Non-Bayesian Classifiers
Part II: Linear Discriminants
and Support Vector Machines

Selim Aksoy

Department of Computer Engineering
Bilkent University
saksoy@cs.bilkent.edu.tr

CS 551, Spring 2010

Linear Discriminant Functions

- A classifier that uses *discriminant functions* assigns a feature vector $\mathbf{x}$ to class w_i if

$$g_i(\mathbf{x}) > g_j(\mathbf{x}) \quad \forall j \neq i$$

  where $g_i(\mathbf{x}), i = 1, \ldots, c$, are the discriminant functions for c classes.
- A discriminant function that is a linear combination of the components of $\mathbf{x}$ is called a *linear discriminant function* and can be written as

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

  where $\mathbf{w}$ is the *weight vector* and w_0 is the *bias* (or threshold weight).

The Two-Category Case

- For the two-category case, the decision rule can be written as

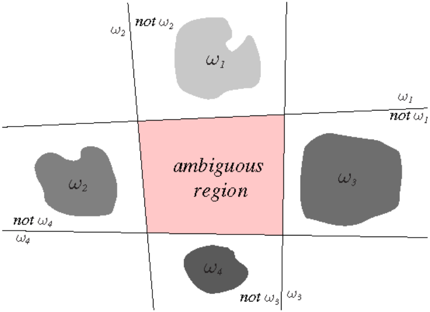
$$\text{Decide} \quad \begin{cases} w_1 & \text{if } g(\mathbf{x}) > 0 \\ w_2 & \text{otherwise} \end{cases}$$

- The equation $g(\mathbf{x}) = 0$ defines the decision boundary that separates points assigned to w_1 from points assigned to w_2.
- When $g(\mathbf{x})$ is linear, the decision surface is a hyperplane whose orientation is determined by the normal vector $\mathbf{w}$ and location is determined by the bias w_0.

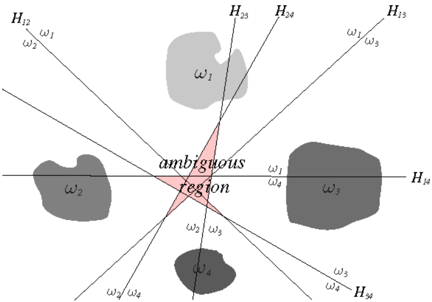
The Multicategory Case

- There is more than one way to devise multicategory classifiers with linear discriminant functions.
- For example, we can pose the problem as c two-class problems, where the i'th problem is solved by a linear discriminant that separates points assigned to w_i from those not assigned to w_i.
- Alternatively, we can use $c(c-1)/2$ linear discriminants, one for every pair of classes.
- Also, we can use c linear discriminants, one for each class, and assign $\mathbf{x}$ to w_i if $g_i(\mathbf{x}) > g_j(\mathbf{x})$ for all $j \neq i$.

The Multicategory Case



(a) Boundaries separate w_i from $\neg w_i$.



(b) Boundaries separate w_i from w_j.

Figure 1: Linear decision boundaries for a four-class problem devised as four two-class problems (left figure) and six pairwise problems (right figure). The pink regions have ambiguous category assignments.

The Multicategory Case

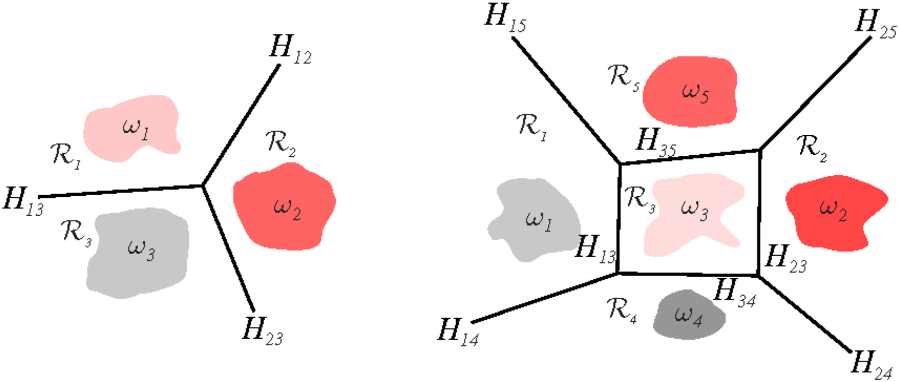


Figure 2: Linear decision boundaries produced by using one linear discriminant for each class. $\mathbf{w}_i - \mathbf{w}_j$ is the normal vector for the decision boundary that separates the decision region for class w_i from class w_j.

Generalized Linear Discriminant Functions

- The linear discriminant function $g(\mathbf{x})$ can be written as

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{d} \mathbf{w}_i \mathbf{x}_i$$

  where $\mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_d)^T$.

- We can obtain the *quadratic discriminant function* by adding second-order terms as

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{d} \mathbf{w}_i \mathbf{x}_i + \sum_{i=1}^{d} \sum_{j=1}^{d} \mathbf{w}_{ij} \mathbf{x}_i \mathbf{x}_j$$

  which result in more complicated decision boundaries (hyperquadrics).

Generalized Linear Discriminant Functions

- Adding higher-order terms gives the *generalized linear discriminant function*

$$g(\mathbf{x}) = \sum_{i=1}^{d'} \mathbf{a}_i \mathbf{y}_i(\mathbf{x}) = \mathbf{a}^T \mathbf{y}$$

  where $\mathbf{a}$ is a d'-dimensional weight vector and d' functions $\mathbf{y}_i(\mathbf{x})$ are arbitrary functions of $\mathbf{x}$.
- The physical interpretation is that the functions $\mathbf{y}_i(\mathbf{x})$ map point $\mathbf{x}$ in d-dimensional space to point $\mathbf{y}$ in d'-dimensional space.

Generalized Linear Discriminant Functions

- Then, the discriminant $g(\mathbf{x}) = \mathbf{a}^T\mathbf{y}$ separates points in the transformed space using a hyperplane passing through the origin.
- This mapping to a higher dimensional space brings problems and additional requirements for computation and data.
- However, certain assumptions can make the problem tractable.

Generalized Linear Discriminant Functions

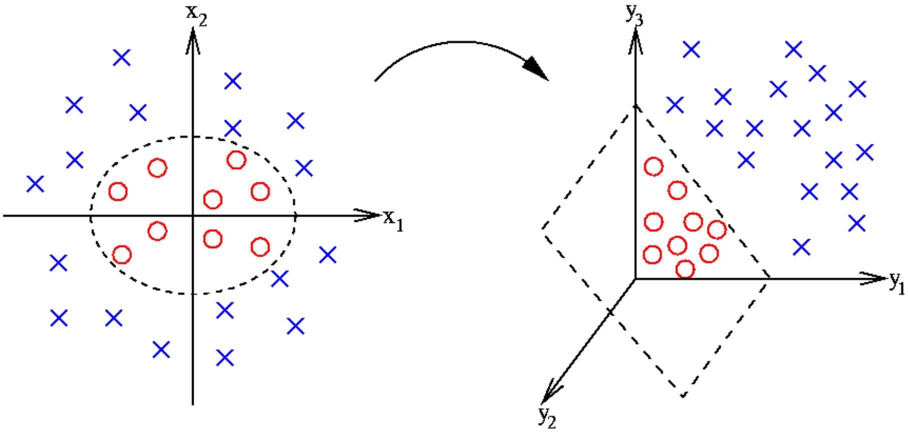


Figure 3: Mapping from $\mathbb{R}^2$ to $\mathbb{R}^3$ where points $(x_1, x_2)^T$ in the original space become $(y_1, y_2, y_3)^T = (x_1^2, \sqrt{2}x_1x_2, x_2^2)^T$ in the new space. The planar decision boundary in the new space corresponds to a non-linear decision boundary in the original space.

Generalized Linear Discriminant Functions

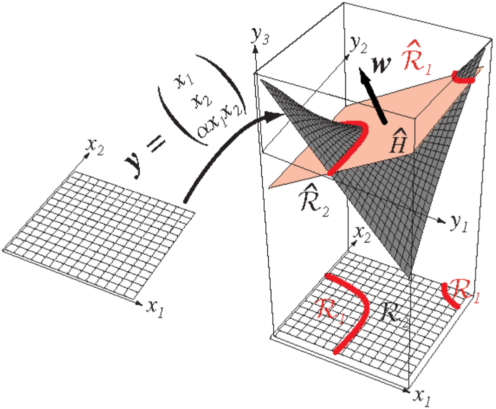


Figure 4: Mapping from $\mathbb{R}^2$ to $\mathbb{R}^3$ where points $(x_1, x_2)^T$ in the original space become $(y_1, y_2, y_3)^T = (x_1, x_2, \alpha x_1 x_2)^T$ in the new space. The decision regions $\hat{\mathcal{R}}_1$ and $\hat{\mathcal{R}}_2$ are separated by a plane in the new space where the corresponding regions $\mathcal{R}_1$ and $\mathcal{R}_2$ in the original space are separated by non-linear boundaries ($\mathcal{R}_1$ is also not connected).

Support Vector Machines

- We have seen that linear discriminant functions are optimal if the underlying distributions are Gaussians having equal covariance for each class.
- In the general case, the problem of finding linear discriminant functions can be formulated as a problem of optimizing a criterion function.
- Among all hyperplanes separating the data, there exists a unique one yielding the maximum margin of separation between the classes.

Support Vector Machines

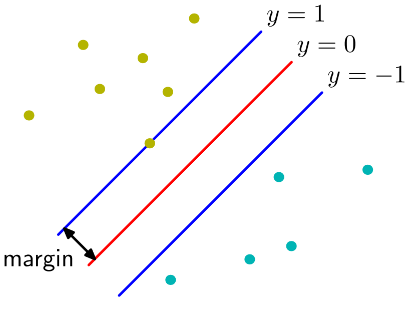


Figure 5: The margin is defined as the perpendicular distance between the decision boundary and the closest of the data points (left). Maximizing the margin leads to a particular choice of decision boundary (right). The location of this boundary is determined by a subset of the data points, known as the support vectors, which are indicated by the circles.

Support Vector Machines

- Given a set of training patterns and class labels as $(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_n}, y_n) \in \mathbb{R}^d \times \{\pm 1\}$, the goal is to find a classifier function $f : \mathbb{R}^d \rightarrow \{\pm 1\}$ such that $f(\mathbf{x}) = y$ will correctly classify new patterns.
- *Support vector machines* are based on the class of hyperplanes

$$(\mathbf{w} \cdot \mathbf{x}) + b = 0, \quad \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$$

  corresponding to decision functions

$$f(\mathbf{x}) = \operatorname{sign}((\mathbf{w} \cdot \mathbf{x}) + b).$$

Support Vector Machines

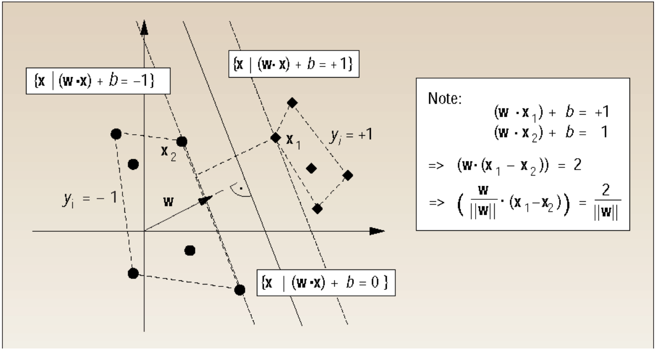


Figure 6: A binary classification problem of separating balls from diamonds. The optimal hyperplane is orthogonal to the shortest line connecting the convex hulls of the two classes (dotted), and intersects it half way between the two classes. There is a weight vector $\mathbf{w}$ and a threshold b such that the points closest to the hyperplane satisfy $|(\mathbf{w} \cdot \mathbf{x_i}) + b| = 1$ corresponding to $y_i((\mathbf{w} \cdot \mathbf{x_i}) + b) \geq 1$. The margin, measured perpendicularly to the hyperplane, equals $2/\|\mathbf{w}\|$.

Support Vector Machines

- To construct the optimal hyperplane, we can define the following optimization problem:

  minimize $\frac{1}{2}\|\mathbf{w}\|^2$

  subject to $y_i((\mathbf{w} \cdot \mathbf{x}_{\boldsymbol{i}}) + b) \geq 1, \quad i = 1, \ldots, n.$

- This constrained optimization problem is solved using Lagrange multipliers $\alpha_i \geq 0$ and the Lagrangian

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i (y_i((\mathbf{w} \cdot \mathbf{x}_{\boldsymbol{i}}) + b) - 1)$$

  where L has to be minimized w.r.t. the prime variables $\mathbf{w}$ and b, and maximized w.r.t. the dual variables α_i.

Support Vector Machines

- The solution can be obtained using quadratic programming techniques where the solution vector

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i \, y_i \, \mathbf{x}_{\boldsymbol{i}}$$

  is the summation of a subset of the training patterns, called the *support vectors*, whose α_i are non-zero.
- The support vectors lie on the margin and carry all relevant information about the classification problem (the remaining patterns are irrelevant).

Support Vector Machines

- The value of b can be computed as the solution of

$$\alpha_i(y_i((\mathbf{w} \cdot \mathbf{x}_{\boldsymbol{i}}) + b) - 1) = 0$$

  using any of the support vectors but it is numerically safer to take the average value of b resulting from all such equations.
- In many real-world problems there will be no linear boundary separating the classes and the problem of searching for an optimal separating hyperplane is meaningless.
- However, we can extend the above ideas to handle non-separable data by relaxing the constraints.

Support Vector Machines

- The new optimization problem becomes:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to } & (\mathbf{w}\cdot\mathbf{x}_{\boldsymbol{i}}) + b \geq +1 - \xi_i \quad \text{for } y_i = +1, \\ & (\mathbf{w}\cdot\mathbf{x}_{\boldsymbol{i}}) + b \leq -1 + \xi_i \quad \text{for } y_i = -1, \\ & \xi_i \geq 0 \qquad\qquad\qquad\qquad\quad i = 1,\ldots,n \end{aligned}$$

  where ξ_i, $i = 1,\ldots,n$, are called the slack variables and C is a regularization parameter.

- The term $C\sum_{i=1}^{n}\xi_i$ can be thought of as measuring some amount of misclassification where lowering the value of C corresponds to a smaller penalty for misclassification (see references).

Support Vector Machines

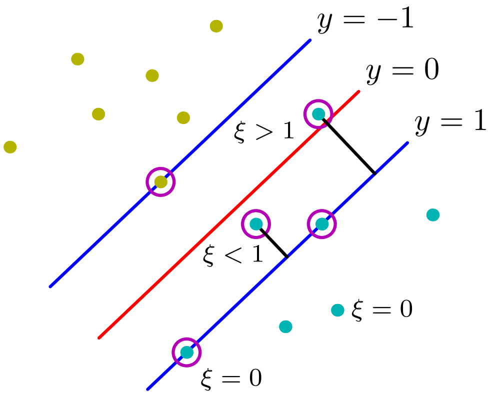


Figure 7: Illustration of the slack variables $\xi_i \geq 0$. Data points with circles around them are support vectors.

Support Vector Machines

- Both the quadratic programming problem and the final decision function

$$f(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{n} \alpha_i \, y_i \, (\mathbf{x} \cdot \mathbf{x}_{\boldsymbol{i}}) + b\right)$$

  depend only on the dot products between patterns.
- We can generalize this result to the non-linear case by mapping the original input space into some other space $\mathcal{F}$ using a non-linear map $\Phi : \mathbb{R}^d \rightarrow \mathcal{F}$ and perform the linear algorithm in the $\mathcal{F}$ space which only requires the dot products

$$k(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x})\Phi(\mathbf{y}).$$

Support Vector Machines

- Even though $\mathcal{F}$ may be high-dimensional, a simple *kernel* $k(\mathbf{x}, \mathbf{y})$ such as the following can be computed efficiently.

Table 1: Common kernel functions.

Polynomial	$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^p$
Sigmoidal	$k(\mathbf{x}, \mathbf{y}) = \tanh(\kappa(\mathbf{x} \cdot \mathbf{y}) + \theta)$
Radial basis function	$k(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2/(2\sigma^2))$

- Once a kernel function is chosen, we can substitute $\Phi(\mathbf{x}_i)$ for each training example $\mathbf{x}_i$, and perform the optimal hyperplane algorithm in $\mathcal{F}$.

Support Vector Machines

- This results in the non-linear decision function of the form

$$f(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{n} \alpha_i \, y_i \, k(\mathbf{x}, \mathbf{x}_{\boldsymbol{i}}) + b\right)$$

  where the parameters α_i are computed as the solution of the quadratic programming problem.
- In the original input space, the hyperplane corresponds to a non-linear decision function whose form is determined by the kernel.

Support Vector Machines

- SVMs are quite popular because of their intuitive formulation using computational learning theory and their high performances in practical applications.
- However, we must be careful about certain issues such as the following during implementation.
- *Choice of kernel functions:* We can use training data to find the best performing kernel.
- *Computational requirements of the quadratic program:* Several algorithms exist for speeding up the optimization problem (see references).

Support Vector Machines

- *Extension to multiple classes:* We can train a separate SVM for each class, compute the output value using each SVM, and select the class that assigns the unknown pattern the furthest into the positive region.
- *Converting the output of an SVM to a posterior probability for post-processing:* We can fit a sigmoid model to the posterior probability $P(y = 1|f(\mathbf{x}))$ as

$$P(y = 1|f(\mathbf{x})) = \frac{1}{1 + \exp(a\, f(\mathbf{x}) + b)}$$

where the parameters a and b are learned using maximum likelihood estimation from a training set.