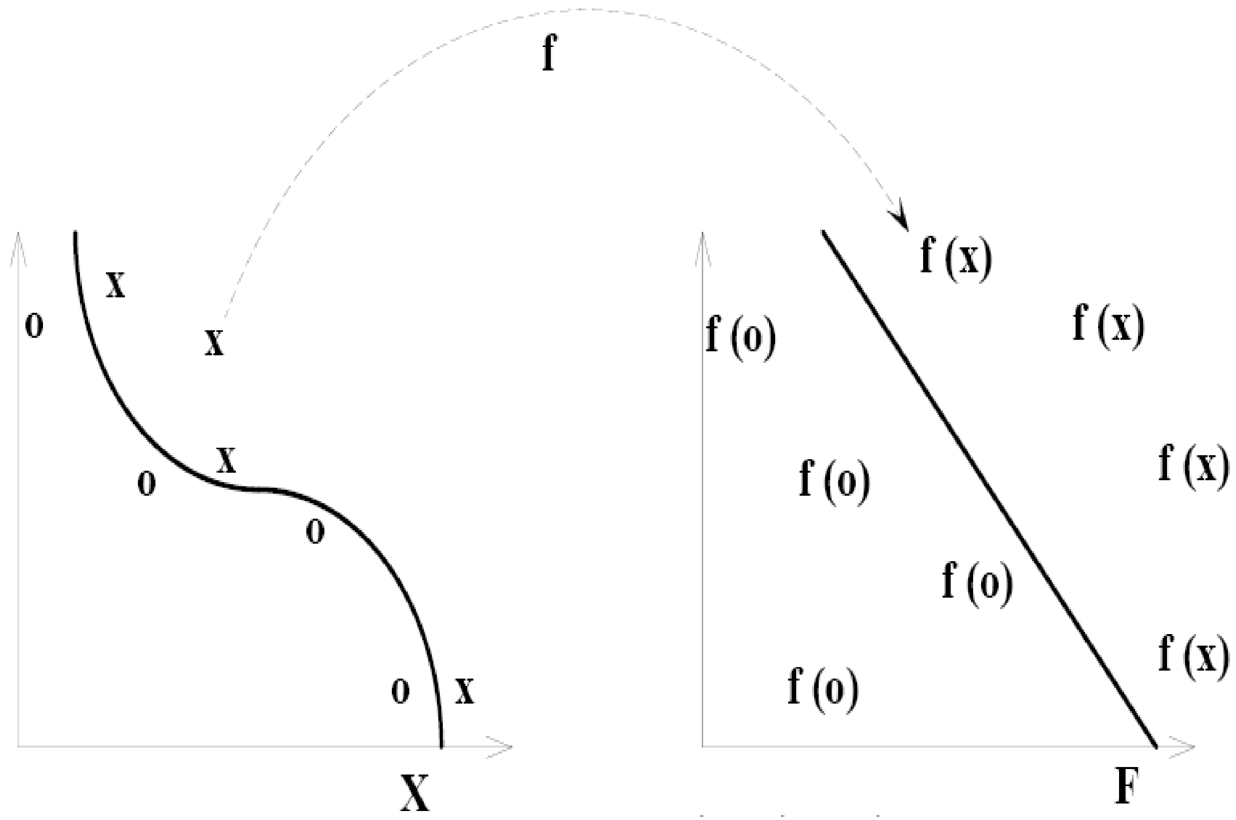
- Si las distribución de clases es uniforme, al menos tan rápido como cualquier otro método multiclase
  - Si algoritmo lineal con nº instancias:
    - $k(k-1)/2 * 2n/k = (k-1)n$
  - Más ventajoso aún si algoritmo más que lineal

# 5. Clasificación no lineal con modelos lineales

- Los modelos lineales solo pueden obtener fronteras de decisión lineales

- ¿Cómo usarlos para problemas no linealmente separables?
  - Realizar una transformación no lineal del espacio de entrada
  - Construir un clasificador lineal en el espacio transformado
  - **Una frontera lineal en el espacio transformado puede representar una frontera no lineal en el espacio original**

# Transformación no lineal del espacio de entrada

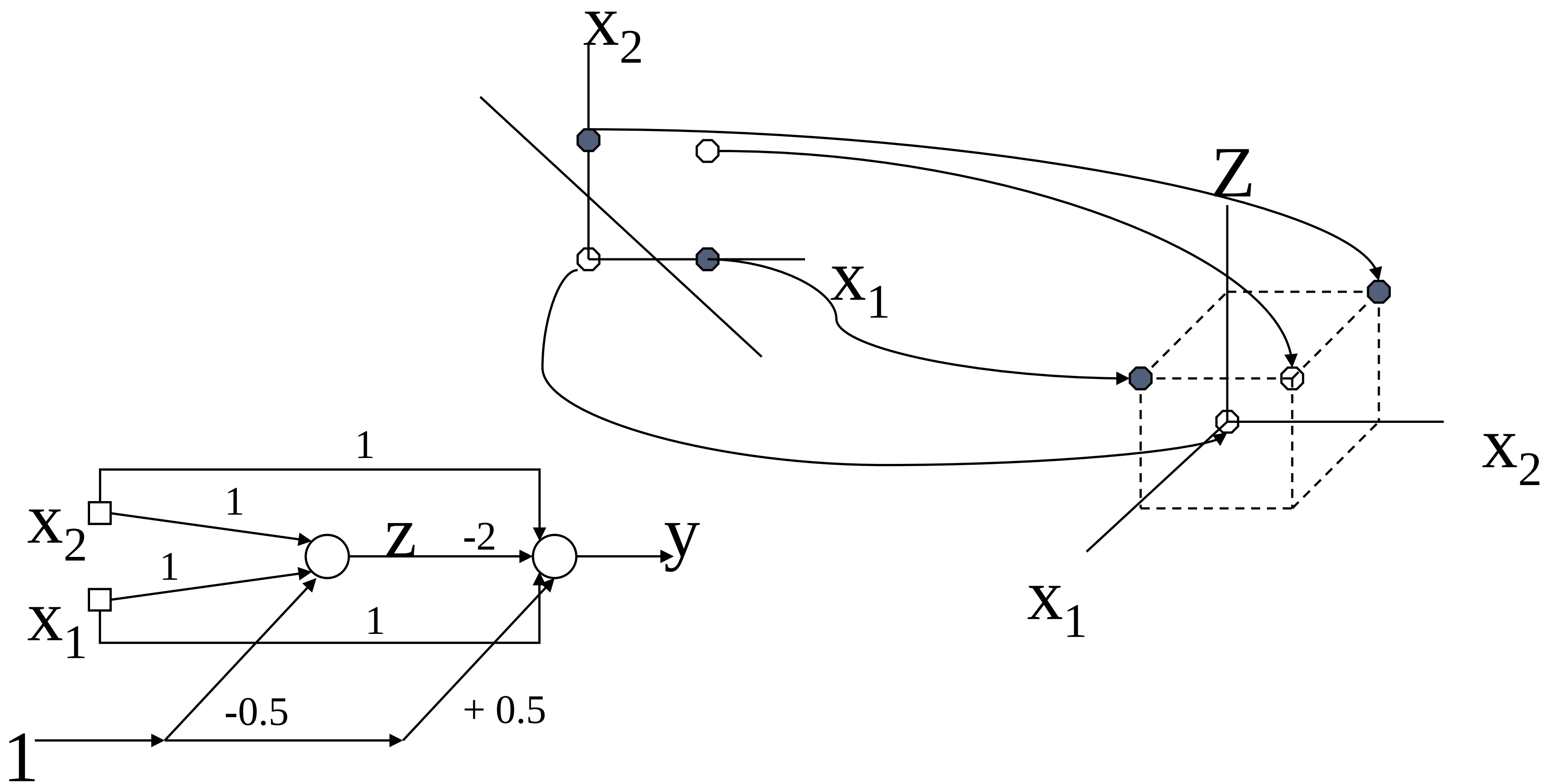



Espacio de entrada: no linealmente separables

Espacio transformado: linealmente separables

[N. Cristianini, ICML01]

# Transformación no lineal aumentando dimensionalidad




Perceptrón unipolar discreto

# Ejemplo transformación no lineal directa

- Transformar el espacio de entrada reemplazando los atributos originales por todos los productos de *n* factores
- Para 2 atributos y 3 factores:

$$x = w_0 + w_1 a_1^3 + w_2 a_1^2 a_2 + w_3 a_1 a_2^2 + w_4 a_2^3$$

- Requiere:
  - Entrenamiento: transformar ejemplos de entrenamiento y crear modelo lineal
  - Prueba: transformar nueva instancia, aplicar modelo lineal

# Problemas transformación no lineal directa

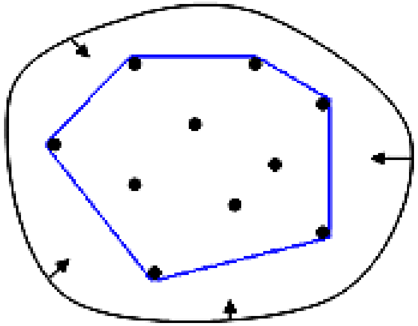
- Complejidad computacional
  - 10 atributos, $n=5$: 2000 coeficientes
  - Impracticable incluso para regresión lineal (cúbica en el número de atributos)
- Sobreajuste
  - Siempre que el número de parámetros es comparable al número de ejemplos

# 6. Máquinas de vectores soporte

- SVM es un algoritmo para encontrar clasificadores lineales en espacios transformados
- Resistentes al sobreajuste porque buscan una frontera de decisión específica:
  - Hiperplano de margen máximo
- Eficiente en el caso no lineal
  - No crean explícitamente el espacio transformado
  - La trasformación no lineal es implícita

# Envolvente convexa

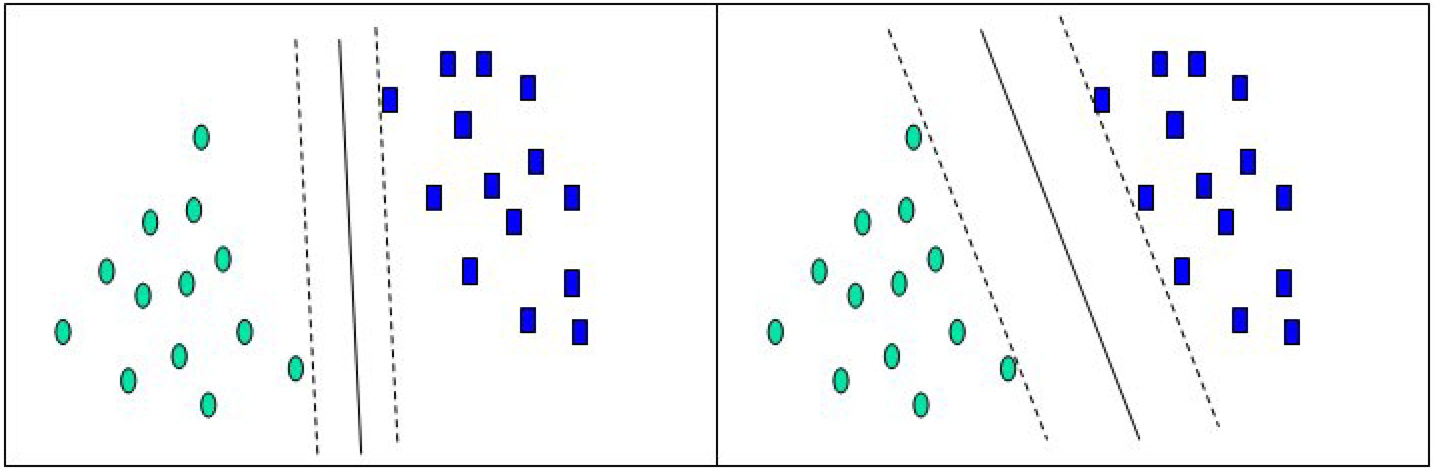
- Dado un conjunto de puntos, su envolvente convexa es el conjunto convexo minimal que contiene al conjunto
  - Símil bidimensional: banda elástica que rodea un conjunto de puntos (polígono)



# Envolvente convexa de dos clases linealmente separables

- La intersección de las envolventes convexas de dos clases linealmente separables es vacía
  - Las envolventes convexas no se solapan

# Hiperplano de margen máximo

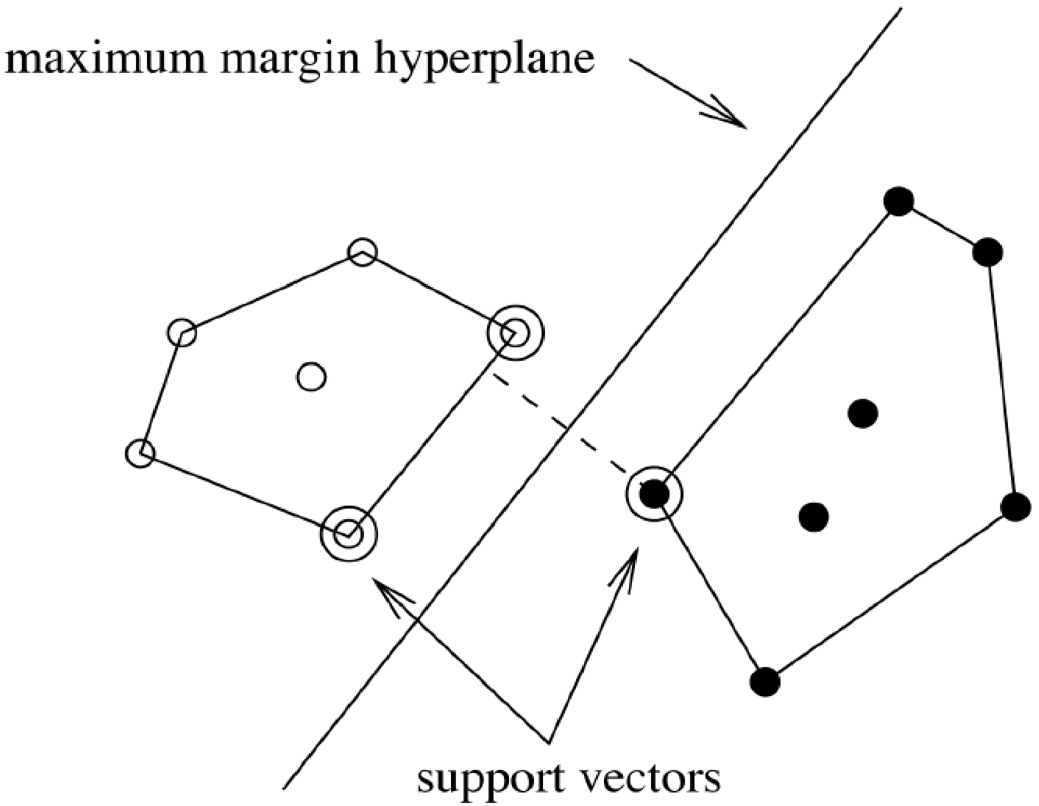


[WWW.dtreeg.com]

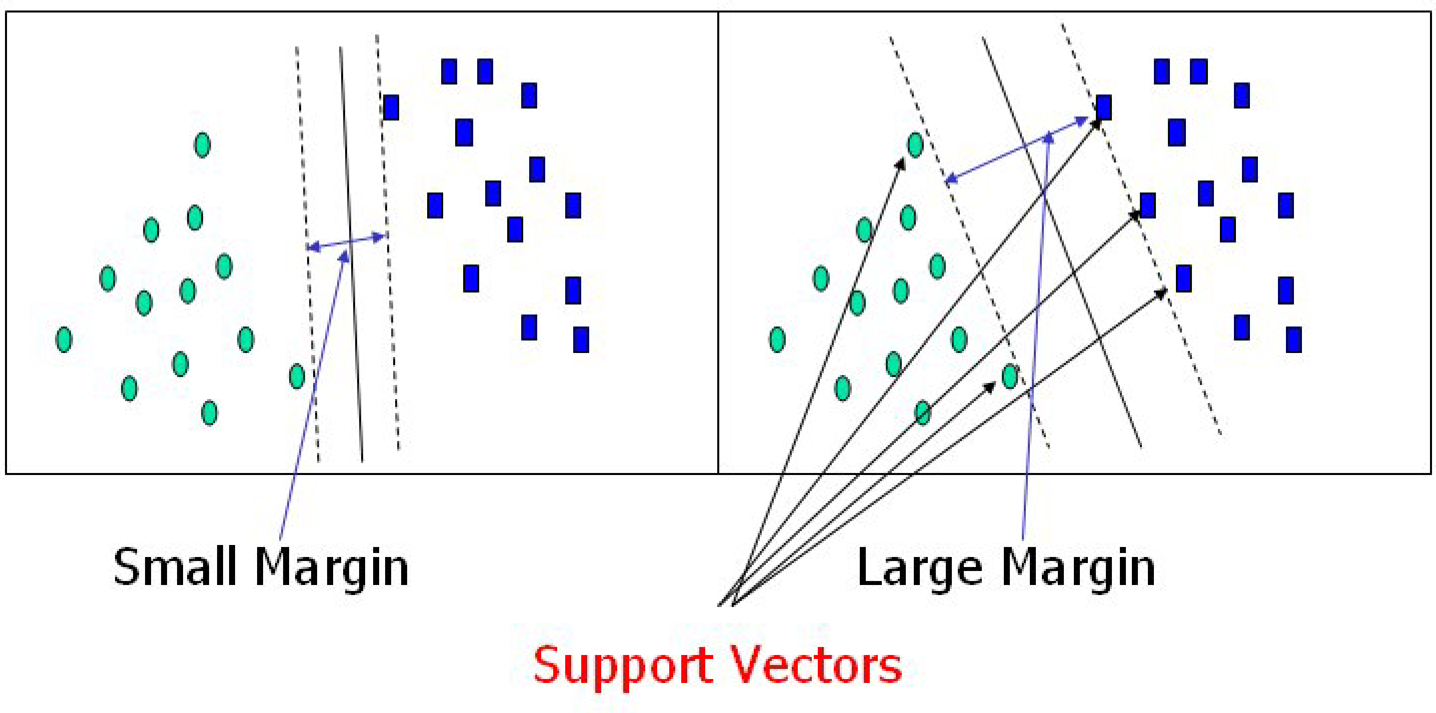
# Hiperplano de margen máximo

- Hiperplano que proporciona la máxima separación entre dos clases linealmente separables
  - Calcular la envolvente convexa de cada clase
  - Determinar el segmento mas corto que une ambas envolventes
  - Hiperplano de margen máximo : perpendicular al segmento anterior, cortándolo en su punto medio

# Hiperplano de margen máximo



# Hiperplano de margen máximo

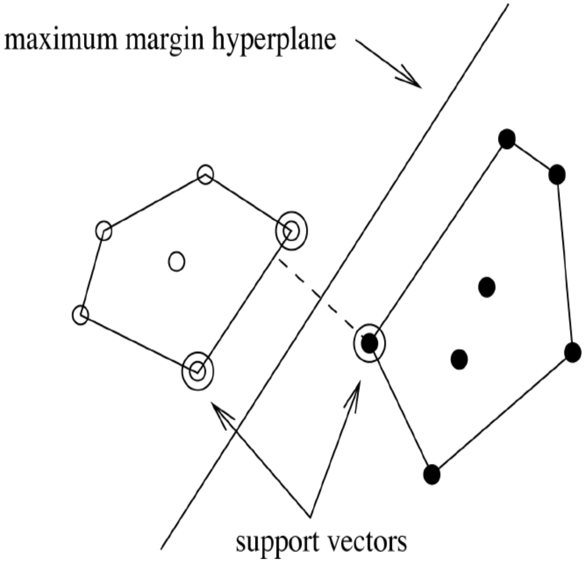


# Vectores Soporte

- Instancias más próximas, de cada clase, al hiperplano de margen máximo.
  - Al menos uno por clase
  - Posiblemente más

- El conjunto de Vectores Soporte define de forma única el hiperplano de margen máximo
  - Las restantes instancias son irrelevantes

# Hiperplano de margen máximo y vectores soporte (I)




- Ecuación de un hiperplano

  $x = w_0 + w_1 a_1 + w_2 a_2 + \ldots + w_k a_k = \underline{w} \cdot \underline{a}$

- Ecuación hiperplano margen máximo en términos de los vectores soporte (formulación dual)

  $x = b + \Sigma_{i \text{ vector soporte}} \alpha_i y_i \underline{a}(i) \cdot \underline{a}$

# Hiperplano de margen máximo y vectores soporte (II)

$x = b + \Sigma_{i\ \text{vector soporte}}\ \alpha_i y_i \underline{a}(i) \cdot \underline{a}$

Con:

- $b, \alpha_i$, parámetros numéricos a determinar
- $\underline{a}$, instancia (a clasificar)
- $\underline{a}(i)$, vector soporte i-esimo
- $y_i$, clase de $\underline{a}(i)$ (que también es una instancia, de entrenamiento), con valores +1, -1

- Obtención de $b, \alpha_i$:
  - Problema de optimización cuadrática con restricciones
    - No hay mínimos locales
    - Herramientas optimización
    - Algoritmos específicos para entrenamiento SVM más eficientes
- Suponiendo clases linealmente separables

# SVM y sobreajuste

- Resistente al sobreajuste
- El sobreajuste esta relacionado con la flexibilidad con que se ajustan las fronteras de decisión (mayor con el número de parámetros)
  - Añadir o eliminar un pequeño número de instancias puede, con otras técnicas de ajuste, modificar drásticamente las fronteras de decisión
- EL hiperplano de margen máximo es relativamente estable: solo depende de los vectores soporte
  - Añadir o eliminar instancias sólo afecta si son vectores soporte

# SVM y clases no linealmente separables

- Realizar una transformación no lineal del espacio de entrada
  - Sobreajuste: menos problemático, pues los vectores soporte son relativamente estables
  - Complejidad: sigue siendo un problema si se transforman las instancias y se busca el hiperplano de máximo margen en el espacio transformado
  - **Solución: buscar vectores soporte en espacio original y no realizar de forma explicita la transformación**

# Ejemplo transformación no lineal implícita

$x = b + \Sigma_{i\ \text{vector soporte}}\ \alpha_i y_i(\underline{a}(i)\cdot\underline{a})^n$

- Similar a producto de *n* factores (= salvo coeficientes constantes)
- El **producto escalar se realiza en el espacio original**
  - No se generan nuevos atributos (no aumenta la dimensión)
  - Sin embargo, el efecto es el mismo que si hubiésemos realizado la transformación y luego realizado el producto $\underline{a}(i)^n\cdot\underline{a}^n$
  - Los ejemplos de entrenamiento (incluidos vectores soporte) y los ejemplos de prueba permanecen en el espacio original (de menor dimensión)

# Kernel

- La aplicación no lineal se denomina *Kernel*
- Kernel polinomial: $K(\underline{x}_i , \underline{x}_j) = (\underline{x}_i \cdot \underline{x}_j)^n$
- Requisito: $K(\underline{x}_i , \underline{x}_j) = \Phi(\underline{x}_i) \cdot \Phi(\underline{x}_j)$

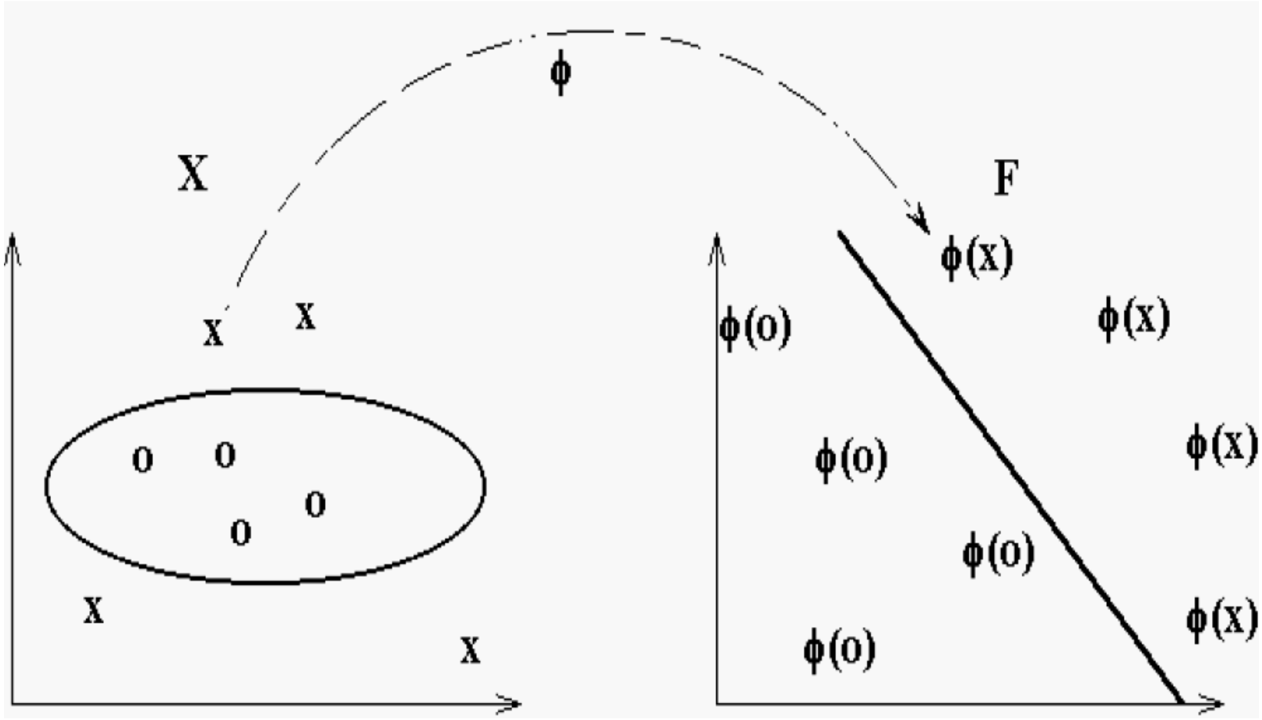
- Kernel gausiano, radial:

  $K(\underline{x}_i , \underline{x}_j) = \exp(- (\underline{x}_i - \underline{x}_j)^2 / 2\sigma^2)$
- Kernel perceptrón:

  $K(\underline{x}_i , \underline{x}_j) = || \underline{x}_i - \underline{x}_j ||$
- Kernel sigmoidal:

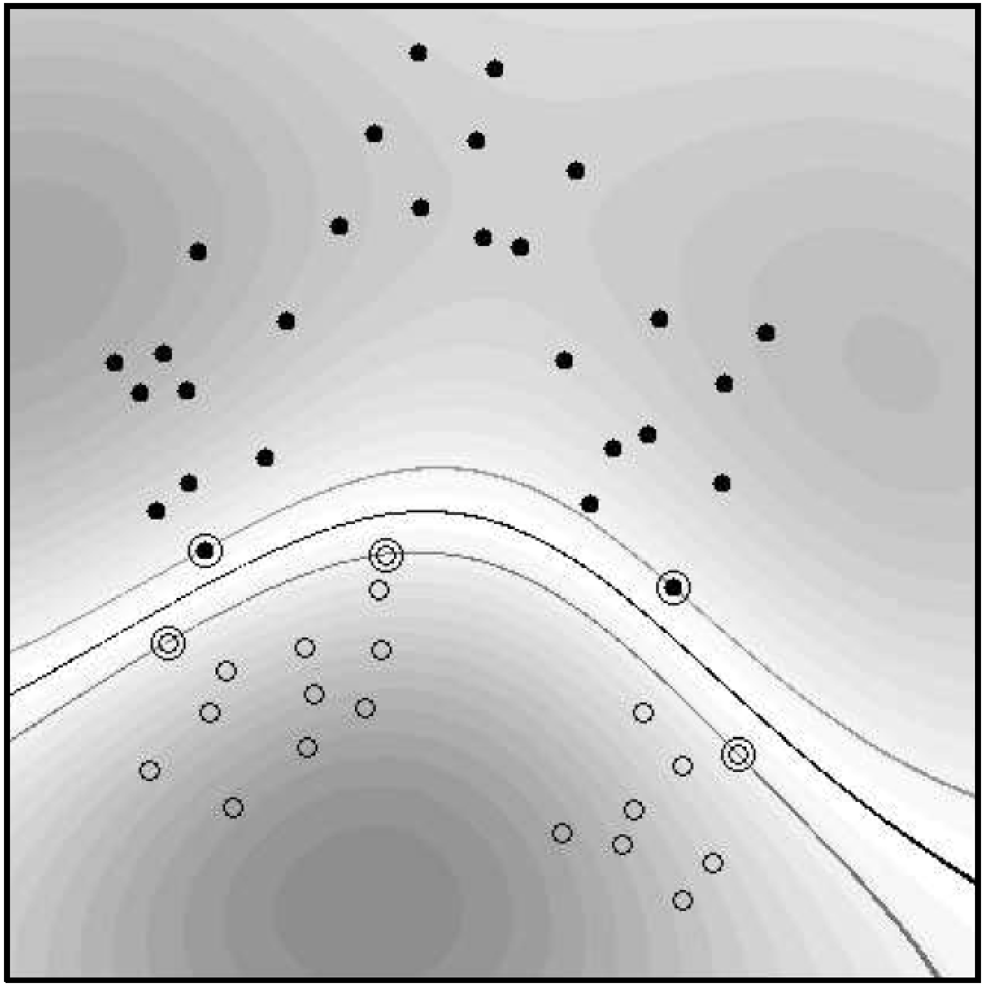
  $K(\underline{x}_i, \underline{x}_j) = \tanh(\underline{x}_i \cdot \underline{x}_j - \theta)$

# Ejemplo de Kernel: polinomial



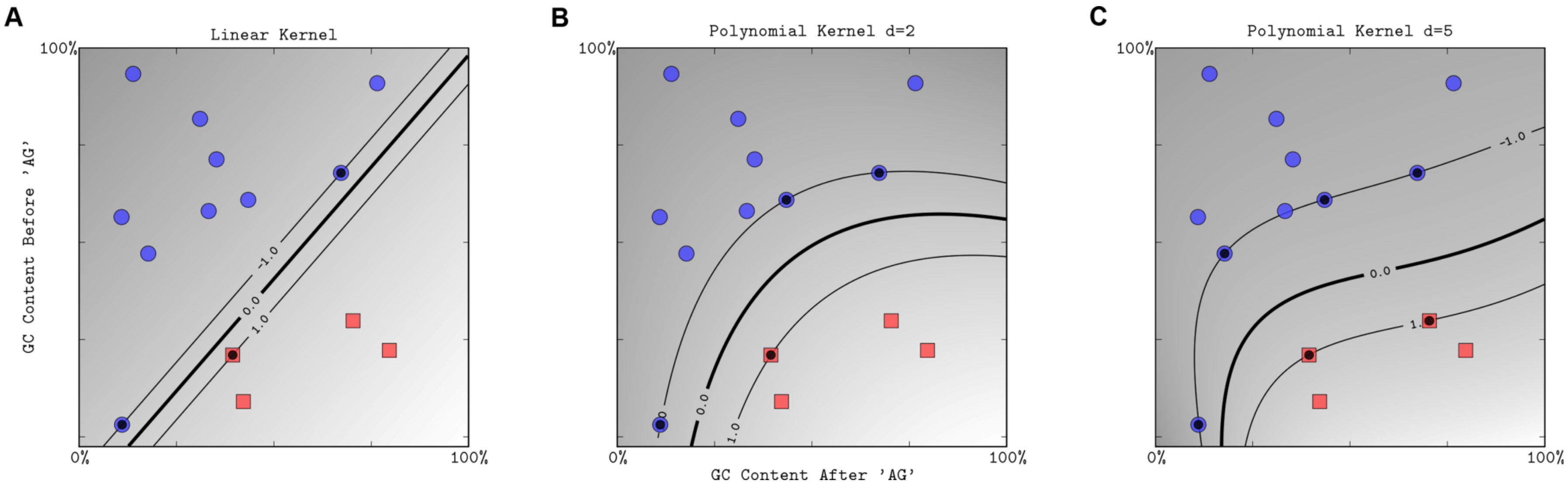

[N. Cristianini, ICML01]

# Ejemplo: Kernel perceptrón



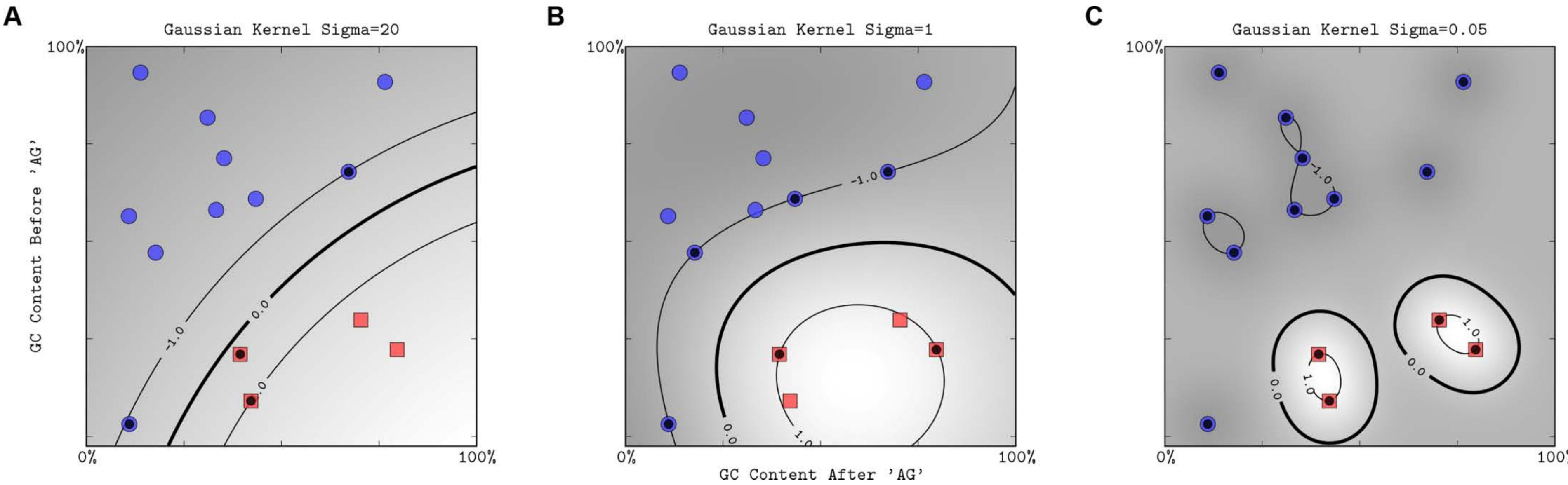
[Schölkopf, NIPS 2001]

# Kernel Polinomial: efecto del grado

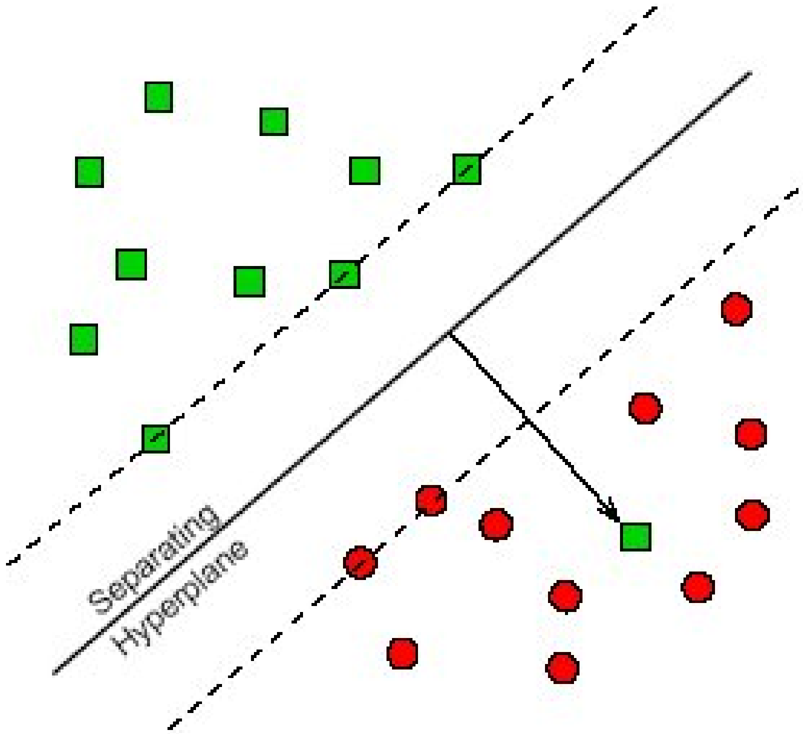



Ben-Hur A, Ong CS, Sonnenburg S, Schölkopf B, Rätsch G, 2008 Support Vector Machines and Kernels for Computational Biology. PLoS Comput Biol 4(10): e1000173. doi:10.1371/journal.pcbi.1000173

# Kernel Gausiano: efecto del radio



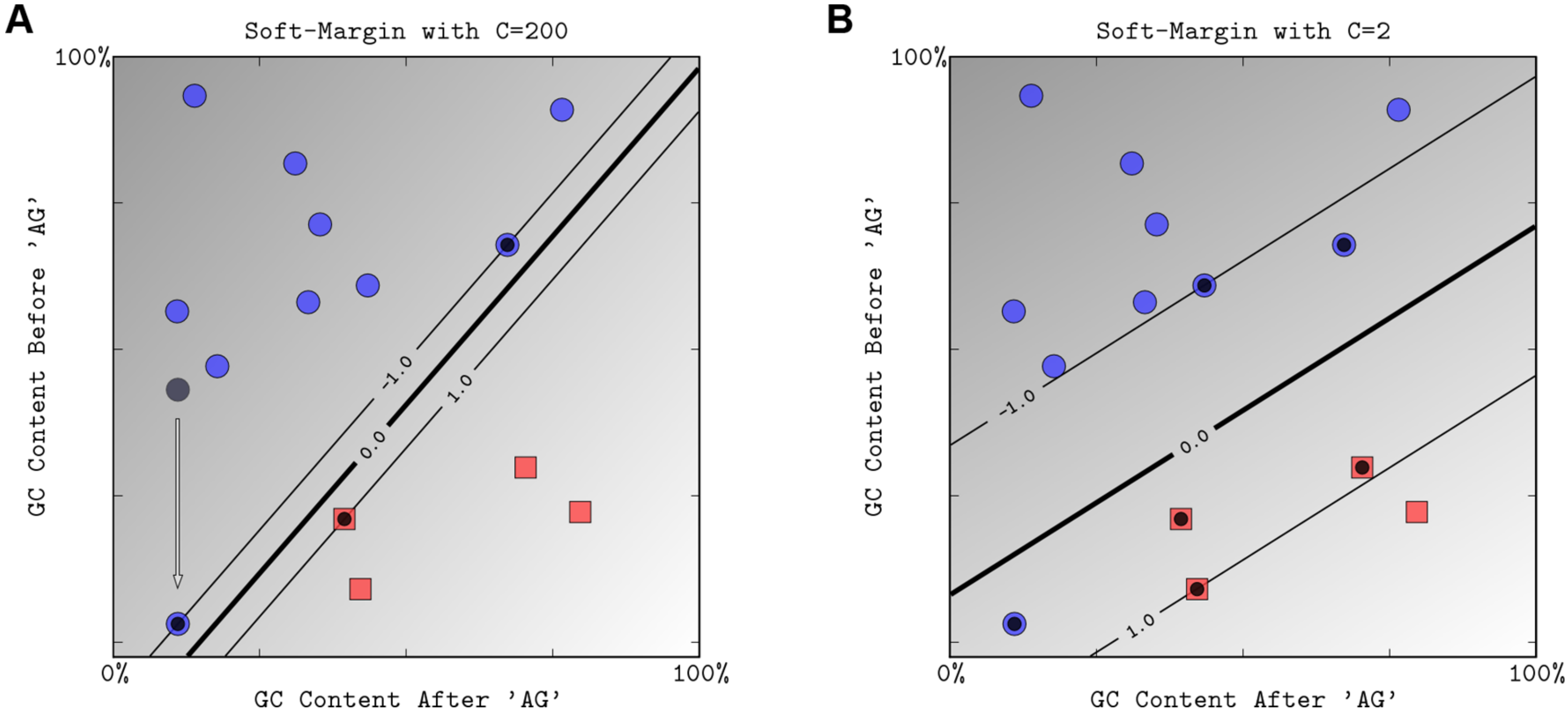
# Ruido (clases no linealmente separables)



# Clases no linealmente separables

- Hemos asumido clases linealmente separables (en espacio original o transformado)
- Si clases no linealmente separables
  - Admitir error, penalizándolo según constante C
  - Introducir parámetro C, que limita el efecto de cualquier instancia de entrenamiento en la superficie de decisión
  - Se traduce en añadir restricciones $0 \leq \alpha_i \leq C$
  - Sigue siendo un problema de optimización cuadrática
  - C se determina experimentalmente

# Efecto de C


A
Soft-Margin with C=200
100%
0%
100%
GC Content After 'AG'
GC Content Before 'AG'
-1.0
0.0
1.0
B
Soft-Margin with C=2
100%
0%
100%
GC Content After 'AG'
GC Content Before 'AG'
-1.0
0.0
1.0

# Datos dispersos (Sparse)

- Los algoritmos para SVM mejoran notablemente su eficiencia si datos dispersos
- Motivo: se calculan productos escalares
- Datos dispersos
  - Cálculo eficiente de productos escalares
  - Iterando sólo sobre los datos no nulos
- SVM pueden procesar datos dispersos con 10.000s de atributos

# Algoritmos: (formulación original, lineal)

- Función discriminante:

  $f(x) = \underline{w}.\underline{x} + b$

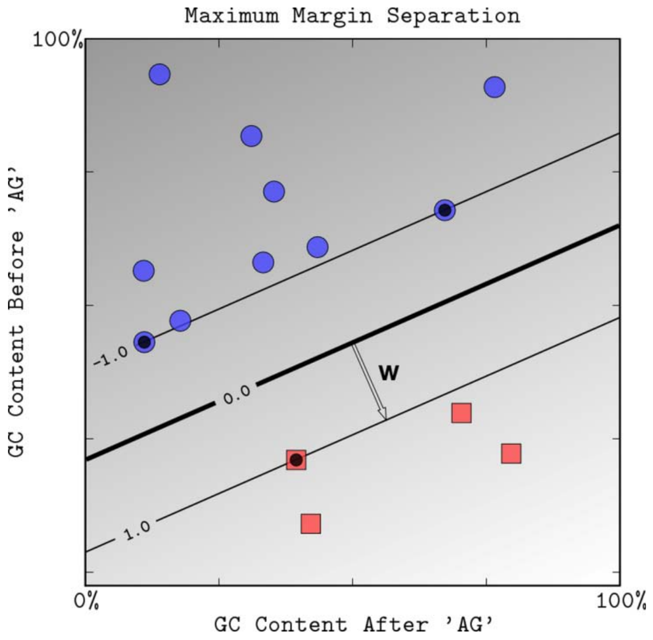
- Margen: $1/||\underline{w}||$

- **Maximizar margen:**

  $\underset{w,b}{\text{Minimizar}}\ \frac{1}{2}\ ||w||^2$

  Sujeto a $y_i(\underline{w}.\underline{x_i} + b) \geq 1$

  Para i= 1, 2,… ,n



# Algoritmos: (formulación original, Kernel, C)

- Introducir variables de holgura ($\xi$ slack) y penalizar con constante C

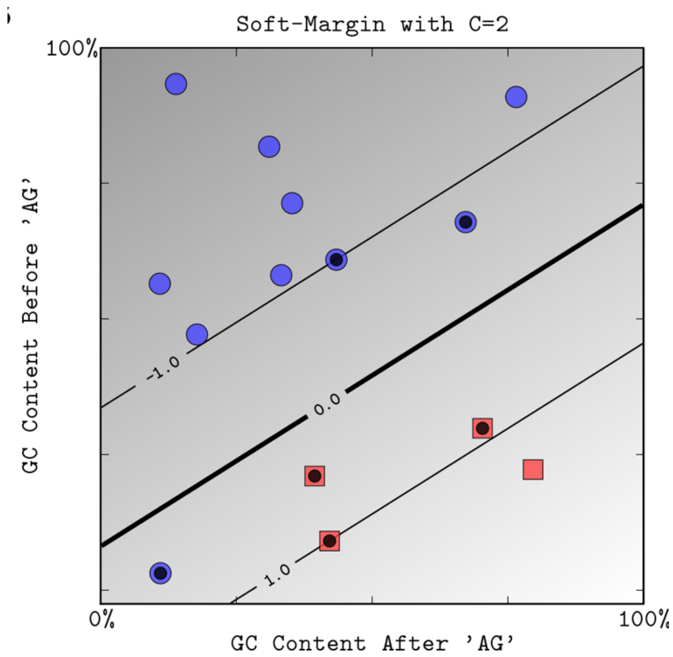
- **Maximizar margen:**

  Minimizar $\frac{1}{2}\ ||w||^2 + C\ \Sigma_{i=1,n}\ \xi_i$

  $w,b,\ \xi$

  Sujeto a $y_i(K(\underline{w},\underline{x_i}) + b) \geq 1 - \xi_i$

  Para i= 1, 2,… ,n



# Algoritmos (formulación dual)

$$\min_{\vec{\alpha}} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j K(\vec{x}_i,\vec{x}_j)\alpha_i\alpha_j - \sum_{i=1}^{N}\alpha_i,$$

$$0 \le \alpha_i \le C, \forall i,$$

$$\sum_{i=1}^{N} y_i\alpha_i = 0.$$

[J. Platt, 1998]

- Problema de Optimización Cuadrática
- Herramientas estándar de optimización cuadrática
  - Poco eficientes
- Memoria: cuadrática con conjunto de entrenamiento
- Tiempo: entre lineal y cúbico

# Algoritmos: SMO

- Sequential Minimal Optimization
- J. Platt, 1998
- Divide el problema de Optimización Cuadrática en subproblemas Pequeños (OCP)
- Los problemas OCP se resuelven analíticamente, evitando las iteraciones internas de los algoritmos de optimización
- Memoria: lineal con conjunto de entrenamiento (frente cuadrático)
- Tiempo: entre lineal y cuadrático
- Muy eficiente si datos dispersos

# Problemas multiclase

- pairwise classification
- Si $k$ clases $k(k-1)/2$ clasificadores binarios
- Clasificación final: voto

# Discusión SVM

- Clasificadores muy precisos en muchos dominios de aplicación
- Resistentes al sobreajuste
- Elevado coste computacional
  - Temporal:
    - polinomial , ¿$n^{2.2}$? (experimental)
    - exponencial?
  - Espacial
    - Al menos ($\beta^2 n^2$), $\beta$ mínimo error alcanzable con el Kernel
    - ¿SMO lineal?
- Mejora notable con datos dispersos (Sparse)

# Aplicaciones

- Reconocimiento facial
- OCR
- Bioinformática
- Minería de texto
- Series temporales

# Referencias


- C. Bishop. Pattern Recognition and Machine Learning. Springer, N.Y., 2005
- N. Cristianini and J. Shawe-Taylor. An introduction to Support Vector Machines and other kernel-based learning methods. Cambridge University Press, 2000.
- B. Schölkopf and A. J. Smola. Learning with kernels. MIT Press, Cambridge, MA, 2002.
- I. H. Witten and E. Frank. Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, 2nd edition, 2005.
- Ben-Hur A, Ong CS, Sonnenburg S, Schölkopf B, Rätsch G, 2008 Support Vector Machines and Kernels for Computational Biology. PLoS Comput Biol 4(10): e1000173. doi:10.1371/journal.pcbi.1000173
- www.kernel-machines.net (tutoriales, bibliografía...)