Proyecto Final: Support Vector Machines

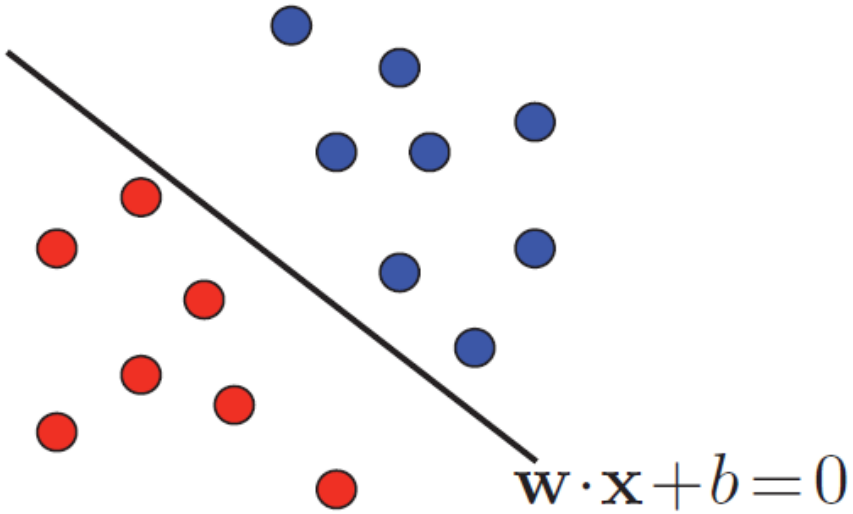
Julián Bayardo

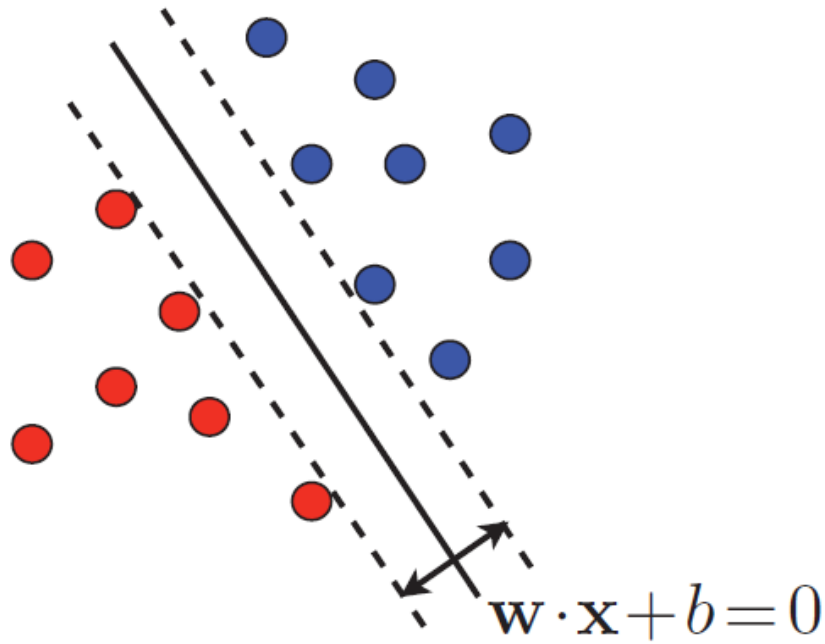
Universidad de Buenos Aires

23 de mayo de 2018

Contenidos

Support Vector Machines

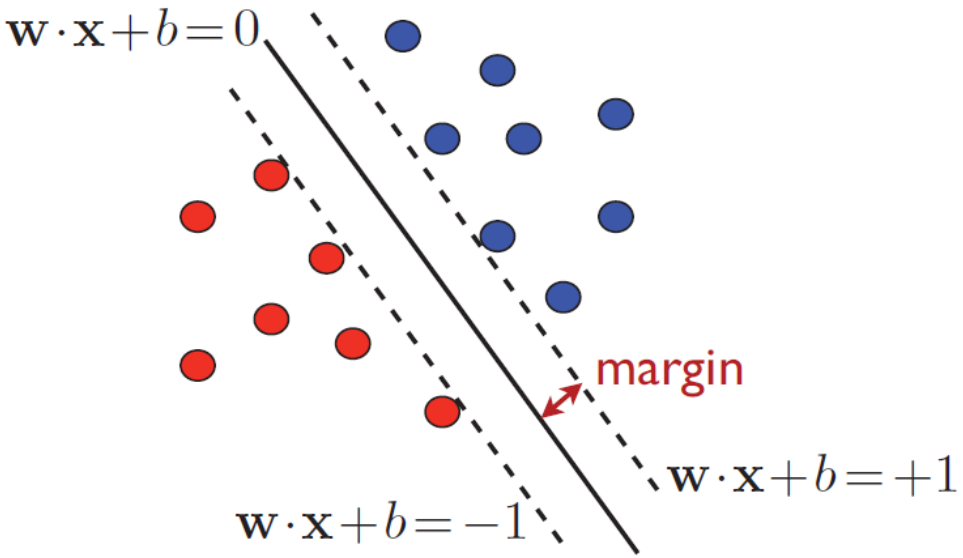




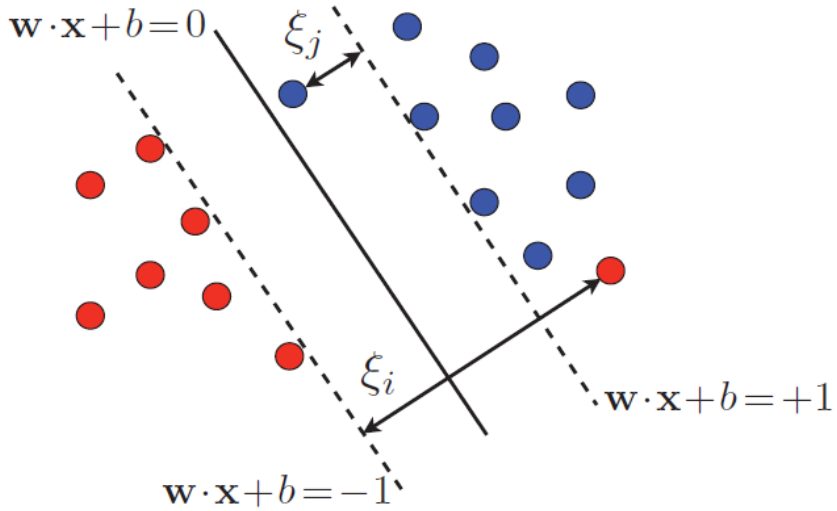
¿Por Qué?

- Originalmente: Navaja de Ockham.
- Con el Kernel Trick: capacidad de representar decision boundaries no lineales, y de manejar espacios de muy alta dimensionalidad.
- Formalmente muy elegantes y bien estudiadas, tienen muchas garantías que otros métodos no tienen (por ejemplo, sobre el error de generalización).
- Entrenamiento e inferencia pueden hacerse eficientemente, ya que reducen a problemas que están muy estudiados.

Dos Formulaciones



Cuando los datos son linealmente separables, podemos formular el problema en términos de maximizar el margen con respecto a los datos.



Cuando no lo son, deja de haber un hiperplano que separa. La formulación se adapta para permitir un error sobre el los márgenes.

Contenidos

Producto Interno

Definición

Dado $\mathbb{K} = \mathbb{R}$ o $\mathbb{C}$, $\mathbb{V}$ un espacio vectorial sobre $\mathbb{K}$, un producto interno es una función $\langle \cdot, \cdot \rangle : \mathbb{V} \times \mathbb{V} \to \mathbb{K}$ tal que:

- $\langle \boldsymbol{\alpha} + \boldsymbol{\beta}, \boldsymbol{\gamma} \rangle = \langle \boldsymbol{\alpha}, \boldsymbol{\gamma} \rangle + \langle \boldsymbol{\beta}, \boldsymbol{\gamma} \rangle$
- $\langle c\boldsymbol{\alpha}, \boldsymbol{\beta} \rangle = c\langle \boldsymbol{\alpha}, \boldsymbol{\beta} \rangle$
- $\langle \boldsymbol{\beta}, \boldsymbol{\alpha} \rangle = \overline{\langle \boldsymbol{\alpha}, \boldsymbol{\beta} \rangle}$
- $\langle \boldsymbol{\alpha}, \boldsymbol{\alpha} \rangle > 0$ si $\boldsymbol{\alpha} \neq 0$

Observación

$\langle \boldsymbol{\alpha}, c\boldsymbol{\beta} + \boldsymbol{\gamma} \rangle = \overline{c}\langle \boldsymbol{\alpha}, \boldsymbol{\beta} \rangle + \langle \boldsymbol{\alpha}, \boldsymbol{\gamma} \rangle$

Producto Interno: Ejemplo

Si tomamos $\mathbb{V} = \{\boldsymbol{f} : [a, b] \to \mathbb{R} / \boldsymbol{f} \in \mathcal{C}([a, b])\}$ como un $\mathbb{R}$-E.V,

$$\langle \boldsymbol{f}, \boldsymbol{g} \rangle = \int_a^b f(x)g(x)dx$$

Es un producto interno. Si fijamos $(a, b) = (0, T)$, $\mathbb{W} = \{\sin(\frac{\pi nx}{T}) / n \in \mathbb{N}\}$ es un conjunto de funciones ortogonales, y además:

$$\langle \sin(\frac{\pi nx}{T}), \sin(\frac{\pi nx}{T}) \rangle = \frac{T}{2}$$

Por lo que es trivial volverlo un conjunto ortonormal.

Hiperplanos

Definición

Dado un $\mathbb{K}$-E.V. $\mathbb{V}$ de dimensión finita, un hiperplano $\mathbb{H}$ es un subespacio de dimensión $\dim(\mathbb{V}) - 1$

Observemos entonces que $\mathbb{V} = \mathbb{H} \oplus \mathbb{H}^{\perp}$, donde $\dim(\mathbb{H}^{\perp}) = 1$. Además, vale que $(\mathbb{H}^{\perp})^{\perp} = \mathbb{H}$. Por ende, podemos pensar que

$$\mathbb{H} = \{\boldsymbol{v} \in \mathbb{V} / \langle \boldsymbol{w}, \boldsymbol{v} \rangle = 0, \text{donde } \boldsymbol{w} \in \mathbb{H}^{\perp}\}$$

Distancia Mínima a un Hiperplano

Es fácil ver que dado un vector $\boldsymbol{x}$, la proyección ortogonal sobre un hiperplano de normal $\boldsymbol{w}$ es $\boldsymbol{x} - \langle \boldsymbol{w}, \boldsymbol{x} \rangle \boldsymbol{w}$. Más aun, la distancia de $\boldsymbol{x}$ a este punto es

$$\frac{|\langle \boldsymbol{w}, \boldsymbol{x} \rangle|}{||\boldsymbol{w}||}$$

En el caso genérico en que el plano está corrido del eje, esta expresión se convierte en

$$\frac{|\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b|}{||\boldsymbol{w}||}$$

Contenidos

Support Vector Machines: Caso Separable

Dado un sample $S = \{(\boldsymbol{x_1}, y_1), ..., (\boldsymbol{x_m}, y_m)\}$ y un hiperplano de ecuación $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0$, el margen se puede definir como:

$$\rho = \min_{(\boldsymbol{x},y) \in S} \frac{|\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b|}{||\boldsymbol{w}||}$$

Cualquier hiperplano puede tomarse de forma canónica **con respecto a un sample** de manera tal que

$$\min_{(\boldsymbol{x},y) \in S} |\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b| = 1$$

Simplemente escalando a $\boldsymbol{w}$ y b por el valor original. Cuando un hiperplano está expresado de manera canónica, vale que:

$$\rho = \frac{1}{||\boldsymbol{w}||}$$

$$\min_{\boldsymbol{w},b} \quad F(\boldsymbol{w}) = \frac{1}{2}||\boldsymbol{w}||^2$$
$$\text{sujeto a} \quad y_i(\langle \boldsymbol{w}, \boldsymbol{x_i} \rangle + b) \geq 1, \ \forall i \in [1, m].$$

La función objetivo es infinitamente diferenciable y estrictamente convexa ($\nabla_{\boldsymbol{w}} F = \boldsymbol{w}$, $\nabla^2_{\boldsymbol{w}} F = \mathbf{I}$). Las restricciones son funciones afines, y por ende calificadas. Esto implica que el problema de optimización tiene una **solución única**.

Además, como la función objetivo es cuadrática y las restricciones afines, este problema así es una instancia particular de QP (*quadratic programming*), que ha sido muy estudiada y dispone de algoritmos muy eficientes.

Support Vectors Fantásticos y Dónde Encontrarlos

El Lagrangiano para el problema anterior puede escribirse como:

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}||\boldsymbol{w}||^2 - \sum_{i=1}^{m} \alpha_i (y_i(\langle \boldsymbol{w}, \boldsymbol{x_i} \rangle + b) - 1)$$

Con $\alpha \geq 0$. En el punto crítico aplican las condiciones de Karush-Kuhn-Tucker, y por ende tenemos:

$$\nabla_{\boldsymbol{w}} \mathcal{L} = \boldsymbol{w} - \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x_i} = 0$$

$$\nabla_b \mathcal{L} = -\sum_{i=1}^{m} \alpha_i y_i = 0$$

$$\forall i, \alpha_i (y_i(\langle \boldsymbol{w}, \boldsymbol{x} \rangle \boldsymbol{x_i} + b) - 1) = 0$$

Los $\boldsymbol{x_i}/\alpha_i > 0$ se llaman *support vectors*. Pertenecen a los hiperplanos marginales, y determinan la solución. **Ojo:** los vectores de soporte no son únicos. $N + 1$ vectores definen un hiperplano.

Y la solución? Y candela? Y la moto?

El Lagrangiano del problema dual es

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x_i}, \boldsymbol{x_j} \rangle$$

Que se corresponde con

$$\max_{\boldsymbol{\alpha}} \quad G(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x_i}, \boldsymbol{x_j} \rangle$$

$$\text{sujeto a} \quad \alpha_i \geq 0 \wedge \sum_{i=1}^{m} \alpha_i y_i = 0, \ \forall i \in [1, m].$$

La función objetiva $G(\boldsymbol{\alpha})$ es infinitamente diferenciable, su hessiano es $\nabla^2 G = -\boldsymbol{A}$ con

$$\boldsymbol{A}_{ij} = \langle y_i \boldsymbol{x_i}, y_j \boldsymbol{x_j} \rangle$$

El mismo es semidefinido positivo, por lo que G es concava. Las restricciones son afines y convexas, y por ende el problema es equivalente a uno de optimización convexa. Como G es cuadrática en $\boldsymbol{\alpha}$, es un problema de QP igual que antes.
Más aun, como las restricciones son afines, también son calificadas y vale dualidad fuerte. Es decir, el $\boldsymbol{\alpha}$ que resuelve el problema dual también resuelve el problema original.

La importancia es la siguiente. Supongamos que tenemos ese $\boldsymbol{\alpha}$. Entonces,

$$\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x_i}$$

$$b = y_i - \sum_{j=i}^{m} \alpha_j y_j \langle \boldsymbol{x_j}, \boldsymbol{x_i} \rangle$$

Pero lo más importante es que

$$\begin{aligned} h(\boldsymbol{x}) &= \text{sign}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b) \\ &= \text{sign}(\sum_{i=1}^{m} \alpha_i y_i \langle \boldsymbol{x_i}, \boldsymbol{x} \rangle + b) \end{aligned}$$

Recapitulando

1. Empezamos con un problema de optimización cuyo objetivo era maximizar el márgen
2. Lo reducimos a resolver un Lagrangiano. Esto nos dijo:
   - Cómo se va a ver la solución una vez que sepamos cuál es el α.
   - Los vectores de soporte están en los hiperplanos marginales y no son únicos.
3. Convertimos al Lagrangiano en el de su problema Dual, y sabemos que resolver el problema dual nos va a dar una solución del problema original.
4. Miramos la solución del problema original asumiendo que tenemos el α, y encontramos:
   - Que la solución de SVM depende sólo de los vectores de soporte.
   - Además, sólo depende de productos internos.

Support Vector Machines: Caso No Separable

Cuando los datos no son linealmente separables, tenemos que $\exists i \in [1, m]$ tal que:

$$y_i(\langle w, x_i \rangle + b) \ngeq 1$$

Intentar minimizar el número de errores en los datos es NP-hard en función de la cantidad de dimensiones N. Es decir, hay que buscar una formulación alternativa.

Se definen un conjunto de *slack variables* $\xi_i \geq 0$, tal que

$$y_i(\langle w, x_i \rangle + b) \geq 1 - \xi_i$$

Ojo: El error se mide respecto al hiperplano marginal, no con respecto la *decision surface*. Penalizamos a elementos que estén del lado incorrecto del margen, no sólo a los incorrectamente clasificados.

$$\min_{w,b,\xi} \quad \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\xi_i^p$$

$$\text{sujeto a} \quad y_i(\langle w, x_i\rangle + b) \geq 1 - \xi_i \wedge \xi_i \geq 0,\ \forall i \in [1, m].$$

- C controla cuánta importancia darle al error cometido por el *slack*.
- p qué tan severa será la penalización del *slack*.
- La función objetivo es convexa y diferenciable para cualquier valor de $p \geq 1$; las restricciones son afines, por ende convexas y diferenciables. Es decir, el problema es de **optimización convexa**, y **existe solución, aunque puede no ser única**.

Ojo: Seguimos encontrando un hiperplano. Lo único que incorporamos en esta formulación es la *posibilidad de que se cometan errores*.

Solución

El *Foundations* [1, 4.3, p. 71] y el Bishop [2, 7.1, p. 326] tienen la solución en el caso en que $p = 1$.
No vamos a ver la solución de este problema acá:

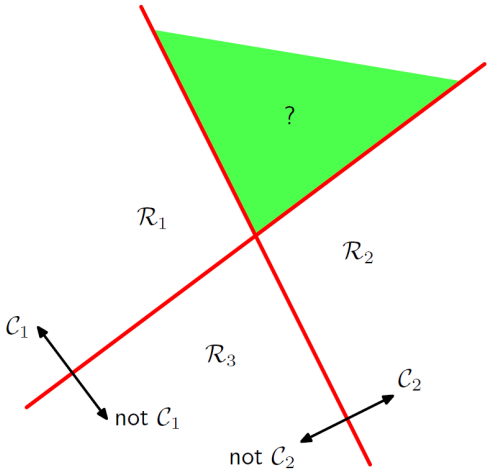
- La metodología a seguir es la misma, sólo se introduce un β adicional en el Lagrangiano por las restricciones de no-negatividad de los ξ_is.
- La solución final termina siendo exactamente la misma.

Contenidos

Problema

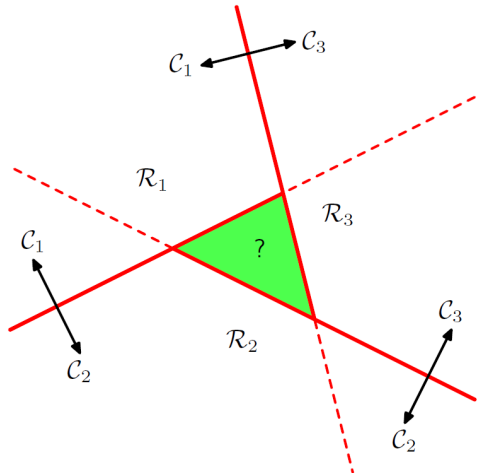
Hay tres soluciones comunes a cómo construir un clasificador multiclase a partir de un clasificador lineal binario [2, 4.1, p. 183]:

One-Vs-Rest



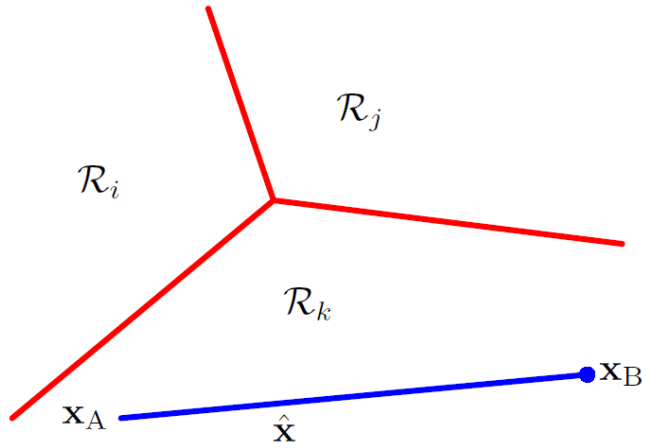
K clasificadores, diciendo si pertenece a la clase i o no.

One-Vs-One



$\frac{K(K-1)}{2}$ clasificadores, y se toma el voto mayoritario.

Single Discriminant



Un único clasificador, con K salidas, se toma la mayor.

Conclusión

No hay una buena respuesta:

- *One-Vs-Rest*:
  - Tiene regiones donde no sabemos cuál es la clase.
  - Los clasificadores se entrenan son instancias distintas; no hay garantía sobre la escala entre sus salidas, lo que invalida soluciones del tipo $\max y_k(\boldsymbol{x})$.
  - El entrenamiento es des-balanceado por naturaleza.
  - Introducir una única función objetivo que entrena K SVMs es muy costoso. La complejidad pasa de $O(KN^2)$ a $O(K^2N^2)$.
- *One-Vs-One* también tiene regiones donde no sabemos cuál es la clase, pero además tiene un costo de $O(K^2N^2)$ de entrada.

En la práctica se utiliza *One-Vs-Rest*. Ejemplo (parámetros: `-s 0 -t 0 -c 100`).

Contenidos

Idea

SVM sólo depende del producto interno entre los vectores de entrada.
Idea: aplicar una transformación de datos $\Phi(\cdot)$ a un nuevo espacio vectorial, con otro producto interno.
Ojo: el hiperplano que encontremos va a cumplir con las restricciones, pero en un espacio de dimensión potencialmente mucho mayor (inclusive infinita).
Observación: Este truco puede ser, y en efecto es, aplicado a cualquier algoritmo cuya salida se pueda determinar únicamente como productos internos (por ejemplo, el perceptrón!)

Qué es un *Kernel*

Definición

Una función $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ se denomina un *kernel* sobre $\mathcal{X}$, si cumple que:

$$\forall x, x' \in \mathcal{X}, K(x, x') = \langle \Phi(x), \Phi(x') \rangle$$

Para alguna función $\Phi : \mathcal{X} \to \mathbb{H}$, con $\mathbb{H}$ un espacio de Hilbert que denominamos *feature space*.

Pro-Tips:

- El *kernel* tiene que ser rápido de calcular, ya que reemplaza al producto interno en SVM.
- Si $\phi(\cdot)$ es no-lineal, en el espacio vectorial original la superficie decisión también es no-lineal.

La Leyenda del *Kernel* Perdido

El problema reduce a buscar *kernels* lleven nuestros datos a espacios donde sean linealmente separables. Observemos que este problema se divide en dos:

- Ver que en el espacio de llegada los datos son lo más linealmente separables posible. No podemos hacerlo formalmente, se hace *cross-validation* con distintos *kernels*.
- Cómo encontrar *kernels*. Es lo que vamos a ver ahora, qué condiciones son necesarias y suficientes, y cómo podemos combinar *kernels* para crear nuevos.

La Condición de Mercer

Teorema

Sea $\mathcal{X} \subset \mathbb{R}^N$ un conjunto compacto, $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ continua y simétrica. Entonces K admite una expansión uniformemente convergente de la forma:

$$K(x, x') = \sum_{n=0}^{\infty} a_n \Phi_n(x) \Phi_n(x')$$

Con $a_n > 0$ sí y sólo sí para toda función integrable cuadrada c (con $c \in L_2(\mathcal{X})$ se cumple la siguiente condición:

$$\int \int_{\mathcal{X} \times \mathcal{X}} c(x) c(x') K(x, x') dx dx' \geq 0$$

Si K satisface la condición de Mercer, entonces existe Φ. Es decir, K es un kernel y, por ende, existe una solución al problema de SVM.

Positive Definite Symmetric Kernels

Definición

$K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ se dice *positive definite symmetric (PDS)* si $\forall \{x_1, ..., x_m\} \subseteq \mathcal{X}$ la matriz $\boldsymbol{K}_{ij} = K(x_i, x_j)$ es simétrica semidefinida positiva (recordatorio: semidefinida positiva es que $c^T \boldsymbol{A} c \geq 0$)

Teorema

Sea $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ PDS, entonces K es un kernel. Es decir:

$$K(x, x') = \langle \Phi(x), \Phi(x') \rangle$$

Con $\Phi : \mathcal{X} \to \mathbb{H}$, y $\mathbb{H}$ un espacio de Hilbert.

Construyendo *Kernels*

Dados k_1, k_2 *kernels*, los siguientes también lo son [2, 6.2, p. 296]:

$$k(\boldsymbol{x}, \boldsymbol{x'}) = ck_1(\boldsymbol{x}, \boldsymbol{x'}) \qquad k(\boldsymbol{x}, \boldsymbol{x'}) = k_1(\boldsymbol{x}, \boldsymbol{x'}) + k_2(\boldsymbol{x}, \boldsymbol{x'})$$
$$k(\boldsymbol{x}, \boldsymbol{x'}) = f(\boldsymbol{x})k_1(\boldsymbol{x}, \boldsymbol{x'})f(\boldsymbol{x'}) \qquad k(\boldsymbol{x}, \boldsymbol{x'}) = k_1(\boldsymbol{x}, \boldsymbol{x'})k_2(\boldsymbol{x}, \boldsymbol{x'})$$
$$k(\boldsymbol{x}, \boldsymbol{x'}) = q(k_1(\boldsymbol{x}, \boldsymbol{x'})) \qquad k(\boldsymbol{x}, \boldsymbol{x'}) = \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x'}$$
$$k(\boldsymbol{x}, \boldsymbol{x'}) = e^{k_1(\boldsymbol{x}, \boldsymbol{x'})} \qquad k(\boldsymbol{x}, \boldsymbol{x'}) = k_a(\boldsymbol{x_a}, \boldsymbol{x'_a}) + k_b(\boldsymbol{x_b}, \boldsymbol{x'_b})$$
$$k(\boldsymbol{x}, \boldsymbol{x'}) = k_3(\Phi(\boldsymbol{x}), \Phi(\boldsymbol{x'})) \qquad k(\boldsymbol{x}, \boldsymbol{x'}) = k_a(\boldsymbol{x_a}, \boldsymbol{x'_a})k_b(\boldsymbol{x_b}, \boldsymbol{x'_b})$$

Donde $c > 0$ es una constante, $f(\cdot)$ es cualquier función, $q(\cdot)$ es un polinomio de coeficientes positivos, $\Phi(x)$ es una función de $\mathcal{X}$ en $\mathbb{R}^M$, $k_3(\cdot, \cdot)$ es un kernel en $\mathbb{R}^M$, $\boldsymbol{A}$ es una matriz simétrica semidefinida positiva, y $\boldsymbol{x_a}, \boldsymbol{x_b}$ son proyecciones de $\boldsymbol{x}$, con k_a y k_b *kernels* sobre sus proyecciones.

Contenidos

Disclaimer

- Claramente, hay infinitos *kernels*.
- Usar SVM está en saber cómo encontrar el *kernel* que sirve para el problema a resolver.
- Sólo un sitio web cuenta con 25 *kernels* utilizados en la práctica.
- Vamos a ver los 4 que están disponibles en todas las librerías (lineal, polinomial, gaussiano, y sigmoideo), y dos ejemplos que me parecieron interesantes.
- La idea de esta parte es que logren llevarse consejos prácticos para poder usar SVM.

Ejemplo 0: *Linear*

$$k(\boldsymbol{x}, \boldsymbol{x'}) = \boldsymbol{x}^T \boldsymbol{x'} + c$$

Con $c \geq 0$.

- El no-ejemplo: la superficie de decisión a seguir siendo un hiperplano.
- Útil cuando los datos son linealmente separables. Esto sucede en general cuando hay muchas *features* (es decir, cuando la dimensión del espacio vectorial es muy alta).
- Es muy rápido de computar (sólo un `for`!).
- Un buen uso de *kernels* lineales son, por ejemplo, las tareas de clasificación de texto [4].

Ejemplo 1: *Polynomial*

$$k(\boldsymbol{x}, \boldsymbol{x'}) = (\boldsymbol{x}^T \boldsymbol{x'} + c)^M$$
$$= \sum_{i=0}^{M} \binom{M}{i} (\boldsymbol{x}^T \boldsymbol{x'})^i c^{M-i}$$

Con $c \geq 0$. Por otro lado,

$$(\boldsymbol{x}^T \boldsymbol{x'})^i = \sum_{k_1+\ldots+k_N=i} \binom{i}{k_1, \ldots, k_N} (x_1 x_1')^{k_1} \cdots (x_N x_N')^{k_N}$$

- $k(\boldsymbol{x}, \boldsymbol{x'})$ tiene todos los términos hasta grado M posibles.
- Si $c = 0$, entonces tenemos sólo monomios.

Ejemplo 1: *Polynomial* (Cont)

$$k(\boldsymbol{x}, \boldsymbol{x'}) = (\boldsymbol{x}^T \boldsymbol{x'} + c)^M$$

- Problemas de inestabilidad numérica. Si $\boldsymbol{x}^T \boldsymbol{x'} + c < 1$ la función colapsa a 0, y si es mayor diverge.

Ejemplo 2: *Gaussian* o *Radial Basis Function (RBF)*

$$k(\boldsymbol{x}, \boldsymbol{x'}) = \exp(-\gamma||\boldsymbol{x} - \boldsymbol{x'}||^2)$$
$$= \exp(-\frac{||\boldsymbol{x} - \boldsymbol{x'}||^2}{2\sigma^2})$$

- γ o σ funcionan como control de **granularidad**. Ejemplo.
- Un σ demasiado pequeño va a generar *overfitting*.
- $\lim_{x \to 0} \frac{e^x}{1+x} = 1$. Es decir, un σ demasiado grande va a convertir el *kernel* en un kernel lineal.

Ejemplo 2: *Gaussian* o *Radial Basis Function (RBF)* (Cont)

$$k(\boldsymbol{x}, \boldsymbol{x'}) = \exp(-\frac{||\boldsymbol{x} - \boldsymbol{x'}||^2}{2\sigma^2})$$

- El *kernel* gaussiano es un producto interno en una espacio de dimensión infinita. Se ve expandiendo la exponencial que a una suma infinita sobre *kernels* polinomiales [3].
- Tiene buenas propiedades, en el sentido de que la relación entre los hiperparámetros C y σ está muy bien estudiada [6].
- El *kernel* lineal es una versión degenerada del *kernel RBF*, por lo que en cualquier problema (asumiendo que todo es exacto), el *RBF* debería tener mejor o igual *accuracy* que el lineal [6].
- Es el más utilizado en la práctica, y lo que se recomienda para comenzar a atacar un problema.

Ejemplo 3: *Sigmoid*

$$K(\boldsymbol{x}, \boldsymbol{x'}) = \tanh(\alpha \boldsymbol{x}^T \boldsymbol{x'} + c)$$

- No necesariamente es un *kernel* válido. Las condiciones para que sí sea se imponen sobre α y c, y son muy específicas [7, 3, p. 7].
- Se puede ver que, por diversos motivos, $\alpha > 0, c < 0$ es el caso en que mejor funciona. Pero también, si $\alpha, c \approx 0$, *kernel* es básicamente equivalente a *RBF* [7, 4, p. 11].
- Es equivalente (en términos de **capacidad**) a una red neuronal de 2 capas con función de activación dada por la tangente hiperbólica.
- En general se recomienda utilizar *RBF*, pero hay trabajos donde dió buenos resultados.

Ejemplo 4: Modelo Generativo

En un modelo generativo se modela $P(X, Y)$, en lugar de $P(Y|X)$. Entonces, tenemos formas de, por ejemplo, obtener $P(X)$ (marginalizando sobre Y). En este contexto, podemos definir:

$$k(\boldsymbol{x}, \boldsymbol{x'}) = P(\boldsymbol{x})P(\boldsymbol{x'})$$

Más todavía, podemos definir

$$k(\boldsymbol{x}, \boldsymbol{x'}) = \sum_i P(\boldsymbol{x}|i)P(\boldsymbol{x'}|i)P(i)$$

Que es equivalente, módulo una constante multiplicativa, a una *mixture* donde los componentes se factorizan e i toma el rol de una variable latente [2, 6.2, p. 297]

Ejemplo 5: *Kernels* Simbólicos

- Si suponemos un alfabeto finito Σ, podemos definir un *kernel* que dice que dos *strings* son similares si comparten muchas *substrings* lo suficientemente contiguas, y se puede evaluar eficientemente usando programación dinámica [8, 3, p. 422].
- Hay otro montón de ejemplos, tomando árboles, grafos genéricos, etcétera [9].

Cuándo Usar Qué *Kernels*

Basándome en [10]:

- Si la cantidad de *features* es mayor o igual a el tamaño del conjunto de entrenamiento, suele ser mejor un *kernel* lineal. Simplemente por el hecho de que en un espacio de dimensión alta es más probable que los datos estén bien separados, y la no-linealidad difícilmente agregue información.
- Si la cantidad de observaciones es mucho mayor a la cantidad de *features*, muy probablemente sea útil un *kernel* no lineal. En este caso intentar con el gaussiano para empezar.
- De cualquier forma, se recomienda fuertemente siempre intentar con un *kernel* lineal primero, simplemente porque es rápido de entrenar y funciona como un buen *baseline*.

Manejo de Variables Categóricas

Se recomienda siempre utilizar un *one-hot encoding* para las variables categóricas. Por ejemplo, si tenemos como una categoría Color con posibles valores Rojo, Azul, y Verde, tendríamos 3 vectores:

- $(1, 0, 0)$ para Rojo.
- $(0, 1, 0)$ para Azul.
- $(0, 0, 1)$ para Verde.

Entonces, reemplazamos la *feature* Color por tres *features*, ColorRojo, ColorAzul, y ColorVerde; poniendo los valores como 0 o 1 según sea apropiado.

Sensibilidad a la Escala

- SVM es muy sensible a la escala de las variables de entrada. Se recomienda que los valores de entrada estén entre $[-1, 1]$ o $[0, 1]$, y que el rango sea el mismo para todas las variables.
- Es importante definir una metodología razonable para el problema, y utilizarla consistentemente. Por ejemplo, *standard scaling*, *min-max scaling*, etcétera.
- Hay que tener mucho cuidado de determinar los parámetros de escala **antes** de separar el conjunto de datos en entrenamiento y validación, y escalar a ambos por igual.

No Free Lunch

La complejidad de computo de las funciones de *kernel* no es gratis desde un punto de vista de *performance* al momento de entrenar [5]:

	Linear		RBF Kernel	
Data set	Time	Accuracy	Time	Accuracy
MNIST38	0.1	96.82	38.1	99.70
ijcnn1	1.6	91.81	26.8	98.69
covtype	1.4	76.37	46,695.8	96.11
news20	1.1	96.95	383.2	96.90
real-sim	0.3	97.44	938.3	97.82
yahoo-japan	3.1	92.63	20,955.2	93.31
webspam	25.7	93.35	15,681.8	99.26

Siempre hay que tener en cuenta qué tanto nos cuesta ganar ese extra de *accuracy*, y si vale la pena.

SVM no escala

- Los algoritmos de entrenamiento son cuadráticos en la cantidad de elementos, entrenar en millones de elementos es un problema.
- Si hay muchos vectores de soporte, tanto el proceso de entrenamiento como el de clasificación se vuelven muy lentos.
- En los casos donde los datos tienen ruido, la cantidad de vectores de soporte se incrementa linealmente con los datos [11].

SVM no escala (Cont)

Hay varias soluciones a estos problemas

- Aproximar el *kernel*.
- Paralelizar el entrenamiento.
- Introducir *features* no-lineales. Por ejemplo, aleatorias [13].
- Utilizar sólo una parte del *kernel*.
- Hacer *early stopping* en la optimización cuando ya estamos en un punto bueno.
- Etcétera...

Un buen recurso para leer es [12].


"That's all Folks!"
Isberg©

Contenidos

Espacio Vectorial

Definición

Un espacio vectorial consiste en

- Un cuerpo $\mathbb{K}$ de escalares y un conjunto $\mathbb{V}$ de vectores
- Una operación de suma vectorial, que asocia a $\alpha, \beta \in \mathbb{V}$ un vector $\alpha + \beta \in \mathbb{V}$. Debe ser conmutativa, asociativa, tener un elemento neutro (llamado 0), y un elemento inverso (denotado $-\alpha$) tal que $\alpha + (-\alpha) = 0$.
- Una operación de producto por un escalar, que asocia a $c \in \mathbb{K}$ y $\alpha \in \mathbb{V}$ un vector $c\alpha \in \mathbb{V}$. Debe tener un elemento neutro (1), ser asociativa ($(c_1 c_2)\alpha = c_1(c_2\alpha)$) y distributiva sobre la suma (en ambos sentidos).

Vectores Ortogonales

Definición

Decimos que $\alpha, \beta \in \mathbb{V}$ son ortogonales si $\langle \alpha, \beta \rangle = 0$.

Valen toda una serie de teoremas:

- Un conjunto de vectores distintos de 0 y ortogonales entre sí es linealmente independiente.
- Dado un conjunto de vectores linealmente independientes, puede construirse otro conjunto de vectores linealmente independientes que generan al mismo subespacio, y además son ortogonales entre sí (Graham-Schmidt).
- Todo espacio vectorial de dimensión finita tiene una base ortogonal.

Complemento Ortogonal

Definición

Dado un subespacio $\mathbb{W} \subset \mathbb{V}$, definimos el complemento ortogonal de $\mathbb{W}$ como:

$$\mathbb{W}^{\perp} = \{v \in \mathbb{V} : \langle v, w \rangle = 0 \forall w \in \mathbb{W}\}$$

Teorema

Dado un $\mathbb{K}$-E.V. $\mathbb{V}$ de dimensión finita, y un subespacio $\mathbb{W} \subset \mathbb{V}$, podemos escribir

$$\mathbb{V} = \mathbb{W} \oplus \mathbb{W}^{\perp}$$

Proyección

Definición

Sea $\mathbb{V}$ un $\mathbb{K}$-E.V. de dimensión finita. $P : \mathbb{V} \to \mathbb{V}$ una transformación lineal. Decimos que P es una proyección si $P^2 = P$

Observación

Vale que

- $\mathbb{V} = \text{Im}(P) \oplus \text{Ker}(P)$
- P es la identidad sobre los vectores en $\text{Im}(P)$

Teorema

Si $\mathbb{V} = \mathbb{R} \oplus \mathbb{N}$, *existe una única proyección* P *tal que* $Im(P) = \mathbb{R}$ *y* $Ker(P) = \mathbb{N}$. *A* P *la llamamos la proyección en* $\mathbb{R}$ *sobre* $\mathbb{N}$.

La mejor aproximación

Definición

$\beta \in \mathbb{V}$, $\mathbb{W} \subset \mathbb{V}$ subespacio. $\alpha \in \mathbb{W}$ es la mejor aproximación de β por un vector de $\mathbb{W}$ si:

$$||\beta - \alpha|| \leq ||\beta - \gamma|| \forall \gamma \in \mathbb{W}$$

Teorema

- *α es la mejor aproximación de β* $\iff \langle \beta - \alpha, w \rangle = 0 \forall w \in \mathbb{W}$
- *La mejor aproximación es única*
- *$\mathbb{W}$ finito dimensional y $\alpha_1, ..., \alpha_n$ es una base ortogonal de $\mathbb{W}$, entonces la única mejor aproximación para β es:*

$$\alpha = \sum_k \frac{\langle \beta, \alpha_k \rangle}{||\alpha_k||^2} \alpha_k$$

Contenidos

Problema de Optimización

Un problema de optimización es

$$\begin{aligned} \underset{\boldsymbol{x}\in\mathcal{X}}{\text{minimizar}} \quad & f(\boldsymbol{x}) \\ \text{sujeto a} \quad & g(\boldsymbol{x}) \leq 0,\ i = 1, \ldots, m. \end{aligned}$$

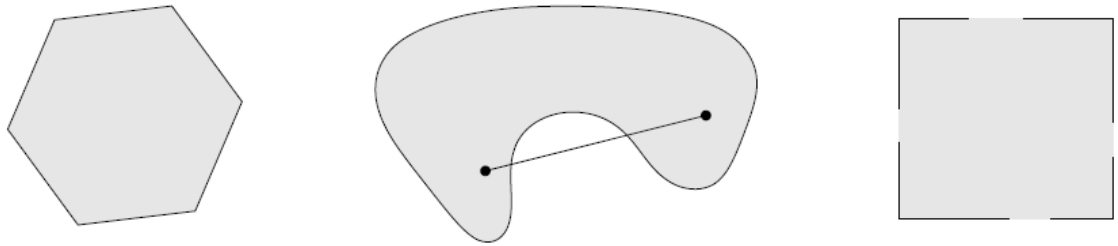
Con $f : \mathbb{R}^n \to \mathbb{R}$, $f : \mathbb{R}^n \to \mathbb{R}^m$. f se denomina función objetivo, y las componentes de g se denominan restricciones. Un vector p^* se llama óptimo, o una solución del problema, si es el que minimiza la función objetivo dentro de los que cumplen las restricciones.

Conjunto Convexo

Un conjunto S es convexo si $\forall x, y \in S$ y $\theta \in [0, 1]$ tenemos que

$$\theta x + (1 - \theta)y \in S$$

Es decir, si la recta que une a x con y está contenida dentro del conjunto. Equivalentemente, se puede pedir que contenga cualquier combinación pesada de puntos. Un par de ejemplos:

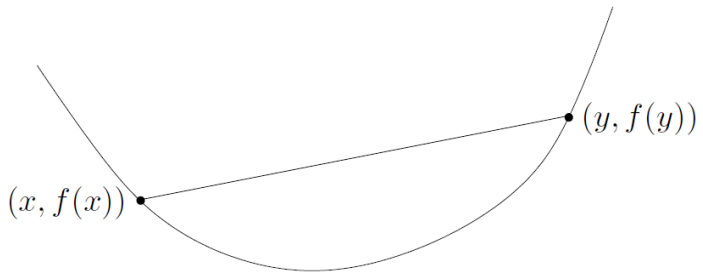


Funciónes Concavas y Convexas

Una función $f : \mathbb{R}^n \to \mathbb{R}$ se dice convexa si Dom(f) es un conjunto convexo y, además, $\forall x, y \in$ Dom(f) y $\alpha, \beta \in \mathbb{R}$

$$f(\alpha x + \beta y) \leq \alpha f(x) + \beta f(y)$$

Gráficamente, una función es convexa si cumple esto para cualquier par de puntos:



El caso de cóncava es dar vuelta la desigualdad.

Lagrangiano

Definición

El Lagrangiano asociado a un problema de optimización es una función $\mathcal{L} : \mathcal{X} \times \mathbb{R}_+ \to \mathbb{R}$ tal que:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\alpha}) = f(\boldsymbol{x}) + \sum_{i=1}^{m} \alpha_i g(\boldsymbol{x})$$

Donde las variables α_i se denominan variables de Lagrange o duales.

Función Dual

Definición

La función dual de Lagrange asociada a un problema de optimización restringido está definida por $F : \mathbb{R}_+ \to \mathbb{R}$ tal que:

$$F(\boldsymbol{\alpha}) = \inf_{\boldsymbol{x} \in \mathcal{X}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\alpha})$$

F siempre es cóncava, y además:

$$F(\boldsymbol{\alpha}) \leq p*$$

Porque el sumando a $f(\boldsymbol{x})$ en el ínfimo siempre es negativo.

Problema Dual

La función dual introduce un problema de optimización:

$$\begin{aligned} \underset{\boldsymbol{\alpha} \in \mathbb{R}_+}{\text{maximizar}} \quad & F(\boldsymbol{\alpha}) \\ \text{sujeto a} \quad & \boldsymbol{\alpha} \geq 0. \end{aligned}$$

Que siempre es un problema de optimización convexa, porque F es cóncava. Además, si tomamos a d^* como su óptimo, vale la **dualidad débil**:

$$d^* \leq p^*$$

Por la propiedad del slide anterior.

Dualidad Fuerte y Calificación

El caso en que $d^* = p^*$ se denomina **dualidad fuerte**. Eso sucede cuando los problemas satisfacen ciertas condiciones sobre sus restricciones.

Definición

Supongamos que interior$(\mathcal{X}) \neq \emptyset$. Entonces, la Condición de Restricción Fuerte, o condición de Slater, dice que:

$$\exists \boldsymbol{x} \in \text{interior}(\mathcal{X}) : g(\boldsymbol{x}) < 0$$

Definición

Dada una función afín (es decir, $h(x) = w \cdot x + b$), y si suponemos que interior$(\mathcal{X}) \neq \emptyset$, la Condición de Restricción Débil, o condición de Slater débil, dice que:

$$\exists \boldsymbol{x} \in \text{interior}(\mathcal{X}) : \forall i \in [1, m], (g_i(\boldsymbol{x}) < 0) \vee (g_i(\boldsymbol{x}) = 0 \wedge g_i \text{es afín})$$

Puntos Silla

Teorema

Sea P un problema de optimización restringido sobre $\mathcal{X} = \mathbb{R}^N$. Si $(\boldsymbol{x}^*, \boldsymbol{\alpha}*)$ es un punto silla del Lagrangiano asociado, es decir:

$$\forall \boldsymbol{x} \in \mathbb{R}^N, \forall \alpha \geq 0, \mathcal{L}(\boldsymbol{x}^*, \boldsymbol{\alpha}) \leq \mathcal{L}(\boldsymbol{x}^*, \boldsymbol{\alpha}^*) \leq \mathcal{L}(\boldsymbol{x}, \boldsymbol{\alpha}^*)$$

Entonces $(\boldsymbol{x}^*, \boldsymbol{\alpha}^*)$ es una solución del problema P.

Teorema

f y g_is convexas, vale Slater. Si $\boldsymbol{x}$ es una solución al problema de optimización restringida, $\exists \boldsymbol{\alpha} / (\boldsymbol{x}, \boldsymbol{\alpha})$ es punto silla del Lagrangiano.

Teorema

f y g_is convexas y diferenciables, vale Slater **débil**. *Si $\boldsymbol{x}$ es una solución al problema de optimización restringida, $\exists \boldsymbol{\alpha} / (\boldsymbol{x}, \boldsymbol{\alpha})$ es punto silla del Lagrangiano.*

Condiciones de Karush-Kuhn-Tucker

Teorema

Si la función objetivo f y las restricciones g son convexas y diferenciables, y las restricciones son calificadas, entonces x es una solución del problema sí y sólo sí $\exists \alpha \geq 0$ tal que

$$\nabla_x \mathcal{L}(x, \alpha) = \nabla_x f(x) + \alpha \nabla_x g(x) = 0$$

$$\nabla_\alpha \mathcal{L}(x, \alpha) = g(x) \leq 0$$

$$\alpha \cdot g(x) = \sum_{i=1}^{m} \alpha_i g(x_i) = 0$$

Contenidos

Bibliografía I


Mehryar Mohri, Afshihn Rostamizadeh, Ameet Talwalkar
Foundations of Machine Learning.
MIT Press, 2012.


Christopher M. Bishop
Pattern Recognition and Machine Learning
Springer, 2006.


Yaser Abu-Mostafa
Lecture 15 - Kernel Methods
Caltech University, 2012.


Thorsten Joachims
Text Categorization with Suport Vector Machines: Learning with Many Relevant Features
Springer-Verlag, 1998.

Bibliografía II


Chih-Jen Lin
Machine Learning Software: Design and Practical Use
Machine Learning Summer School, 2012.


S. Sathiya Keerthi, Chih-Jen Lin
Asymptotic Behaviors of Support Vector Machines with Gaussian Kernel
National University of Singapore, National Taiwan University, 2003.


Hsuan-Tien Lin, Chih-Jen Lin
A Study on Sigmoid Kernels for SVM and the Training of non-PSD Kernels by SMO-type Methods
National Taiwan University, 2003.

Bibliografía III

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, Chris Watkins

Text Classification using String Kernels

The Journal of Machine Learning Research, 2002.

S. V. N. Vishwanathan, Nicol N. Schraudolph, Risi Kondor, Karsten M. Borgwardt

Graph Kernels

The Journal of Machine Learning Research, 2010.

Hsin-Yuan Huang, Chih-Jen Lin

Linear and Kernel Classification: When to Use Which?

National Taiwan University.

Bibliografía IV


Ingo Steinwart
Sparseness of support vector machines
The Journal of Machine Learning Research, 2003.


Chih-Jen Lin
Support Vector Machines and Kernel Methods: Status and Challenges
K. U. Leuven Optimization in Engineering Center, 2013.


Ali Rahimi, Benjamin Recht
Random Features for Large-Scale Kernel Machines
Advances in Neural Information Processing Systems 20, 2007.