BIM488 Introduction to Pattern Recognition

Classification Algorithms - Part II

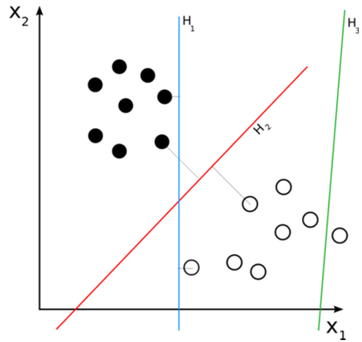
Outline

- Introduction
- Linear Discriminant Functions
- The Perceptron Algorithm
- Performance Assessment

Introduction

- Previously, our major concern was to design classifiers based on probability density functions.
- Now, we will focus on the design of linear classifiers, regardless of the underlying distributions describing the training data.
- The major advantage of linear classifiers is their simplicity and computational attractiveness.
- Here, our assumption is that all feature vectors from the available classes can be classified correctly using a linear classifier, and we will develop techniques for the computation of the corresponding linear functions.

Introduction



The solid and empty dots can be correctly classified by any number of linear classifiers. H1 (blue) classifies them correctly, as does H2 (red). H2 could be considered "better" in the sense that it is also furthest from both groups. H3 (green) fails to correctly classify the dots.

Introduction

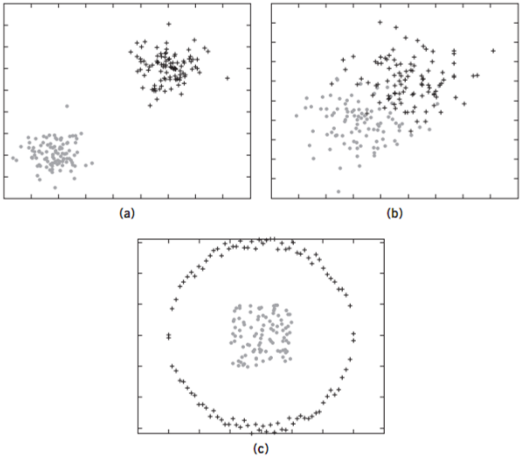


FIGURE 2.1

(a) Linearly separable 2-class classification problem; (b)–(c) 2-class classification problems that are not linearly separable.

Linear Discriminant Functions

- A classifier that uses discriminant functions assigns a feature vector x to class ω_i if

$$g_i(\mathbf{x}) > g_j(\mathbf{x}) \quad \text{for all } j \neq i$$

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

$$g(x) = \mathbf{w}^T\mathbf{x} + w_0 = w_1x_1 + w_1x_2 + \ldots + w_dx_d + w_0$$

  where **w** is the weight vector and w_0 is the bias (or threshold weight).

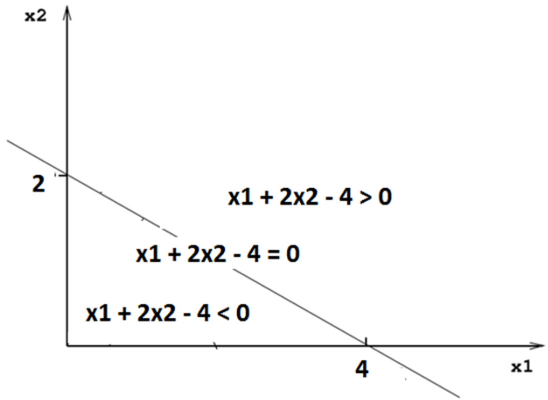
Linear Discriminant Functions

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

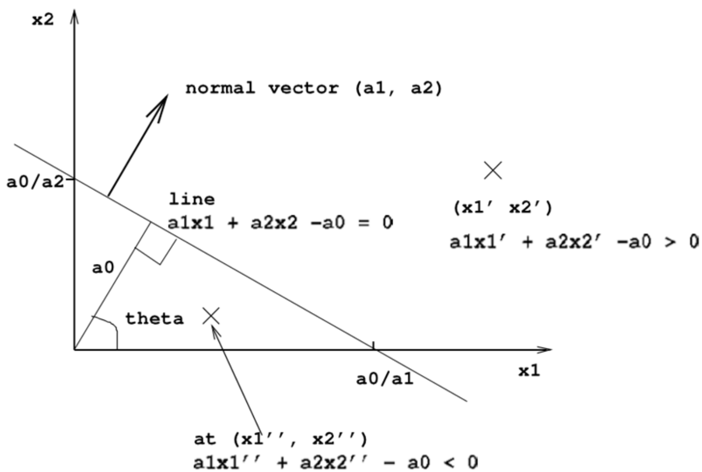
Decide : ω_1 if $g(x) > 0$
ω_2 otherwise

- The equation $g(x) = 0$ defines the decision boundary that separates points assigned to ω_1 from points assigned to ω_2.
- When $g(x)$ is linear, the decision surface is a hyperplane whose orientation is determined by the normal vector **w** and location is determined by the bias ω_0.

Linear Discriminant Functions



Linear Discriminant Functions

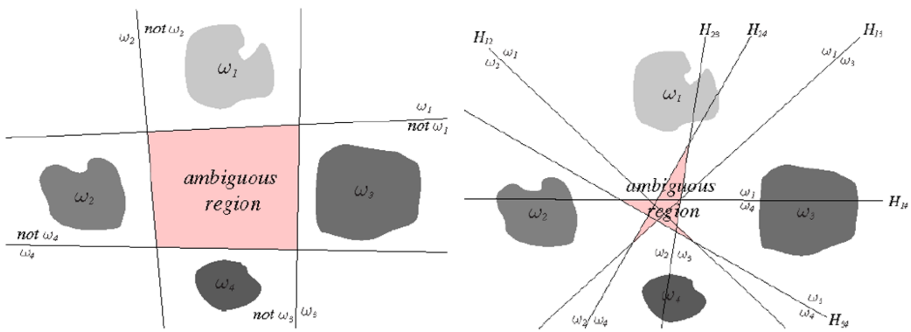


Linear Discriminant Functions

Multicategory Case:

- There is more than one way to devise multicategory classifiers with linear discriminant functions.
- One against all: 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 ω_i from those not assigned to ω_i.
- One against one: 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 x to ω_i if $g_i(x) > g_j(x)$ for all $j \neq i$.

Linear Discriminant Functions



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

Figure: Linear decision boundaries for a 4-class problem devised as (a) four 2-class problems (b) 6 pairwise problems.The pink regions have ambiguous category assignments.

Linear Discriminant Functions

- To avoid the problem of ambiguous regions:
  - Define c linear discriminant functions
  - Assign $\mathbf{x}$ to ω_i if $g_i(\mathbf{x}) > g_j(\mathbf{x})$ for all $j \neq i$.
- The resulting classifier is called a linear machine

Linear Discriminant Functions

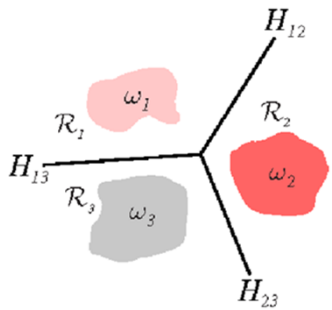


Figure: Linear decision boundaries produced by using one linear discriminant for each class.

Linear Discriminant Functions

- The boundary between two regions R_i and R_j is a portion of the hyperplane given by:

$$g_i(\mathbf{x}) = g_j(\mathbf{x}) \quad or$$

$$(\mathbf{w}_i - \mathbf{w}_j)^t \mathbf{x} + (w_{i0} - w_{j0}) = 0$$

- The decision regions for a linear machine are convex.

The Perceptron Algorithm

- The perceptron algorithm is appropriate for the 2-class problem and for classes that are linearly separable.
- The perceptron algorithm computes the values of the weights **w** of a linear classifier, which separates the two classes.
- The algorithm is iterative. It starts with an initial estimate in the extended (d +1)-dimensional space and converges to a solution in a finite number of iteration steps.
- The solution **w** correctly classifies all the training points assuming linearly separable classes.
- Note that the perceptron algorithm converges to one out of infinite possible solutions.
- Starting from different initial conditions, different hyperplanes result.

The Perceptron Algorithm

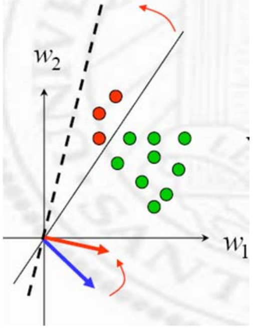
- The update at the *i* th iteration step has the simple form

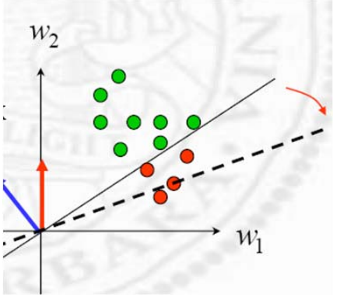
$$\boxed{\underline{w}(t+1) = \underline{w}(t) - \rho_t \sum_{\underline{x} \in Y} \delta_x \underline{x}}$$

- Y is the set of wrongly classified samples by the current estimate w(t),
- δ_x is −1 if x Є ω1, and +1 if x Є ω2,
- ρ_t is a user-defined parameter that controls the convergence speed and must obey certain requirements to guarantee convergence (for example, ρ_t can be chosen to be constant, $\rho_t = \rho$).
- The algorithm converges when Y becomes empty.

The Perceptron Algorithm

- Move the hyperplane so that training samples are on its positive side.





The Perceptron Algorithm

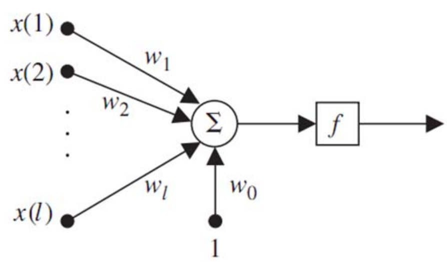
- Once the classifier has been computed, a point, x, is classified to either of the two classes depending on the outcome of the following operation:

$$f\,(w^T x) = f\,(w_1 x(1) + w_2 x(2) + \cdots + w_d x(d) + w_0)$$

- The function $f\,(\cdot)$ in its simplest form is the step or sign function ($f\,(z) = 1$ if $z > 0$; $f\,(z) = -1$ if $z < 0$).
- However, it may have other forms; for example, the output may be either 1 or 0 for $z > 0$ and $z < 0$, respectively.
- In general, it is known as the **activation function**.

The Perceptron Algorithm

- The basic network model, known as perceptron or neuron, that implements the classification operation is shown below:



The Perceptron Algorithm

Example

Figure shows the dashed line

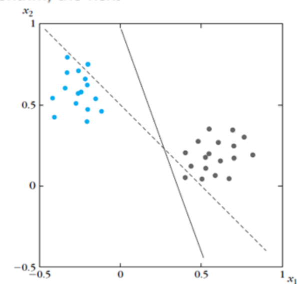
$$x_1 + x_2 - 0.5 = 0$$

corresponding to the weight vector $[1, 1, -0.5]^T$, which has been computed from the latest iteration step of the perceptron algorithm (3.9), with $\rho_t = \rho = 0.7$. The line classifies correctly all the vectors except $[0.4, 0.05]^T$ and $[-0.20, 0.75]^T$. According to the algorithm, the next weight vector will be

$$w(t+1) = \begin{bmatrix} 1 \\ 1 \\ -0.5 \end{bmatrix} - 0.7(-1) \begin{bmatrix} 0.4 \\ 0.05 \\ 1 \end{bmatrix} - 0.7(+1) \begin{bmatrix} -0.2 \\ 0.75 \\ 1 \end{bmatrix}$$

or

$$w(t+1) = \begin{bmatrix} 1.42 \\ 0.51 \\ -0.5 \end{bmatrix}$$



The resulting new (solid) line $1.42x_1 + 0.51x_2 - 0.5 = 0$ classifies all vectors correctly, and the algorithm is terminated.

The Perceptron Algorithm

Some important points related to perceptron:

- For a fixed learning parameter, the number of iterations (in general) increases as the classes move closer to each other (i.e., as the problem becomes more difficult).
- The algorithm fails to converge for a data set that is not linearly separable. **Then, what should we do?**
- Different initial estimates for w may lead to different final estimates for it (although all of them are optimal in the sense that they separate the training data of the two classes).

Performance Assessment

- We can use **accuracy** or **error rate** to assess performance of classifiers.
- *Accuracy* is the ratio of correct classifications.
- *Error rate* is the ratio of incorrect classifications.
- *Accuracy = 1 - Error rate*.
- Example:

  10 images belonging to the same class

  Number of correctly classified images = 8

  Number of incorrectly classified images = 2

  Accuracy = 8 / 10 = 0.8 = 80%

  Error Rate = 2 / 10 = 0.2 = 20%

Performance Assessment

- Performance is evaluated on a testing set.
- Therefore, entire dataset should be divided into
  - training set
  - testing set
- Classification model is obtained using the training set.
- Classification performance is assessed using the testing set.

Performance Assessment

- For objective evaluation, ***k-fold cross validation*** technique is used. Why ?
- Example: k = 3

Fold 1	Fold 2	Fold 3
Training	Training	Testing
Training	Testing	Training
Testing	Training	Training
Accuracy1	*Accuracy2*	*Accuracy3*

Overall accuracy = (Accuracy1 + Accuracy2 + Accuracy3) / 3

Performance Assessment

- We can also use a **confusion matrix** during assessment
- The example below shows predicted and true class labels for a 10-class recognition problem.

	class j predicted by a classifier										
true class i	'0'	'1'	'2'	'3'	'4'	'5'	'6'	'7'	'8'	'9'	'R'
'0'	97	0	0	0	0	0	1	0	0	1	1
'1'	0	98	0	0	1	0	0	1	0	0	0
'2'	0	0	96	1	0	1	0	1	0	0	1
'3'	0	0	2	95	0	1	0	0	1	0	1
'4'	0	0	0	0	98	0	0	0	0	2	0
'5'	0	0	0	1	0	97	0	0	0	0	2
'6'	1	0	0	0	0	1	98	0	0	0	0
'7'	0	0	1	0	0	0	0	98	0	0	1
'8'	0	0	0	1	0	0	1	0	96	1	1
'9'	1	0	0	0	3	1	0	0	0	95	0

Summary

- Introduction
- Linear Discriminant Functions
- The Perceptron Algorithm
- Performance Assessment

References

- S. Theodoridis, A. Pikrakis, K. Koutroumbas, D. Cavouras, *Introduction to Pattern Recognition: A MATLAB Approach,* Academic Press, 2010.
- S. Theodoridis and K. Koutroumbas, *Pattern Recognition* (4th Edition), Academic Press, 2009.
- R. O. Duda, P. E. Hart, D. G. Stork, Pattern Classification (2nd Edition), Wiley, 2001.