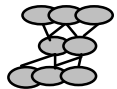


# Support Vector Machines (SVM)



# Outline

Linear pattern classifiers and optimal hyperplane

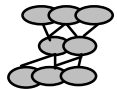
Optimization problem formulation

Statistical properties of optimal hyperplane

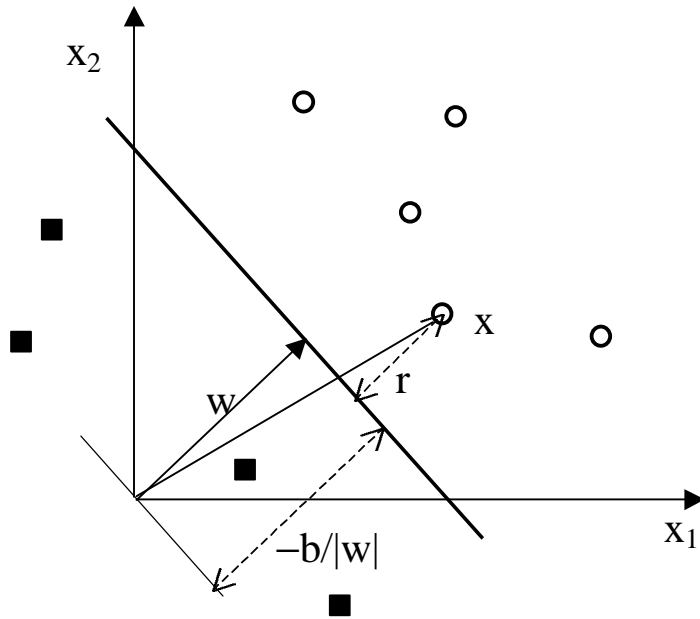
The case of non-separable patterns

Applications to general pattern classification

Mercer's Theorem



# Linear Hyperplane Classifier



Given:  $\{(x_i, d_i); i = 1 \text{ to } N, d_i \in \{+1, -1\}\}$ .

A linear hyperplane classifier is a hyperplane consisting of points  $x$  such that

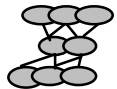
$$H = \{x \mid g(x) = w^T x + b = 0\}$$

$g(x)$ : discriminant function!.

For  $x$  in the side of  $o$  :  $w^T x + b \geq 0$ ;  $d = +1$ ;

For  $x$  in the side of  $\blacksquare$  :  $w^T x + b \leq 0$ ;  $d = -1$ .

Distance from  $x$  to  $H$ :  $r = w^T x / |w| - (-b / |w|) = g(x) / |w|$



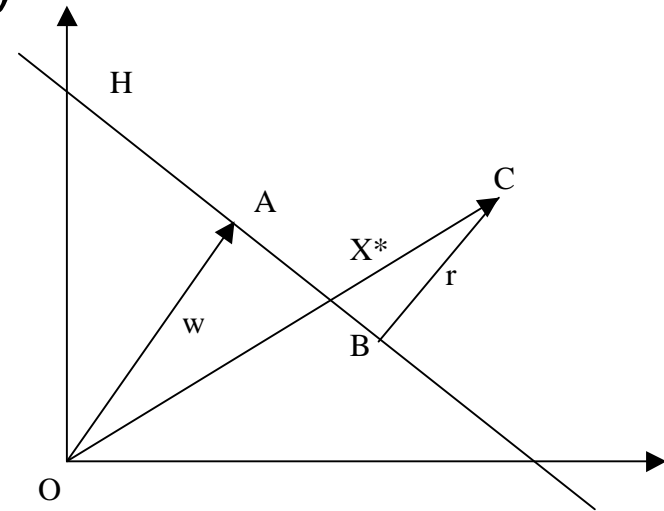
# Distance from a Point to a Hyper-plane

The hyperplane  $H$  is characterized by

$$(*) \quad w^T x + b = 0$$

$w$ : normal vector perpendicular to  $H$

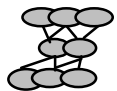
(\*) says any vector  $x$  on  $H$  that project to  $w$  will have a length of  $\overline{OA} = -b/|w|$ .



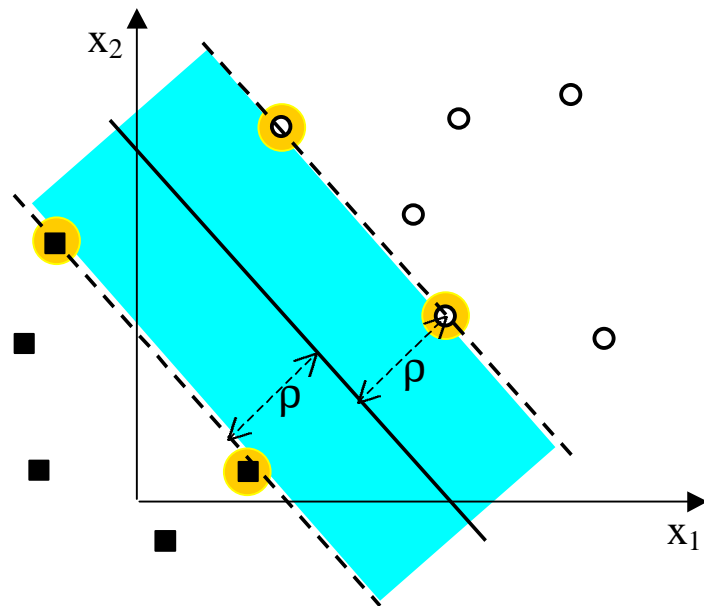
Consider a special point  $C$  corresponding to vector  $x^*$ . Its projection onto vector  $w$  is

$$w^T x^* / |w| = \overline{OA} + \overline{BC}. \quad \text{Or equivalently, } w^T x^* / |w| = r - b / |w|.$$

$$\text{Hence } r = (w^T x^* + b) / |w| = g(x^*) / |w|$$



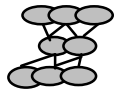
# Optimal Hyperplane: Linearly Separable Case



- Optimal hyperplane should be in the center of the gap.
- Supporting Vectors – Samples on the boundaries. Supporting vectors alone can determine optimal hyperplane.
- Question: How to find optimal hyperplane?

$$\text{For } d_i = +1, \quad g(x_i) = w^T x_i + b \geq \rho |w| \quad \Rightarrow \quad w_o^T x_i + b_o \geq 1$$

$$\text{For } d_i = -1, \quad g(x_i) = w^T x_i + b \leq -\rho |w| \quad \Rightarrow \quad w_o^T x_i + b_o \leq -1$$



## Separation Gap

For  $x_i$  being a supporting vector,

$$\text{For } d_i = +1, \quad g(x_i) = w^T x_i + b = \rho|w| \quad \Rightarrow \quad w_o^T x_i + b_o = 1$$

$$\text{For } d_i = -1, \quad g(x_i) = w^T x_i + b = -\rho|w| \quad \Rightarrow \quad w_o^T x_i + b_o = -1$$

Hence  $w_o = w/(\rho|w|)$ ,  $b_o = b/(\rho|w|)$ . But distance from  $x_i$  to hyperplane is  $\rho = g(x_i)/|w|$ . Thus  $w_o = w/g(x_i)$ , and  $\rho = 1/|w_o|$ .

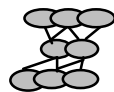
The maximum distance between the two classes is

$$2\rho = 2/|w_o|.$$

Hence the objective is to find  $w_o$ ,  $b_o$  to minimize  $|w_o|$  (so that  $\rho$  is maximized) subject to the constraints that

$$w_o^T x_i + b_o \geq 1 \text{ for } d_i = +1; \quad \text{and } w_o^T x_i + b_o \leq -1 \text{ for } d_i = -1.$$

Combine these constraints, one has:  $d_i \bullet (w_o^T x_i + b_o) \geq 1$



# Quadratic Optimization Problem Formulation

Given  $\{(x_i, d_i); i = 1 \text{ to } N\}$ , find  $w$  and  $b$  such that

$$\phi(w) = w^T w / 2$$

is minimized subject to  $N$  constraints

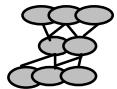
$$d_i \bullet (w^T x_i + b) \geq 0; \quad 1 \leq i \leq N.$$

## Method of Lagrange Multiplier

$$J(w, b, \alpha) = \phi(w) - \sum_{i=1}^N \alpha_i [d_i (w^T x_i + b) - 1]$$

$$\text{Set} \quad \frac{\partial J(w, b, \alpha)}{\partial w} = 0 \quad \Rightarrow \quad w = \sum_{i=1}^N \alpha_i d_i x_i$$

$$\frac{\partial J(w, b, \alpha)}{\partial \alpha} = 0 \quad \Rightarrow \quad \sum_{i=1}^N \alpha_i d_i = 0$$



## Optimization (continued)

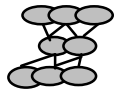
The solution of Lagrange multiplier problem is at a saddle point where the minimum is sought w.r.t.  $w$  and  $b$ , while the maximum is sought w.r.t.  $\alpha_i$ .

Kuhn-Tucker Condition: at the saddle point,

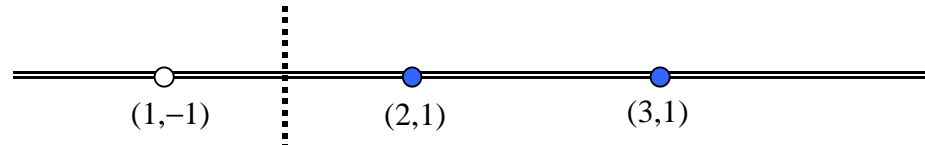
$$\alpha_i [d_i (w^T x_i + b) - 1] = 0 \quad \text{for } 1 \leq i \leq N.$$

$\Rightarrow$  If  $x_i$  is NOT a support vector, the corresponding  $\alpha_i = 0$ !

$\Rightarrow$  Hence, only support vector will affect the result of optimization!



# A Numerical Example



3 inequalities:  $1 \bullet w + b \leq -1$ ;  $2 \bullet w + b \geq +1$ ;  $3 \bullet w + b \geq +1$

$$J = w^2/2 - \alpha_1(-w-b-1) - \alpha_2(2w+b-1) - \alpha_3(3w+b-1)$$

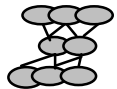
$$\partial J / \partial w = 0 \Rightarrow w = -\alpha_1 + 2\alpha_2 + 3\alpha_3$$

$$\partial J / \partial b = 0 \Rightarrow 0 = \alpha_1 - \alpha_2 - \alpha_3$$

Solve: (a)  $-w-b-1 = 0$  (b)  $2w+b-1 = 0$  (c);  $3w + b - 1 = 0$

(b) and (c) conflict each other. Solve (a), (b) yield  $w = 2$ ,

$b = -3$ . From the Kuhn-Tucker condition,  $\alpha_3 = 0$ . Thus,  $\alpha_1 = \alpha_2 = 2$ . Hence the solution of decision boundary is:  $2x - 3 = 0$ . or  $x = 1.5$ ! This is shown as the dash line in above figure.



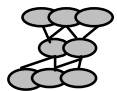
## Primal/Dual Problem Formulation

Given a constrained optimization problem with a convex cost function and linear constraints; a *dual problem* with the Lagrange multipliers providing the solution can be formulated.

### Duality Theorem (Bertsekas 1995)

- (a) If the primal problem has an optimal solution, then the dual problem has an optimal solution with the same optimal values.
- (b) In order for  $w_o$  to be an optimal solution and  $\alpha_o$  to be an optimal dual solution, it is necessary and sufficient that  $w_o$  is feasible for the primal problem and

$$\Phi(w_o) = J(w_o, b_o, \alpha_o) = \text{Min}_w J(w, b_o, \alpha_o)$$



## Formulating the Dual Problem

$$J(w, b, \mathbf{a}) = \frac{1}{2} w^T w - \sum_{i=1}^N \mathbf{a}_i d_i w^T x_i - b \sum_{i=1}^N \mathbf{a}_i d_i + \sum_{i=1}^N \mathbf{a}_i$$

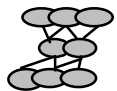
With  $w = \sum_{i=1}^N \mathbf{a}_i d_i x_i$  and  $\sum_{i=1}^N \mathbf{a}_i d_i = 0$ . These lead to a

### Dual Problem

$$\text{Maximize } Q(\mathbf{a}) = \sum_{i=1}^N \mathbf{a}_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \mathbf{a}_i \mathbf{a}_j d_i d_j x_i^T x_j$$

Subject to:  $\sum_{i=1}^N \mathbf{a}_i d_i = 0$  and  $\alpha_i \geq 0$  for  $i = 1, 2, \dots, N$ .

$$\text{Note } Q(\mathbf{a}) = \sum_{i=1}^N \mathbf{a}_i - \frac{1}{2} \begin{bmatrix} \mathbf{a}_1 d_1 & \cdots & \mathbf{a}_N d_N \end{bmatrix} \begin{bmatrix} x_1^2 & \cdots & x_1 x_N \\ \vdots & \ddots & \vdots \\ x_N x_1 & \cdots & x_N^2 \end{bmatrix} \begin{bmatrix} \mathbf{a}_1 d_1 \\ \vdots \\ \mathbf{a}_N d_N \end{bmatrix}$$



## Numerical Example (cont'd)

$$Q(\mathbf{a}) = \sum_{i=1}^3 a_i - \frac{1}{2} \begin{bmatrix} -a_1 & a_2 & a_3 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{bmatrix} \begin{bmatrix} -a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

$$\text{or } Q(\alpha) = \alpha_1 + \alpha_2 + \alpha_3 - [0.5\alpha_1^2 + 2\alpha_2^2 + 4.5\alpha_3^2 - 2\alpha_1\alpha_2 - 3\alpha_1\alpha_3 + 6\alpha_2\alpha_3]$$

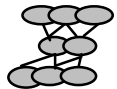
subject to constraints:  $-\alpha_1 + \alpha_2 + \alpha_3 = 0$ , and

$$\alpha_1 \geq 0, \alpha_2 \geq 0, \text{ and } \alpha_3 \geq 0.$$

Use Matlab™ Optimization tool box command:

`x=fmincon('qalpha',X0, A, B, Aeq, Beq)`

The solution is  $[\alpha_1 \ \alpha_2 \ \alpha_3] = [2 \ 2 \ 0]$  as expected.



## Implication of Minimizing $\|w\|$

Let  $D$  denote the diameter of the smallest hyper-ball that encloses all the input training vectors  $\{x_1, x_2, \dots, x_N\}$ . The set of optimal hyper-planes described by the equation

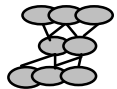
$$W_o^T x + b_o = 0$$

has a VC-dimension  $h$  bounded from above as

$$h \leq \min \{ \lceil D^2/\rho^2 \rceil, m_0 \} + 1$$

where  $m_0$  is the dimension of the input vectors, and  $\rho = 2/\|w_o\|$  is the margin of the separation of the hyper-planes.

VC-dimension determines the *complexity of the classifier structure*, and usually the smaller the better.



## Non-separable Cases

Recall that in linearly separable case, each training sample pair  $(x_i, d_i)$  represents a linear inequality constraint

$$d_i(w^T x_i + b) \geq 1, \quad i = 1, 2, \dots, N$$

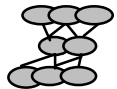
If the training samples are not linearly separable, the constraint can be modified to yield a *soft constraint*.

$$d_i(w^T x_i + b) \geq 1 - \zeta_i, \quad i = 1, 2, \dots, N$$

$\{\zeta_i; 1 \leq i \leq N\}$  are known as *slack variables*. If  $\zeta_i > 1$ , then the corresponding  $(x_i, d_i)$  will be mis-classified.

The minimum error classifier would minimize  $\sum_{i=1}^N I(z_i - 1)$ , but it is non-convex w.r.t.  $w$ . Hence an approximation is to minimize

$$\Phi(w, \mathbf{z}) = \frac{1}{2} w^T w + C \sum_{i=1}^N z_i$$



# Primal and Dual Problem Formulation

Primal Optimization Problem Given  $\{(x_i, \zeta_i); 1 \leq i \leq N\}$ . Find  $w, b$  such that

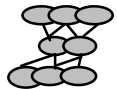
$$\Phi(w, \mathbf{z}) = \frac{1}{2} w^T w + C \sum_{i=1}^N z_i$$

is minimized subject to the constraints (i)  $\zeta_i \geq 0$ , and (ii)  $d_i(w^T x_i + b) \geq 1 - \zeta_i$  for  $i = 1, 2, \dots, N$ .

Dual Optimization Problem Given  $\{(x_i, \zeta_i); 1 \leq i \leq N\}$ . Find Lagrange multipliers  $\{\alpha_i; 1 \leq i \leq N\}$  such that

$$Q(\mathbf{a}) = \sum_{i=1}^N \mathbf{a}_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \mathbf{a}_i \mathbf{a}_j d_i d_j x_i^T x_j$$

is maximized subject to the constraints (i)  $0 \leq \mathbf{a}_i \leq C$  (a user-specified positive number) and (ii)  $\sum_{i=1}^N \mathbf{a}_i d_i = 0$



## Solution to the Dual Problem

Optimal Solution to the Dual problem is:

$$w_o = \sum_{i=1}^{N_s} \mathbf{a}_{o,i} d_i x_i \quad N_s: \# \text{ of support vectors.}$$

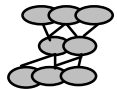
Kuhn-Tucker condition implies for  $i = 1, 2, \dots, N$ ,

$$(i) \quad \alpha_i [d_i (w^T x_i + b) - 1 + \zeta_i] = 0 \quad (*)$$

$$(ii) \quad \mu_i \zeta_i = 0$$

$\{\mu_i; 1 \leq i \leq N\}$  are Lagrange multipliers to enforce the condition  $\zeta_i \geq 0$ . At optimal point of the primal problem,  $\partial \Phi / \partial \zeta_i = 0$ . One may deduce that  $\zeta_i = 0$  if  $\alpha_i \leq C$ . Solving (\*), we have

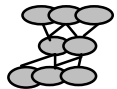
$$b_o = \frac{1}{N} \sum_{i=1}^N \mathbf{a}_i (w^T x_i - d_i)$$



# Matlab Implementation

```
% svm1.m: basic support vector machine
% X: N by m matrix. i-th row is x_i
% d: N by 1 vector. i-th element is d_i
% X, d should be loaded from file or read from input.
% call MATLAB optimization tool box function fmincon
a0=eps*ones(N,1);
C = 1;
a=fmincon('qfun',a0,[],[],d',0,zeros(N,1),C*ones(N,1),...
         [],[],X,d)
wo=X'*(a.*d)
bo=sum(diag(a)*(X*wo-d))/sum([a > 10*eps])

function y=qfun(a,X,d);
% the Q(a) function. Note that it is actually -Q(a)
% because we call fmincon to minimize -Q(a) is
% the same as to maximize Q(a)
[N,m]=size(X);
y=-ones(1,N)*a+0.5*a'*diag(d)*X*X'*diag(d)*a;
```



# Inner Product Kernels

In general, if the input is first transformed via a set of nonlinear functions  $\{\phi_i(x)\}$  and then subject to the hyperplane classifier

$$g(x) = \sum_{j=1}^p w_j \mathbf{j}_j(x) + b = \sum_{j=0}^p w_j \mathbf{j}_j(x) = \mathbf{w}^T \mathbf{j} \quad b = w_0; \quad \mathbf{j}_0(x) = 1$$

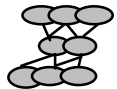
Define the inner product kernel as

$$K(x, y) = \sum_{j=0}^p \mathbf{j}_j(x) \mathbf{j}_j(y) = \mathbf{j} \mathbf{j}^T$$

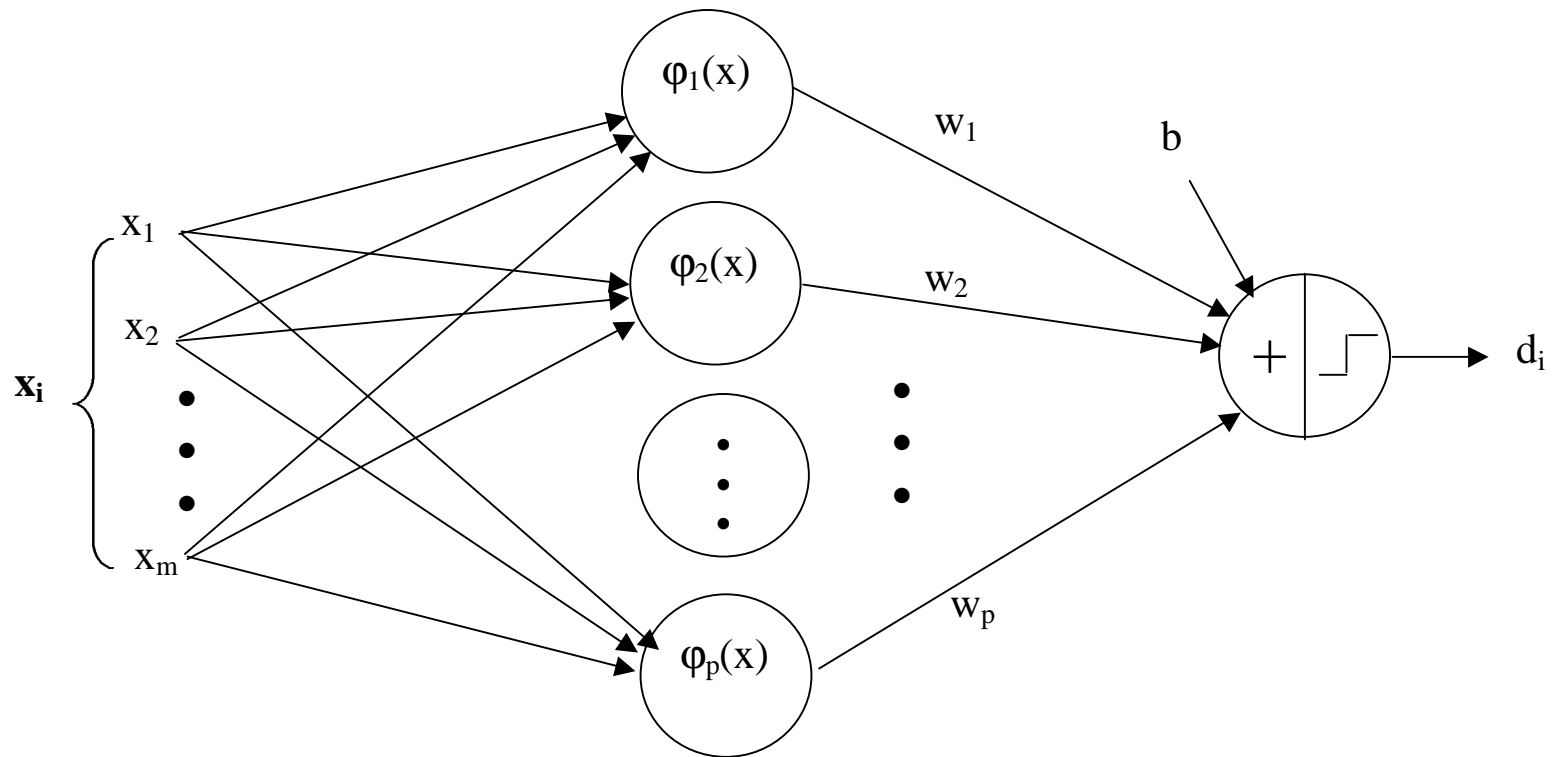
one may obtain a dual optimization problem formulation as:

$$Q(\mathbf{a}) = \sum_{i=1}^N a_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N a_i a_j d_i d_j K(x_i, x_j)$$

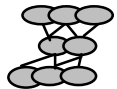
Often,  $\dim$  of  $\phi$  ( $=p+1$ )  $\gg$   $\dim$  of  $x$ !



# General Pattern Recognition with SVM



By careful selection of the nonlinear transformation  $\{\phi_j(x); 1 \leq j \leq p\}$ , any pattern recognition problem can be solved.



# Polynomial Kernel

Consider a polynomial kernel

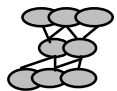
$$K(x, y) = (1 + x^T y)^2 = 1 + 2 \sum_{i=1}^m x_i y_i + 2 \sum_{i=1}^m \sum_{j=i+1}^m x_i y_i x_j y_j + \sum_{i=1}^m x_i^2 y_i^2$$

Let  $K(x, y) = \phi^T(\mathbf{x}) \phi(\mathbf{x})$ , then

$$\begin{aligned} \phi(\mathbf{x}) &= [1 \quad x_1^2, \dots, x_m^2, \sqrt{2} x_1, \dots, \sqrt{2} x_m, \sqrt{2} x_1 x_2, \dots, \sqrt{2} x_1 x_m, \\ &\quad \sqrt{2} x_2 x_3, \dots, \sqrt{2} x_2 x_m, \dots, \sqrt{2} x_{m-1} x_m] \\ &= [1 \quad \phi_1(x), \dots, \phi_p(x)] \end{aligned}$$

where  $p = 1 + m + m + (m-1) + (m-2) + \dots + 1 = (m+2)(m+1)/2$

Hence, using a kernel, a low dimensional pattern classification problem (with dimension  $m$ ) is solved in a higher dimensional space (dimension  $p+1$ ). But only  $\phi_j(x)$  corresponding to support vectors are used for pattern classification!

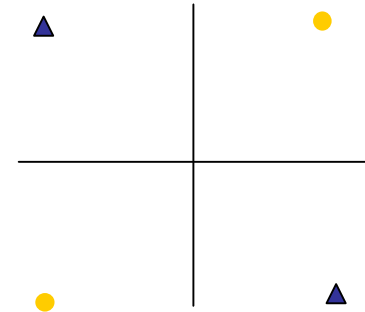


# Numerical Example: XOR Problem

Training samples:

$$(-1 \ -1; -1), (-1 \ 1 \ +1),$$

$$(1 \ -1 \ +1), (1 \ 1 \ -1)$$

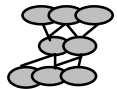


$\mathbf{x} = [x_1, x_2]^T$ . Use  $K(\mathbf{x}, \mathbf{x}_i) = (1 + \mathbf{x}^T \mathbf{x}_i)^2$  one has

$$\varphi(\mathbf{x}) = [1 \ x_1^2 \ x_2^2 \ \sqrt{2} x_1, \ \sqrt{2} x_2, \ \sqrt{2} x_1 x_2]^T$$

$$\Phi = \begin{bmatrix} 1 & 1 & 1 & -\sqrt{2} & -\sqrt{2} & \sqrt{2} \\ 1 & 1 & 1 & -\sqrt{2} & \sqrt{2} & -\sqrt{2} \\ 1 & 1 & 0 & \sqrt{2} & -\sqrt{2} & -\sqrt{2} \\ 1 & 1 & 1 & \sqrt{2} & \sqrt{2} & \sqrt{2} \end{bmatrix}; \quad K(\mathbf{x}_i, \mathbf{x}_j) = \Phi \Phi^T = \begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{bmatrix};$$

Note  $\dim[\varphi(\mathbf{x})] = 6 > \dim[\mathbf{x}] = 2!$



## XOR Problem (Continued)

Note that  $K(x_i, x_j)$  can be calculated directly without using  $\Phi$ !

$$\text{E.g. } K_{1,1} = (1 + [-1 \ -1] \begin{bmatrix} -1 \\ -1 \end{bmatrix})^2 = 9; \quad K_{1,2} = (1 + [-1 \ -1] \begin{bmatrix} -1 \\ 1 \end{bmatrix})^2 = 1$$

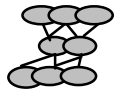
The corresponding Lagrange multiplier  $\alpha = (1/8)[1 \ 1 \ 1 \ 1]^T$ .

$$\mathbf{w} = \sum_{i=1}^N \alpha_i d_i \mathbf{j}(\mathbf{x}_i) = \Phi^T [\alpha_1 d_1 \quad \alpha_2 d_2 \quad \cdots \quad \alpha_N d_N]^T$$

$$\begin{aligned} &= (1/8)(-1) \varphi(\mathbf{x}_1) + (1/8)(1) \varphi(\mathbf{x}_2) + (1/8)(1) \varphi(\mathbf{x}_3) + (1/8)(-1) \varphi(\mathbf{x}_4) \\ &= [0 \ 0 \ 0 \ 0 \ 0 \ -1/\sqrt{2}]^T \end{aligned}$$

Hence the hyperplane is:  $y = \mathbf{w}^T \varphi(\mathbf{x}) = -x_1 x_2$

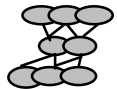
$(x_1, x_2)$	$(-1, -1)$	$(-1, +1)$	$(+1, -1)$	$(+1, +1)$
$y = -1 x_1 x_2$	$-1$	$+1$	$+1$	$-1$



## Other Types of Kernels

type of SVM	K(x,y)	Comments
Polynomial learning machine	$(\mathbf{x}^T \mathbf{y} + 1)^p$	p: selected a priori
Radial basis function	$\exp\left(-\frac{1}{2\sigma^2} \ \mathbf{x} - \mathbf{y}\ ^2\right)$	$\sigma^2$ : selected a priori
Two-layer perceptron	$\tanh(\beta_0 \mathbf{x}^T \mathbf{y} + \beta_1)$	only some $\beta_0$ and $\beta_1$ values are feasible.

What kernel is feasible? It must satisfy the "Mercer's theorem"!



# Mercer's Theorem

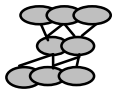
Let  $K(\mathbf{x}, \mathbf{y})$  be a continuous, symmetric kernel, defined on  $a \leq \mathbf{x}, \mathbf{y} \leq b$ .  $K(\mathbf{x}, \mathbf{y})$  admits an eigen-function expansion

$$K(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{\infty} \lambda_i \mathbf{j}_i(\mathbf{x}) \mathbf{j}_i(\mathbf{y})$$

with  $\lambda_i > 0$  for each  $i$ . This expansion converges absolutely and uniformly if and only if

$$\int_b^a \int_b^a K(\mathbf{x}, \mathbf{y}) \mathbf{y}(\mathbf{x}) \mathbf{y}(\mathbf{y}) d\mathbf{x} d\mathbf{y} \geq 0$$

for all  $\mathbf{y}(\mathbf{x})$  such that  $\int_b^a \mathbf{y}^2(\mathbf{x}) d\mathbf{x} < \infty$ .



# Testing with Kernels

For many types of kernels,  $\phi(\mathbf{x})$  can not be explicitly represented or even found. However,

$$\mathbf{w} = \sum_{i=1}^N \mathbf{a}_i d_i \mathbf{j}(\mathbf{x}_i) = \Phi^T [\mathbf{a}_1 d_1 \quad \mathbf{a}_2 d_2 \quad \cdots \quad \mathbf{a}_N d_N]^T = \Phi^T \mathbf{f}$$

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) = \mathbf{f}^T \Phi \phi(\mathbf{x}) = \mathbf{f}^T \mathbf{K}(\mathbf{x}_i, \mathbf{x}) = \mathbf{K}(\mathbf{x}, \mathbf{x}_i) \mathbf{f}$$

Hence there is no need to know  $\phi(\mathbf{x})$  explicitly! For example, in the XOR problem,  $\mathbf{f} = (1/8)[-1 \quad +1 \quad +1 \quad -1]^T$ . Suppose that  $\mathbf{x} = (-1, +1)$ , then

$$y(\mathbf{x}) = \mathbf{K}(\mathbf{x}, \mathbf{x}_j) \mathbf{f} = \begin{bmatrix} 1 + [-1 \quad -1] \begin{bmatrix} -1 \\ 1 \end{bmatrix} \\ 1 + [-1 \quad 1] \begin{bmatrix} -1 \\ 1 \end{bmatrix} \\ 1 + [1 \quad -1] \begin{bmatrix} -1 \\ 1 \end{bmatrix} \\ 1 + [1 \quad 1] \begin{bmatrix} -1 \\ 1 \end{bmatrix} \end{bmatrix} \begin{bmatrix} -1/8 \\ 1/8 \\ 1/8 \\ -1/8 \end{bmatrix}$$

$$= [1 \quad 9 \quad 1 \quad 1] [-1/8 \quad 1/8 \quad 1/8 \quad -1/8]^T = 1$$

