

Learning From Data

Lecture 25

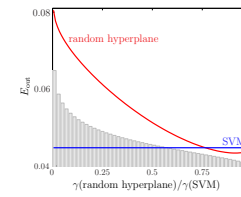
The Kernel Trick

Learning with only inner products
The Kernel

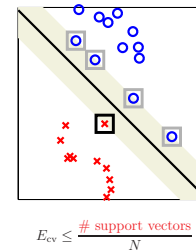
M. Magdon-Ismail
CSCI 4100/6100

RECAP: Large Margin is Better

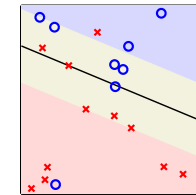
Controlling Overfitting



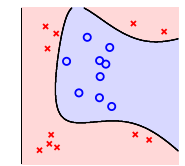
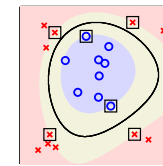
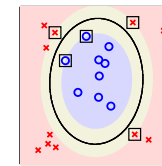
Theorem. $d_{vc}(\gamma) \leq \left\lceil \frac{R^2}{\gamma^2} \right\rceil + 1$



Non-Separable Data

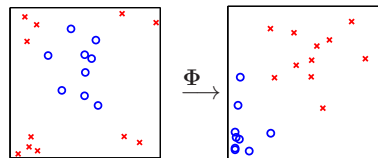


minimize $\frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{n=1}^N \xi_n$
 subject to: $y_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1 - \xi_n$
 $\xi_n \geq 0$ for $n = 1, \dots, N$

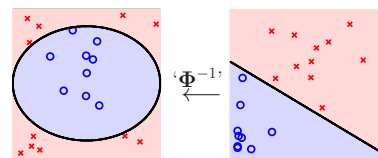


Complex hypothesis that does not overfit because it is 'simple', controlled by only a few support vectors.

Recall: Mechanics of the Nonlinear Transform



1. Original data $\mathbf{x}_n \in \mathcal{X}$
2. Transform the data $\mathbf{z}_n = \Phi(\mathbf{x}_n) \in \mathcal{Z}$



4. Classify in \mathcal{X} -space $g(\mathbf{x}) = \hat{g}(\Phi(\mathbf{x})) = \text{sign}(\hat{\mathbf{w}}^T \Phi(\mathbf{x}))$
3. Separate data in \mathcal{Z} -space $\hat{g}(\mathbf{z}) = \text{sign}(\hat{\mathbf{w}}^T \mathbf{z})$

\mathcal{X} -space is \mathbb{R}^d

$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix}$$

$\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$

y_1, y_2, \dots, y_N

no weights

$d_{vc} = d + 1$

$g(\mathbf{x}) = \text{sign}(\hat{\mathbf{w}}^T \Phi(\mathbf{x}))$

\mathcal{Z} -space is $\mathbb{R}^{\hat{d}}$

$$\mathbf{z} = \Phi(\mathbf{x}) = \begin{bmatrix} 1 \\ \Phi_1(\mathbf{x}) \\ \vdots \\ \Phi_{\hat{d}}(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 1 \\ z_1 \\ \vdots \\ z_{\hat{d}} \end{bmatrix}$$

$\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_N$

y_1, y_2, \dots, y_N

$$\hat{\mathbf{w}} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{\hat{d}} \end{bmatrix}$$

$d_{vc} = \hat{d} + 1$

Have to **transform** the data to the \mathcal{Z} -space.

This Lecture

How to use nonlinear transforms without **physically transforming** data to \mathcal{Z} -space.

Primal Versus Dual

Primal

$$\text{minimize}_{\mathbf{w}, b} \frac{1}{2} \mathbf{w}^T \mathbf{w}$$

$$\text{subject to: } y_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1 \text{ for } n = 1, \dots, N$$

$$g(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x} + b)$$

$d + 1$ optimization variables \mathbf{w}, b

Dual

$$\text{minimize}_{\alpha} \frac{1}{2} \sum_{n,m=1}^N \alpha_n \alpha_m y_n y_m (\mathbf{x}_n^T \mathbf{x}_m) - \sum_{n=1}^N \alpha_n$$

$$\text{subject to: } \sum_{n=1}^N \alpha_n y_n = 0$$

$$\alpha_n \geq 0 \text{ for } n = 1, \dots, N$$

$$\mathbf{w}^* = \sum_{n=1}^N \alpha_n^* y_n \mathbf{x}_n$$

$$b^* = y_s - \mathbf{w}^{*T} \mathbf{x}_s \quad (\alpha_s^* > 0)$$

$$\begin{aligned} g(\mathbf{x}) &= \text{sign}(\mathbf{w}^{*T} \mathbf{x} + b^*) \\ &= \text{sign} \left(\sum_{n=1}^N \alpha_n^* y_n \mathbf{x}_n^T (\mathbf{x} - \mathbf{x}_s) + y_s \right) \end{aligned}$$

N optimization variables α

Kernel Trick: 5 / 18

Vector-matrix form \rightarrow

Primal Versus Dual - Matrix Vector Form

Primal

$$\text{minimize}_{\mathbf{w}, b} \frac{1}{2} \mathbf{w}^T \mathbf{w}$$

$$\text{subject to: } y_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1 \text{ for } n = 1, \dots, N$$

$$g(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x} + b)$$

$d + 1$ optimization variables \mathbf{w}, b

Dual

$$\text{minimize}_{\alpha} \frac{1}{2} \alpha^T G \alpha - \mathbf{1}^T \alpha \quad (G_{nm} = y_n y_m \mathbf{x}_n^T \mathbf{x}_m)$$

$$\text{subject to: } \mathbf{y}^T \alpha = 0$$

$$\alpha \geq 0$$

$$\mathbf{w}^* = \sum_{n=1}^N \alpha_n^* y_n \mathbf{x}_n$$

$$b^* = y_s - \mathbf{w}^{*T} \mathbf{x}_s \quad (\alpha_s^* > 0)$$

$$\begin{aligned} g(\mathbf{x}) &= \text{sign}(\mathbf{w}^{*T} \mathbf{x} + b^*) \\ &= \text{sign} \left(\sum_{n=1}^N \alpha_n^* y_n \mathbf{x}_n^T (\mathbf{x} - \mathbf{x}_s) + y_s \right) \end{aligned}$$

N optimization variables α

Kernel Trick: 6 / 18

The Lagrangian \rightarrow

Deriving the Dual: The Lagrangian

$$\mathcal{L} = \frac{1}{2} \mathbf{w}^T \mathbf{w} + \sum_{n=1}^N \alpha_n \cdot (1 - y_n(\mathbf{w}^T \mathbf{x}_n + b))$$

\uparrow Lagrange multipliers \uparrow the constraints

minimize w.r.t. b, \mathbf{w} \leftarrow unconstrained

maximize w.r.t. $\alpha \geq 0$

Intuition

- $1 - y_n(\mathbf{w}^T \mathbf{x}_n + b) > 0 \implies \alpha_n \rightarrow \infty$ gives $\mathcal{L} \rightarrow \infty$
- Choose (b, \mathbf{w}) to min \mathcal{L} , so $1 - y_n(\mathbf{w}^T \mathbf{x}_n + b) \leq 0$
- $1 - y_n(\mathbf{w}^T \mathbf{x}_n + b) < 0 \implies \alpha_n = 0$ (max \mathcal{L} w.r.t. α_n)

non support vectors

Conclusion

At the optimum, $\alpha_n (y_n(\mathbf{w}^T \mathbf{x}_n + b) - 1) = 0$, so

$$\mathcal{L} = \frac{1}{2} \mathbf{w}^T \mathbf{w}$$

is minimized and the constraints are satisfied

$$1 - y_n(\mathbf{w}^T \mathbf{x}_n + b) \leq 0$$

Formally: use KKT conditions to transform the primal.

Kernel Trick: 7 / 18

\rightarrow

Unconstrained Minimization w.r.t. (b, \mathbf{w})

$$\mathcal{L} = \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{n=1}^N \alpha_n \cdot (y_n(\mathbf{w}^T \mathbf{x}_n + b) - 1)$$

$$\text{Set } \frac{\partial \mathcal{L}}{\partial b} = 0:$$

$$\frac{\partial \mathcal{L}}{\partial b} = \sum_{n=1}^N \alpha_n y_n \implies \sum_{n=1}^N \alpha_n y_n = 0$$

$$\text{Set } \frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0:$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \sum_{n=1}^N \alpha_n y_n \mathbf{x}_n \implies \mathbf{w} = \sum_{n=1}^N \alpha_n y_n \mathbf{x}_n$$

Substitute into \mathcal{L} to maximize w.r.t. $\alpha \geq 0$

$$\begin{aligned} \mathcal{L} &= \frac{1}{2} \mathbf{w}^T \mathbf{w} - \mathbf{w}^T \sum_{n=1}^N \alpha_n y_n \mathbf{x}_n - b \sum_{n=1}^N \alpha_n y_n + \sum_{n=1}^N \alpha_n \\ &= -\frac{1}{2} \mathbf{w}^T \mathbf{w} + \sum_{n=1}^N \alpha_n \\ &= -\frac{1}{2} \sum_{n,m=1}^N \alpha_n \alpha_m y_n y_m \mathbf{x}_n^T \mathbf{x}_m + \sum_{n=1}^N \alpha_n \end{aligned}$$

$$\text{minimize}_{\alpha} \frac{1}{2} \alpha^T G \alpha - \mathbf{1}^T \alpha \quad (G_{nm} = y_n y_m \mathbf{x}_n^T \mathbf{x}_m)$$

$$\text{subject to: } \mathbf{y}^T \alpha = 0$$

$$\alpha \geq 0$$

$$\mathbf{w} = \sum_{n=1}^N \alpha_n^* y_n \mathbf{x}_n$$

$$\alpha_s > 0 \implies y_s(\mathbf{w}^T \mathbf{x}_s + b) - 1 = 0$$

$$\implies b = y_s - \mathbf{w}^T \mathbf{x}_s$$

Kernel Trick: 8 / 18

Example \rightarrow

© Creator: Malik Magdon-Essam

© Creator: Malik Magdon-Essam

Example — Our Toy Data Set

$$X = \begin{bmatrix} 0 & 0 \\ 2 & 2 \\ 2 & 0 \\ 3 & 0 \end{bmatrix} \quad y = \begin{bmatrix} -1 \\ -1 \\ +1 \\ +1 \end{bmatrix} \quad \rightarrow \quad X_s = \begin{bmatrix} 0 & 0 \\ -2 & -2 \\ 2 & 0 \\ 3 & 0 \end{bmatrix} \quad \rightarrow \quad G = X_s X_s^T = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 8 & -4 & -6 \\ 0 & -4 & 4 & 6 \\ 0 & -6 & 6 & 9 \end{bmatrix}$$

signed data matrix
↓

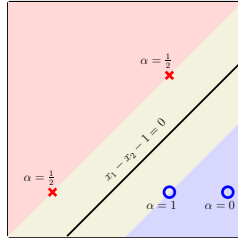
Quadratic Programming

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & \frac{1}{2} u^T Q u + p^T z \\ \text{subject to:} \quad & A u \geq c \end{aligned}$$

Dual SVM

$$\begin{aligned} \underset{\alpha}{\text{minimize}} \quad & \frac{1}{2} \alpha^T G \alpha - \mathbf{1}^T \alpha \\ \text{subject to:} \quad & y^T \alpha = 0 \\ & \alpha \geq 0 \end{aligned}$$

$$\left. \begin{aligned} u &= \alpha \\ Q &= G \\ p &= -\mathbf{1}_N \\ A &= \begin{bmatrix} y^T \\ -y^T \\ \mathbf{1}_N \end{bmatrix} \\ c &= \begin{bmatrix} 0 \\ 0 \\ \mathbf{0}_N \end{bmatrix} \end{aligned} \right\} \text{QP}(Q, p, A, c) \rightarrow \begin{aligned} \alpha^* &= \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 1 \\ 0 \end{bmatrix} \\ w &= \sum_{n=1}^4 \alpha_n^* y_n x_n = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \\ b &= y_1 - w^T x_1 = -1 \\ \gamma &= \frac{1}{\|w\|} = \frac{1}{\sqrt{2}} \end{aligned}$$



non-support vectors $\Rightarrow \alpha_n = 0$
only support vectors can have $\alpha_n > 0$

© 2015 Creator: Malik Magdon-Ismail

Kernel Trick: 9 / 18

Dual linear-SVM QP algorithm \rightarrow

Dual QP Algorithm for Hard Margin linear-SVM

- 1: **Input:** X, y .
- 2: Let $\mathbf{p} = -\mathbf{1}_N$ be the N -vector of ones and $\mathbf{c} = \mathbf{0}_{N+2}$ the N -vector of zeros. Construct matrices Q and A , where

$$X_s = \underbrace{\begin{bmatrix} -y_1 x_1^T \\ \vdots \\ -y_N x_N^T \end{bmatrix}}_{\text{signed data matrix}}, \quad Q = X_s X_s^T, \quad A = \begin{bmatrix} y^T \\ -y^T \\ \mathbf{1}_{N \times N} \end{bmatrix}$$

- 3: $\alpha^* \leftarrow \text{QP}(Q, \mathbf{c}, A, \mathbf{a})$.

- 4: Return

$$\begin{aligned} w^* &= \sum_{\alpha_n^* > 0} \alpha_n^* y_n x_n \\ b^* &= y_s - w^{*T} x_s \quad (\alpha_s^* > 0) \end{aligned}$$

- 5: The final hypothesis is $g(\mathbf{x}) = \text{sign}(w^{*T} \mathbf{x} + b^*)$.

$$\begin{aligned} \underset{\alpha}{\text{minimize}} \quad & \frac{1}{2} \alpha^T G \alpha - \mathbf{1}^T \alpha \\ \text{subject to:} \quad & y^T \alpha = 0 \\ & \alpha \geq 0 \end{aligned}$$

↑
Some packages allow **equality** and bound constraints to directly solve this type of QP

© 2015 Creator: Malik Magdon-Ismail

Kernel Trick: 10 / 18

Primal versus dual (non-separable) \rightarrow

Primal Versus Dual (Non-Separable)

Primal

$$\begin{aligned} \underset{b, w, \xi}{\text{minimize}} \quad & \frac{1}{2} w^T w + C \sum_{n=1}^N \xi_n \\ \text{subject to:} \quad & y_n (w^T x_n + b) \geq 1 - \xi_n \\ & \xi_n \geq 0 \text{ for } n = 1, \dots, N \end{aligned}$$

$$g(\mathbf{x}) = \text{sign}(w^T \mathbf{x} + b)$$

$N + d + 1$ optimization variables b, w, ξ

Dual

$$\begin{aligned} \underset{\alpha}{\text{minimize}} \quad & \frac{1}{2} \alpha^T G \alpha - \mathbf{1}^T \alpha \\ \text{subject to:} \quad & y^T \alpha = 0 \\ & C \geq \alpha \geq 0 \end{aligned}$$

$$\begin{aligned} w^* &= \sum_{n=1}^N \alpha_n^* y_n x_n \\ b^* &= y_s - w^{*T} x_s \quad (C > \alpha_s^* > 0) \end{aligned}$$

$$\begin{aligned} g(\mathbf{x}) &= \text{sign}(w^{*T} \mathbf{x} + b^*) \\ &= \text{sign} \left(\sum_{n=1}^N \alpha_n^* y_n x_n^T (\mathbf{x} - \mathbf{x}_s) + y_s \right) \end{aligned}$$

N optimization variables α

© 2015 Creator: Malik Magdon-Ismail

Kernel Trick: 11 / 18

Inner product algorithm \rightarrow

Dual SVM is an Inner Product Algorithm

\mathcal{X} -Space

$$\begin{aligned} \underset{\alpha}{\text{minimize}} \quad & \frac{1}{2} \alpha^T G \alpha - \mathbf{1}^T \alpha \\ \text{subject to:} \quad & y^T \alpha = 0 \\ & C \geq \alpha \geq 0 \end{aligned}$$

$$G_{nm} = y_n y_m (x_n^T x_m)$$

$$g(\mathbf{x}) = \text{sign} \left(\sum_{\alpha_n^* > 0} \alpha_n^* y_n (x_n^T \mathbf{x}) + b^* \right) \quad \begin{aligned} C &> \alpha_s^* > 0 \\ b^* &= y_s - \sum_{\alpha_n^* > 0} \alpha_n^* y_n (x_n^T x_s) \end{aligned}$$

© 2015 Creator: Malik Magdon-Ismail

Kernel Trick: 12 / 18

\mathcal{X} -space inner product algorithm \rightarrow

Dual SVM is an Inner Product Algorithm

Z-Space

$$\text{minimize}_{\alpha} \quad \frac{1}{2} \alpha^T G \alpha - \mathbf{1}^T \alpha$$

$$\text{subject to: } \mathbf{y}^T \alpha = 0$$

$$C \geq \alpha \geq 0$$

$$G_{nm} = y_n y_m (\mathbf{z}_n^T \mathbf{z}_m)$$

$$g(\mathbf{x}) = \text{sign} \left(\sum_{\alpha_n^* > 0} \alpha_n^* y_n (\mathbf{z}_n^T \mathbf{z}) + b^* \right) \quad \begin{array}{l} C > \alpha_n^* > 0 \\ b^* = y_s - \sum_{\alpha_n^* > 0} \alpha_n^* y_n (\mathbf{z}_n^T \mathbf{z}_s) \end{array}$$

Dual SVM is an Inner Product Algorithm

Z-Space

$$\text{minimize}_{\alpha} \quad \frac{1}{2} \alpha^T G \alpha - \mathbf{1}^T \alpha$$

$$\text{subject to: } \mathbf{y}^T \alpha = 0$$

$$C \geq \alpha \geq 0$$

$$G_{nm} = y_n y_m (\mathbf{z}_n^T \mathbf{z}_m)$$

$$g(\mathbf{x}) = \text{sign} \left(\sum_{\alpha_n^* > 0} \alpha_n^* y_n (\mathbf{z}_n^T \mathbf{z}) + b \right) \quad \begin{array}{l} C > \alpha_n^* > 0 \\ b = y_s - \sum_{\alpha_n^* > 0} \alpha_n^* y_n (\mathbf{z}_n^T \mathbf{z}_s) \end{array}$$

Can we compute $\mathbf{z}^T \mathbf{z}'$ without needing $\mathbf{z} = \Phi(\mathbf{x})$ to visit \mathcal{Z} -space?

The Kernel $K(\cdot, \cdot)$ for a Transform $\Phi(\cdot)$

The Kernel tells you how to compute the inner product in \mathcal{Z} -space

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

Example: 2nd-order polynomial transform

$$\Phi(\mathbf{x}) = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix} \quad K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x})^T \Phi(\mathbf{x}') = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix} \cdot \begin{bmatrix} x_1' \\ x_2' \\ x_1'^2 \\ \sqrt{2}x_1'x_2' \\ x_2'^2 \end{bmatrix} \quad \leftarrow O(d^2)$$

$$= x_1x_1' + x_2x_2' + x_1^2x_1'^2 + 2x_1x_2x_1'x_2' + x_2^2x_2'^2$$

$$= \left(\frac{1}{2} + \mathbf{x}^T \mathbf{x}' \right)^2 - \frac{1}{4}$$

↑
computed quickly
in \mathcal{Z} -space, in $O(d)$

The Gaussian Kernel is Infinite-Dimensional

$$K(\mathbf{x}, \mathbf{x}') = e^{-\gamma \|\mathbf{x} - \mathbf{x}'\|^2}$$

Example: Gaussian Kernel in 1-dimension

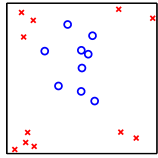
$$\Phi(\mathbf{x}) = \begin{bmatrix} e^{-x^2} \sqrt{\frac{2^0}{0!}} \\ e^{-x^2} \sqrt{\frac{2^1}{1!}} x \\ e^{-x^2} \sqrt{\frac{2^2}{2!}} x^2 \\ e^{-x^2} \sqrt{\frac{2^3}{3!}} x^3 \\ e^{-x^2} \sqrt{\frac{2^4}{4!}} x^4 \\ \vdots \end{bmatrix} \quad K(x, x') = \Phi(x)^T \Phi(x') = \begin{bmatrix} e^{-x^2} \sqrt{\frac{2^0}{0!}} \\ e^{-x^2} \sqrt{\frac{2^1}{1!}} x \\ e^{-x^2} \sqrt{\frac{2^2}{2!}} x^2 \\ e^{-x^2} \sqrt{\frac{2^3}{3!}} x^3 \\ e^{-x^2} \sqrt{\frac{2^4}{4!}} x^4 \\ \vdots \end{bmatrix} \cdot \begin{bmatrix} e^{-x'^2} \sqrt{\frac{2^0}{0!}} \\ e^{-x'^2} \sqrt{\frac{2^1}{1!}} x' \\ e^{-x'^2} \sqrt{\frac{2^2}{2!}} x'^2 \\ e^{-x'^2} \sqrt{\frac{2^3}{3!}} x'^3 \\ e^{-x'^2} \sqrt{\frac{2^4}{4!}} x'^4 \\ \vdots \end{bmatrix}$$

$$= e^{-x^2} e^{-x'^2} \sum_{i=0}^{\infty} \frac{(2xx')^i}{i!}$$

(infinite dimensional Φ)

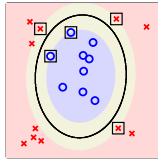
$$= e^{-(x-x')^2}$$

The Kernel Allows Us to Bypass \mathcal{Z} -space



$\mathbf{x}_n \in \mathcal{X}$

$\downarrow K(\cdot, \cdot)$



$$g(\mathbf{x}) = \text{sign} \left(\sum_{\alpha_n^* > 0} \alpha_n^* y_n K(\mathbf{x}_n, \mathbf{x}) + b^* \right)$$

$$b^* = y_s - \sum_{\alpha_n^* > 0} \alpha_n^* y_n K(\mathbf{x}_n, \mathbf{x}_s)$$

1: **Input:** X, \mathbf{y} , regularization parameter C

2: Compute G : $G_{nm} = y_n y_m K(\mathbf{x}_n, \mathbf{x}_m)$.

3: Solve (QP):

$$\left. \begin{array}{l} \text{minimize: } \frac{1}{2} \boldsymbol{\alpha}^T G \boldsymbol{\alpha} - \mathbf{1}^T \boldsymbol{\alpha} \\ \text{subject to: } \mathbf{y}^T \boldsymbol{\alpha} = 0 \\ \mathbf{C} \geq \boldsymbol{\alpha} \geq \mathbf{0} \end{array} \right\} \rightarrow \boldsymbol{\alpha}^* \text{ index } s : C > \alpha_s^* > 0$$

4: $b^* = y_s - \sum_{\alpha_n^* > 0} \alpha_n^* y_n K(\mathbf{x}_n, \mathbf{x}_s)$

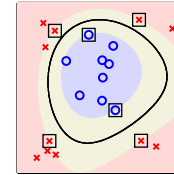
5: The final hypothesis is

$$g(\mathbf{x}) = \text{sign} \left(\sum_{\alpha_n^* > 0} \alpha_n^* y_n K(\mathbf{x}_n, \mathbf{x}) + b^* \right)$$

The Kernel-Support Vector Machine

Overfitting

SVM



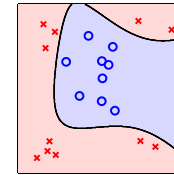
high $\tilde{d} \rightarrow$ complicated separator

small # support vectors \rightarrow low effective complexity

Can go to high (infinite) \tilde{d}

Computation

Regression



Inner products with Kernel

$$K(\cdot, \cdot)$$

high $\tilde{d} \rightarrow$ expensive or infeasible computation

kernel \rightarrow computationally feasible to go to high \tilde{d}

Can go to high (infinite) \tilde{d}

