SVM: Kernels y SMO

Lección 12

Dr. Pablo Alvarado Moya

CE5506 Introducción al reconocimiento de patrones
Área de Ingeniería en Computadores
Tecnológico de Costa Rica

II Semestre, 2019

Contenido

Repaso (1)

- Iniciamos definiendo márgenes funcional y geométrico:

$$\hat{\gamma}^{(i)} = y^{(i)}(\underline{\mathbf{w}}^T \underline{\mathbf{x}} + b) \qquad \gamma^{(i)} = \frac{\hat{\gamma}^{(i)}}{\|\underline{\mathbf{w}}\|}$$

  asociados a distancia de cada dato a frontera de decisión.

- Con estos definimos el clasificador de margen máximo

$$\begin{aligned} &\underset{\gamma,\underline{\mathbf{w}},b}{\text{máx}} \gamma \\ &\text{sujeto a } y^{(i)}(\underline{\mathbf{w}}^T \underline{\mathbf{x}}^{(i)} + b) \geq \gamma, \quad i = 1,\ldots,m \\ &\|\underline{\mathbf{w}}\| = 1 \end{aligned}$$

  que es difícil de trabajar.

Repaso (2)

- Este problema se reformuló hasta llegar al clasificador de margen óptimo, que sí es convexo:

$$\min_{\gamma,\underline{\mathbf{w}},b} \frac{1}{2}\|\underline{\mathbf{w}}\|^2$$

$$\text{sujeto a } y^{(i)}(\underline{\mathbf{w}}^T\underline{\mathbf{x}}^{(i)} + b) \geq 1, \quad i = 1,\ldots,m$$

que tiene como problema dual:

$$\max_{\underline{\boldsymbol{\alpha}}} \left(\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \left\langle \underline{\mathbf{x}}^{(i)}, \underline{\mathbf{x}}^{(j)} \right\rangle \right)$$

$$\text{sujeto a } \alpha_i \geq 0, \quad i = 1,\ldots,m$$

$$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$$

Repaso (3)

- Por condiciones KKT solo pocos $\alpha_i > 0$, corresponden a vectores de soporte $\underline{\mathbf{x}}^{(i)}$.
- Para predecir luego de varias manipulaciones llegamos a que

$$y = \underline{\mathbf{w}}^T \underline{\mathbf{x}} + b = \sum_{i=1}^{m} \alpha_i y^{(i)} \left\langle \underline{\mathbf{x}}^{(i)}, \underline{\mathbf{x}} \right\rangle + b$$

  que solo depende del producto punto de la entrada con los vectores de soporte
- Tanto entrenamiento como predicción dependen solo de productos internos, por lo que podemos usar el **truco del kernel** y usar $K(\underline{\mathbf{x}}, \underline{\mathbf{z}}) = \phi(\underline{\mathbf{x}})^T \phi(\underline{\mathbf{z}})$

Clasificador de margen óptimo con kernels

- Problema de optimización:

$$\max_{\underline{\boldsymbol{\alpha}}} \left(\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j K(\underline{\mathbf{x}}^{(i)}, \underline{\mathbf{x}}^{(j)}) \right)$$

$$\text{sujeto a } \alpha_i \geq 0, \quad i = 1, \ldots, m$$

$$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$$

- Predicción:

$$y = \sum_{i=1}^{m} \alpha_i y^{(i)} K(\underline{\mathbf{x}}^{(i)}, \underline{\mathbf{x}}) + b$$

Kernels

Kernels

- Kernels evitan evaluación explícita de mapeo $\phi(\underline{\mathbf{x}})$ entre espacios de entrada y de características
- Evaluación del kernel es computacionalmente viable para hacer mapeo **implicito** de dos vectores al espacio de características y hacer allí su producto punto.
- Revisaremos varios mapeos usuales
- Visualización

Kernel cuadrático

- Por ejemplo, revisemos el kernel $K(\underline{\mathbf{x}},\underline{\mathbf{z}}) = (\underline{\mathbf{x}}^T\underline{\mathbf{z}})^2$

$$K(\underline{\mathbf{x}},\underline{\mathbf{z}}) = \left(\sum_{i=1}^{n} x_i z_i\right)\left(\sum_{j=1}^{n} x_j z_j\right) = \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j z_i z_j$$
$$= \sum_{i,j=1}^{n} (x_i x_j)(z_i z_j)$$

- Con $n = 3$ si $K(\underline{\mathbf{x}},\underline{\mathbf{z}}) = \phi(\underline{\mathbf{x}})^T\phi(\underline{\mathbf{z}})$ entonces

$$\phi(\underline{\mathbf{x}}) = \begin{bmatrix} x_1x_1 & x_1x_2 & x_1x_3 & x_2x_1 & x_2x_2 & x_2x_3 & x_3x_1 & x_3x_2 & x_3x_3 \end{bmatrix}^T$$

- Calcular $\phi(\underline{\mathbf{x}})$ tiene $\mathcal{O}(n^2)$ pero evaluar $K(\underline{\mathbf{x}},\underline{\mathbf{z}})$ es $\mathcal{O}(n)$

Kernel cuadrático con desplazamiento (1)

- Otro kernel similar $K(\underline{\mathbf{x}},\underline{\mathbf{z}}) = (\underline{\mathbf{x}}^T\underline{\mathbf{z}} + c)^2$ se reexpresa:

$$K(\underline{\mathbf{x}},\underline{\mathbf{z}}) = \sum_{i,j}^{n}(x_i x_j)(z_i z_j) + \sum_{i=1}^{n}(\sqrt{2c}x_i)(\sqrt{2c}z_i) + c^2$$

- Para $n = 3$ el mapeo correspondiente es:

$$\phi(\underline{\mathbf{x}}) = \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \\ \sqrt{2c}x_1 \\ \sqrt{2c}x_2 \\ \sqrt{2c}x_3 \\ c \end{bmatrix}$$

Kernel cuadrático con desplazamiento (2)

- El parámetro c controla el peso relativo entre los términos de primer orden x_i y los de segundo orden $x_i x_j$

Kernel polinomial

- Si usamos $K(\underline{\mathbf{x}},\underline{\mathbf{z}}) = (\underline{\mathbf{x}}^T\underline{\mathbf{z}} + c)^q$ realizamos un mapeo a un espacio de características con $\binom{n+q}{q} = \frac{(n+q)!}{n!q!}$ dimensiones
- ¡Evaluar el kernel sigue siendo $\mathcal{O}(n)$!

Similitud por producto interno

- Otra manera de ver los kernels tiene que ver con su relación con **similitud** entre vectores
- Si $\phi(\underline{\mathbf{x}})$ y $\phi(\underline{\mathbf{z}})$ son muy similares, entonces su producto $K(\underline{\mathbf{x}},\underline{\mathbf{z}}) = \phi(\underline{\mathbf{x}})^T \phi(\underline{\mathbf{z}})$ se espera sea grande.
- Por otro lado, si $\phi(\underline{\mathbf{x}})$ y $\phi(\underline{\mathbf{z}})$ son distantes (casi ortogonales), entonces $K(\underline{\mathbf{x}},\underline{\mathbf{z}}) = \phi(\underline{\mathbf{x}})^T \phi(\underline{\mathbf{z}})$ será muy pequeño
- De este modo, hasta cierto punto $K(\underline{\mathbf{x}},\underline{\mathbf{z}})$ codifica la similitud entre $\phi(\underline{\mathbf{x}})$ y $\phi(\underline{\mathbf{z}})$, e indirectamente qué tan similares son $\underline{\mathbf{x}}$ y $\underline{\mathbf{z}}$
- Con esta idea podemos diseñar kernels que explicitamente codifiquen similitud

Kernels de base radial

- Un kernel que busca codificar similitud es:

$$K(\underline{\mathbf{x}},\underline{\mathbf{z}}) = \exp\left(-\frac{\|\underline{\mathbf{x}} - \underline{\mathbf{z}}\|^2}{2\sigma^2}\right)$$

  que toma valores cercanos a 1 si los vectores $\underline{\mathbf{x}}$ y $\underline{\mathbf{z}}$ son similares, y 0 si se encuentran aparte
- Este llamado kernel **gaussiano** mapea el espacio de entrada a un espacio de características de infinitas dimensiones.

Matriz de Kernel (1)

- ¿Qué propiedades debe tener $K(\underline{\mathbf{x}},\underline{\mathbf{z}})$ para que exista un mapeo asociado ϕ tal que $K(\underline{\mathbf{x}},\underline{\mathbf{z}}) = \phi(\underline{\mathbf{x}})^T \phi(\underline{\mathbf{z}})$?
- Considérese un conjunto de m puntos cualquiera $\{\underline{\mathbf{x}}^{(1)}, \underline{\mathbf{x}}^{(2)}, \ldots, \underline{\mathbf{x}}^{(m)}\}$ y defínase la **matriz de kernel** como

$$\mathbf{K} = \begin{bmatrix} K_{11} & K_{12} & \ldots & K_{1m} \\ K_{21} & K_{22} & \ldots & K_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ K_{m1} & K_{m2} & \ldots & K_{mm} \end{bmatrix}$$

  con $K_{ij} = K(\underline{\mathbf{x}}^{(i)},\underline{\mathbf{x}}^{(j)})$
- Puesto que el producto escalar es conmutativo, la matriz $\mathbf{K}$ es simétrica.

Matriz de Kernel (2)

- Si $\phi_k(\underline{\mathbf{x}})$ denota la k-ésima componente del vector en el espacio de características entonces, para cualquier vector $\underline{\mathbf{z}}$ se cumple:

$$\begin{aligned}
\underline{\mathbf{z}}^T \mathbf{K} \underline{\mathbf{z}} &= \sum_i \sum_j z_i K_{ij} z_j = \sum_i \sum_j z_i \phi(\underline{\mathbf{x}}^{(i)})^T \phi(\underline{\mathbf{x}}^{(j)}) z_j \\
&= \sum_i \sum_j z_i \sum_k \phi_k(\underline{\mathbf{x}}^{(i)}) \phi_k(\underline{\mathbf{x}}^{(j)}) z_j \\
&= \sum_k \sum_i \sum_j z_i \phi_k(\underline{\mathbf{x}}^{(i)}) \phi_k(\underline{\mathbf{x}}^{(j)}) z_j \\
&= \sum_k \left(\sum_i z_i \phi_k(\underline{\mathbf{x}}^{(i)}) \right)^2 \geq 0
\end{aligned}$$

lo que implica que $\mathbf{K}$ es positiva semi-definida.

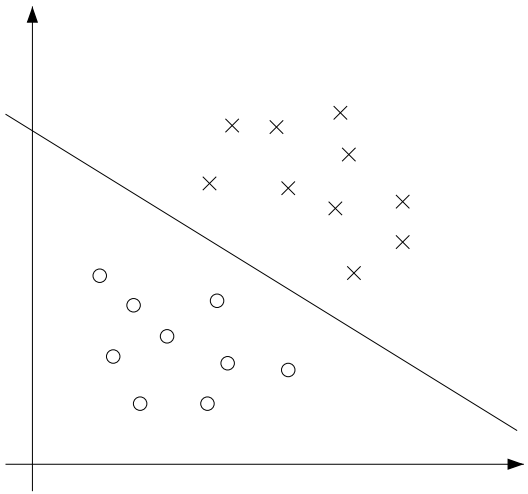
Matriz de Kernel (3)

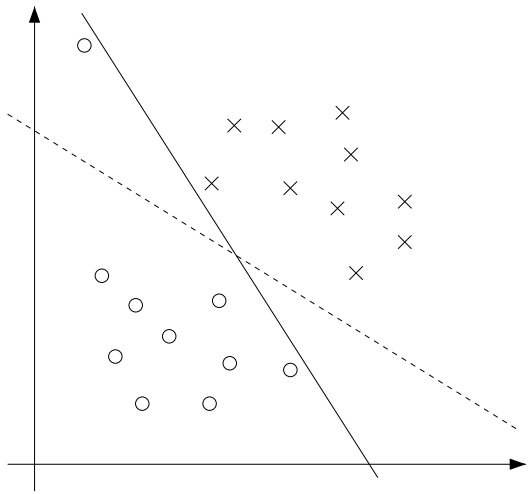
- Mercer demostró que basta con que **K** sea positiva semi-definida, para que K sea un kernel válido (kernel de Mercer)
- Esto implica: no es necesario conocer $\phi(\underline{\mathbf{x}})$. Basta con saber que **K** (para un conjunto de datos) es simétrica y positiva semidefinida, para que el mapeo ϕ exista, aunque nunca deberá calcularse explícitamente.
- Este concepto de kernel se aplica a gran cantidad de otros problemas adicionalmente a los SVM: basta con que los problemas se puedan plantear en términos de productos escalares únicamente.

Regularización y el caso no separable

Regularización ℓ_1 (1)

- Hasta ahora hemos supuesto datos linealmente separables
- Datos perdidos o atípicos (*outliers*) pueden arruinar el cálculo de la frontera de decisión, aún con el truco del kernel





Regularización ℓ_1 (2)

- Para aumentar robustez al ruido vamos a replantear el problema de optimización una vez más:

$$\min_{\gamma,\underline{\mathbf{w}},b} \frac{1}{2}\|\underline{\mathbf{w}}\|^2 + C\sum_{i=1}^{m}\xi_i$$

$$\text{sujeto a } y^{(i)}\left(\underline{\mathbf{w}}^T\underline{\mathbf{x}}^{(i)} + b\right) \geq 1-\xi_i, \quad i = 1,\ldots,m$$

$$\xi_i \geq 0, \quad i = 1,\ldots,m$$

- Con parámetros **de holgura** ξ_i (*slack variables*) permitimos márgenes funcionales menores a 1, pero pagando un costo $C\xi_i$
- C controla la ponderación del castigo de usar márgenes funcionales menores a 1.

Lagrangiano de problema regularizado (1)

- El Langrangiano es ahora

$$\mathcal{L}(\underline{\mathbf{w}}, b, \underline{\boldsymbol{\xi}}, \underline{\boldsymbol{\alpha}}, \underline{\mathbf{r}}) = \frac{1}{2}\underline{\mathbf{w}}^T\underline{\mathbf{w}} + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\alpha_i\left[y^{(i)}(\underline{\mathbf{x}}^T\underline{\mathbf{w}} + b) - 1 + \xi_i\right] - \sum_{i=1}^{m} r_i\xi_i$$

- $\alpha_i \geq 0$ y $r_i \geq 0$ son los multiplicadores de Lagrange

Problema dual del problema regularizado (1)

- El problema dual está dado por

$$\max_{\underline{\boldsymbol{\alpha}}} (W(\underline{\boldsymbol{\alpha}})) = \max_{\underline{\boldsymbol{\alpha}}} \left(\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \left\langle x^{(i)}, x^{(j)} \right\rangle \right)$$

$$\text{sujeto a } 0 \leq \alpha_i \leq C, i = 1, \ldots, m$$

$$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$$

- De nuevo $\underline{\mathbf{w}} = \sum_{i=1}^{m} \alpha_i y^{(i)} \underline{\mathbf{x}}^{(i)}$, de modo que podemos predecir con:

$$\underline{\mathbf{w}}^T \underline{\mathbf{x}} + b = \sum_{i=1}^{m} \alpha_i y^{(i)} \left\langle \underline{\mathbf{x}}^{(i)}, \underline{\mathbf{x}} \right\rangle + b$$

Problema dual del problema regularizado (2)

- Interesante es el hecho de que al agregar la regularización ℓ_1 el **único** cambio al problema dual es que la restricción original $0 \leq \alpha_i$ pasó a ser ahora $0 \leq \alpha_i \leq C$
- Convergencia de los α_i se deriva de las condiciones KKT:

$$\begin{aligned} \alpha_i = 0 &\implies y^{(i)}(\underline{\mathbf{w}}^T \underline{\mathbf{x}}^{(i)} + b) \geq 1 \\ \alpha_i = C &\implies y^{(i)}(\underline{\mathbf{w}}^T \underline{\mathbf{x}}^{(i)} + b) \leq 1 \\ 0 < \alpha_i < C &\implies y^{(i)}(\underline{\mathbf{w}}^T \underline{\mathbf{x}}^{(i)} + b) = 1 \end{aligned}$$

- El cálculo de b también cambia
- Este algoritmo se denomina SVM regularizado con ℓ_1 con márgenes suaves (ℓ_1 norm soft margin SVM)
- Permite fronteras no lineales y manejar puntos atípicos

Algoritmo de Optimización Mínima Secuencial (SMO)

[John Platt, 1998]

Optimización mínima secuencial

- Nos queda definir un algoritmo capaz de resolver los problemas duales presentados hasta ahora
- Antes de entrar a revisar el SMO, presentaremos el algoritmo de ascenso de coordenadas

Ascenso de coordinadas (1)

Coordinate ascent

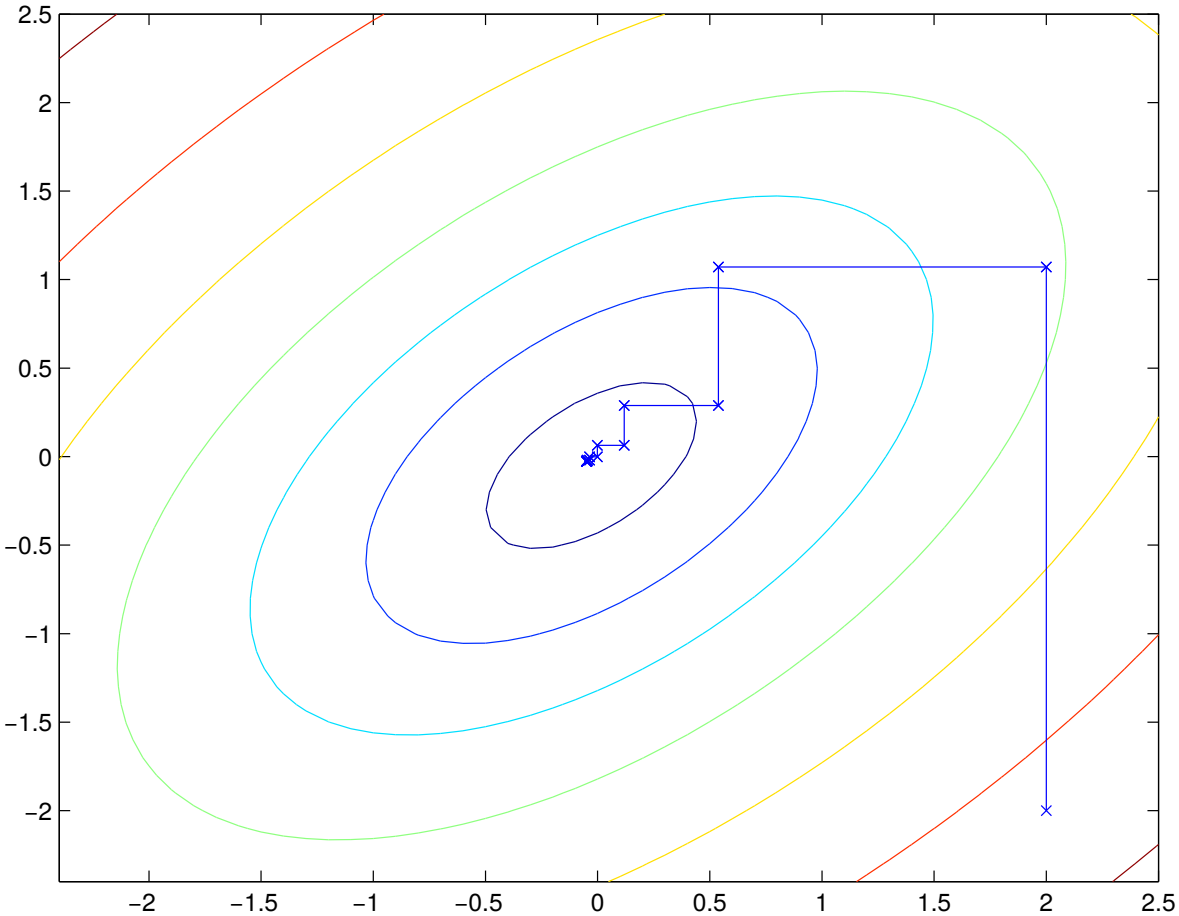
- Considérese el problema sin restricciones

$$\max_{\underline{\boldsymbol{\alpha}}} W(\alpha_1, \alpha_2, \ldots, \alpha_m)$$

- Esto podría solucionarse por ascenso de gradiente o el método de Newton.
- Otro método más simple se denomina **ascenso de coordenadas**

Ascenso de coordinadas (2)

Coordinate ascent



Ascenso de coordinadas (3)

Coordinate ascent

- Cada componente del vector $\underline{\boldsymbol{\alpha}}$ se optimiza aislada

repeat
  for $i = 1, \ldots, m$ **do**
    $\alpha_i \leftarrow \arg\max_{\hat{\alpha}_i} W(\alpha_1, \ldots, \alpha_{i-1}, \hat{\alpha}_i, \alpha_{i+1}, \ldots, \alpha_m)$
  end
until *convergence*;

- Existen varias estrategias para el orden de las componentes:
  - Componentes en orden predeterminado (ver seudocódigo)
  - Orden dinámico maximizando probabilidad de ascenso
- Si la función W tiene forma tal que el arg máx se puede realizar eficientemente, este algoritmo es eficiente

Optimización mínima secuencial (1)

Sequential minimal optimization (SMO)

- Recordemos problema dual a optimizar:

$$\max_{\underline{\boldsymbol{\alpha}}} \left(W(\underline{\boldsymbol{\alpha}}) \right) = \max_{\underline{\boldsymbol{\alpha}}} \underbrace{\left(\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j K(\underline{\mathbf{x}}^{(i)}, \underline{\mathbf{x}}^{(j)}) \right)}_{W(\underline{\boldsymbol{\alpha}})}$$

sujeto a $0 \leq \alpha_i < C, \quad i = 1, \ldots, m$

$$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$$

Optimización mínima secuencial (2)

Sequential minimal optimization (SMO)

- Considerando que $y^{(i)} \in \{-1,1\}$ y por tanto $(y^{(i)})^2 = 1$, de la segunda condición tenemos

$$y^{(j)}\alpha_j = -\sum_{\substack{i=1 \\ i\neq j}}^{m} \alpha_i y^{(i)}$$

$$\alpha_j = -\frac{1}{y^{(j)}} \sum_{\substack{i=1 \\ i\neq j}}^{m} \alpha_i y^{(i)} = -y^{(j)} \sum_{\substack{i=1 \\ i\neq j}}^{m} \alpha_i y^{(i)}$$

  es decir, un solo α_j está restringido por los otros términos y no puede optimizarse de forma aislada

- Necesitamos entonces optimizar **al menos dos** α_j y α_k simultáneamente

Optimización mínima secuencial (3)

Sequential minimal optimization (SMO)

- El algoritmo SMO (Optimización mínima secuencial) en principio hace:

  repeat

  1. $[j,k] \leftarrow$Select$(\underline{\boldsymbol{\alpha}}, W(\cdot))$;
  2. $\underline{\boldsymbol{\alpha}} \leftarrow$Reoptimize$(j,k,\underline{\boldsymbol{\alpha}}, W(\cdot))$;

  until *convergence*;

- La verificación de convergencia utiliza la condición dual complementaria KKT:

$$\begin{aligned} \alpha_i = 0 &\implies y^{(i)}(\underline{\mathbf{w}}^T \underline{\mathbf{x}}^{(i)} + b) \geq 1 \\ \alpha_i = C &\implies y^{(i)}(\underline{\mathbf{w}}^T \underline{\mathbf{x}}^{(i)} + b) \leq 1 \\ 0 < \alpha_i < C &\implies y^{(i)}(\underline{\mathbf{w}}^T \underline{\mathbf{x}}^{(i)} + b) = 1 \end{aligned}$$

Optimización mínima secuencial (4)

Sequential minimal optimization (SMO)

- SMO es eficiente porque la actualización de α_i y α_j se puede hacer de forma muy eficiente.
- Seleccionemos α_j y α_k y el resto las dejamos fijas:

$$\alpha_j y^{(j)} + \alpha_k y^{(k)} = - \sum_{\substack{i=1 \\ i \neq j, i \neq k}}^{m} \alpha_i y^{(i)} = \zeta = \text{cte}$$

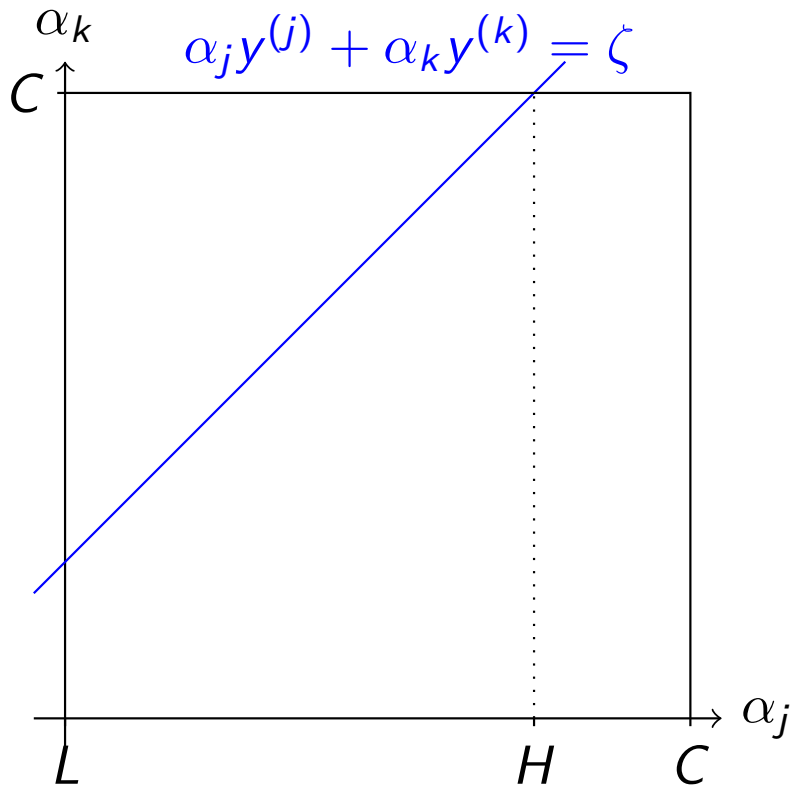
de donde se despeja por ejemplo

$$\alpha_k = \zeta y^{(k)} - \alpha_j (y^{(j)} y^{(k)})$$

que tiene pendiente $(y^{(j)} y^{(k)}) = \pm 1$

Optimización mínima secuencial (5)

Sequential minimal optimization (SMO)



- $\alpha_j, \alpha_k \in [0, C]$ obliga a $\alpha_j \in [L, H] \subseteq [0, C]$

Optimización mínima secuencial (6)

Sequential minimal optimization (SMO)

- Con todas las componentes de $\underline{\boldsymbol{\alpha}}$ constantes excepto α_j y $\alpha_k = y^{(k)}(\zeta - \alpha_j y^{(j)})$, entonces $W(\underline{\boldsymbol{\alpha}})$ es una función cuadrática de α_j

$$W(\underline{\boldsymbol{\alpha}}) = a\alpha_j^2 + b\alpha_j + c$$

con máximo en $\hat{\alpha}_j = -\frac{b}{2a}$

- Para encontrar el máximo que satisface las condiciones verificamos:

$$\alpha_j^* = \begin{cases} H & \text{si } \hat{\alpha}_j > H \\ \hat{\alpha}_j & \text{si } L \leq \hat{\alpha}_j \leq H \\ L & \text{si } \hat{\alpha}_j < L \end{cases}$$

y con α_j^* obtenemos $\alpha_k^* = y^{(k)}(\zeta - \alpha_j^* y^{(j)})$

Optimización mínima secuencial (7)

Sequential minimal optimization (SMO)

- El algoritmo de Platt especifica heurísticos para determinar j y k en cada iteración, y cómo actualizar además b

Más de dos clases

- Para problemas con k clases distintas usan dos estrategias:
  - Uno-contra-todos (*one-against-all*)
  - Uno-contra-uno (*one-against-one*)
- **Uno-contra-uno**: entrena $k(k-1)/2$ clasificadores, uno para cada par de clases distintas. Cada clasificador emite un voto por la clase ganadora cuando se predice.
- **Uno-contra-todos**: entrena k clasificadores, uno para cada clase contra el resto. La elección del ganador en predicción utiliza un clasificador softmax sobre las salidas de los k clasificadores, que debe ser previamente entrenado.

Aplicaciones de SVM

- MNIST fue clasificado exitosamente con SVM con kernels polinomiales y gaussianos, que se comparaba a redes neuronales de la época
- En aplicaciones de biología, se diseñan kernels que calculan eficientemente la comparación de secuencias de aminoácidos

Resumen

Este documento ha sido elaborado con software libre incluyendo LaTeX, Beamer, GNUPlot, GNU/Octave, XFig, Inkscape, GNU-Make y Subversion en GNU/Linux