EE 583
PATTERN RECOGNITION

Bayes Decision Theory
Supervised Learning
Linear Discriminant Functions

- Definitions, Decision Surfaces
- Two-category Linearly Separable : Perceptron Criterion
- Non-separable case : MSE & Ho-Kashyap
- Support Vector Machines

Unsupervised Learning

Linear Discriminant Functions

- In previous parametric supervised approaches, it is assumed that the form of probability density is known
- Now, assume the form of the discriminant function is known
- Assume this form is linear either in components or functions of x
- In such cases, LDF are relatively easy to compute and analytically attractive

LDF and Decision Surfaces (1/2)

- Assume a two-class problem, then LDF :

$$g(x) = \underbrace{\vec{w}^t}_{\text{weight vector}}\vec{x} + \underbrace{w_0}_{\text{threshold weight}}$$

$$\begin{aligned} &Decide\ \ w_1\ \ if\ \ g(\vec{x}) > 0 \\ &\qquad\qquad w_2\ \ if\ \ g(\vec{x}) < 0 \\ &either\ \ class\ \ if\ \ g(\vec{x}) = 0 \end{aligned}$$

- $g(x)=0$ is a *decision surface*
  - $g(x)$ is linear ➔ surface is a hyperplane
- This hyperplane, H, divides the feature space into two subspaces, R_1 & R_2
- Vector w is normal to any vector on H

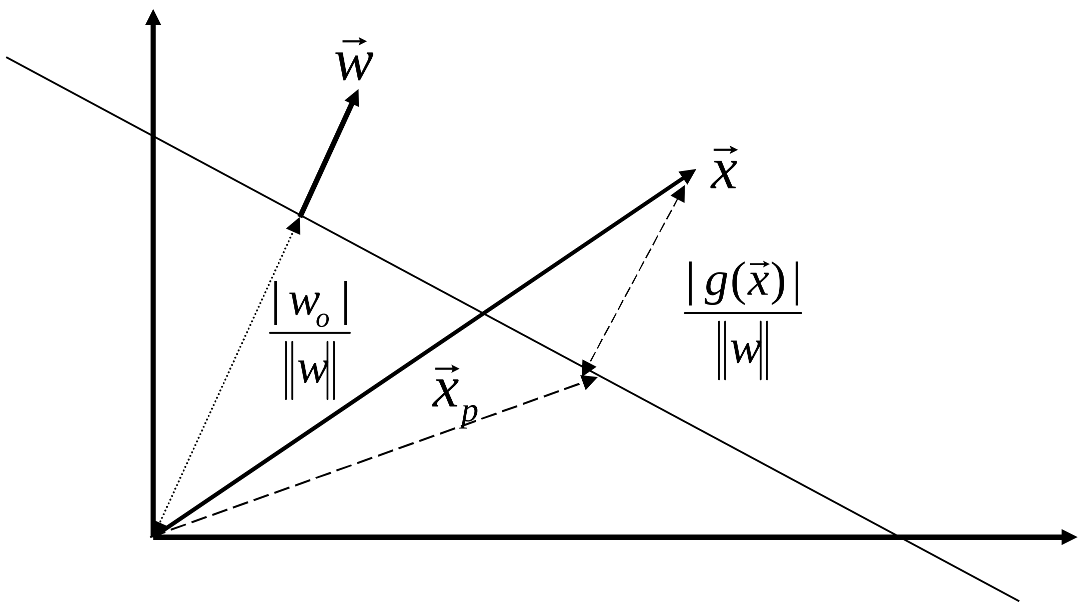
LDF and Decision Surfaces (2/2)

$$g(x) = \underbrace{\vec{w}^t}_{\text{weight vector}} \vec{x} + \underbrace{w_0}_{\text{threshold weight}}$$

$$\begin{aligned} &Decide \;\; w_1 \;\; if \;\; g(\vec{x}) > 0 \\ &\qquad\quad\;\; w_2 \;\; if \;\; g(\vec{x}) < 0 \\ &either \;\; class \;\; if \;\; g(\vec{x}) = 0 \end{aligned}$$

- Note that $g(x)$ gives a measure of distance from x to H

$$\vec{x} = \vec{x}_p + r\frac{\vec{w}}{\|\vec{w}\|} \qquad g(\vec{x}_p) = 0$$

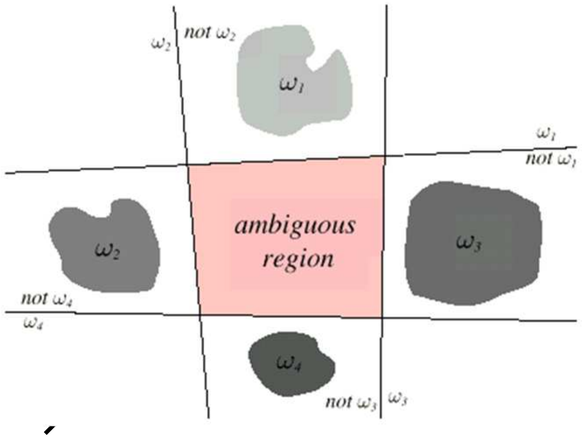


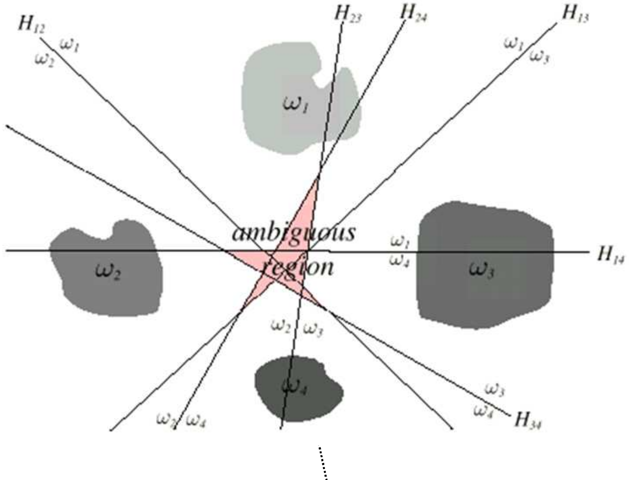
$$g(\vec{x}) = \vec{w}^t\vec{x} + w_0 = r\|w\|$$

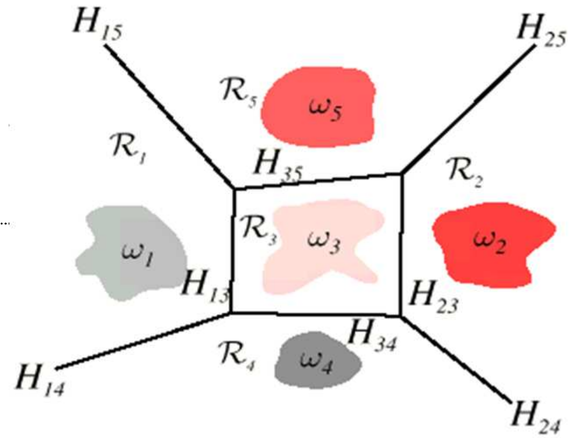
$$\Rightarrow r = \frac{g(\vec{x})}{\|w\|}$$

LDF for Multi-class Problems

There are 3 ways to classify <u>multi-category problems</u> :







<u>Wi/not-Wi dichotomies</u>

(c-1 2-class problems)

Find a LDF that will separate i-th class from rest of the classes

<u>Wi/Wj dichotomies</u>

(c(c-1)/2 2-class problems)

Find a LDF that will separate i-th class from j-th class

<u>Linear Machine</u>

Divides the feature space into c while $g_i(x)$ being the largest DF in i-th class/region

Two-category Linearly Separable Case (1/3)

- If a vector that classifies correctly all the samples of two classes exits, than the samples are called *linearly separable.*

- In order to simplify analysis, perform the conversion

$$g(\vec{x}) = w_0 + \sum_{i=1}^{d} w_i x_i \text{ , } let\ \vec{y} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix}, \vec{a} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} \rightarrow g(\vec{x}) = \vec{a}^t \vec{y}$$

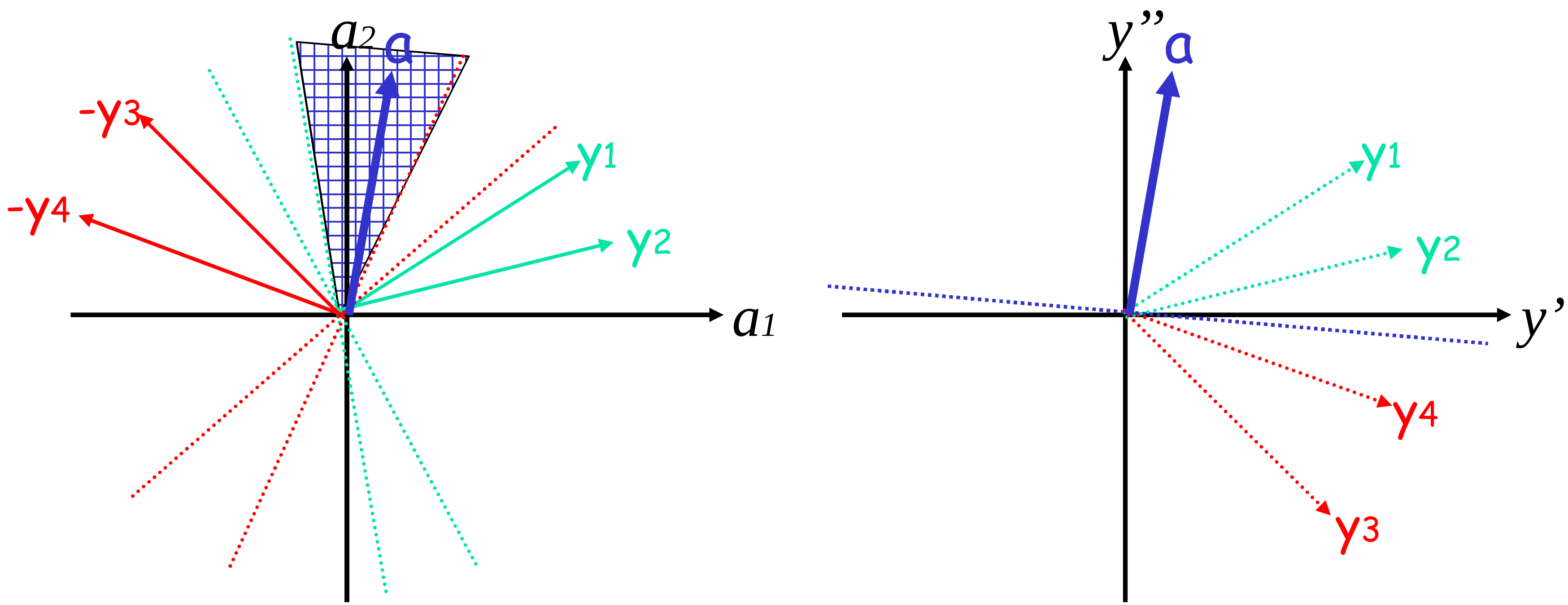
- Assume n-samples of *y* vector, some labeled w_1 and some w_2; then the unknown a vector has the following constraints according to the correct classifications :

$$\vec{y}_i \text{ is labelled } w_1 \quad \Rightarrow \vec{a}^t \vec{y}_i > 0$$

$$\vec{y}_i \text{ is labelled } w_2 \quad \Rightarrow \vec{a}^t \vec{y}_i < 0 \text{ or } \vec{a}^t (-\vec{y}_i) > 0$$

Two-category Linearly Separable Case (2/3)

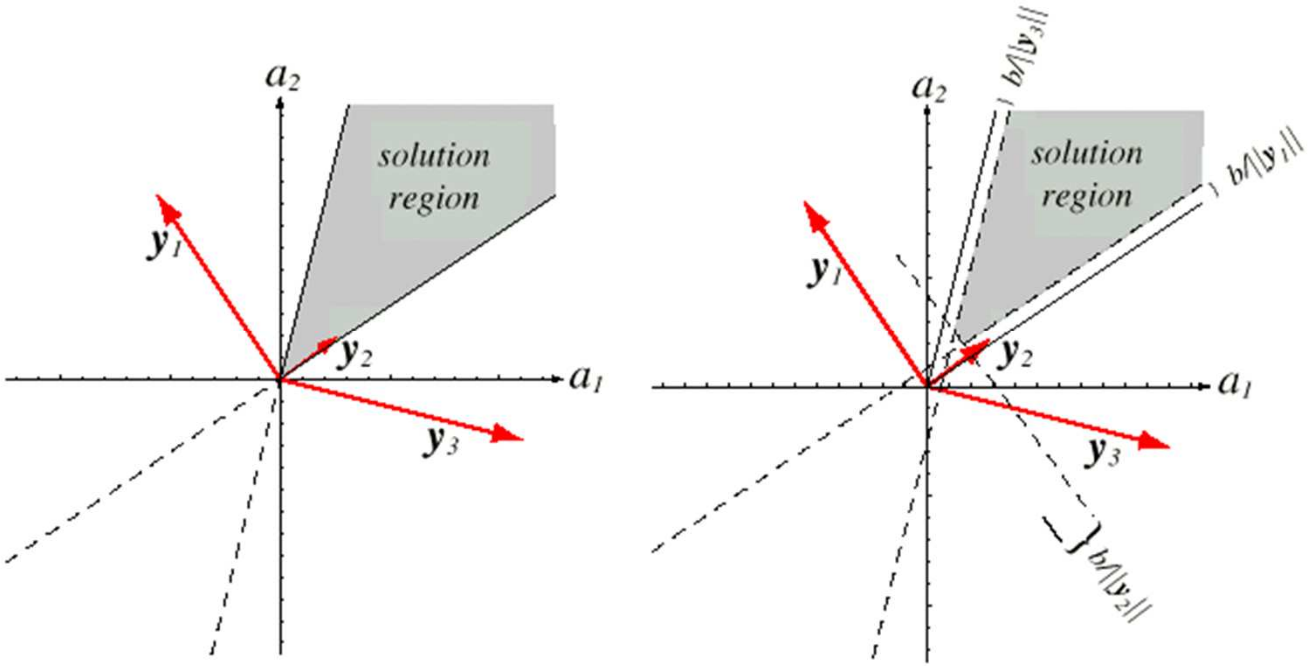
- In the "solution space", each sample y is a constraint to find a solution for vector a, such that y vector is normal to the hyperplane $a^t y=0$



Since there is a (shaded) solution region, solution vector is not unique;

Two-category Linearly Separable Case (3/3)

- Since solution vector is not unique, one option is to choose this vector such that $a^t y_i > b > 0$ for all i



- Motivation for going to the `middle' portion is due to natural belief for better classification of new samples

Minimizing Perceptron Criterion (1/3)

- Lets define a criterion function for solving $a^t y_i > 0$

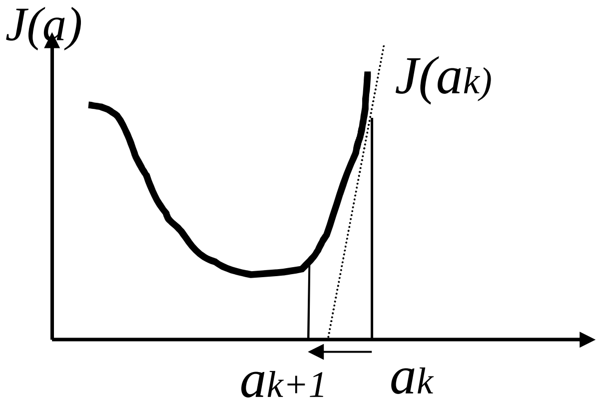
$$J_p(\vec{a}) = \sum_{y \in Y(\vec{a})} (-\vec{a}^t \vec{y}) \qquad Y(\vec{a}): set\ of\ misclassified\ samples$$

- Note that
  - $-a^t y_i$ is always positive for misclassified data and equal to zero, if all samples are correctly classified
  - *Perceptron* criterion is proportional to the sum distances of misclassified samples to decision boundary
- Using one of the descent procedures, minimize $J_p(a)$

Minimizing Perceptron Criterion (2/3)

- In order to find a "solution vector" a, the criterion $J(a)$ should be minimized using an optimization method

- *<u>Steepest Descent</u>* is such an optimization technique:



$$a_{k+1} = a_k - \underbrace{\rho_k}_{\text{step size}} \nabla J(a_k)$$

- Step size choice is critical :
  - if it is too small → slow convergence
  - if is is too large → convergence overshoot, diverge

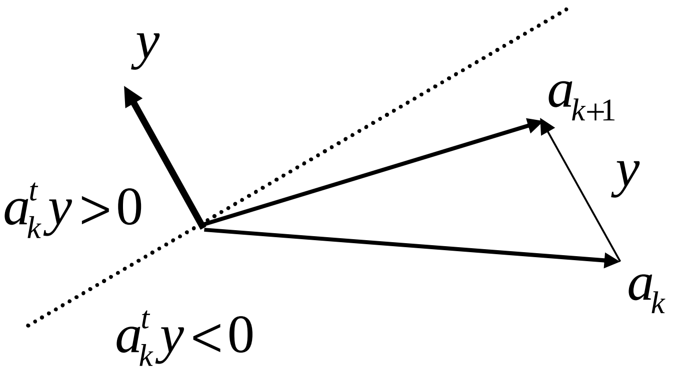
Minimizing Perceptron Criterion (3/3)

$$J_p(\vec{a}) = \sum_{y \in Y(\vec{a})} (-\vec{a}^t \vec{y}) \qquad Y(\vec{a}): set\ of\ misclassified\ samples$$

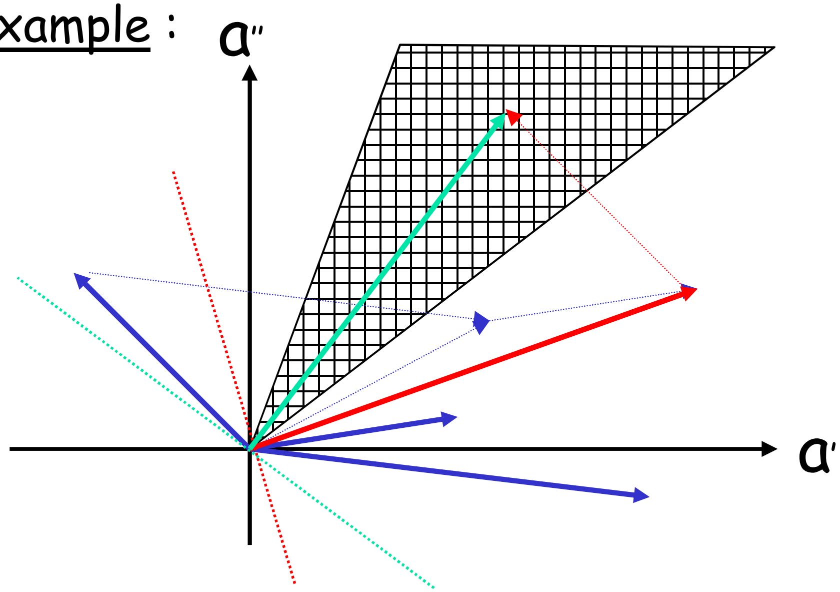
- Gradient of $J_p(a)$ is obtained as : $\nabla J_p(\vec{a}) = \sum_{y \in Y(\vec{a})} (-\vec{y})$
- Using gradient descent, a value for the k^th iteration : $\vec{a}_{k+1} = \vec{a}_k + \rho_k \sum_{y \in Y_k} \vec{y}$
- If stepsize is constant → *fixed increment* case

<u>Geometrical Interpretation</u> :

Angle between a_k and y should be < π/2



<u>Example</u> :



- If the samples are linearly separable, convergence to a solution is guaranteed by the Perceptron Method (read the proof at Duda&Hart)

Non-separable Behavior

- Approaches based on separability assumption, relentlessly search for an error-free solution
- In practice, if there is no a priori info about separability,
  - such procedures should be modified with an appropriate termination rule so that divergence is avoided
  - one should seek for other approaches that do not require separability condition
    - MSE procedures
    - Ho-Kashyap approach

Minimum Squared Error Procedures (1/2)

- Rather than trying to make $a^t y_i > 0$ for all i, lets make $a^t y_i = b_i$ for an arbitrary constant $b_i > 0$,

$$Y \equiv \begin{bmatrix} \vec{y}_1^{\,t} \\ \vdots \\ \vec{y}_i^{\,t} \\ \vdots \\ \vec{y}_n^{\,t} \end{bmatrix} \quad ; \quad \vec{b} \equiv \begin{bmatrix} b_1 \\ \vdots \\ b_i \\ \vdots \\ b_n \end{bmatrix} \quad \Rightarrow \quad \textit{Find } \vec{a} \textit{ such that } \quad Y\vec{a} = \vec{b}$$

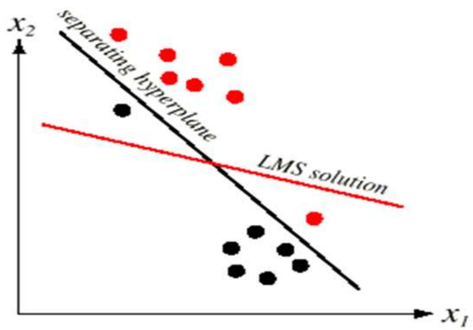
- Since the relation above is usually overdetermined, a solution can be obtained by minimizing the square of the error, $e = |Ya - b|^2$,

Minimum Squared Error Procedures (2/2)

- $e=|Ya-b|^2$ is minimized by finding the *pseudo inverse* of Y :

$$\hat{a} = Y^{*}\vec{b} = \underbrace{(Y^{t}Y)^{-1}Y^{t}}_{pseudo-inverse}\,\vec{b}$$

- Note that such approaches do not try to find a separating plane, but rather minimize the average error



- Selection of b is important ➔ MSE method will be equivalent to Fisher's LD with appropriate b

Ho-Kashyap Procedures (1/2) :

- Perceptron procedure finds separating plane, if samples are linearly separable,
  - but do not converge for non-separable problems
- MSE procedure yields a weight vector in both separable and non-separable cases,
  - but there is no guarantee to have a separating plane, even if the samples are linearly separable
- If margin vector, b, is chosen arbitrarily, all one can guarantee is minimization of $|Ya-b|^2$,
  - but for a linearly separable problem, all the elements of b must be greater than zero; i.e. there exists a' and b' such that $Ya' = b' > 0$

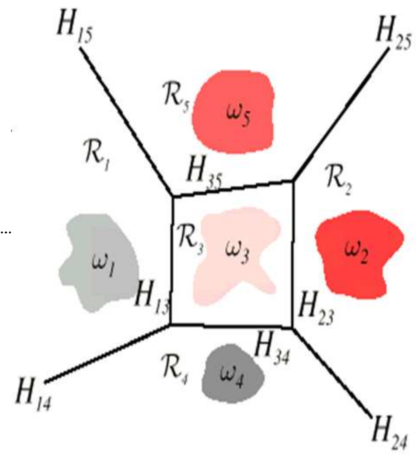
Ho-Kashyap Procedures (2/2) :

- Minimize $|Ya\text{-}b|^2$ varying both a and b within the criterion function, J_s $\quad J_s(a,b) = \|Ya - b\|^2$
- In order to use a modified version of gradient descent procedure, find the gradients as
$$\nabla_a J_s(a,b) = 2Y^t(Ya - b) \quad , \nabla_b J_s(a,b) = -2(Ya - b)$$
- For any value of b, $a=Y^*b$, but for any value of a, the same is not true, since we have constraint $b>0$
- <u>Algorithm</u> : $\quad b_1 > 0 \quad \textit{but arbitrary}$
$$a_k = Y^* b_k$$
$$b_{k+1} = b_k - \rho_k \nabla_b J_s(a_k, b_k) = b_k + \rho_k \underbrace{(Ya_k - b_k)}_{\textit{should always } (+)}$$
- For non-separable case, $Ya_k\text{-}b_k<0$ for all elements of b

Multi-category Generalizations (1/4)

- All the methods, we have examined so far are proposed for <u>two-class problems</u>

- *Linear Machine* approach can be utilized to generalize these algorithms to multi-category
- If a Linear Machine exits that classifies all the samples correctly, these samples are called linearly separable
- Assume the samples are linearly separable, then for c classes there exist a set of weight vectors, satisfying



<u>Linear Machine</u>

Divides the feature space into c while *gi(x)* being the largest DF in i-th class/region

$$\vec{\hat{a}}_1,\ldots,\vec{\hat{a}}_c \quad such\ that\ \ for\ \vec{y}_k \in Y_i,\ \vec{\hat{a}}_i^{\,t}\vec{y}_k > \vec{\hat{a}}_j^{\,t}\vec{y}_k \quad for\ all\ i \neq j$$

Multi-category Generalizations (2/4)

- Multi-category problems can be reduced to two-class problems by
- Assume sample y belongs to class-1 :

$$(\hat{\vec{a}}_1^{\,t} - \hat{\vec{a}}_j^{\,t})\vec{y} > 0 \quad for \;\; j = 2, \ldots, c$$

$$\hat{\alpha} = \begin{bmatrix} \vec{a}_1 \\ \vec{a}_2 \\ \vdots \\ \vec{a}_c \end{bmatrix} \quad should\ classify \quad \eta_{12} = \begin{bmatrix} \vec{y} \\ -\vec{y} \\ \vdots \\ 0 \end{bmatrix}, \ldots, \eta_{1c} = \begin{bmatrix} \vec{y} \\ 0 \\ \vdots \\ -\vec{y} \end{bmatrix} \quad correctly$$

$$\Rightarrow \hat{\alpha}^t \eta_{1j} > 0 \;\; for\ all \;\; j \neq 1$$

Multi-category Generalizations (3/4)

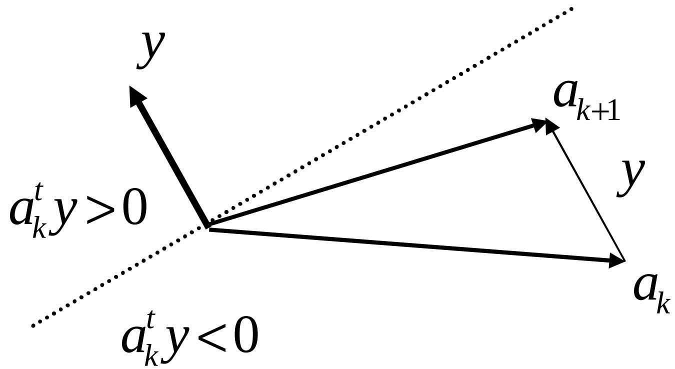
- Fixed Increment Rule for multi-category problems :

$$Let\ \vec{y}_k \in Y_i \quad and \quad \vec{a}_i(k)^t \vec{y}_k \le \vec{a}_j(k)^t \vec{y}_k \quad i \ne j$$

$$\vec{a}_i(k+1) = \vec{a}_i(k) + \vec{y}_k$$

$$\vec{a}_j(k+1) = \vec{a}_j(k) - \vec{y}_k$$

$$\vec{a}_l(k+1) = \vec{a}_l(k), \quad l \ne i\ and\ l \ne j$$



Angle between a_k and y should be < $\pi/2$

- For linearly separable problems, it can be shown that fixed increment rule is guaranteed to converge

Multi-category Generalizations (4/4)

- MSE for multi-category problems : min $|Ya\text{-}b|^2$

Find $\vec{a}_i$ such that $\vec{a}_i^t\vec{y}=1$ for all $\vec{y}\in Y_i$ and $\vec{a}_i^t\vec{y}=0$ for all $\vec{y}\notin Y_i$

$$\text{Let}\quad A_{\hat{d}xc}=[\vec{a}_1\cdots\vec{a}_c],\quad Y_{nx\hat{d}}=\begin{bmatrix}Y_1\\ \vdots\\ Y_c\end{bmatrix},\quad B_{nxc}=\begin{bmatrix}B_1\\ \vdots\\ B_c\end{bmatrix}\quad\begin{pmatrix}Y_i:\text{samples labelled }\omega_i\\ B_i:\text{all zeros except i}^{\text{th}}\text{column}\end{pmatrix}$$

$$A_{\hat{d}xc}=\begin{bmatrix}\begin{bmatrix}\\ \vec{a}_1\\ \\ \end{bmatrix}\begin{bmatrix}\\ \vec{a}_2\\ \\ \end{bmatrix}\cdots\begin{bmatrix}\\ \vec{a}_c\\ \\ \end{bmatrix}\end{bmatrix},\quad Y_{nx\hat{d}}=\begin{bmatrix}\begin{bmatrix}\vec{y}_{11}^t\\ \vdots\\ \vec{y}_{1k_1}^t\end{bmatrix}\\ \vdots\\ \begin{bmatrix}\vec{y}_{c1}^t\\ \vdots\\ \vec{y}_{ck_c}^t\end{bmatrix}\end{bmatrix},\quad B_{nxc}=\begin{bmatrix}\begin{bmatrix}1&\cdots&0\\ \vdots&\vdots&\vdots\\ 1&\cdots&0\end{bmatrix}\\ \vdots\\ \begin{bmatrix}0&\cdots&1\\ \vdots&\vdots&\vdots\\ 0&\cdots&1\end{bmatrix}\end{bmatrix}$$

$$\min_A\left\{(YA-B)^t(YA-B)\right\}\Rightarrow A=Y^*B$$

Generalized LDF (1/2)

- Linear Discriminant Function :

$$g(\vec{x}) = w_0 + \sum_{i=1}^{d} w_i x_i$$

- Quadratic Discriminant Function :

$$g(\vec{x}) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=1}^{d} w_{ij} x_i x_j$$

- Polynomial Discriminant Function :

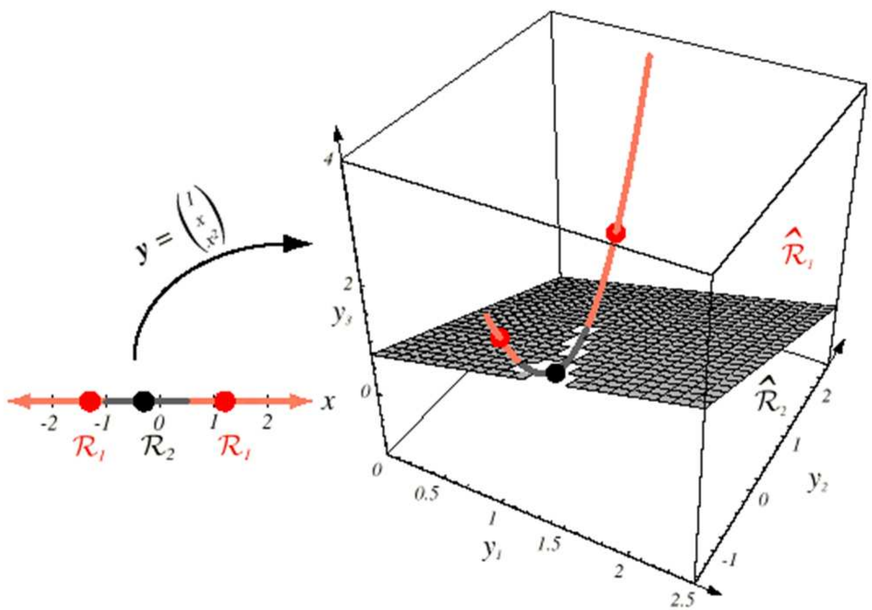
$$g(\vec{x}) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=1}^{d} w_{ij} x_i x_j + \sum_{i=1}^{d} \sum_{j=1}^{d} \sum_{k=1}^{d} w_{ijk} x_i x_j x_k + \cdots$$

- Generalized Linear Discriminant Function :

$$g(\vec{x}) = w_0 + \sum_{i=1}^{d} a_i \phi_i(\vec{x}) \qquad e.g. \ \Phi(\vec{x}) = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \\ x_2^2 \end{bmatrix}, \quad \vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Generalized LDF (2/2)

- Assume a simple quadratic DF : $g(x)=a_1+a_2x+a_3x^2$
- In order to make DF linear, let $\Phi(x)=[\ 1\ x\ x^2\ ]^T$
- Note that $a=[-1\ \ 1\ \ 2]^T$ ➔ $g(x)>0$ for $x<-1$ & $x>0.5$



Kernel Methods (1/2)

- Lets generalize the quadratic DF $g(x)=a_1+a_2x+a_3x^2$ which has the mapping $\Phi(x)=[\ 1\ x\ x^2\ ]$
- Let $\Phi(x)$ be any nonlinear feature space mapping
- A *kernel function* is defined by the relation

$$k(\vec{x},\vec{x}') \equiv \Phi(\vec{x})^T \Phi(\vec{x}')$$

- A typical kernel $k(\vec{x},\vec{z}) = (\vec{x}^T\vec{z})^2 = (x_1 z_1 + x_2 z_2)^2$

$$\begin{aligned} &= x_1^2 z_1^2 + 2x_1 z_1 x_2 z_2 + x_2^2 z_2^2 \\ &= (x_1^2,\ \sqrt{2}x_1 x_2,\ x_2^2)(z_1^2,\ \sqrt{2}z_1 z_2,\ z_2^2)^T \\ &= \Phi(\vec{x})^T \Phi(\vec{z}) \end{aligned}$$

- A kernel is called *homogenous* (e.g. radial basis functions), if it only depends on distance between features

$$k(\vec{x},\vec{x}') = k(\|\vec{x}-\vec{x}'\|)$$

Kernel Methods (2/2)

$$\sum_{i=1}^{n}\sum_{j=1}^{n} K(x_i, x_j)c_i c_j \geq 0$$

- The necessary & sufficient condition for a function to be a valid kernel is positive semi-definiteness of matrix K

$$K \equiv \Phi\Phi^T \quad \text{where} \quad \Phi = [\cdots \Phi(x_i)\cdots], \;\; for \;\; any \;\; x_i$$

- Kernel functions, $k(x,x')$, <u>avoid explicit utilization of $\Phi(x)$</u> vectors, as nonlinear feature space mappings with high-Ds

- Some well-known kernels

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

$$k(\vec{x},\vec{x}') = e^{-\|\vec{x}-\vec{x}'\|^2/2\sigma^2} \quad : \text{Gaussian} \quad \text{kernel}$$

$$k(\vec{x},\vec{x}') = \tanh(a\vec{x}^T\vec{x}' + b) : \text{Sigmoid} \quad \text{kernel}$$

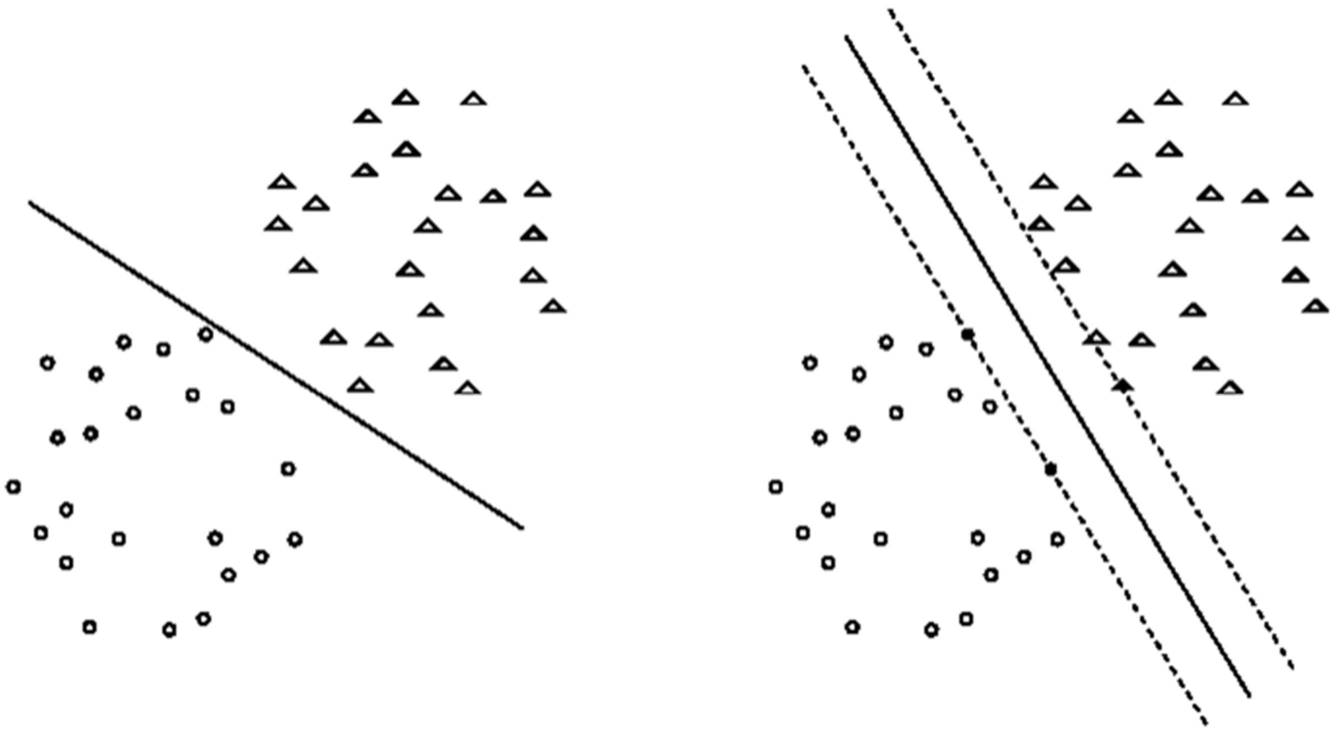
Support Vector Machine (SVM)

- SVM performs classification between two classes by finding a decision surface that is based on the <u>most "informative" points of the training set</u>

- SVM differs from classical classifiers in the way that it handles the <u>risk concept</u>
  - Empirical risk : minimize error on training data
  - Structural risk : minimize probability of misclassifying future test data

- SVM tries to <u>maximize the margin</u> between samples for different classes

SVM : Decision Boundary



(a) (b)

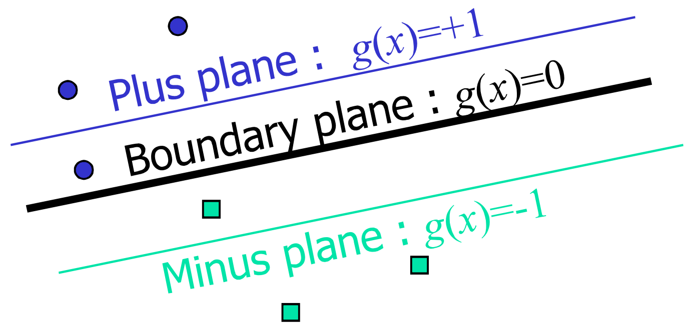
Decision boundary obtained by (a) an ordinary classifier and (b) SVM

SVM : Formulation (1/7)

- Assume the following is given
  - a training data set $\{x_1,\ldots,x_n\}$, consisting of vectors
  - their corresponding labels $\{y_1,\ldots,y_n\}$, taking values $+1$ or -1.
- LDF is defined $g(\vec{x}_i) = \vec{w}^t\vec{x}_i + w_0 \quad i = 1,\ldots,n$

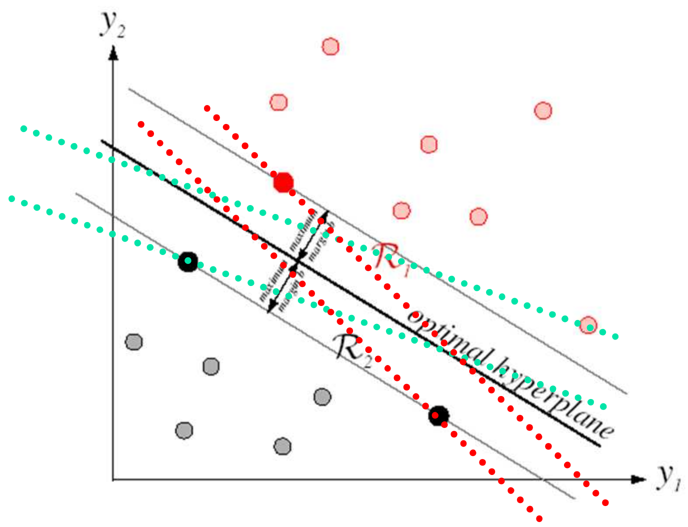
$$\text{Decide } \begin{array}{l} y_i = +1 \ \ if \ \ g(\vec{x}_i) \geq +1 \\ y_i = -1 \ \ if \ \ g(\vec{x}_i) \leq -1 \end{array} \Rightarrow y_i\left(\vec{w}^t\vec{x}_i + w_0\right) > +1 \qquad i = 1,\ldots,n$$

either class otherwise



SVM : Formulation (2/7)

- *Optimal Seperating Hyperplane* (OSH) separates feature space, while maximizing the distance from the nearest point :
- *Support Vectors* (SV) are the training patterns nearest to OSH, defining OSH

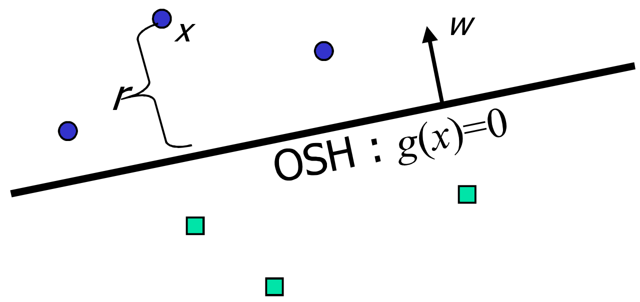


- SVs are the most difficult samples to classify
- SVs are the most informative for classification

SVM : Formulation (3/7)

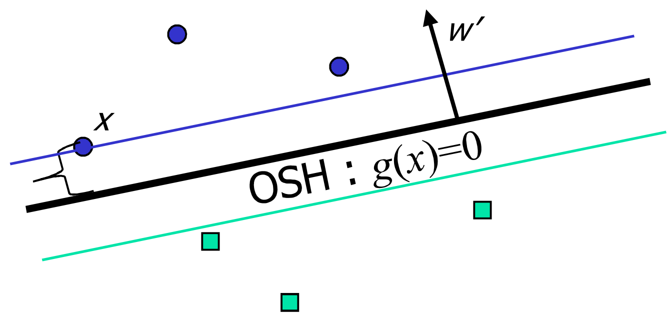
- Distance of point x_i from the decision boundary is equal to

$$r = \frac{\vec{w}^t \vec{x}_k + w_0}{\|\vec{w}\|}$$



- Note that $g(\vec{x}) = 0 \quad \Rightarrow \vec{w}^t \vec{x} + w_0 = k\vec{w}^t \vec{x} + kw_0 = \vec{w}'^t \vec{x} + w'_0 = 0$
- Lets normalize (w, w_0) so that distance for <u>nearest point</u> becomes $1/|w'|$

$$\frac{\vec{w}'^t \vec{x}_i + w'_0}{\|\vec{w}'\|} = \frac{1}{\|\vec{w}'\|}$$



SVM : Formulation (4/7)

- For a linearly separable problem, OSH can be obtained as a result of an optimization by maximizing distance of samples closest to OSH

$$\max \frac{1}{\|\vec{w}'\|^2} \quad \left(\text{or } \min \|\vec{w}'\|^2\right)$$

$$\textit{subject to} \quad y_i\left(\vec{w}'^t \vec{x}_i + w_0'\right) \geq +1 \qquad i = 1, \ldots, n$$

- This constrained optimization problem can be solved by using the method of *Lagrange multipliers*

$$L(\vec{w}', w_0', \alpha) = \frac{1}{2}\vec{w}'^t \vec{w}' - \sum_{i=1}^{n} \alpha_i \left(y_i\left(\vec{w}'^t \vec{x}_i + w_0'\right) - 1\right)$$

$$\alpha_i : \text{Lagrange multiplier}, \alpha_i \geq 0$$

SVM : Formulation (5/7)

$$L(\vec{w}', w_0', \alpha) = \frac{1}{2}\vec{w}'^t \vec{w}' - \sum_{i=1}^{n} \alpha_i \left(y_i \left(\vec{w}'^t \vec{x}_i + w_0' \right) - 1 \right)$$

- Solution satisfying (Kuhn-Tucker) conditions below provides the minimum & the Lagrange multipliers

$$\frac{\partial L(\vec{w}', w_0', \alpha)}{\partial \vec{w}'} = 0, \quad \frac{\partial L(\vec{w}', w_0', \alpha)}{\partial w_0'} = 0, \quad \alpha_i \geq 0$$

$$\alpha_i \left(y_i \left(\vec{w}'^t \vec{x}_i + w_0' \right) - 1 \right) = 0 \quad i = 1, \ldots, n \Rightarrow \alpha_i = 0 \; or \; \left(y_i \left(\vec{w}'^t \vec{x}_i + w_0' \right) - 1 \right) = 0$$

- Derivatives wrt w' and w_0' yields

$$\Rightarrow \vec{w}' = \sum_{i=1}^{n} \overline{\alpha}_i y_i \vec{x}_i \text{ where } \overline{\alpha}_i \text{ nonzero for only SV's} \qquad \sum_{i=1}^{n} \alpha_i y_i = 0$$

- The solution is obtained thru convex programming in (Wolfe) dual representation

SVM : Formulation (6/7)

- If the problem is <u>non-separable</u>:

$$\min\left\{ \left\| \vec{w}' \right\|^2 + C\sum_i \xi_i \right\}, \quad \xi_i \geq 0 \quad C : \text{trade-off parameter}$$

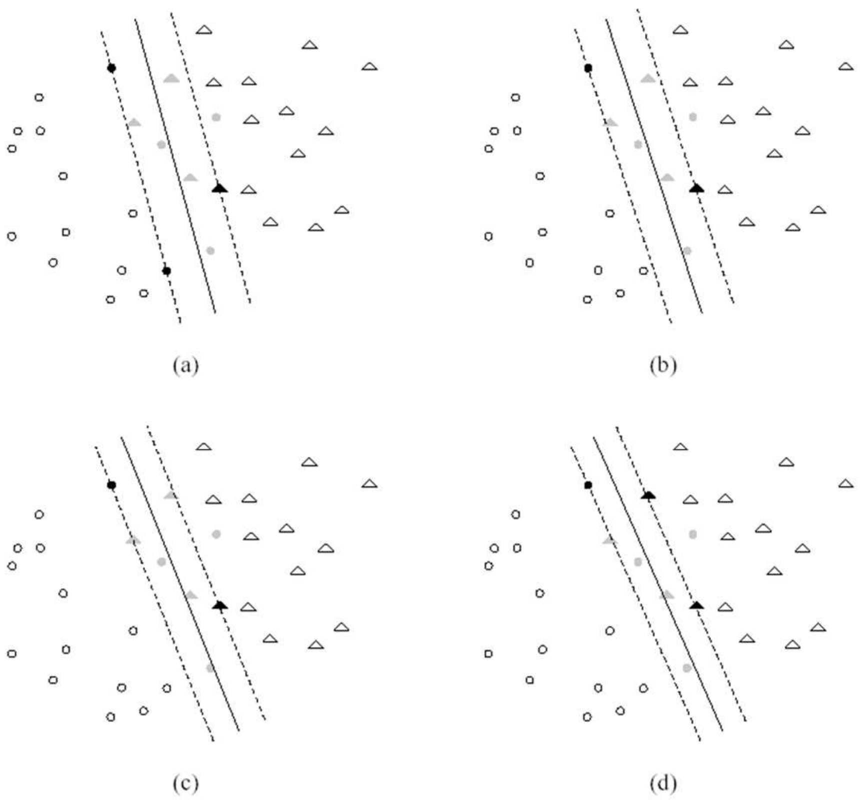
$$\textit{subject to} \quad y_i\left(\vec{w}'^t \vec{x}_i + w_0'\right) \geq +1 - \xi_i \qquad i = 1, \ldots, n$$

- ζ_i is used to compensate for misclassified samples
- C gives a compromise between distance of the nearest point and data
- The non-separable problem can be similarly solved

$$L(\vec{w}', w_0', \alpha, \beta, \xi) = \frac{1}{2}\vec{w}' \cdot \vec{w}' + C\sum_i \xi_i - \sum_{i=1}^{n} \beta_i \xi_i - \sum_{i=1}^{n} \alpha_i\left(y_i\left(\vec{w}'^t \vec{x}_i + w_0'\right) - 1 + \xi_i\right)$$

SVM : Formulation (7/7)

$$\min\left\{ \left\| \vec{w}' \right\|^2 + C\sum_i \xi_i \right\} \quad \xi_i \geq 0 \qquad C : \text{trade-off parameter}$$



Optimal separating hyperplane for $C = 4.0$ (a), $C = 4.8$ (b), $C = 6.7$ (c), and $C = 7.5$ (d) respectively.

SVM : Nonlinear Kernels (1/2)

- Linear separability assumption can especially be useful after projecting feature vectors into higher dimensional feature spaces by *mapping functions, Φ*

$$\vec{x} \rightarrow \Phi(\vec{x}) \quad \Phi : \Re^n \rightarrow \Re^m$$

- Define a LDF, $f(\vec{x}) = \vec{w} \cdot \Phi(\vec{x})$

$$\text{where } \underbrace{\vec{w} = \sum_{i=1}^{n} \alpha_i \Phi(\vec{x}_i)}_{\text{obtained as a solution}} \text{ and } \alpha_i\text{'s non-zero for only SV's}$$

- Define a *kernel, k,* in terms of the mappings, *Φ*

$$k(\vec{x}_i, \vec{x}) = \Phi(\vec{x}) \cdot \Phi(\vec{x}_i) \Rightarrow f(\vec{x}) = \sum_{i=1}^{n} \alpha_i k(\vec{x}_i, \vec{x})$$

SVM : Nonlinear Kernels (2/2)

- Without having full information for Φ, *K* can still be utilized, as long as K is positive, symmetric and continuous (Mercer's theorem).

- Kernel, *k*, is usually chosen as one of the following

$$k(\vec{x}_i, \vec{x}) = (\vec{x}_i \cdot \vec{x} + 1)^d \quad \text{(polynomial type)}$$

$$k(\vec{x}_i, \vec{x}) = e^{-\frac{(\vec{x}_i - \vec{x}).(\vec{x}_i - \vec{x})}{2\sigma^2}} \quad \text{(radial - basis style)}$$

$$k(\vec{x}_i, \vec{x}) = \tanh(\kappa \, \vec{x}_i \cdot \vec{x} - \delta) \quad \text{(neural net type)}$$