Métodos kernel para clasificación

S. Van Vaerenbergh, I. Santamaría

GTAS, Universidad de Cantabria

20 de marzo de 2018

Contents

Aprendizaje Estadístico

Métodos Kernel
- Introducción

SVM lineal
- Introducción
- Formulación

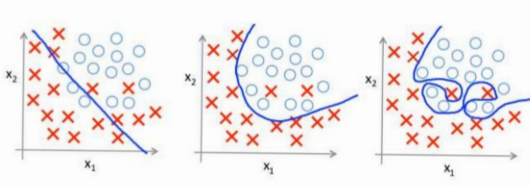
SVM No lineal
- Formulación
- Kernels
- Implementación

Extensiones
- Extensiones

Conclusiones
- Conclusiones

¿Qué es el aprendizaje estadístico?

Suponga tres clasificadores entrenados sobre el **conjunto de entrenamiento** de la figura



¿Qué clasificador funcionará mejor sobre el **conjunto de test**?
Es evidente que existe un compromiso entre:

- Error en el entrenamiento/Error de generalización (test)
- Sesgo/varianza del modelo (clasificador) entrenado

El **aprendizaje estadístico** formaliza estas ideas, caracterizando las propiedades matemáticas de las máquinas de aprendizaje

Aprendizaje Estadístico

- En un problema supervisado de clasificación (binario) queremos inferir una función

$$f(\mathbf{x}) : \mathcal{X} \rightarrow \{\pm 1\}$$

- Conjunto de entrenamiento: $(\mathcal{X}, \mathcal{Y}) = \{(\mathbf{x}_i, y_i)\}$
- Función de pérdidas (**loss function**) $l(\mathbf{x}, y, f)$ (p.ej., $l(\mathbf{x}, y, f) = \frac{1}{2}|f(\mathbf{x}) - y|$)
- Un buen clasificador debería minimizar el **risk or test error**

$$R[f] = \int \frac{1}{2}|f(\mathbf{x}) - y| dP(\mathbf{x}, y)$$

- Sin embargo, sólo podemos estimar el **empirical risk or training error**

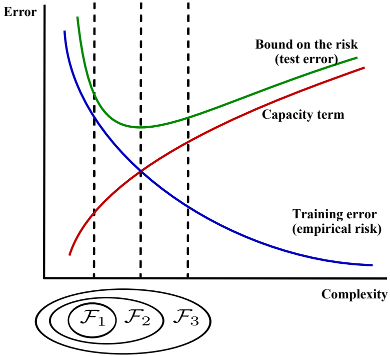
$$R_{emp}[f] = \sum_{i=1}^{n} \frac{1}{2}|f(\mathbf{x}_i) - y_i|$$

- El error de test se puede acotar como

$$R[f] \leq R_{emp}[f] + \phi(f)$$

donde $\phi(f)$ es un término de capacidad que mide la complejidad de las funciones que puede aprender nuestra máquina

- Es imperativo restringir el conjunto de funciones $f(\mathbf{x})$



La idea anterior conduce al principio del **Structural Risk Minimization** o **Regularized Empirical Risk Minimization**: es necesario minimizar una versión regularizada del error de entrenamiento

$$\text{minimize } R_{emp}[f] + \lambda\Omega(f),$$

donde $\Omega(f)$ mide la complejidad de la máquina de aprendizaje, y λ es un parámetro de regularización

- $\lambda \uparrow$ Modelos o fronteras simples
- $\lambda \downarrow$ Modelos o fronteras complejas (riesgo de sobreajuste)

Habitualmente λ se estima mediante validación cruzada

Introducción

- El secreto del éxito de muchos algoritmos de machine learning se basa en la búsqueda de un espacio de características efectivo/adecuado para nuestro problema
- Numerosas aplicaciones aplican una etapa previa de reducción de la dimensionalidad (PCA, LDA)

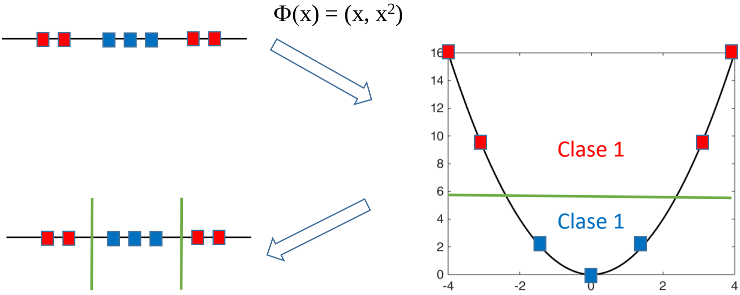
$$\mathbf{x}_i \in \mathcal{R}^d \longrightarrow \mathbf{y}_i \in \mathcal{R}^r, \qquad r < d$$

- Los **métodos kernel** siguen una aproximación distinta en la que se realiza (habitualmente de manera implícita) una expansión de la dimensionalidad

$$\mathbf{x}_i \in \mathcal{R}^d \longrightarrow \Phi(\mathbf{x}_i) \in \mathcal{R}^r, \qquad r >> d$$

¿Qué ventaja puede tener ir a un espacio de dimensión más alta?

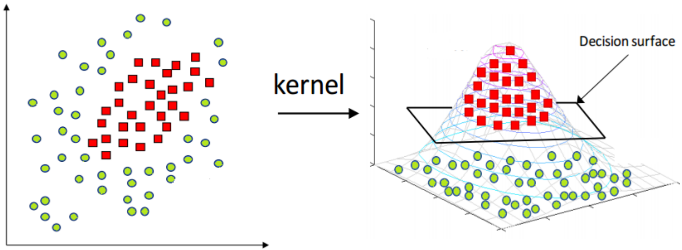
- Considere un problema de clasificación binaria en $\mathcal{R}$
- Conjunto de entrenamiento: { -4, -3,-1, 0, 1, 3, 4 }



- El mapping $\Phi(x) = [x, x^2]^T$ produce un problema lineal en el espacio expandido (espacio de características o **feature space**)

- Habitualmente no es necesario conocer explícitamente el mapping $\Phi(\mathbf{x})$
- Basta con conocer la función núcleo o **kernel** asociado

$$K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x})^T \Phi(\mathbf{x}')$$



- Los métodos kernel obtienen una solución lineal en el espacio de características (que se convierte en una solución no lineal en el espacio de entrada)

Support Vector Machine (SVM)

- La **SVM** es el método kernel estándar en clasificación
- Resuelve un problema de clasificación lineal en el espacio de características aplicando el principio SRM

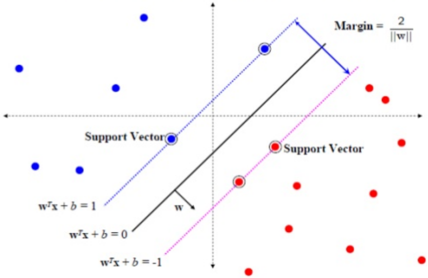
$$\min_{\mathbf{w},b} \sum_{i=1}^{n} \frac{1}{2}|f(\mathbf{x}_i) - y_i| + \lambda\Omega(f)$$

- Para entender los principios de funcionamiento y el problema de optimización asociado, analizaremos en primer lugar la SVM lineal $\rightarrow$ hiperplano óptimo de separación

SVM lineal

- Problema binario de clasificación $\{(\mathbf{x}_i, y_i = \pm 1)\}$
- Clases linealmente separables
- Clasificador lineal: $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = \langle \mathbf{w}, \mathbf{x} \rangle + b$
- Los vectores soporte $\mathbf{w}^T\mathbf{x}_j + b = \pm 1$ actúan como separación entre clases
- Maximizamos la distancia entre hiperplanos (**margen**)

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}||\mathbf{w}||^2 \\ \text{s.t.} \quad & y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1, \quad \forall i \end{aligned}$$



Solución

El problema anterior es **convexo** $\rightarrow$ Solución única

- Lagrangiano (problema dual)

$$\mathcal{L}(\mathbf{w}, b, \alpha) = \frac{1}{2}||\mathbf{w}||^2 + \sum_{i=1}^{n} \alpha_i \left(1 - y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right)\right)$$

- Strong duality $\Rightarrow$ KKT optimality
  1. El hiperplano óptimo es una combinación lineal de los patrones de entrada

$$\nabla \mathcal{L}_{\mathbf{w}}(\mathbf{w}, b, \alpha) = \mathbf{w} + \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

  2. El hiperplano óptimo sólo depende de los puntos que están sobre hiperplanos soporte: los **vectores soporte**

$$\alpha_i \left(1 - y_i(\mathbf{w}^T \mathbf{x}_i + b)\right) = 0, \forall i \quad \Rightarrow \quad y_i(\mathbf{w}^T \mathbf{x}_i + b) = 1$$

  3. b se puede obtener de cualquier vector soporte

Sustituyendo $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$ en el Lagrangiano, se obtiene el **problema dual**, que es el que típicamente se resuelve

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \tfrac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j - \sum_i \alpha_i \\ \text{s.t.} \quad & \sum_i \alpha_i y_i = 0, \\ & \alpha_i \geq 0, \quad \forall i \end{aligned}$$

Definiendo $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)^T$, $\mathbf{1} = (1, \ldots, 1)^T$, $\mathbf{Y} = \operatorname{diag}(y_1, \ldots, y_n)$ y $\mathbf{K}$ una matriz $n \times n$ con elementos $k(i,j) = \mathbf{x}_i^T \mathbf{x}_j = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$, obtenemos

Problema QP (Quadratic Programming)

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \tfrac{1}{2} \boldsymbol{\alpha}^T \mathbf{YKY} \boldsymbol{\alpha} - \mathbf{1}^T \boldsymbol{\alpha} \\ \text{s.t.} \quad & \boldsymbol{\alpha}^T \mathbf{y} = 0, \\ & \boldsymbol{\alpha} \geq 0 \end{aligned}$$

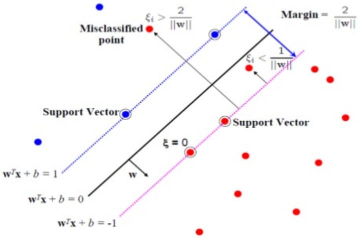
Soft-margin SVM

- Clases no separables
- Permitimos errores de clasificación introduciendo "holguras" (**slack variables**) en el problema de optimización: ξ_i
- Parámetro de regularización $C \rightarrow$ penalización
- El dual es también un problema QP (con $0 \leq \alpha_i \leq C$)

$$\min_{\mathbf{w},b} \quad \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i$$

$$\text{s.t.} \quad y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 - \xi_i, \quad \forall i$$

$$\xi_i, \geq 0 \quad \forall i$$



SVM No Lineal

- Mapeamos los datos a un espacio de caraterísticas de dimensión mayor (probablemente ∞): $\mathbf{x}_i \to \Phi(\mathbf{x}_i)$
- Resolvemos una SVM lineal en el espacio de características
- Hiperplano óptimo en el espacio de características

$$\mathbf{w} = \sum_i \alpha_i y_i \Phi(\mathbf{x}_i)$$

- El problema dual es el mismo !!

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \tfrac{1}{2}\boldsymbol{\alpha}^T \mathbf{YKY}\boldsymbol{\alpha} - \mathbf{1}^T\boldsymbol{\alpha} \\ \text{s.t.} \quad & \boldsymbol{\alpha}^T\mathbf{y} = 0, \\ & 0 \leq \boldsymbol{\alpha} \leq C \end{aligned}$$

pero empleando ahora una matriz kernel $\mathbf{K}$ con elementos $k(i,j) = \Phi(\mathbf{x}_i)^T\Phi(\mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_i) \rangle$

- En el espacio transformado la función de decisión es lineal

$$f(\mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x}) + b$$

- Pero en el espacio de entrada la función es no lineal, y se expresa nuevamente en función del kernel

$$\begin{aligned} f(\mathbf{x}) &= \underbrace{\left(\sum_i \alpha_i y_i \Phi(\mathbf{x}_i)\right)^T}_{\mathbf{w}^T} \Phi(\mathbf{x}) + b \\ &= \sum_i \alpha_i y_i \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}) + b = \sum_i \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b \end{aligned}$$

- Esta es la idea básica del **kernel trick**

Ejemplo: kernel polinómico

- Problema bi-dimensional $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
- Definimos un mapeo polinómico a una espacio 3D

$$\Phi(\mathbf{x}) = \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1 x_2 \end{bmatrix}$$

- La función kernel asociada es

$$\begin{aligned} k(\mathbf{x}, \mathbf{y}) = & \quad \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle = \Phi(\mathbf{x})^T \Phi(\mathbf{y}) = \\ = & \quad x_1^2 y_1^2 + x_2^2 y_2^2 + 2x_1 y_1 x_2 y_2 \end{aligned}$$

Funciones kernel

Teorema de Mercer (informal)

Cualquier función $k(\cdot,\cdot)$ tal que la matriz de kernel **K** para cualquier conjunto de entrenamiento sea positiva semidefinida, es decir

$$\mathbf{x}^T\mathbf{K}\mathbf{x} \geq 0, \qquad \forall \mathbf{x},$$

induce un producto escalar en un espacio transformado. Es decir $k(\mathbf{x}_i,\mathbf{x}_j)$ puede escribirse como

$$k(\mathbf{x}_i,\mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j)\rangle = \Phi(\mathbf{x}_i)^T\Phi(\mathbf{x}_j)$$

- La transformación $\Phi(\mathbf{x})$ es todavía desconocida
- Pero no la necesitamos siempre que elijamos un kernel positivo semidefinido $\Rightarrow$ Problema dual QP

Kernels típicos

- **Lineal**

$$k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$$

- **Polinómico** (parámetros p y c)

$$k(\mathbf{x}_i, \mathbf{x}_j) = \left(\mathbf{x}_i^T \mathbf{x}_j + c\right)^p$$

- **Gaussiano** (parámetro σ^2)

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$$

- Podemos crear nuevos kernel mediante transformaciones
  1. $k_1(\mathbf{x}, \mathbf{y}) + k_2(\mathbf{x}, \mathbf{y})$
  2. $k_1(\mathbf{x}, \mathbf{y}) k_2(\mathbf{x}, \mathbf{y})$
  3. $\exp(k_1(\mathbf{x}, \mathbf{y}))$

Nota: La sigmoide tanh $\left(\mathbf{x}^T \mathbf{y} + b\right)$ no es un kernel válido!

String kernel

También se pueden definir funciones kernel sobre datos en espacios no vectoriales (e.g, strings)

- Dadas dos secuencias

$$\begin{aligned} s &= \textit{statistics} \\ t &= \textit{computation} \end{aligned}$$

- Generamos todos los substrings de una determinada longitud (p.e. 3)

$$\begin{aligned} s &\rightarrow \{\textit{sta}, \textit{tat}, \textit{ati}, \textit{tis}, \textit{ist}, \textit{sti}, \textit{tic}, \textit{ics}\} \\ t &\rightarrow \{\textit{com}, \textit{omp}, \textit{mpu}, \textit{put}, \textit{uta}, \textit{tat}, \textit{ati}, \textit{tio}, \textit{ion}\} \end{aligned}$$

- El kernel se define contando en número de substrings comunes a las dos secuencias

$$k(s, t) = 2$$

También se pueden definir kernels sobre grafos, texto, para genómica, etc.

La matriz de kernel

Para resolver un problemas de clasificación con SVMs sólo se necesita la matriz de kernel **K** (Gramm matrix)

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \cdots & k(\mathbf{x}_1, \mathbf{x}_n) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \cdots & k(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \ddots & \ddots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & k(\mathbf{x}_n, \mathbf{x}_2) & \cdots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}$$

- $k(\mathbf{x}_i, \mathbf{x}_j)$ es una medida de similitud entre patrones
- **K** es $n \times n \rightarrow$ Dificultades de computacionales y de almacenamiento

Distancia para el kernel Gaussiano

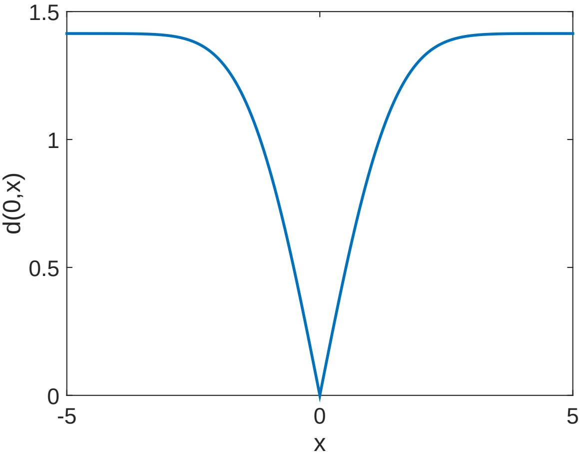
- El kernel Gaussiano es un producto escalar (similitud) en un espacio transformado de dimensión infinita

$$k(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x})^T \Phi(\mathbf{y}) = e^{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}}$$

- La distancia entre $\Phi(\mathbf{x})$ y $\Phi(\mathbf{y})$ es

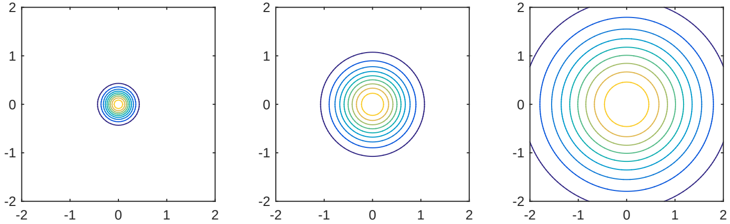
$$\begin{aligned} d(\Phi(\mathbf{x}), \Phi(\mathbf{y})) &= \sqrt{\|\Phi(\mathbf{x}) - \Phi(\mathbf{y})\|^2} = \sqrt{2\left(1 - e^{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}}\right)} \\ &= \sqrt{2\,(1 - k(\mathbf{x}, \mathbf{y}))} \end{aligned}$$

Ejemplo: Caso 1D $\sigma^2 = 1$



Ejemplo: Caso 2D

$\sigma^2 = 0,2$ $\sigma^2 = 0{,}5$ $\sigma^2 = 1$



- $\sigma^2 \downarrow\downarrow$ distancia muy localizada: todos los puntos fuera de un radio están igualmente lejos
- $\sigma^2 \uparrow\uparrow$ distancia global, equivalente a un kernel lineal

Ajuste de una SVM

- Consideramos una SVM con kernel Gaussiano

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \tfrac{1}{2}\boldsymbol{\alpha}^T \mathbf{YKY}\boldsymbol{\alpha} - \mathbf{1}^T\boldsymbol{\alpha} \\ \text{s.t.} \quad & \boldsymbol{\alpha}^T \mathbf{y} = 0, \\ & 0 \leq \boldsymbol{\alpha} \leq C \end{aligned} \qquad k(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma\|\mathbf{x}-\mathbf{y}\|^2}$$

  donde hemos definido $\gamma = \frac{1}{2\sigma^2}$

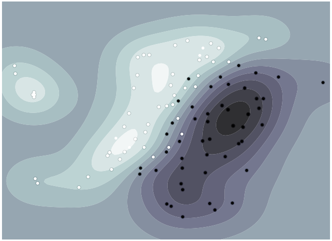
- La elección de unos parámetros γ y C es esencial para obtener buenas prestaciones
- Habitualmente se emplea **cross-validation**

Influencia de C

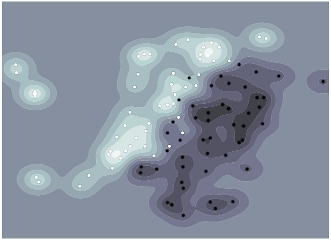
- El parámetro de regularizacion C establece un compromiso entre el error de entrenamiento y la complejidad del modelo
- $C \downarrow$ modelo sencillo, mayor error en el entrenamiento, suavidad en la frontera de decisión
- $C \uparrow$ modelo complejo, poca suavidad de la frontera de decisión, riesgo de **sobreajuste**

Ejemplo

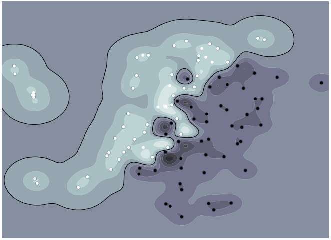
$C = 0.001$



$C = 0.01$



$C = 100$



Influencia de γ

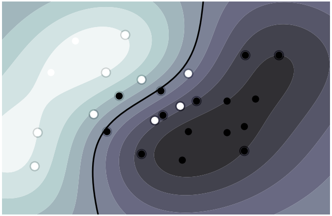
- El parámetro del kernel Gaussiano λ (a.k.a. bandwidth) controla la velocidad a la que $k(\mathbf{x}, \mathbf{y}) \to 0$ en función de la distancia
- Recuerde que para clasificar un nuevo patrón **x** la SVM computa

$$f(\mathbf{x}) = \sum_i \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b \underset{C_0}{\overset{C_1}{\gtrless}} 0$$

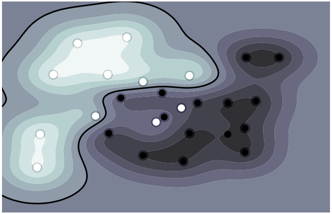
- $\gamma \downarrow$ mayor solape entre Gaussianas, suavidad en la frontera de decisión
- $\gamma \uparrow$ todos los puntos tienden a ser ortogonales unos a otros **sobreajuste**

Ejemplo

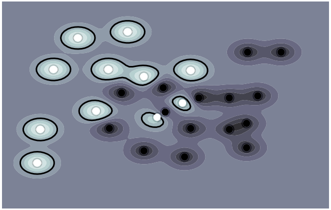
$\gamma = 0.001$



$\gamma = 0.01$

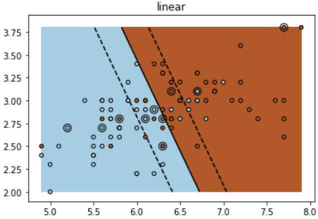


$\gamma = 100$

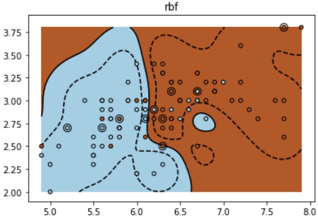


Comparación kernels

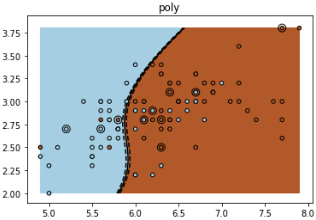
Lineal $C = 1$



Gaussiano $C = 1, \gamma = 10$



Polinómico $C = 1, orden = 10$



Implementación: SVM solvers

- Problema QP $\rightarrow$ **Interior Point Methods**
  1. Requiere almacenar **K** en memoria: $\mathcal{O}(n^2)$
  2. Convergencia lenta, gasto computacional $\mathcal{O}(n^3)$
- Se han desarrollado algoritmos específicos más eficientes para este problema
- **Sequential Minimal Optimization (SMO)**: Resuelve una serie de subproblemas más pequeños
- **LIBSVM**:
  - Paquete estándar para SVMs
  - Implementa una versión del algoritmo SMO
  - Interfaces en R, Matlab, Python,...

Multi-class SVM

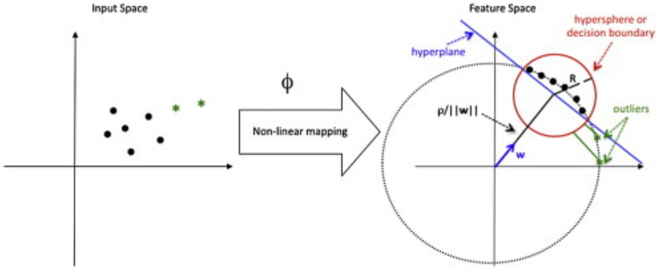
- Metodología estándard: **One-Versus-All**
- En un problema con K clases resolvemos K problemas binarios
- Cada SVM está entrenada para separar una clase del resto de patrones
- Sobre un nuevo patrón de test, **x**, la SVM k-ésima proporciona una salida (score)

$$f^k(\mathbf{x}) = \sum_i \alpha_i^k y_i^k k(\mathbf{x}_i^k, \mathbf{x}) + b^k, \qquad k = 1, \ldots, K$$

- La clase finalmente elegida es

$$k^* = \operatorname*{argmax}_k f^k(\mathbf{x})$$

One-class SVM



- **Objetivo**: encontrar una SVM que englobe una región del espacio donde los datos "viven"
- Problema de clasificación: separar **datos** de **outliers**
- Separación en el espacio transformado
  - Un hiperplano (Schölkopf et al)
  - Una hiperesfera (Tax and Duin)

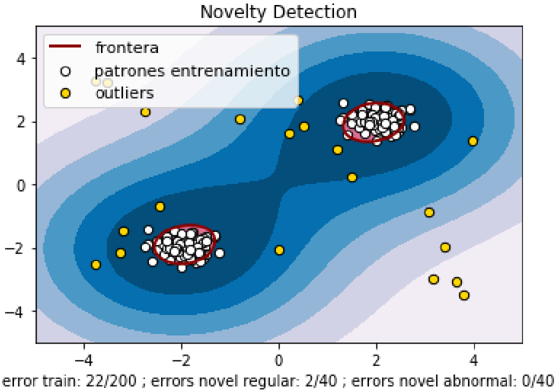
One-class SVM

$$\begin{array}{lll} \min\limits_{\mathbf{w},\xi_i,\rho} & \frac{1}{2}||\mathbf{w}||^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho & \\ \text{s.t.} & \mathbf{w}^T\Phi(\mathbf{x}_i) \geq \rho - \xi_i, & \forall i \\ & \xi_i \geq 0 & \forall i \end{array}$$

- El problema dual es equivalente a una SVM convencional
- El parámetro ν caracteriza la solución $\rightarrow$ ν**-SVM**
  - Es una cota superior de la fracción de patrones de entrenamiento etiquetados como outliers
  - Es una cota inferior del número de vectores soporte

Ejemplo

- ν-SVM, kernel Gaussiano, $\gamma = 0{,}1$, $\nu = 0{,}1$



Conclusiones

- Una de las máquinas de aprendizaje más populares
- Las **SVMs** implementan el criterio SRM (Structural Risk Minimization)
- **Problema dual QP**: solución única, problema bien definido
- Hay que elegir el **kernel** (medida de similitud entre patrones), sus **hiperparámetros**, y el **parámetro de regularización**
- Solución dispersa (**sparse**) expresada en función de unos pocos vectores soporte
- Proporcionan (todavía) resultados competitivos en muchas aplicaciones. En especial, cuando los datos de entrada no tienen una dimensionalidad muy alta