Kernel Methods and Support Vector Machines

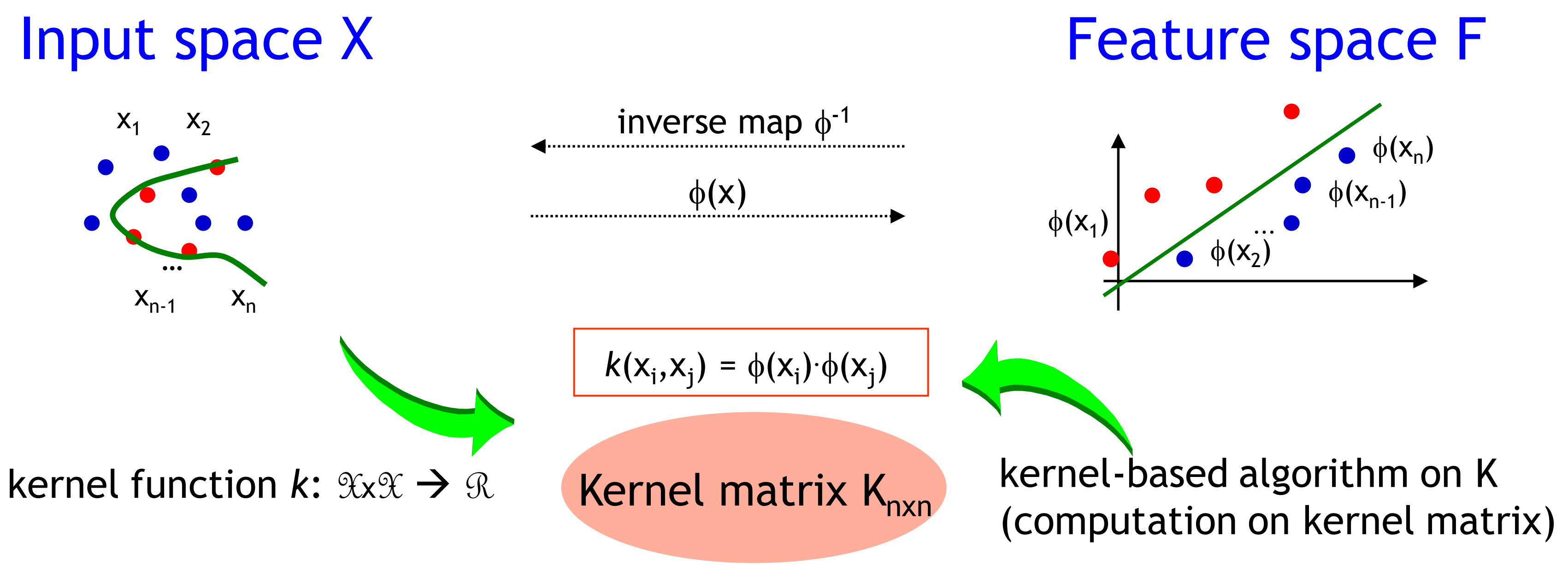
Ho Tu Bao
Japan Advance Institute of Science and Technology
John von Neumann Institute, VNU-HCM

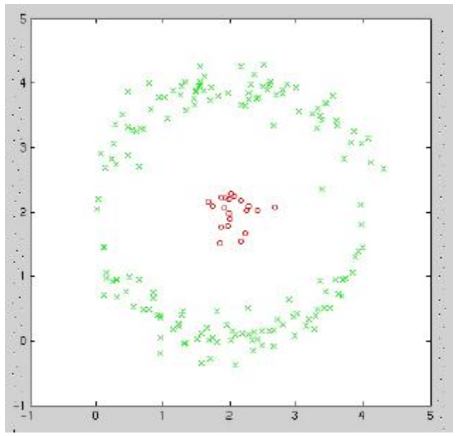
Content

Introduction

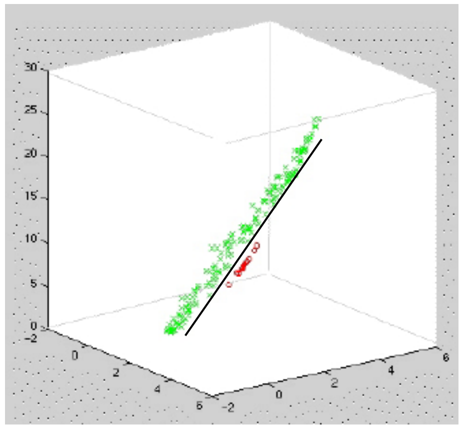
- SVMs are currently of great interest to theoretical researchers and applied scientists.
- By means of the new technology of *kernel methods*, SVMs have been very successful in building highly nonlinear classifiers.
- SVMs have also been successful in dealing with situations in which there are many more variables than observations, and complexly structured data.
- Wide applications in machine learning, natural language processing, boinformatics.

Kernel methods: key idea

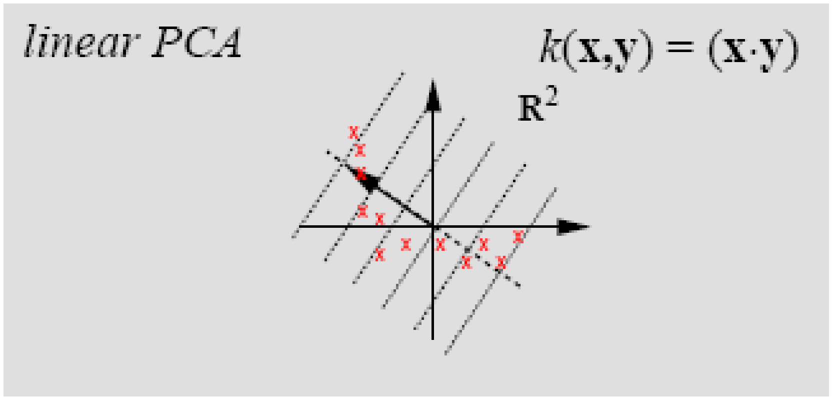


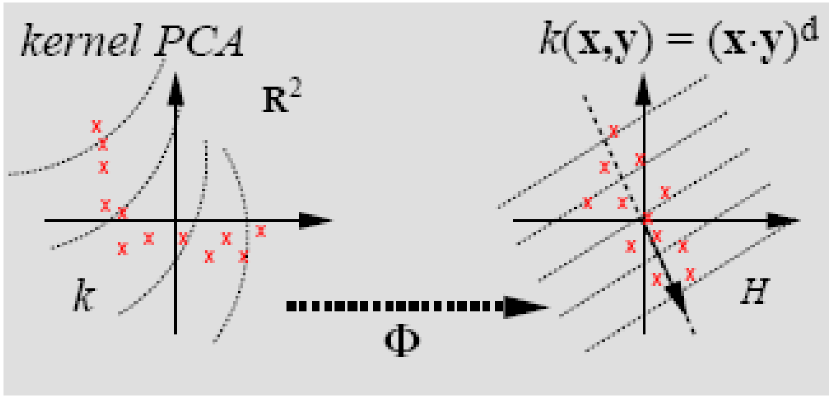


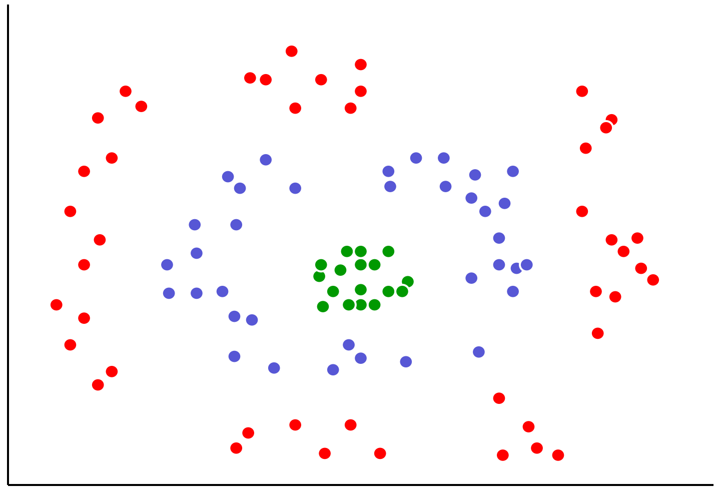
$$\phi : \mathcal{X} = \Re^2 \rightarrow \mathcal{H} = \Re^3$$
$$(x_1, x_2) \mapsto (x_1, x_2, x_1^2 + x_2^2)$$

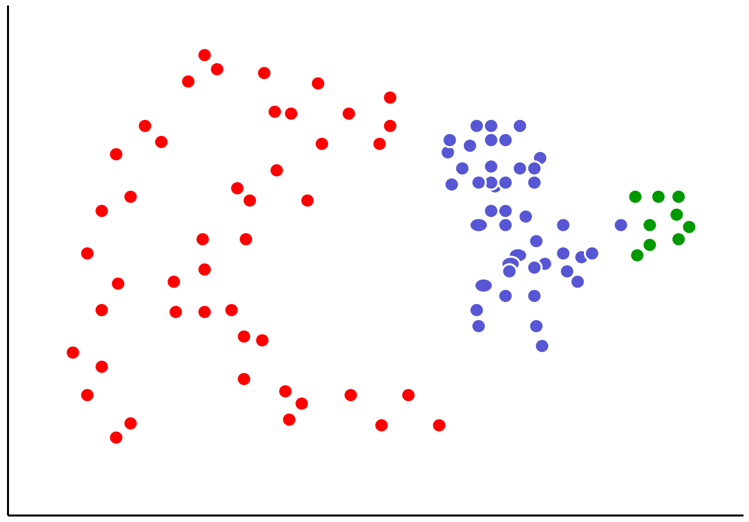


Kernel PCA





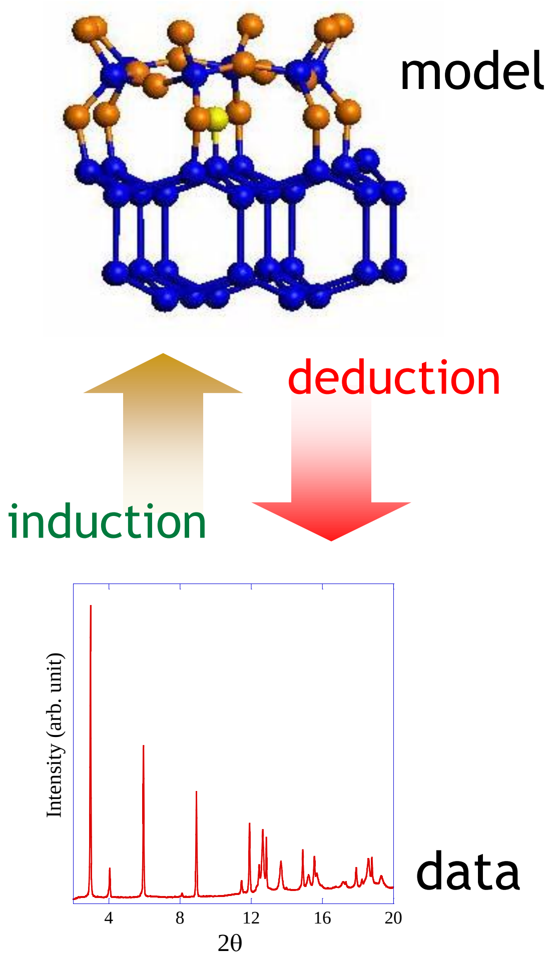




Using kernel function, linear operators of PCA is carried out in a reproducing kernel Hilbert space with a linear mapping.

Regularization (1/4)

- Classification is one inverse problem (induction): Data → Model parameters
- Inverse problems are typically ill posed, as opposed to the well-posed problems typically when modeling physical situations where the model parameters or material properties are known (a unique solution exists that depends continuously on the data).
- To solve these problems numerically one must introduce some additional information about the solution, such as an assumption on the smoothness or a bound on the norm.



Regularization (2/4)

- *Input of the classification problem*: m pairs of training data (x_i, y_i) generated from some distribution P(x,y), $x_i \in$ X, $y_i \in$ C = {C$_1$, C$_2$, ..., C$_k$} (training data).
- *Task:* Predict y given x at a new location, i.e., to find a function f (model) to do the task, f: X → C.
- *Training error* (empirical risk): Average of a loss function on the training data, for example

$$R_{emp}[f] = \frac{1}{m}\sum_{i=1}^{m} c(\mathbf{x}_i, y_i, f(\mathbf{x}_i)) \qquad \text{for example, } c(\mathbf{x}_i, y_i, f(\mathbf{x}_i)) = \begin{cases} 0, & y_i = f(\mathbf{x}_i) \\ 1, & y_i \neq f(\mathbf{x}_i) \end{cases}$$

- *Target:* (risk minimization) to find a function f that minimizes the test error (expected risk)

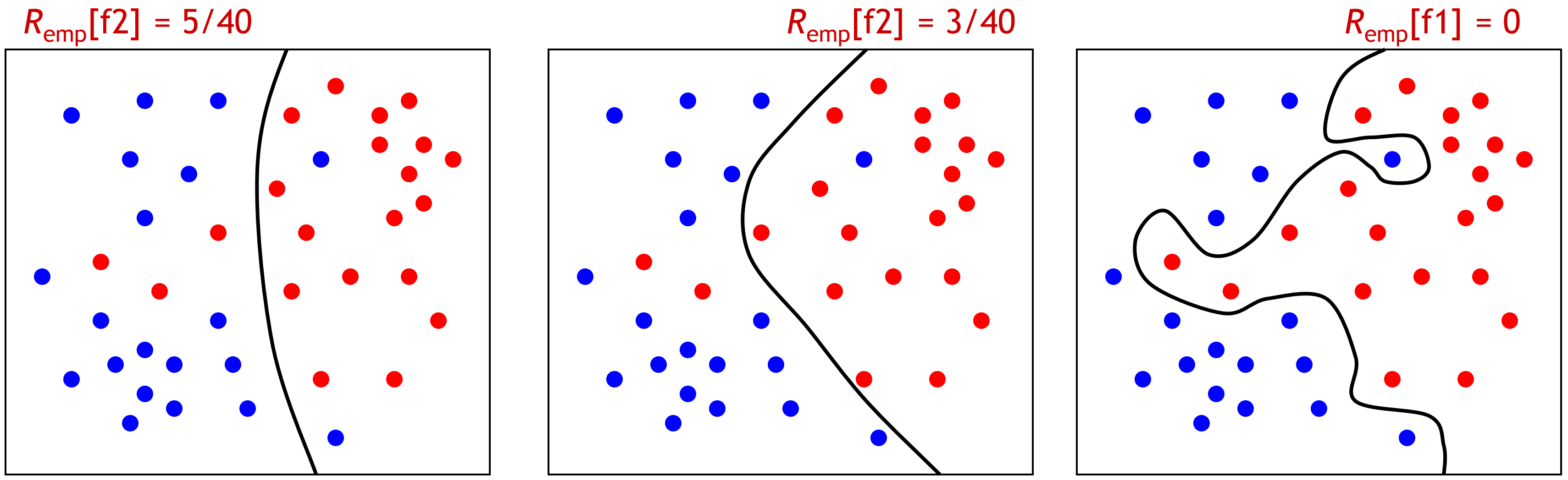
$$R[f] := E[R_{test}[f]] = E[c(x, y, f(x))] = \int_{\mathcal{X}\times\mathcal{Y}} c(x, y, f(x))\, dP(x, y)$$

Regularization (3/4)

- *Problem:* Small $R_{\text{emp}}[f]$ does not always ensure small $R[f]$ (overfitting), i.e., we may get small

$$\text{Prob}\{\sup_{f \in \mathfrak{F}} \left| R_{emp}[f] - R[f] \right| < \varepsilon\}$$

- *Fact 1*: Statistical learning theory says the difference is small if $\mathfrak{F}$ is small.
- *Fact 2*: Practical work says the difference is small if f is smooth.



Regularization (4/4)

- Regularization is restriction of a class $\mathfrak{F}$ of possible minimizers (with $f \in \mathfrak{F}$) of the empirical risk functional $R_{\text{emp}}[f]$ such that $\mathfrak{F}$ becomes a compact set.
- *Key idea*: Add a regularization (stabilization) term $\Omega[f]$ such that small $\Omega[f]$ corresponds to smooth f (or otherwise simple f) and minimize

$$R_{reg}[f] := R_{emp}[f] + \lambda\Omega[f]$$

  - $R_{\text{reg}}[f]$: regularized risk functionals;
  - $R_{\text{emp}}[f]$: empirical risk;
  - $\Omega[f]$: regularization term; and
  - λ: regularization parameter that specifies the trade-off between minimization of $R_{\text{emp}}[f]$ and the smoothness or simplicity enforced by small $\Omega[f]$ (i.e., complexity penalty).
- We need someway to measure if the set $\mathfrak{F}_C = \{f \mid \Omega[f] < C\}$ is a "small" class of functions.

Content

Linear support vector machines
The linearly separable case

- Learning set of data L = $\{(\boldsymbol{x}_i, y_i): i = 1, 2, \ldots, n\}$, $\boldsymbol{x}_i \in \Re^r$, $y_i \in \{-1, +1\}$.
- The binary classification problem is to use L to construct a function $f: \Re^r \rightarrow \Re$ so that C($\boldsymbol{x}$) = sign($f(\boldsymbol{x})$) is a classifier.
- Function f classifies each $\boldsymbol{x}$ in a test set T into one of two classes, Π+ or Π−, depending upon whether C($\boldsymbol{x}$) is +1 (if $f(\boldsymbol{x}) \geq 0$) or −1 (if $f(\boldsymbol{x}) < 0$), respectively. The goal is to have f assign all positive points in T (i.e., those with $y = +1$) to Π+ and all negative points in T ($y = -1$) to Π−.
- The simplest situation: positive ($y_i = +1$) and negative ($y_i = -1$) data points from the learning set L can be separated by a hyperplane,

$$\{\boldsymbol{x}: f(\boldsymbol{x}) = \beta_0 + \boldsymbol{x}^\tau \boldsymbol{\beta} = 0\} \qquad (1)$$

$\boldsymbol{\beta}$ is the *weight vector* with Euclidean norm $\|\boldsymbol{\beta}\|$, and β_0 is the *bias*.

Linear support vector machines

The linearly separable case

- If no error, the hyperplane is called a *separating hyperplane*.
- Let d_- and d_+ be the shortest distance from the separating hyperplane to the nearest negative and positive data points. Then, the *margin* of the separating hyperplane is defined as $d = d_- + d_+$.
- We look for *maximal margin classifier* (optimal separating hyperplane).
- If the learning data are linearly separable, $\exists\ \beta_0$ and $\boldsymbol{\beta}$ such that

$$\beta_0 + \boldsymbol{x}_i^{\tau}\boldsymbol{\beta} \geq +1, if\ y_i = +1 \qquad (2) \qquad\qquad \beta_0 + \boldsymbol{x}_i^{\tau}\boldsymbol{\beta} \leq -1, if\ y_i = -1 \qquad (3)$$

- If there are data vectors in L such that equality holds in (1), then they lie on the hyperplane H_{+1}: $(\beta_0 - 1) + \boldsymbol{x}^{\tau}\boldsymbol{\beta} = 0$; similarly, for hyperplane H_{-1}: $(\beta_0 + 1) + \boldsymbol{x}^{\tau}\boldsymbol{\beta} = 0$. Points in L that lie on either one of the hyperplanes H_{-1} or H_{+1}, are said to be *support vectors*.

Linear support vector machines

The linearly separable case

- If $\boldsymbol{x}_{-1}$ lies on H$_{-1}$, and if $\boldsymbol{x}_{+1}$ lies on H$_{+1}$, then

  $\beta_0 + \boldsymbol{x}_{-1}^{\tau}\boldsymbol{\beta} = -1$ and $\beta_0 + \boldsymbol{x}_{+1}^{\tau}\boldsymbol{\beta} = +1$

  the difference between them is $\boldsymbol{x}_{+1}^{\tau}\boldsymbol{\beta} - \boldsymbol{x}_{-1}^{\tau}\boldsymbol{\beta} = 2$ and their sum is $\beta_0 = -\frac{1}{2}(\boldsymbol{x}_{+1}^{\tau}\boldsymbol{\beta} - \boldsymbol{x}_{-1}^{\tau}\boldsymbol{\beta})$. The perpendicular distances of the hyperplane $\beta_0 + \boldsymbol{x}^{\tau}\boldsymbol{\beta} = 0$ to $\boldsymbol{x}_{-1}$ and $\boldsymbol{x}_{+1}$ are

$$d_{-} = \frac{|\beta_0 + \boldsymbol{x}_{-1}^{\tau}\boldsymbol{\beta}|}{\|\boldsymbol{\beta}\|} = \frac{1}{\|\boldsymbol{\beta}\|}$$

$$d_{+} = \frac{|\beta_0 + \boldsymbol{x}_{+1}^{\tau}\boldsymbol{\beta}|}{\|\boldsymbol{\beta}\|} = \frac{1}{\|\boldsymbol{\beta}\|}$$

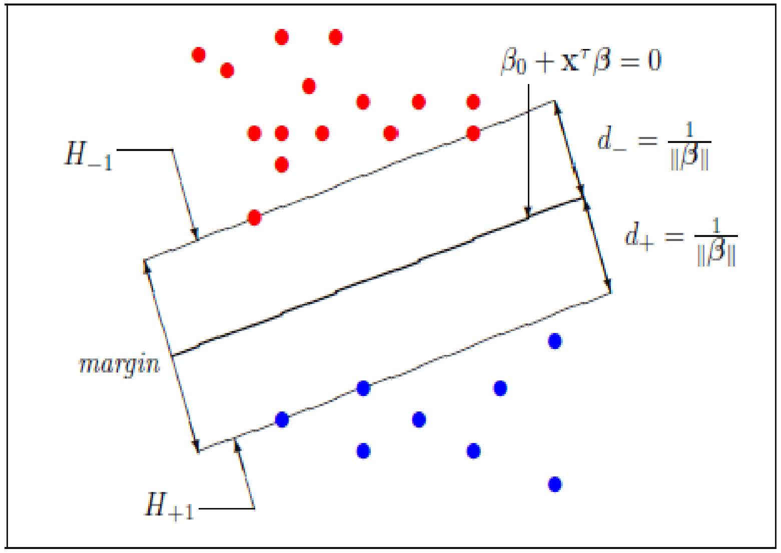

$\beta_0 + x^{\tau}\beta = 0$
H_{-1}
$d_{-} = \frac{1}{\|\beta\|}$
$d_{+} = \frac{1}{\|\beta\|}$
margin
H_{+1}

FIGURE 11.1. *Support vector machines: the linearly separable case. The red points correspond to data points with $y_i = -1$, and the blue points correspond to data points with $y_i = +1$. The separating hyperplane is the line $\beta_0 + x^{\tau}\beta = 0$. The support vectors are those points lying on the hyperplanes H_{-1} and H_{+1}. The margin of the separating hyperplane is $d = 2/\|\beta\|$.*

Linear support vector machines

The linearly separable case

- Combine (2) and (3) into a single set of inequalities

$$y_i(\beta_0 + \boldsymbol{x}_i^\tau \boldsymbol{\beta}) \geq +1, \text{i} = 1, 2, \ldots, \text{n}.$$

- The quantity $y_i(\beta_0 + \boldsymbol{x}_i^\tau \boldsymbol{\beta})$ is called the *margin of* $(\boldsymbol{x}_i, y_\text{i})$ *with respect to the hyperplane* (1), i = 1, …, n and $\boldsymbol{x_i}$ is the support vectors wrt to (1) if $y_i(\beta_0 + \boldsymbol{x}_i^\tau \boldsymbol{\beta}) = 1$.
- Problem: Find the hyperplane that miximizes the margin $\frac{2}{\|\boldsymbol{\beta}\|}$.
- Equivalently, find β_0 and $\boldsymbol{\beta}$ to

$$\begin{aligned} &\textit{minimize} \quad \frac{1}{2}\|\boldsymbol{\beta}\|^2 \\ &\textit{subject to} \quad y_i(\beta_0 + \boldsymbol{x}_i^\tau \boldsymbol{\beta}) \geq 1, i = 1, 2, \ldots, n \end{aligned} \qquad (4)$$

- Solve this *primal optimization problem* using Lagrangian multipliers.

Linear support vector machines

The linearly separable case

- Multiply the constraints, $y_i(\beta_0 + \boldsymbol{x}_i^\tau \boldsymbol{\beta}) - 1 \geq 0$, by positive Lagrangian multipliers and subtract each product from the objective function …

- Dual optimization problem: Find $\boldsymbol{\alpha}$ to,

$$\begin{aligned} &\textit{maximize} \quad F_D(\boldsymbol{\alpha}) = \mathbf{1}_n^\tau \boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^\tau \mathbf{H}\boldsymbol{\alpha} \\ &\textit{subject to} \quad \boldsymbol{\alpha} \geq 0, \boldsymbol{\alpha}^\tau \boldsymbol{y} = 0 \end{aligned} \tag{5}$$

where $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)^\tau$, $\mathbf{H} = (H_{\mathrm{ij}}) = y_i y_j (\boldsymbol{x}_i^\tau \boldsymbol{x}_j)$.

- If $\boldsymbol{\alpha}^*$solves this problem, then $\boldsymbol{\beta}^* = \sum_{i=1}^n \alpha_i^* y_i \boldsymbol{x}_i \rightarrow \boldsymbol{\beta}^* = \sum_{i \in sv} \alpha_i^* y_i \, \boldsymbol{x}_i$

$$\beta_0^* = \frac{1}{|sv|} \sum_{i \in sv} \left(\frac{1 - y_i \boldsymbol{x}_i^\tau \boldsymbol{\beta}^*}{y_i} \right)$$

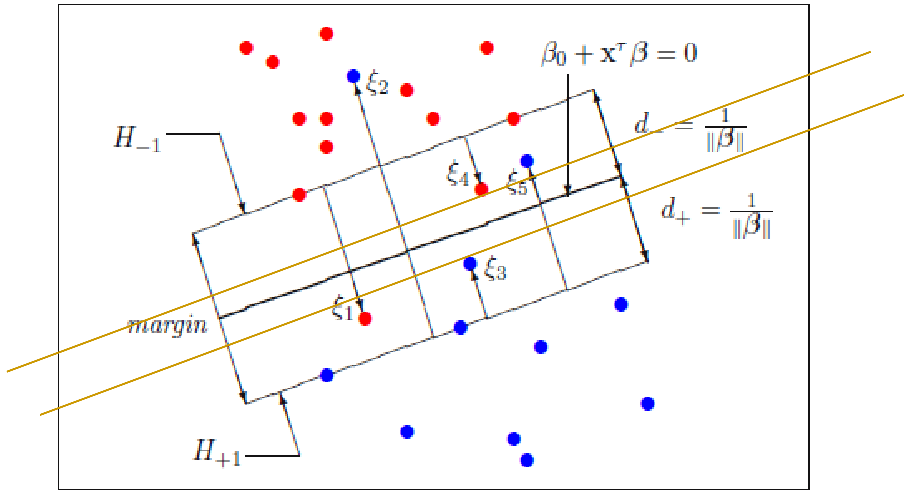
- Optimal hyperplane $\quad f^*(\mathbf{x}) = \beta_0^* + \boldsymbol{x}^\tau \boldsymbol{\beta}^* = \beta_0^* + \sum_{i \in sv} \alpha_i^* \, y_i (\boldsymbol{x}^\tau \boldsymbol{x}_i)$

Linear support vector machines

The linearly nonseparable case

- The *nonseparable case* occurs if either the two classes are separable, but not linearly so, or that no clear separability exists between the two classes, linearly or nonlinearly (caused by, for example, noise).
- Create a more flexible formulation of the problem, which leads to a *soft-margin solution*. We introduce a nonnegative *slack variable*, ξ_i, for each observation $(\boldsymbol{x}_i, y_i)$ in $\mathcal{L}$, i = 1, 2, . . . , n. Let

$$\boldsymbol{\xi} = (\xi_1, \cdots, \xi_n)^\tau \geq \mathbf{0}.$$



$\beta_0 + x^\tau \beta = 0$
H_{-1}
ξ_2
$d_- = \frac{1}{\|\beta\|}$
ξ_4
ξ_5
$d_+ = \frac{1}{\|\beta\|}$
ξ_3
ξ_1
margin
H_{+1}

FIGURE 11.2. *Support vector machines: the nonlinearly separable case. The red points correspond to data points with $y_i = -1$, and the blue points correspond to data points with $y_i = +1$. The separating hyperplane is the line $\beta_0 + x^\tau \beta = 0$. The support vectors are those circled points lying on the hyperplanes H_{-1} and H_{+1}. The slack variables ξ_1 and ξ_4 are associated with the red points that violate the constraint of hyperplane H_{-1}, and points marked by ξ_2, ξ_3, and ξ_5 are associated with the blue points that violate the constraint of hyperplane H_{+1}. Points that satisfy the constraints of the appropriate hyperplane have $\xi_i = 0$.*

Linear support vector machines

The linearly nonseparable case

- The constraints in (5) become $y_i(\beta_0 + \boldsymbol{x}_i^\tau \boldsymbol{\beta}) + \xi_i \geq 1$ for i = 1, 2, ..., n.

- Find the optimal hyperplane that controls both the margin, $2/\|\boldsymbol{\beta}\|$, and some computationally simple function of the slack variables, such as $g_\sigma(\xi)=\sum_{i=1}^{n} \xi_i^\sigma$. Consider "1-norm" ($\sigma = 1$) and "2-norm" ($\sigma = 2$).

- The 1-norm soft-margin optimization problem is to find β_0, $\boldsymbol{\beta}$ and $\boldsymbol{\xi}$ to

$$\begin{aligned} &\textit{minimize} \quad \frac{1}{2}\|\beta\|^2 + \mathrm{C}\sum_{i=1}^{n} \xi^i \\ &\textit{subject to} \quad \xi_i \geq 0, y_i(\beta_0 + \boldsymbol{x}_i^\tau \boldsymbol{\beta}) \geq 1 - \xi_i, \;\; i = 1, 2, \dots, n. \end{aligned} \tag{6}$$

where C > 0 is a *regularization parameter*. C takes the form of a tuning constant that controls the size of the slack variables and balances the two terms in the minimizing function.

Linear support vector machines

The linearly nonseparable case

- We can write the dual maximization problem in matrix notation as follows. Find $\boldsymbol{\alpha}$ to

$$\begin{aligned} &\textit{maximize } F_D(\boldsymbol{\alpha}) = \mathbf{1}_n^\tau \boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^\tau \mathbf{H}\boldsymbol{\alpha} \\ &\textit{subject to} \quad \boldsymbol{\alpha}^\tau \boldsymbol{y} = 0,\ \mathbf{0} \le \boldsymbol{\alpha} \le C\mathbf{1}_n \end{aligned} \tag{7}$$

- The difference between this optimization problem and (4), is that here the coefficients α_i, i = 1.. n, are each bounded above by C; this upper bound restricts the influence of each observation in determining the solution.

- This constraint is referred to as a box constraint because $\boldsymbol{\alpha}$ is constrained by the box of side C in the positive orthant. The feasible region for the solution to this problem is the intersection of hyperplane $\boldsymbol{\alpha}^\tau \boldsymbol{y} = 0$ with the box constraint $\mathbf{0} \le \boldsymbol{\alpha} \le C\mathbf{1}_n$. If $C = \infty \rightarrow$ hard-margin separable case.

- If $\boldsymbol{\alpha}^*$solves (7) then $\boldsymbol{\beta}^* = \sum_{i\in sv} \alpha_i^* y_i \boldsymbol{x}_i$ yields the optimal weight vector.

Content

Nonlinear support vector machines

- What if a linear classifier is not appropriate for the data set?
- Can we extend the idea of linear SVM to the nonlinear case?
- The key to constructing a nonlinear SVM is to observe that the observations in $\mathcal{L}$ only enter the dual optimization problem through the inner products $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle = \boldsymbol{x}_i^\tau \boldsymbol{x}_j$, i, j = 1, 2, …, n.

$$F_D(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \, \alpha_j y_i y_j (\boldsymbol{x}_i^\tau \boldsymbol{x}_j)$$

Nonlinear support vector machines

Nonlinear transformations

- Suppose we transform each observation, $x_i \in \Re^r$, in$\mathcal{L}$using some nonlinear mapping $\mathbf{\Phi}: \Re^r \rightarrow \mathcal{H}$, $\mathcal{H}$ is an $N_{\mathcal{H}}$-dimensional feature space.
- The nonlinear map $\mathbf{\Phi}$ is generally called the *feature map* and the space $\mathcal{H}$ is called the *feature space*.
- The space $\mathcal{H}$ may be very high-dimensional, possibly even infinite dimensional. We will generally assume that $\mathcal{H}$ is a Hilbert space of real-valued functions on with inner product $\langle . , . \rangle$ and norm $\|.\|$.
- Let $\mathbf{\Phi}(x_i) = (\phi_1(\boldsymbol{x}_i), ..., \phi_{N_{\mathcal{H}}}(\boldsymbol{x}_i))^\tau \in \mathcal{H}$, i =1..n. The transformed sample is $\{\mathbf{\Phi}(\boldsymbol{x}_i), y_i\}$, where $y_i \in \{-1, +1\}$ identifies the two classes.
- If substitute $\mathbf{\Phi}(\boldsymbol{x}_i)$ for $\boldsymbol{x}_i$ in the development of the linear SVM, then data would only enter the optimization problem by way of the inner products $\langle \mathbf{\Phi}(\boldsymbol{x}_i), \mathbf{\Phi}(\boldsymbol{x}_j) \rangle = \mathbf{\Phi}(\boldsymbol{x}_i)^\tau \mathbf{\Phi}(\boldsymbol{x}_j)$. The difficulty in using nonlinear transform is computing such inner products in high-dimensional space $\mathcal{H}$.

Nonlinear support vector machines

The "kernel trick"

- The idea behind nonlinear SVM is to find an optimal separating hyperplane in high-dimensional feature space $\mathcal{H}$ just as we did for the linear SVM in input space.
- The "kernel trick" was first applied to SVMs by Cortes & Vapnik (1995).
- Kernel trick: Wonderful idea that is widely used in algorithms for computing inner products $\langle \boldsymbol{\Phi}(\boldsymbol{x}_i), \boldsymbol{\Phi}(\boldsymbol{x}_j) \rangle$ in feature space $\mathcal{H}$.
- The trick: *instead of computing the inner products in $\mathcal{H}$, which would be computationally expensive due to its high dimensionality, we compute them using a nonlinear kernel function, $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{\Phi}(\boldsymbol{x}_i), \boldsymbol{\Phi}(\boldsymbol{x}_j) \rangle$ in input space, which helps speed up the computations.*
- Then, we just compute a linear SVM, but where the computations are carried out in some other space.

Nonlinear support vector machines

Kernels and their properties

- A *kernel* K is a function $K: \Re^r \times \Re^r \rightarrow \Re$ such that $\forall\, \boldsymbol{x}, \boldsymbol{y} \in \Re^r$

$$K(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{\Phi}(\mathrm{x}), \boldsymbol{\Phi}(\boldsymbol{y}) \rangle$$

- The kernel function is designed to compute inner-products in $\mathcal{H}$ by using only the original input data → substitute $\langle \boldsymbol{\Phi}(\boldsymbol{x}), \boldsymbol{\Phi}(\boldsymbol{y}) \rangle$ by $K(\boldsymbol{x}, \boldsymbol{y})$ whenever. Advantage: given K, no need to know the explicit form of $\boldsymbol{\Phi}$.

- K should be symmetric: $K(\boldsymbol{x}, \boldsymbol{y}) = K(\boldsymbol{y}, \boldsymbol{x})$, and $[K(\boldsymbol{x}, \boldsymbol{y})]^2 \leq K(\boldsymbol{x}, \boldsymbol{x}) K(\boldsymbol{y}, \boldsymbol{y})$.

- K is a *reproducing kernel* if $\forall f \in \mathcal{H}: \langle f(.), K(\boldsymbol{x}, .) \rangle = f(\boldsymbol{x})$ (8), K is called the *representer of evaluation*. Particularly, if $f(.) = K(., \boldsymbol{x})$ then $\langle K(\boldsymbol{x}, .), K(y, .) \rangle = K(\boldsymbol{x}, \boldsymbol{y})$.

- Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ be n points in $\Re^r$. The $(n \times n)$-matrix $\mathbf{K} = (Kij) = (K(\boldsymbol{x}_i, \boldsymbol{x}_j))$ is called *Gram* (or *kernel*) *matrix* wrt $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$.

Nonlinear support vector machines

Kernels and their properties

- If for any n-vector $\mathbf{u}$, we have $\mathbf{u}^{\tau}\mathbf{K}\mathbf{u} \geq 0$, $\mathbf{K}$ is said to be *nonnegative-definite* with nonnegative eigenvalues and K is nonnegative-definite kernel (or Mercer kernel).
- If K is a Mercer kernel on $\Re^r \times \Re^r$, we can construct a unique Hilbert space $\mathcal{H}_\mathrm{K}$, say, of real-valued functions for which K is its reproducing kernel. We call $\mathcal{H}_\mathrm{K}$ a (real) *reproducing kernel Hilbert space* (rkhs). We write the inner-product and norm of $\mathcal{H}_\mathrm{K}$ by $\langle .,. \rangle_{\mathcal{H}_\mathrm{K}}$ and $\|.\|_{\mathcal{H}_\mathrm{K}}$.
- Ex: inhomogeneous polynomial kernel of degree d (c, d: parameters)

$$K(\boldsymbol{x},\boldsymbol{y}) = (\langle \boldsymbol{x},\boldsymbol{y}\rangle + \mathrm{c})^{\mathrm{d}},\ \ \boldsymbol{x},\boldsymbol{y}\ \in\ \Re^r$$

- If r = 2, d = 2, $\boldsymbol{x} = (x_1, x_2)^{\tau}$, $\boldsymbol{y} = (y_1, y_2)^{\tau}$,

$$K(\boldsymbol{x},\boldsymbol{y}) = \ (\langle \boldsymbol{x},\boldsymbol{y}\rangle + c)^2 = (x_1y_1 + x_2y_2\ + c)^2 = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{y})\rangle$$

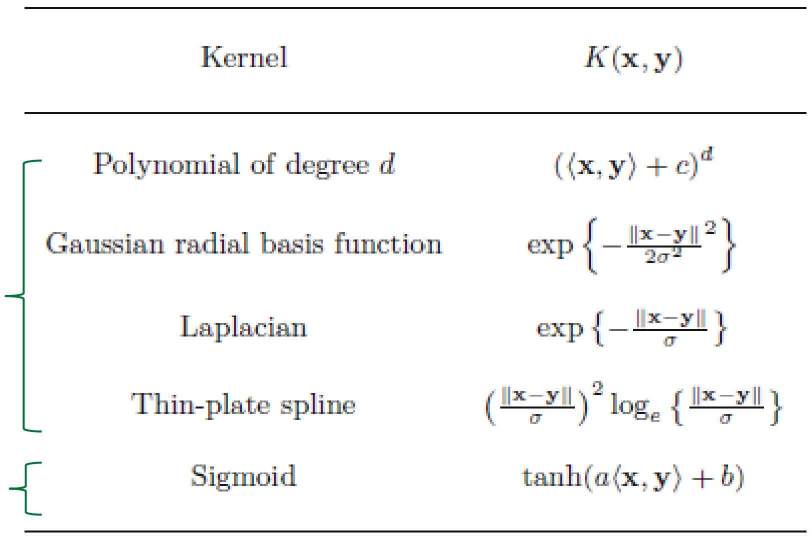
$$\Phi(\boldsymbol{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2c}x_1x_2, \sqrt{2c}x_1, \sqrt{2c}x_2, \mathrm{c})$$

Nonlinear support vector machines

Examples of kernels

- Here $\mathcal{H} = \Re^6$, monomials have degree ≤ 2. In general, $\dim(\mathcal{H}) = \binom{r+d}{d}$ consisting of monomials with degree $\leq d$.
- For 16x16 pixels, r = 256. If d =2, $\dim(\mathcal{H})$ = 33,670; d = 4, $\dim(\mathcal{H})$ = 186,043,585.

TABLE 11.1. *Kernel functions, $K(\mathbf{x},\mathbf{y})$, where $\sigma > 0$ is a scale parameter, $a, b, c \geq 0$, and d is an integer. The Euclidean norm is $\|\mathbf{x}\|^2 = \mathbf{x}^\tau \mathbf{x}$.*



Kernel	$K(\mathbf{x},\mathbf{y})$
Polynomial of degree d	$(\langle \mathbf{x},\mathbf{y}\rangle + c)^d$
Gaussian radial basis function	$\exp\left\{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}\right\}$
Laplacian	$\exp\left\{-\frac{\|\mathbf{x}-\mathbf{y}\|}{\sigma}\right\}$
Thin-plate spline	$\left(\frac{\|\mathbf{x}-\mathbf{y}\|}{\sigma}\right)^2 \log_e\left\{\frac{\|\mathbf{x}-\mathbf{y}\|}{\sigma}\right\}$
Sigmoid	$\tanh(a\langle \mathbf{x},\mathbf{y}\rangle + b)$

Examples of translation-invariant (stationary) kernels having the general form
$K(\boldsymbol{x},\boldsymbol{y}) = k(\boldsymbol{x}-\boldsymbol{y}), k\colon \Re^r \to \Re$

sigmoid kernel is not strictly a kernel but very popular in certain situations

If no information, the best approach is to try either a Gaussian RBF, which has only a single parameter (σ) to be determined, or a polynomial kernel of low degree (d = 1 or 2).

Nonlinear support vector machines

Example: String kernels for text (Lodhi et al., 2002)

- A "string" $s = s_1 s_2 \dots s_{|s|}$ is a finite sequence of elements of a finite alphabet $\mathcal{A}$.
- We call u a *subsequence* of s (written $u = s(\boldsymbol{i})$) if there are indices $\boldsymbol{i} = (i_1, i_2, \dots, i_{|u|})$, $1 \leq i_1 < \cdots < i_{|u|} \leq |s|$, such that $u_j = s_{i_j}$, $j = 1, 2, \dots, |u|$.
- If the indices **i** are contiguous, we say that u is a *substring* of s. The length of u in s is $l(i) = i_{|u|} - i_1 + 1$.
- Let s ="cat" ($s_1 = c$, $s_2 = a$, $s_3 = t$, $|s| = 3$). Consider all possible 2-symbol sequences, "ca," "ct," and "at," derived from s.
  - u = ca has u_1 = c = s_1, u_2 = a = s_2, $u = s(\mathbf{i})$, $\mathbf{i} = (i_1, i_2) = (1, 2)$, $(\mathbf{i}) = 2$.
  - u = ct has u_1 = c = s_1, u_2 = t = s_3, $\mathbf{i} = (i_1, i_2) = (1, 3)$, and $(\mathbf{i}) = 3$.
  - u = at has u_1 = a = s_2, u_2 = t = s_3, $\mathbf{i} = (2, 3)$, and $(\mathbf{i}) = 2$.

Nonlinear support vector machines

Examples: String kernels for text

- If $D = \mathcal{A}^m = \{$all strings of length at most m from $A\}$, then, the feature space for a string kernel is $\Re^D$.
- Using $\lambda \in (0, 1)$ (*drop-off rate* or *decay factor*) to weight the interior gaps in the subsequences, we define the feature map $\Phi_u\colon \Re^D \longrightarrow \Re$

$$\Phi_u(s) = \sum_{\mathbf{i}:u=s(\mathbf{i})} \lambda^{l(\mathbf{i})}, u \in \mathcal{A}^m$$

- $\Phi_u(s)$ is computed as follows: identify all subsequences (indexed by **i**) of s that are identical to u; for each such subsequence, raise λ to the power (**i**); and then sum the results over all subsequences.
- In our example above, $\Phi_{ca}(\text{cat}) = \lambda^2$, $\Phi_{ct}(\text{cat}) = \lambda^3$, and $\Phi_{at}(\text{cat}) = \lambda^2$.
- Two documents are considered to be "similar" if they have many subsequences in common: the more subsequences they have in common, the more similar they are deemed to be.

Nonlinear support vector machines

Examples: String kernels for text

- The kernel associated with the feature maps corresponding to *s* and *t* is the sum of inner products for all *common* substrings of length *m*

$$K_m(s,t) = \sum_{u \in \mathcal{D}} \langle \Phi_u(s), \Phi_u(t) \rangle = \sum_{u \in \mathcal{D}} \sum_{\mathbf{i}:u=s(\mathbf{i})} \sum_{\mathbf{j}:u=s(\mathbf{j})} \lambda^{l(i)+l(j)}$$

  and it is called a *string kernel* (or a *gap-weighted subsequences kernel*).

- Let t = "car" (t_1 = c, t_2 = a, t_3 = r, |t| = 3). The strings "cat" and "car" are both substrings of the string "cart." The three 2-symbol substrings of t are "ca," "cr," and "ar." We have that Φ_{ca}(car) = λ^2,Φ_{cr}(car) = λ^3, Φ_{ar}(car) = λ^2, and thus K_2(cat, car) = Φ_{ca}(cat),Φ_{ca}(car) = λ^4.

- We normalize the kernel by removing any bias by document length

$$K_m^*(s,t) = \frac{K_m(s,t)}{\sqrt{K_m(s,s)K_m(t,t)}}$$

Nonlinear support vector machines

Optimizing in feature space

- Let K be a kernel. Suppose obs. in $\mathcal{L}$ are *linearly separable* in the feature space corr. to K. The dual opt. problem is to find $\boldsymbol{\alpha}$ and β_0 to

$$\textit{maximize } F_D(\boldsymbol{\alpha}) = \mathbf{1}_n^\tau \boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^\tau \mathbf{H}\boldsymbol{\alpha}$$
$$\textit{subject to } \boldsymbol{\alpha} \geq 0, \boldsymbol{\alpha}^\tau \boldsymbol{y} = 0 \qquad (9)$$

where $\boldsymbol{y} = (y_1, y_1, \ldots, y_1)^\tau$, $\mathbf{H} = (H_{\mathrm{ij}}) = y_i y_j K(x_i, x_j) = y_i y_j K_{ij}$.

- Because K is a kernel, the $\mathbf{K} = (K_{ij})$ and so $\mathbf{H}$ are nonnegative-definite → the functional $F_D(\boldsymbol{\alpha})$ is convex → unique solution. If $\boldsymbol{\alpha}$ and β_0 solve this problem, the SVM decision rule is ($f^*(\mathrm{x})$ is optimal in feature space)

$$\mathrm{sign}\{f^*(\mathrm{x})\} = \mathrm{sign}\{\beta_0^* + \textstyle\sum_{i \in sv} \alpha_i^* \, y_i K(\boldsymbol{x}, \boldsymbol{x}_i)\}$$

- In the *nonseparable* case, the dual problem of the 1-norm soft-margin opt. problem is to find $\boldsymbol{\alpha}$ to

$$\textit{maximize } F_D(\boldsymbol{\alpha}) = \mathbf{1}_n^\tau \boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^\tau \mathbf{H}\boldsymbol{\alpha}$$
$$\textit{subject to } \quad \boldsymbol{\alpha}^\tau \boldsymbol{y} = 0, \ \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}_n$$

Nonlinear support vector Machines

Example: E-mail or spam?

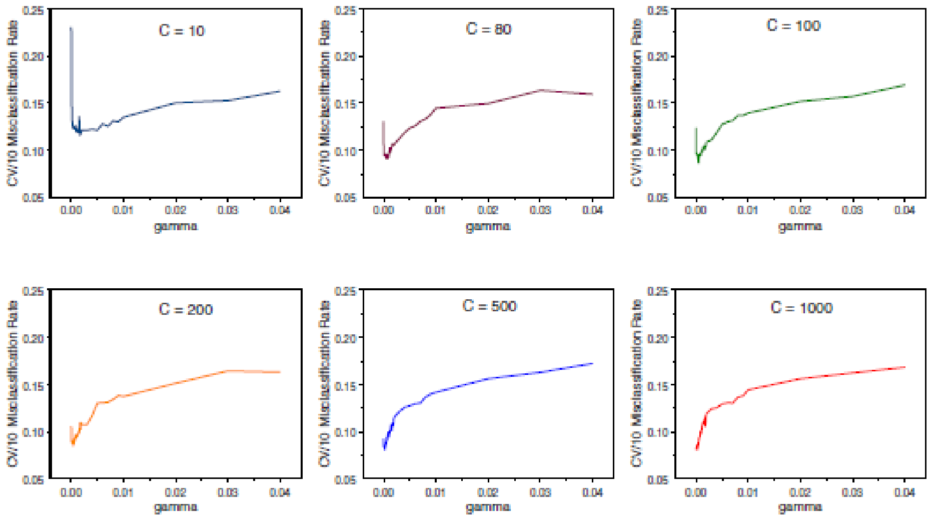
- 4,601 messages: 1,813 spam e-mails and 2,788 non-spam e-mails. There are 57 variables (attributes).
- Apply nonlinear SVM (R package libsvm) using a Gaussian RBF kernel to the 4,601 messages. The solution depends on the cost C of violating the constraints and σ^2 of the Gaussian RBF kernel. After applying a trial-and-error method, we used the following grid of values for C and $\gamma = 1/\sigma^2$:
  - $C = 10, 80, 100, 200, 500, 1{,}000,$
  - $\gamma = 0.00001(0.00001)0.0001(0.0001)0.002(0.001)0.01(0.01)0.04.$
- Plot the 10-fold CV misclassification rate against γ listed above, where each curve (connected set of points) represents a different value of C.
- For each C, we see that the CV/10 misclassification curves have similar shapes: a minimum value for γ very close to zero, and for values of γ away from zero, the curve trends upwards.

Nonlinear support vector Machines

Example: E-mail or spam?

- We find a minimum CV/10 misclassification rate of 8.06% at (C, γ) = (500, 0.0002) and (1,000, 0.0002). The level of the misclassification rate tends to decrease as C increases and γ decreases together.
- misclassification rate of 6.91% at C = 11, 000 and γ = 0.00001, at corresponding to classification rate:
  - 0.9043, 0.9478, 0.9304, 0.9261, 0.9109,
  - 0.9413, 0.9326, 0.9500. 0.9326, 0.9328.

  is better than LDA and QDA.
- 931 support vectors (482 e-mails, 449 spam).



Initial grid search for the minimum 10-fold CV misclassification rate using 0.00001 ≤ γ ≤ 0.04. The curves correspond to C = 10 (dark blue), 80 (brown), 100 (green), 200 (orange), 500 (light blue), and 1,000 (red). Within this intial grid search, the minimum CV/10 misclassification rate is 8.06%, which occurs at (C, γ) = (500, 0.0002) and (1,000, 0.0002).

Nonlinear Support Vector Machines
SVM as a Regularization Method

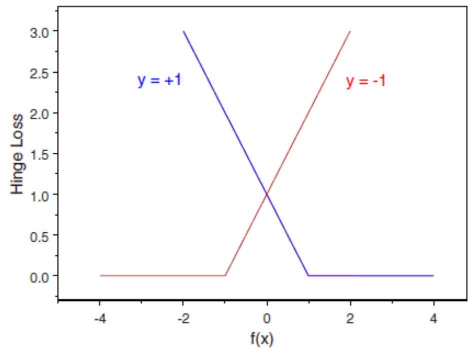
- *Regularization* involves introducing additional information in order to solve an ill-posed problem or to prevent overfitting. This information is usually of the form of a penalty for complexity.

- Let $f \in \mathcal{H}_K$, the reproducing kernel Hilbert space associated with the kernel K, with $\|f\|^2_{\mathcal{H}_K}$ the squared-norm of f in $\mathcal{H}_K$.

- Consider the classification error, $y_i - f(\boldsymbol{x}_i)$, where $y_i \in \{-1, +1\}$. Then

$$|y_i - f(\boldsymbol{x}_i)| = |y_i(1 - y_i f(\boldsymbol{x}_i))| = |1 - y_i f(\boldsymbol{x}_i)| = (1 - y_i f(\boldsymbol{x}_i))_+$$

$i = 1 .. n$, $(x)_+ = \max\ (x, 0)$. The quantity $(1 - y_i f(\boldsymbol{x}_i))_+$, which could be zero if all $\boldsymbol{x}_i$ are correctly classified, called *hinge loss function*. The hinge loss plays a vital role in SVM methodology.



Nonlinear Support Vector Machines

SVM as a Regularization Method

- Want to find $f \in \mathcal{H}_K$ to minimize a penalized version of the hinge loss. Specifically, we wish to find $f \in \mathcal{H}_K$ to

$$minimize \quad \frac{1}{2}\sum_{i=1}^{n}(1 - y_i f(\boldsymbol{x}_i))_+ \; + \; \lambda \|f\|_{\mathcal{H}_K}^2 \qquad (10)$$

- The tuning parameter $\lambda > 0$ balances the trade-off between estimating f (first term: measures the distance of the data from separability) and how well f can be approximated (second term: penalizes overfitting).

- After the minimizing f has been found, the SVM classifier is $C(\boldsymbol{x}) = \text{sign}\{f(\boldsymbol{x})\}, \boldsymbol{x} \in \Re^r$.

- (10) is nondifferentiable, but every $f \in \mathcal{H}$ can be written as sum

$$f(.) = f^{\parallel}(.) + f^{\perp}(.) = \sum_{i=1}^{n} \alpha_i K(\boldsymbol{x}_i, .) + f^{\perp}(.)$$

where $f^{\parallel} \in \mathcal{H}_K$ is the projection of f onto the subspace $\mathcal{H}_K$ of $\mathcal{H}$ and $f^{\perp}$ is in the subspace perpendicular to $\mathcal{H}_K$; that is, $\langle f^{\perp}(.), K(\boldsymbol{x}_i, .)\rangle_{\mathcal{H}} = 0$.

Nonlinear support vector machines

SVM as a regularization method

- We write $f(\boldsymbol{x}_i)$ via the reproducing property

$$f(\boldsymbol{x}_i) = \langle f(.), K(\boldsymbol{x}_i, .)\rangle = \left\langle f^{\parallel}(.), K(\boldsymbol{x}_i, .)\right\rangle + \langle f^{\perp}(.), K(\boldsymbol{x}_i, .)\rangle$$

- We have $$f(\boldsymbol{x}) = \textstyle\sum_{i=1}^{n} \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}) \qquad (11)$$

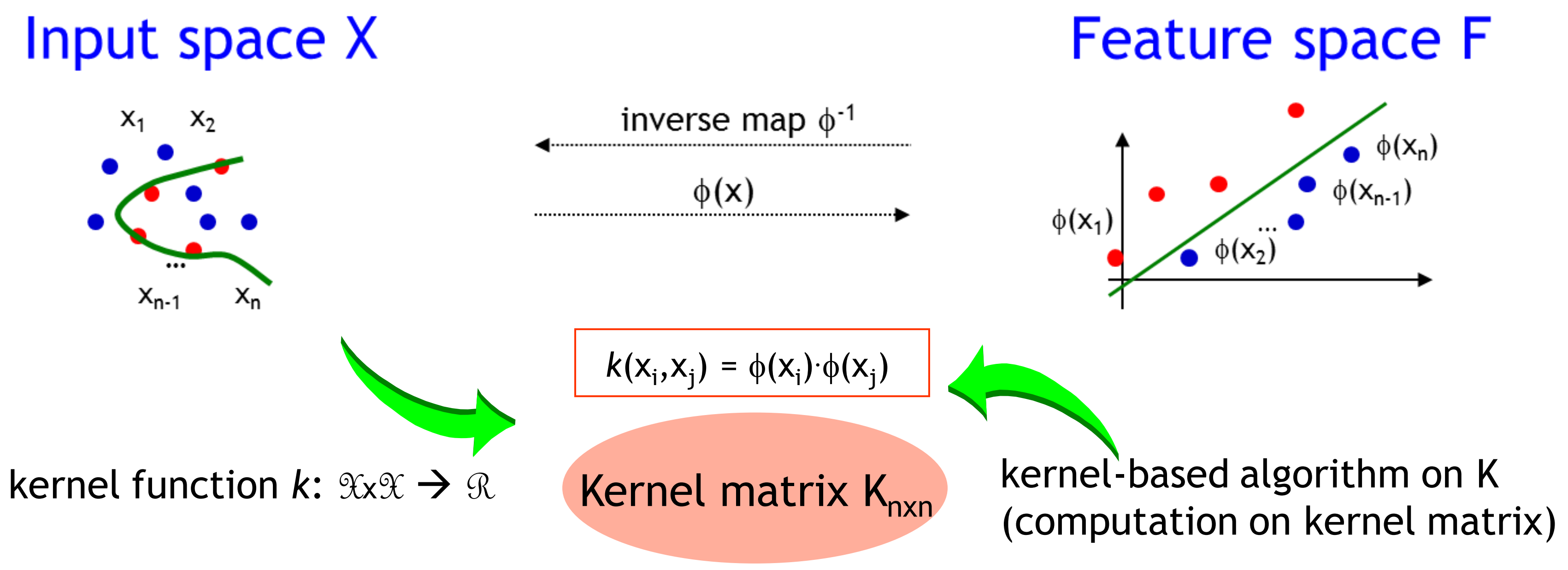
  is independent of $f^{\perp}$ as the second term is zero. We have

$$\|f\|^2_{\mathcal{H}_K} \geq \|\textstyle\sum_i \alpha_i K(\boldsymbol{x}_i, .)\|^2_{\mathcal{H}_K} \qquad (12)$$

This important result is known as the *representer,* says that *the minimizing f can be written as a linear combination of a reproducing kernel evaluated at each of the n data points* (Kimeldorf and Wahba, 1971). Problem (10) is equivalent to find β_0 and $\boldsymbol{\beta}$ to

$$minimize\ \ \frac{1}{n}\textstyle\sum_{i=1}^{n}(1 - y_i(\beta_0 + \boldsymbol{\Phi}(\boldsymbol{x}_i)^{\tau}\beta)_+ + \lambda\|\boldsymbol{\beta}\|^2 \qquad (13)$$

Kernel methods: math background


Input space X
Feature space F
x_1
x_2
...
x_{n-1}
x_n
inverse map ϕ^{-1}
$\phi(x)$
$\phi(x_1)$
$\phi(x_2)$
...
$\phi(x_{n-1})$
$\phi(x_n)$
$k(x_i,x_j) = \phi(x_i)\cdot\phi(x_j)$
kernel function k: $\mathcal{X}\mathrm{x}\mathcal{X} \to \mathcal{R}$
Kernel matrix K_{nxn}
kernel-based algorithm on K
(computation on kernel matrix)

Linear algebra, probability/statistics, functional analysis, optimization

- Mercer theorem: Any positive definite function can be written as an inner product in some feature space.
- Kernel trick: Using kernel matrix instead of inner product in the feature space.
- Representer theorem (Wahba): Every minimizer of $\min_{f\in\mathcal{H}}\{C(f,\{x_i,y_i\})+\Omega(\|f\|_H)\}$ admits a representation of the form $f(.) = \sum_{i=1}^{m}\alpha_i K(.,x_i)$

Content

Multiclass support vector machines

Multiclass SVM as a series of binary problems

- ***One-versus-rest***: Divide the *K*-class problem into *K* binary classification subproblems of the type "*k*th class" vs. "not *k*th class," k = 1, 2, . . .,K.

- ***One-versus-one***: Divide the K-class problem into comparisons of all pairs of classes.

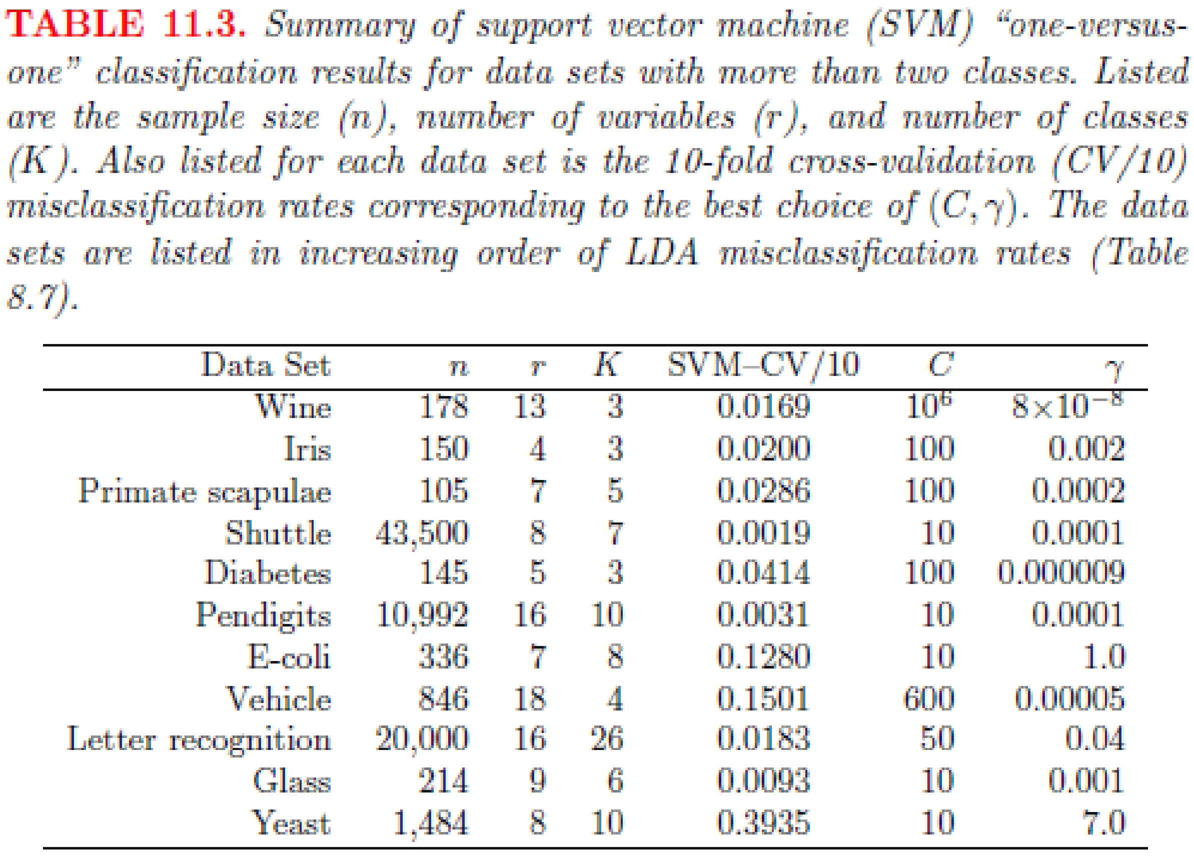


TABLE 11.3. *Summary of support vector machine (SVM) "one-versus-one" classification results for data sets with more than two classes. Listed are the sample size (n), number of variables (r), and number of classes (K). Also listed for each data set is the 10-fold cross-validation (CV/10) misclassification rates corresponding to the best choice of (C, γ). The data sets are listed in increasing order of LDA misclassification rates (Table 8.7).*

Data Set	n	r	K	SVM–CV/10	C	γ
Wine	178	13	3	0.0169	10^6	8×10^{-8}
Iris	150	4	3	0.0200	100	0.002
Primate scapulae	105	7	5	0.0286	100	0.0002
Shuttle	43,500	8	7	0.0019	10	0.0001
Diabetes	145	5	3	0.0414	100	0.000009
Pendigits	10,992	16	10	0.0031	10	0.0001
E-coli	336	7	8	0.1280	10	1.0
Vehicle	846	18	4	0.1501	600	0.00005
Letter recognition	20,000	16	26	0.0183	50	0.04
Glass	214	9	6	0.0093	10	0.001
Yeast	1,484	8	10	0.3935	10	7.0

Multiclass support vector machines

A true multiclass SVM

- To construct a true multiclass SVM classifier, we need to consider all K classes, Π_1,Π_2, . . . ,Π_K, simultaneously, and the classifier has to reduce to the binary SVM classifier if K = 2.
- One construction due to Lee, Lin, and Wahba (2004).
- Provide a unifying framework to multicategory SVM when there are either equal or unequal misclassification costs.

Content

Other issues
Support vector regression

- The SVM was designed for classification. Can we extend (or generalize) the idea to regression?
- How would the main concepts used in SVM — convex optimization, optimal separating hyperplane, support vectors, margin, sparseness of the solution, slack variables, and the use of kernels — translate to the regression situation?
- It turns out that all of these concepts find their analogues in regression analysis and they add a different view to the topic than the views we saw previously.
- *ε-insensitive loss functions*
- *Optimization for linear -insensitive loss*

Other issues

Optimization algorithms for SVMs

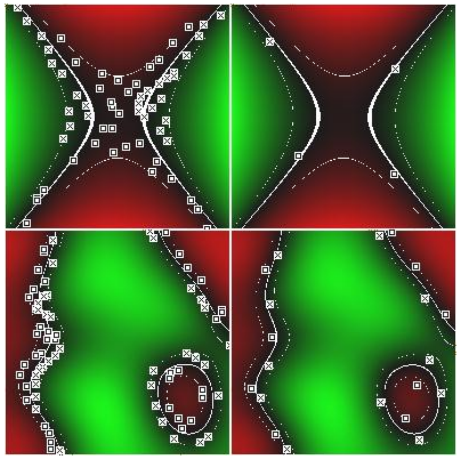
- Quadratic programming (QP) optimizers can solve problems having about a thousand points, general-purpose linear programming (LP) optimizers can deal with hundreds of thousands of points. With large data sets, however, a more sophisticated approach is required.
- *Gradient ascent*: Start with an estimate of the $\boldsymbol{\alpha}$-coefficient then successively update $\boldsymbol{\alpha}$ one α-coefficient by steepest ascent algorithm.
- *Chunking*: Start with a small subset; train an SVM on it, keep only support vectors; apply the resulting classifier to the remaining data.
- *Decomposition*: Similar to chunking, except that at each iteration, the size of the subset is always the same.
- *Sequential minimal optimization* (SMO): An extreme version of the decomposition algorithm.

Other issues

Software packages

TABLE 11.4. *Some implementations of SVM.*

Package	Implementation
SVM^{light}	http://svmlight.joachims.org/
LIBSVM	http://csie.ntu.edu.tw/~cjlin/libsvm/
SVMTorch II	http://www.idiap.ch/machine-learning.php
SVMsequel	http://www.isi.edu/~hdaume/SVMsequel/
TinySVM	http://chasen.org/~taku/TinySVM/



Some of our work on SVM and kernel methods

Nguyen, D.D., Ho, T.B. (2006). A Bottom-up Method for Simplifying Support Vector Solutions, *IEEE Transactions on Neural Networks*, Vol.17, No. 3, 792-796.

Nguyen, C.H., Ho, T.B. (2008). An Efficient Kernel Matrix Evaluation Measure, *Pattern Recognition*, Elsevier, 41 (11), 3366-3372

Content

Some challenges in kernel methods

Scalability and choice of kernels etc.

- The choice of kernel function. In general, there is no way of choosing or constructing a kernel that is optimal for a given problem.
- The complexity of kernel algorithms. Kernel methods access the feature space via the input samples and need to store all the relevant input samples.

  Examples: Store all support vectors or size of the kernel matrices grows quadratically with sample size → scalability of kernel methods.
- Incorporating priors knowledge and invariances in to kernel functions are some of the challenges in kernel methods.
- L1 regularization may allow some coefficients to be zore → hot topic
- Multiple kernel learning (MKL) is initially (2004, Lanckriet) of high computational cost → Many subsequent work, still ongoing, has not been a practical tool yet.