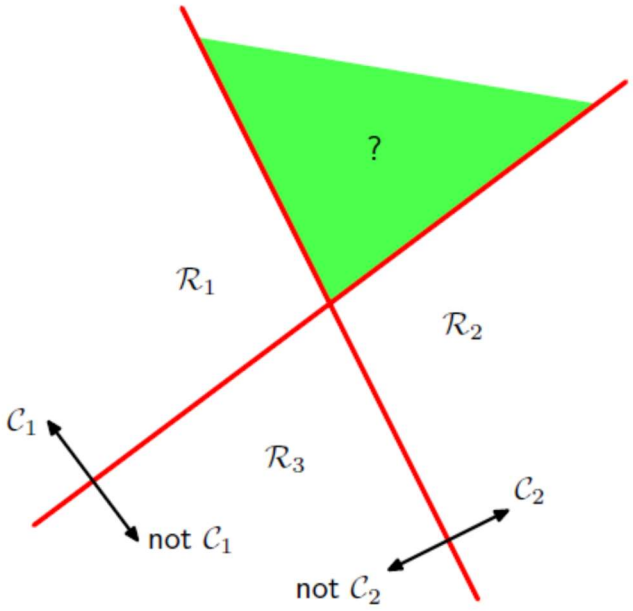
Multiclass Problems: K classes (K>2 classes)

Let's try to build a K-class discriminant by combining a number of 2-class discriminant functions. But this faces some difficulties.

Method I: Try (K-1) Classifiers

S_k vs $\overline{S_k}$ decisions

Solve a two-class problem by separating points into **S_k** or **Not S_k** Regions.
This is also known as a **one-vs.-the rest classifier**.



(Fig. 4.2 C. B.) This method leads to regions of unclassified regions.

Method II: Try K(K-1)/2 Binary Discriminant Functions

S_i vs S_j decisions for every possible pair of classes (one for every pair of classes).

This classifier is known as a **one-vs.-one classifier**.
Each point is classified according to a majority vote amongst the discriminant functions.

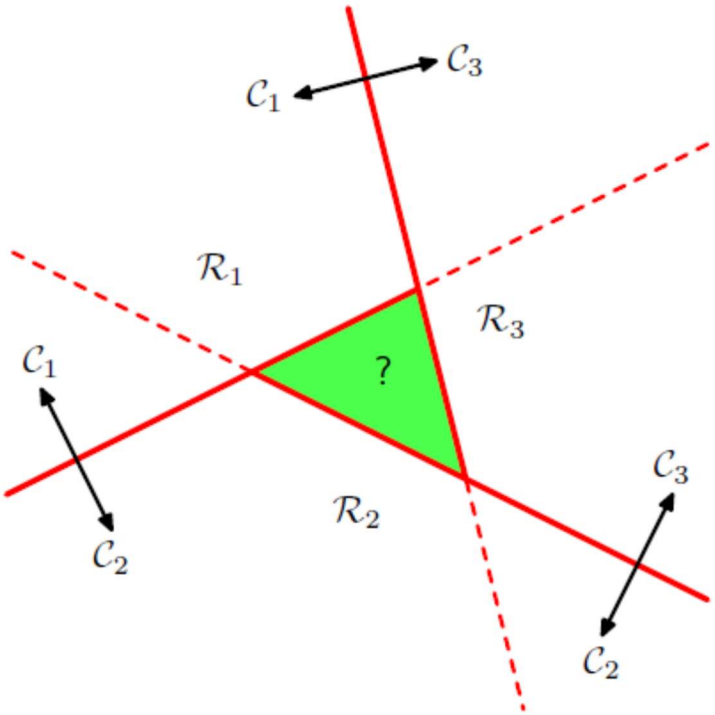


Fig. 4.1 (C.B.) This classifier also leaves some unclassified regions.

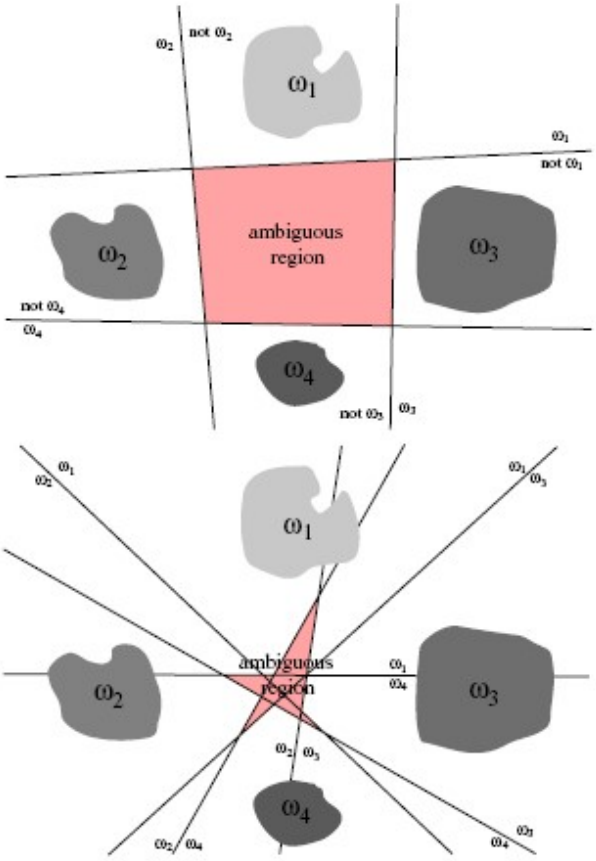


Figure 5.3: Linear decision boundaries for a four-class problem. The top figure shows ω_i/not ω_i dichotomies while the bottom figure shows ω_i/ω_j dichotomies. The pink regions have ambiguous category assigments.

Method III: Try K Discriminant Functions, $g_k(\underline{x})$

Consider a single K-class discriminant function (i.e., K linear functions).

Decision rule:

$$\underline{x} \in S_k \quad \text{iff} \quad g_k(\underline{x}) > g_j(\underline{x}), \quad \forall j \neq k$$

Decision hyperplanes:

$$g_k(\underline{x}) = g_j(\underline{x})$$
$$\underline{w}_k^T \underline{x}^{(a)} = \underline{w}_j^T \underline{x}^{(a)}$$
$$(\underline{w}_k^T - \underline{w}_j^T)\underline{x}^{(a)} = 0$$

Definition

If $g_k(\underline{y}_m^{(k)}) > g_j(\underline{y}_m^{(k)}) \forall m = 1,2,...,M_k; j \neq k$

Then the classes are linearly separable

This classifier is known as a **linear machine.**

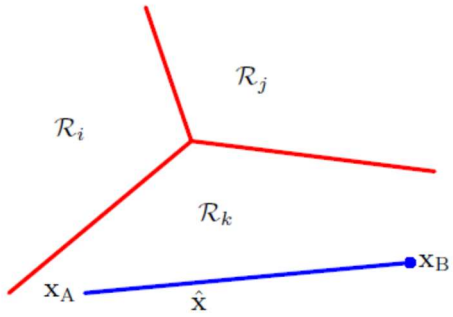


Figure 4.3 Illustration of the decision regions for a multiclass linear discriminant, with the decision boundaries shown in red. If two points $\mathbf{x}_A$ and $\mathbf{x}_B$ both lie inside the same decision region $\mathcal{R}_k$, then any point $\hat{\mathbf{x}}$ that lies on the line connecting these two points must also lie in $\mathcal{R}_k$, and hence the decision region must be singly connected and convex.

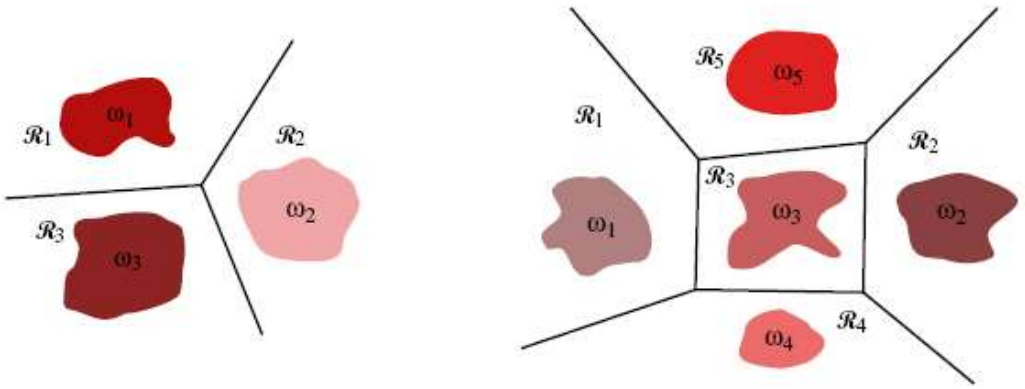


Figure 5.4: Decision boundaries produced by a linear machine for a three-class problem and a five-class problem.

Example: Minimum Distance to Class Means Classifier (Linear)

For S_k: $\left\langle \underline{y}_k \right\rangle \overset{\Delta}{=} \frac{1}{M_k} \sum_{m=1}^{M_k} \underline{y}_m^{(k)}$

Rule: Assign unknown $\underline{x}$ to same class at closest $<\underline{y}_k>$ using Euclidean metric.

Distance d:

$$d^2[\underline{x}, <\underline{y}_k>] = \sum_{n=1}^{N} [x_n - <y_{nk}>]^2$$

$$= [\underline{x} - <\underline{y}_k>]^T [\underline{x} - <\underline{y}_k>]$$

$$= \underline{x}^T \underline{x} - 2\underline{x}^T \left\langle \underline{y}_k \right\rangle + \left\langle \underline{y}_k \right\rangle^T \left\langle \underline{y}_k \right\rangle$$

Let $g_k(\underline{x}) = -\frac{1}{2} d^2[\underline{x}, <\underline{y}_k>] + \frac{1}{2} \underline{x}^T \underline{x}$

$$g_k(\underline{x}) = \underline{x}^T <\underline{y}_k> - \frac{1}{2} <\underline{y}_k>^T <\underline{y}_k> = \underline{w}^T \underline{x} + w_{N+1}$$

Decision surface are perpendicular bisecting hyperplanes of lines joining class means.

S_2-S_4 boundary is redundant.

Example: Minimum Distance to Class Member Classifier

$D(\underline{x}, S_k) = \min_{m=1,...,M_k} \{d(\underline{x}, \underline{y}_m^{(k)})\}$

Decision Rule:

$\underline{x} \in S_j$ if $D(\underline{x},S_j)=\min_k D(\underline{x}, S_k)$

Assign $\underline{x}$ to the same class as the nearest prototype.

All points on the decision surface are equidistant from the closest 2 prototypes from different classes.

Generalized Linear Discriminant Functions (DHS. 5.3)

Quadratic Discriminant Function

$$g_k(\underline{x}) = \sum_{n=1}^{N}[w_{nn}^{(k)} x_n^2 + w_n^{(k)} x_n] + \sum_{n=1}^{N-1} \sum_{j=n+1}^{N} w_{nj}^{(k)} x_n x_j + w_{N+1}^{(k)}$$

$$g_k(\underline{x}) = \underline{x}^T \underline{A}_k \underline{x} + \underline{x}^T \underline{b}_k + w_{N+1}^{(k)}$$

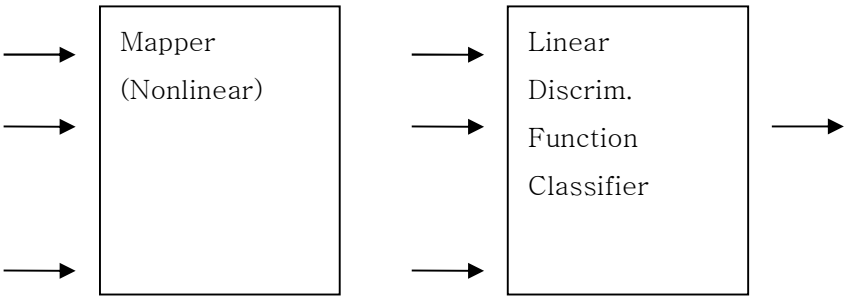
Weights in symmetric matrix $\underline{A}_k$
Vector $\underline{b}_k$
No. of terms – N square terms
N linear
N(N-1)/2 cross product terms
1 constant term

Let'generalize
Nonlinear, but can be cast in the form of a linear classifier

Let $\underline{f}=[x_1^2,...,x_N^2, x_1,...,x_N, x_1x_2, x_1x_3,...,x_{N-1}x_N]^T = [f_1,....,f_N, f_{N+1},..., f_{2N}, f_{2N+1},..., f_T]^T$

where T=N(N+3)/2

$$g_k(\underline{x}) = w_1^{(k)} f_1 + w_2^{(k)} f_2 + ... + w_T^{(k)} f_T + w_{T+1}^{(k)} = w^{(k)} \bullet f + w_{T+1}^{(k)}$$



It is also called as the Φ Machine (Nilsson)

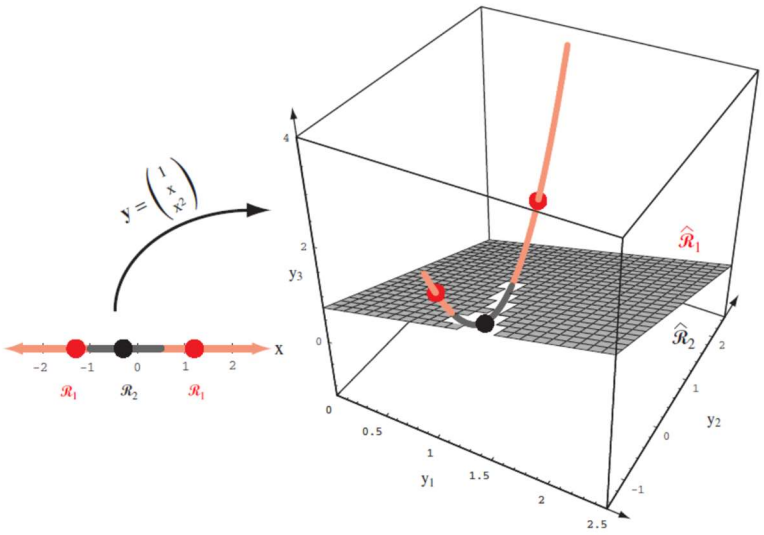


Figure 5.5: The mapping $\mathbf{y} = (1, x, x^2)^t$ takes a line and transforms it to a parabola in three dimensions. A plane splits the resulting $\mathbf{y}$ space into regions corresponding to two categories, and this in turn gives a non-simply connected decision region in the one-dimensional x space.

Higher Order Polynomial Discriminant Functions

Can be extended to any r-th order polynomial discriminant function.

Let $\Phi(x) = w_1 f_1(\underline{x}) + w_2 f_2(\underline{x}) + ... + w_M f_M(\underline{x}) + w_{M+1}$

where $f_i(\underline{x})$ are linearly independent, real, single-valued functions independent of the weights.

Example:

i) $f_i(\underline{x}) = x_i$ -> linear case

ii) $f_i(\underline{x}) = x_j^n x_l^m$ $j,l=1,...,N$ $n,m \in [0,1]$ -> quadratic

iii) $f_i(\underline{x}) = x_{l1}^{n1}, x_{l2}^{n2}, ..., x_{lr}^{nr}$ -> r-th order polymomial

$l_i = 1,..., N$

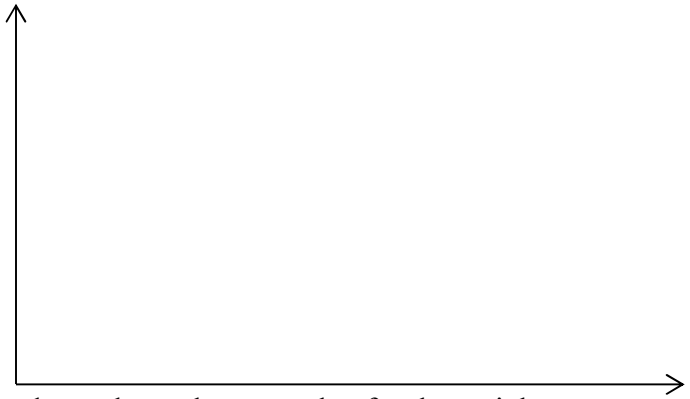
$n_i \in [0,1]$

We can use the Φ machine to map the r-th order polynomial nonlinear discriminant function classifier into a linear classifier that operates in a higher dimensional space.

Nonlinear Discriminant Functions

Piecewise Linear Discriminant Function

Any set of prototypes can be separated by a piecewise linear discriminant function.



However, we do not know how to solve for the weights yet.

It's coming soon! The techniques are known as linear training algorithms. ☺