

Support Vector Machine

- Aprendizaje supervisado
- Clasificadores binarios
- Aspectos teóricos:

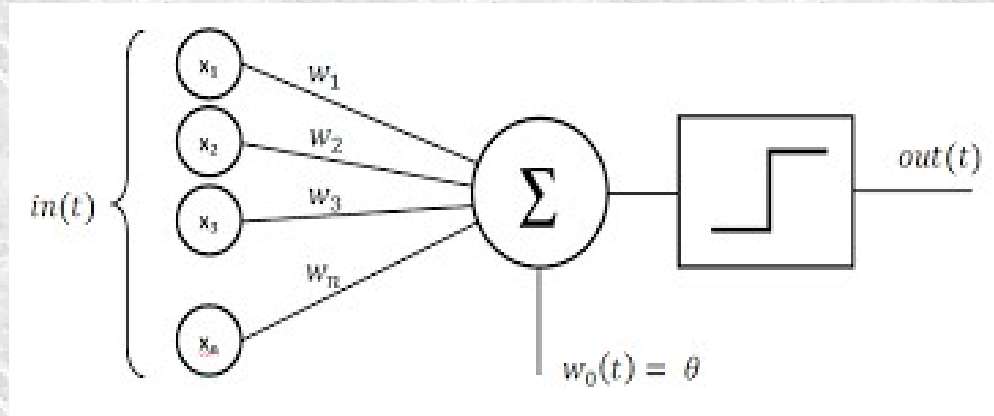
¿Es una red neuronal?

¿Es un sistema lineal o no lineal?

¿Que clase de problemas podemos encarar?

Support Vector Machine

La red neuronal mas simple posible: el *perceptrón*



Support Vector Machine

Si la entrada es el vector \mathbf{x}_i y la salida deseada es y_i :

$$\mathbf{w} \cdot \mathbf{x}_i + b \geq 0 \quad \text{para } y_i = 1$$

$$\mathbf{w} \cdot \mathbf{x}_i + b < 0 \quad \text{para } y_i = -1$$

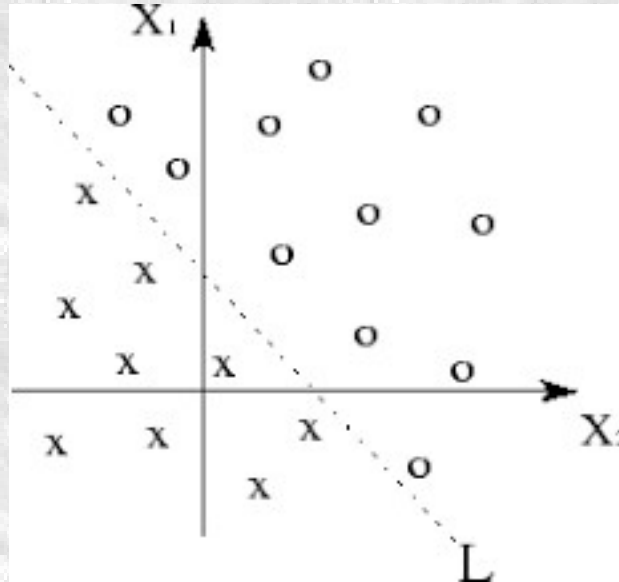
O

$$y_i (\mathbf{w} \cdot \mathbf{x}_i + b) \geq 0$$

Este sistema resuelve problemas de clasificación que son
linealmente separables

Support Vector Machine

Problemas *linealmente separables*

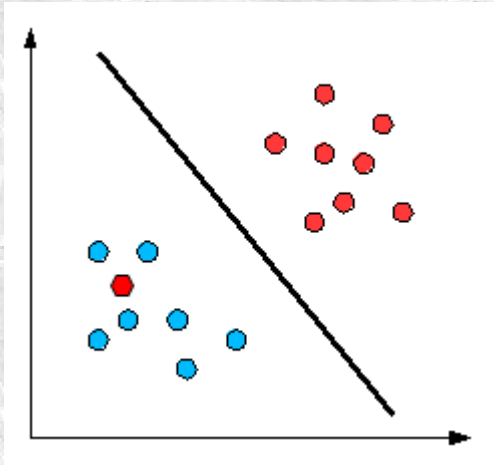


Si el problema es linealmente separable hay un algoritmo (*algoritmo del perceptrón*) que encuentra una solución en tiempo finito

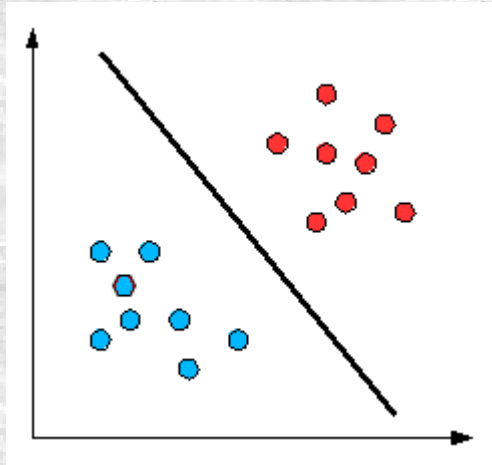
Support Vector Machine

Problemas *linealmente separables*: p entradas en N dimensiones

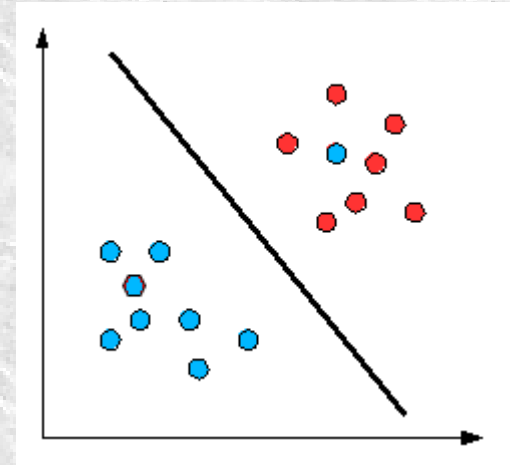
No



Si



No



¿Cual es el número total de problemas que se pueden definir si tengo p entradas?

Support Vector Machine

Problemas *linealmente separables*:

¿Cual es el número total de problemas que se pueden definir si tengo p entradas? $\rightarrow 2^p$

Si estoy p entradas en dimensión N , el número de problemas linealmente separables es denotado por $C(p,N)$

La fracción de problemas linealmente separables es $C(p,N)/2^p$

Esta cantidad se puede calcular con métodos de geometría combinatoria
(para puntos en posición general)

Support Vector Machine

Problemas *linealmente separables*:

Esta cantidad se puede calcular con métodos de geometría combinatoria (Hertz p. 112):

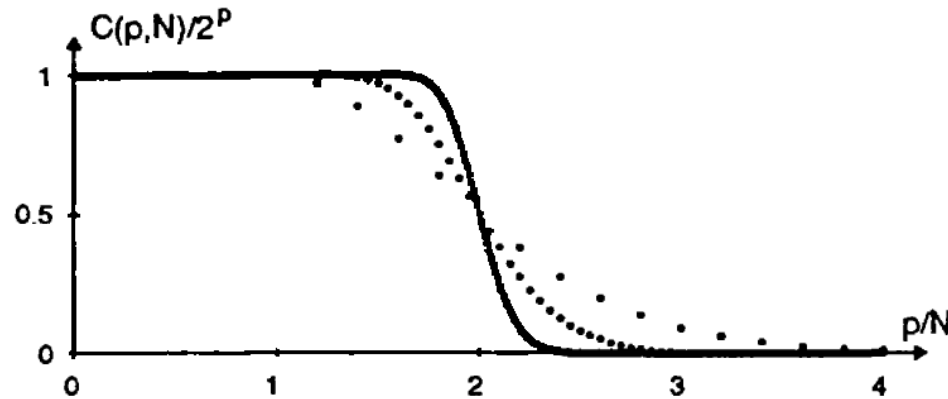


FIGURE 5.11 The function $C(p, N)/2^p$ given by (5.67) plotted versus p/N for $N = 5, 20$, and 100 .

Support Vector Machine

Problemas *linealmente separables*:

Para dimensión grande TODO problema es linealmente separable si $p < 2N$

Si un problema no es linealmente separable se lo puede transformar en uno mapeándolo a dimensión alta

$$\mathbf{x} \rightarrow \varphi(\mathbf{x})$$

$\varphi: \mathbb{R}^N \rightarrow \mathbb{R}^{N'}$ transformación no lineal, $N' > N$ (quizás $N' \gg N$)

Support Vector Machine

Una vez que el problema es *linealmente separable* se puede utilizar algún método de aprendizaje tipo algoritmo del perceptrón

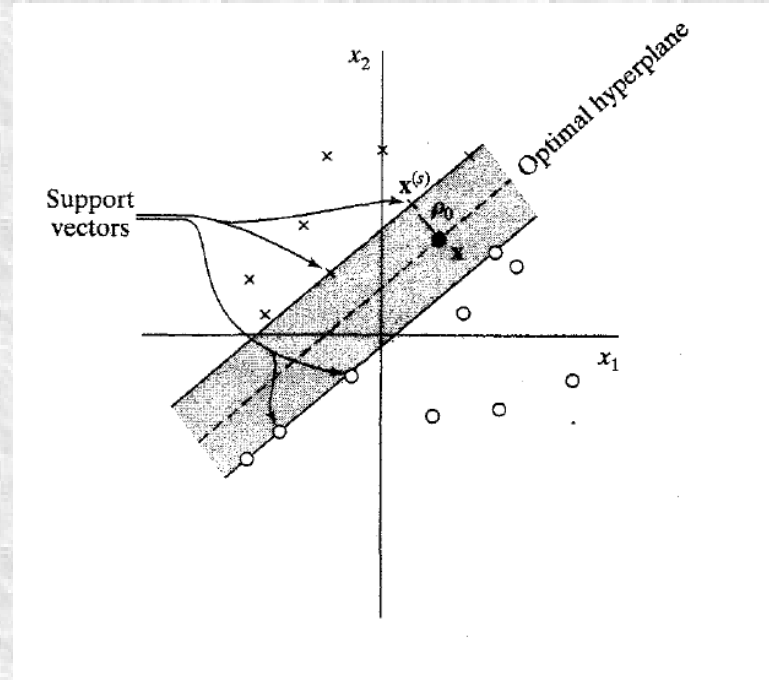
El número de parámetros es tan grande que voy a tener una situación de *overfitting*

Support Vector Machine

La solución no es única

¿Que solución tendrá menor error de generalización?

Es la que tiene el mayor *margen de separación*:



Support Vector Machine

Supongamos que los hiperplanos están definidos por

$$\hat{\mathbf{w}} \cdot \mathbf{x} + \beta = c \quad \hat{\mathbf{w}} \cdot \Delta \mathbf{x} = 2c$$

$$\hat{\mathbf{w}} \cdot \mathbf{x} + \beta = -c$$

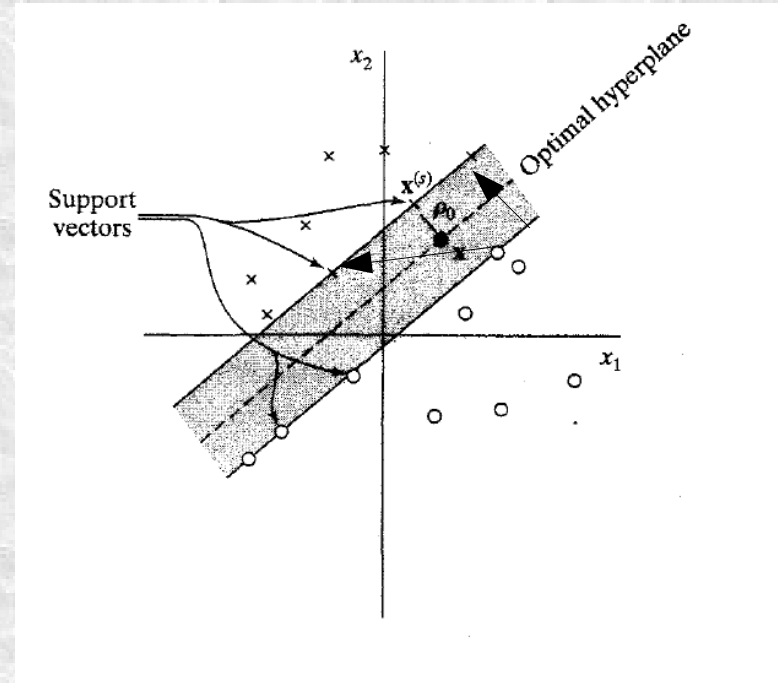
O equivalentemente:

$$\mathbf{w} \cdot \mathbf{x} + b = 1 \quad \mathbf{w} \cdot \Delta \mathbf{x} = 2$$

$$\mathbf{w} \cdot \mathbf{x} + b = -1 \quad \text{donde } \mathbf{w} = \hat{\mathbf{w}}/c, b = \beta/c$$

La distancia entre los hiperplanos es

$$2 \rho_0 = 2/|\mathbf{w}|$$



Support Vector Machine

Es decir que el mejor hiperplano es el que minimiza $|\mathbf{w}|$

En este contexto el clasificador óptimo es que se obtiene de minimizar $|\mathbf{w}|$ (o $|\mathbf{w}|^2$) con los constraints

$$y_i (\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \quad (\text{o } y_i (\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}_i) + b) \geq 1) \quad \text{para } i=1, \dots, p$$

Problema de *programación cuadrática*.

Support Vector Machine

Es decir que el mejor hiperplano es el que minimiza $|\mathbf{w}|$

Esto es compatible con la teoría de generalización

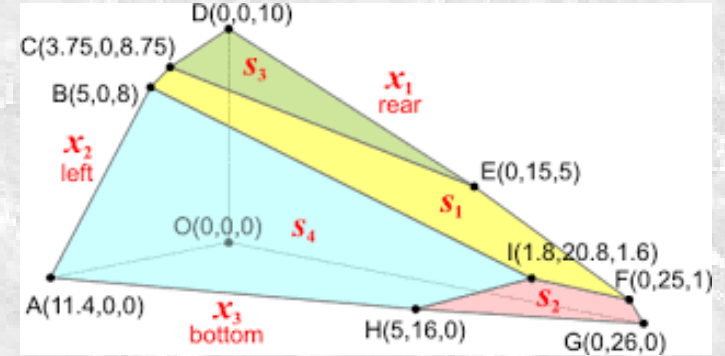
Si $|\mathbf{x}|^2 \leq D$, la dimensión VC está acotada por

$$d_{\text{VC}} \leq \min\{D^2 |\mathbf{w}|^2, N\} + 1$$

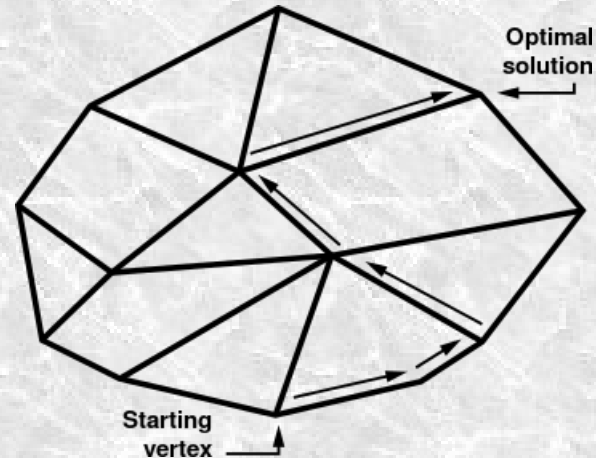
Es decir controlando el tamaño de los pesos controlamos la complejidad de la red

Support Vector Machine

Los constraints definen un poliedro irregular



La solución involucra una búsqueda por las aristas de este objeto



Support Vector Machine

Se puede pasar al problema *dual* introduciendo multiplicadores de Lagrange α_i ($1 \leq i \leq p$) (ver libro Haykin p. 345-346).

El Lagrangiano es

$$L(\mathbf{w}, b, \alpha) = (1/2) \mathbf{w} \mathbf{w}^T - \sum_{1 \leq i \leq p} \alpha_i [y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1]$$

Tomando las derivadas:

$$\partial L / \partial \mathbf{w} = 0 \quad \partial L / \partial b = 0$$

Support Vector Machine

El vector \mathbf{w} puede ser escrito como

$$\mathbf{w} = \sum_{1 \leq i \leq p} \alpha_i y_i \mathbf{x}_i \quad (\text{o} \quad \mathbf{w} = \sum_{1 \leq i \leq p} \alpha_i y_i \boldsymbol{\varphi}(\mathbf{x}_i))$$

Y además

$$\sum_{1 \leq i \leq p} \alpha_i y_i = 0$$

Los α_i maximizan la función

$$Q(\{\alpha_i\}) = \sum_{1 \leq i \leq p} \alpha_i - \frac{1}{2} \sum_{1 \leq i, j \leq p} \alpha_i \alpha_j y_i y_j \boldsymbol{\varphi}(\mathbf{x}_i) \cdot \boldsymbol{\varphi}(\mathbf{x}_j)$$

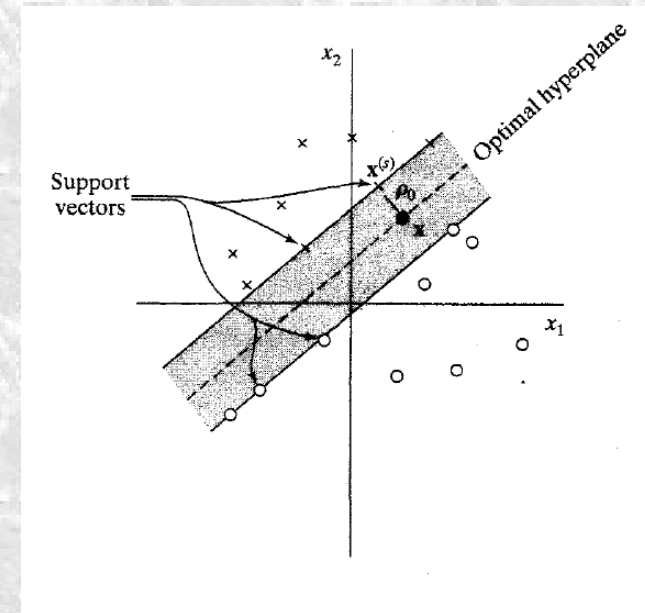
Con los constraints $\alpha_i \geq 0$, $\sum_{1 \leq i \leq p} \alpha_i y_i = 0$ (Haykin, p. 344-346)

Support Vector Machine

Notar que la función $Q(\{\alpha_i\})$ solo depende de los datos de entrenamiento a través de $p \times p$ productos escalares $\mathbf{x}_i \cdot \mathbf{x}_j$ (o $\varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$) ($1 \leq i, j \leq p$)

Se define el *kernel* $k(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$

El vector de peso óptimo es una combinación lineal de los \mathbf{x} para los cuales los α son no nulos
→ vectores de apoyo



Support Vector Machine

$k(\mathbf{X}_i, \mathbf{X}_j)$ es el kernel

Kernels usuales:

- Polynomial (homogeneous): $k(\vec{x}_i, \vec{x}_j) = (\vec{x}_i \cdot \vec{x}_j)^d$.
- Polynomial (inhomogeneous): $k(\vec{x}_i, \vec{x}_j) = (\vec{x}_i \cdot \vec{x}_j + 1)^d$.
- Gaussian radial basis function: $k(\vec{x}_i, \vec{x}_j) = \exp(-\gamma \|\vec{x}_i - \vec{x}_j\|^2)$ for $\gamma > 0$. Sometimes parametrized using $\gamma = 1/(2\sigma^2)$.
- Hyperbolic tangent: $k(\vec{x}_i, \vec{x}_j) = \tanh(\kappa \vec{x}_i \cdot \vec{x}_j + c)$ for some (not every) $\kappa > 0$ and $c < 0$.

Support Vector Machine

Una vez que los coeficientes α_i han sido evaluados la predicción de la red en el punto \mathbf{x} esta dada por

$$\mathbf{w} \cdot \boldsymbol{\varphi}(\mathbf{x}) + b = \sum_{1 \leq i \leq p} \alpha_i y_i \boldsymbol{\varphi}(\mathbf{x}_i) \cdot \boldsymbol{\varphi}(\mathbf{x}) + b = \sum_{1 \leq i \leq p} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b$$

Observar que:

- α_i son no nulos SOLO para los vectores de apoyo: predicción eficiente
- NO** es necesario dar una forma explícita de la transformación no-lineal $\boldsymbol{\varphi}$. La dimensionalidad N' es potencialmente infinita (ver teorema de Mercer, Haykin p. 354)

Support Vector Machine

A veces queremos tolerar tener cierto número de errores, si eso mejora el error de generalización

Podemos buscar cual es la solución que minimiza el número de errores:

Soft-margin SVM linear (o ni linear)

Término de error:

$$E_i = \max(0, 1 - y_i (\mathbf{w} \cdot \mathbf{x}_i + b))$$

$$(o \max(0, 1 - y_i (\mathbf{w} \cdot \boldsymbol{\varphi}(\mathbf{x}_i) + b)))$$

Support Vector Machine

Soft-margin SVM linear (o no linear)

Queremos minimizar

$$E = 1/p \sum_{1 \leq i \leq p} E_i + \lambda \|\mathbf{w}\|^2$$

El parámetro λ controla cuan fuertemente controlamos el tamaño de los pesos

Support Vector Machine

Soft-margin SVM linear (o no linear)

Pasando al problema dual tenemos:

$$\mathbf{w} = \sum_{1 \leq i \leq p} \alpha_i y_i \mathbf{x}_i \quad (\text{o} \quad \mathbf{w} = \sum_{1 \leq i \leq p} \alpha_i y_i \boldsymbol{\phi}(\mathbf{x}_i))$$

Donde α_i maximizan la función

$$Q(\{\alpha_i\}) = \sum_{1 \leq i \leq p} \alpha_i - \frac{1}{2} \sum_{1 \leq i, j \leq p} \alpha_i \alpha_j y_i y_j \boldsymbol{\phi}(\mathbf{x}_i) \cdot \boldsymbol{\phi}(\mathbf{x}_j)$$

Con los constraints $(2p\lambda)^{-1} \geq \alpha_i \geq 0$, $\sum_{1 \leq i \leq p} \alpha_i y_i = 0$

Support Vector Machine

Comparación *SVM* vs. redes multicapa

SVM

Funcion a optimizar
cuadrática ($t=O(p^3)$)

Generalización óptima
garantizada

Requiere memoria
 $O(p^2)$

Redes Multicapa

Función a optimizar
extremadamente complicada,
múltiples mínimos locales

Generalización óptima
determinada empíricamente

Requiere memoria $O(\text{batch size})$

Support Vector Machine

Implementación: scikit-learn

<https://scikit-learn.org/stable/modules/svm.html#svm-mathematical-formulation>

<https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python>

- Que kernel utilizar
- Parámetros del kernel: γ , κ , c , etc
- Parámetro de regularización: $C = (p\lambda)^{-1}$
- Optativo: clases desbalanceadas