Lecture 16 Part 2 - Fisher's Linear Discriminant Function (FLDA or simply LDA)

(Parametric) Linear Models for Classification

So far we designed classifiers based on probability density or probability functions. In some cases, we saw that the resulting classifiers were equivalent to a set of linear discriminant functions.

We will now focus on the design of linear classifiers, irrespective of the underlying distributions describing the training data.

- The major advantage of linear classifiers is their simplicity and computational attractiveness.
- We will develop techniques for the computation of the corresponding linear functions. In the sequel we will focus on a more general problem, in which a linear classifier cannot classify correctly all feature vectors, yet we will seek ways to design an optimal linear classifier by adopting an appropriate optimality criterion.

Before, we designed generative classification algorithm that parametrize each class with a probabilistic model for the posterior probability of each class, i.e., $P(C_k|x) = \frac{P(x|C_k)P(C_k)}{P(x)}$.

Let's now consider a generalization of this model in which we transform the linear function of $\mathbf{w}$ using a nonlinear function $f(\bullet)$ so that:

$$y(\mathbf{x}) = f(\mathbf{w}^T\mathbf{x} + b)$$

In the machine learning literature $f(\bullet)$ is known as an **activation function**. The **decision surfaces** correspond to $y(\mathbf{x}) = \text{constant}$, so that $\mathbf{w}^T\mathbf{x} + b = \text{constant}$ and hence the decision surfaces are linear functions of $\mathbf{x}$, even if the function $f(\bullet)$ is nonlinear. For this reason, the class of models described by equation above are called **generalized linear models**.

Note, however, that in contrast to the models used for regression, they are no longer linear in the parameters due to the presence of the nonlinear function $f(\bullet)$. This will lead to more complex analytical and computational properties than for linear regression models. Nevertheless, these models are still relatively simple compared to the more general nonlinear models that will be studied later in the course.

The algorithm we will study will be equally applicable if we first make a fixed nonlinear transformation of the input variables using a vector of basis functions $\phi(x)$ as we did for

regression models. In what follows, we consider classification directly in the original input space $\mathbf{x}$ but this illustration can be generalizatble to a notation involving basis functions.

Discriminant Functions

A discriminant is a function that takes an input vector x and assigns it to one of K classes, denoted C_k.

Let's restrict our attention to **linear discriminants**, namely those for which the decision surfaces are hyperplanes. To simplify the discussion, we consider first the case of two classes and then investigate the extension to $K > 2$ classes.

Two Classes

The simplest representation of a linear discriminant function is obtained by taking a linear function of the input vector so that

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

where $\mathbf{w}$ is called a **weight vector**, and b is a **bias** (not to be confused with bias in the statistical sense). The negative of the bias is sometimes called a threshold. An input vector $\mathbf{x}$ is assigned to class C_1 if $y(\mathbf{x}) \geq 0$ and to class C_2 otherwise.

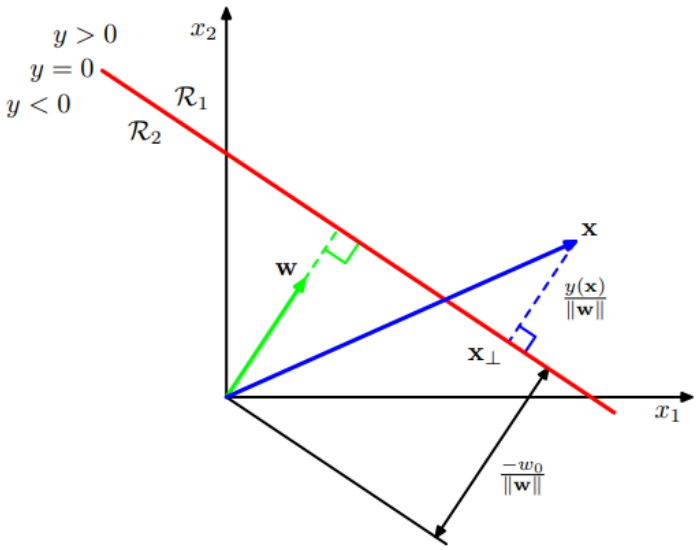
Looks pretty familiar, right? If you are on one side of the line, then you are in class 1. If you are on the other side of the line, then you are in class 2. So, the decision boundary is $y(\mathbf{x}) = 0$.

- The vector $\mathbf{w}$ is orthogonal to every vector lying within the decision surface, and so $\mathbf{w}$ determines the orientation of the decision surface.
- Similarly, if $\mathbf{x}$ is a point on the decision surface, then $y(\mathbf{x}) = 0$, and so the normal distance from the origin to the decision surface is given by: $\frac{y(\vec{\mathbf{x}})}{\|\vec{\mathbf{w}}\|}$

In [1]:
```
from IPython.display import Image
Image('figures/Figure4.1.png', width=800)
# Source: Bishop textbook
```

Out[1]:

Figure 4.1 Illustration of the geometry of a linear discriminant function in two dimensions. The decision surface, shown in red, is perpendicular to $\mathbf{w}$, and its displacement from the origin is controlled by the bias parameter w_0. Also, the signed orthogonal distance of a general point $\mathbf{x}$ from the decision surface is given by $y(\mathbf{x})/\|\mathbf{w}\|$.



Multiple Classes

Considering the extension of linear discriminants to $K > 2$ classes. We might be tempted be to build a K-class discriminant by combining a number of two-class discriminant functions. However, this leads to some serious difficulties.

- Consider the use of $K - 1$ classifiers each of which solves a two-class problem of separating points in a particular class C_k from points not in that class. This is known as a **one-versus-all** classifier.
- An alternative is to introduce $K(K - 1)/2$ binary discriminant functions, one for every possible pair of classes. This is known as a **one-versus-one** classifier. Each point is then classified according to a majority vote amongst the discriminant functions. However, this too runs into the problem of ambiguous regions.

In [2]:
```
Image('figures/Figure4.2.png',width=700)
# Source: Bishop textbook
```

Out[2]:

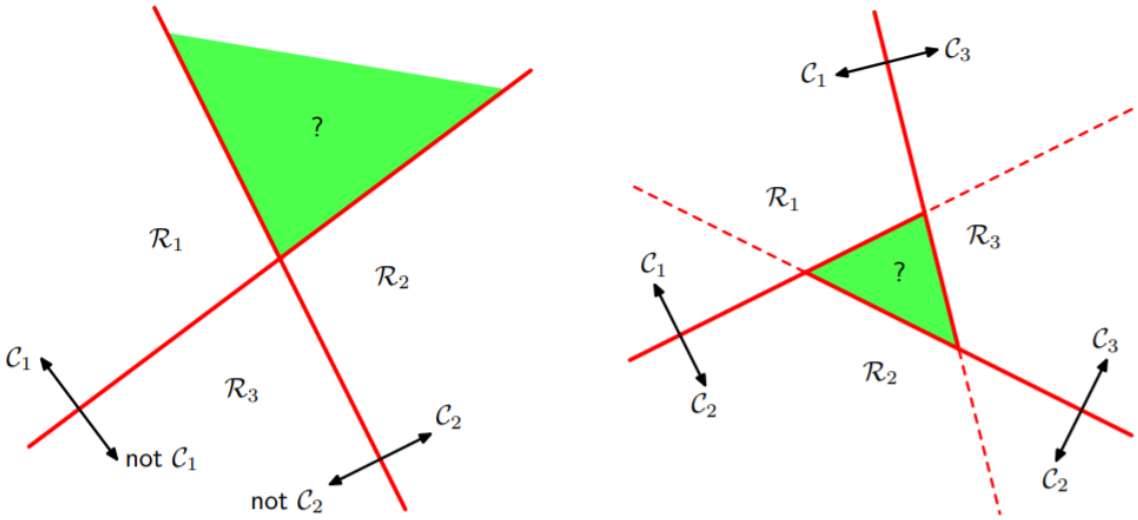


Figure 4.2 Attempting to construct a K class discriminant from a set of two class discriminants leads to ambiguous regions, shown in green. On the left is an example involving the use of two discriminants designed to distinguish points in class $\mathcal{C}_k$ from points not in class $\mathcal{C}_k$. On the right is an example involving three discriminant functions each of which is used to separate a pair of classes $\mathcal{C}_k$ and $\mathcal{C}_j$.

We can avoid these difficulties by considering a **single K-class discriminant** comprising K linear functions of the form

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + b_k$$

and then assigning a point $\mathbf{x}$ to class C_k if $y_k(\mathbf{x}) > y_j(\mathbf{x})$ for all $j \neq k$. The decision boundary between class C_k and class C_j is therefore given by $y_k(\mathbf{x}) = y_j(\mathbf{x})$ and hence corresponds to a $(D-1)$-dimensional hyperplane defined by

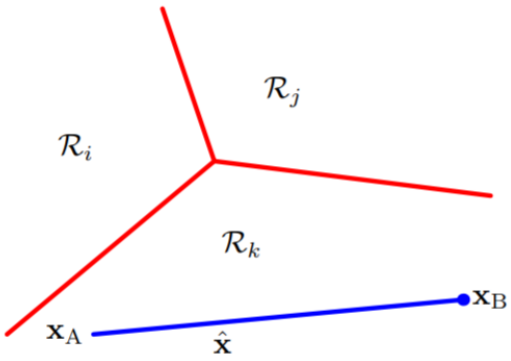
$$(\mathbf{w}_k - \mathbf{w}_j)^T \mathbf{x} + (b_k - bj) = 0$$

This has the same form as the decision boundary for the two-class case.

In [3]:
```
Image('figures/Figure4.3.png',width=800)
# Source: Bishop textbook
```

Out[3]:

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.



The decision regions of such a discriminant are always **singly connected** and **convex**. To see this, consider two points x_A and x_B both of which lie inside decision region R_k. Any point $\hat{\mathbf{x}}$ that lies on the line connecting x_A and x_B can be expressed in the form

$$\hat{\mathbf{x}} = \alpha x_A + (1-\alpha) x_B$$

where $0 \leq \alpha \leq 1$. From the linearity of the discriminant functions, it follows that

$$y_k(\hat{\mathbf{x}}) = \alpha y_k(x_A) + (1 - \alpha) y_k(x_B)$$

Because both x_A and x_B lie inside R_k, it follows that $y_k(x_A) > y_j(x_A)$, and $y_k(x_B) > y_j(x_B)$, for all $j \neq k$, and hence $y_k(\hat{\mathbf{x}}) > y_j(\hat{\mathbf{x}})$, and so $\hat{\mathbf{x}}$ also lies inside R_k. Thus R_k is singly connected and convex.

We will learn 3 methods to optimize the parameters of a linear discriminant function (classifier):

1. Least Squares for Classification
2. Fisher's Linear Discriminant
3. The Perceptron Algorithm
4. Support Vector Machines

Discriminant Functions

A discriminant is a function that takes an input vector x and assigns it to one of K classes, denoted C_k.

Let's restrict our attention to **linear discriminants**, namely those for which the decision surfaces are hyperplanes. To simplify the discussion, we consider first the case of two classes and then investigate the extension to $K > 2$ classes.

Two Classes

The simplest representation of a linear discriminant function is obtained by taking a linear function of the input vector so that

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

where $\mathbf{w}$ is called a **weight vector**, and b is a **bias** (not to be confused with bias in the statistical sense). The negative of the bias is sometimes called a threshold. An input vector $\mathbf{x}$ is assigned to class C_1 if $y(\mathbf{x}) \geq 0$ and to class C_2 otherwise.

Looks pretty familiar, right? If you are on one side of the line, then you are in class 1. If you are on the other side of the line, then you are in class 2. So, the decision boundary is $y(\mathbf{x}) = 0$.

- The vector $\mathbf{w}$ is orthogonal to every vector lying within the decision surface, and so $\mathbf{w}$ determines the orientation of the decision surface.
- Similarly, if $\mathbf{x}$ is a point on the decision surface, then $y(\mathbf{x}) = 0$, and so the normal distance from the origin to the decision surface is given by: $\frac{y(\overrightarrow{\mathbf{x}})}{\|\overrightarrow{\mathbf{w}}\|}$

In [4]:
```
Image('figures/Figure4.1.png', width=800)
# Source: Bishop textbook
```

Out[4]:

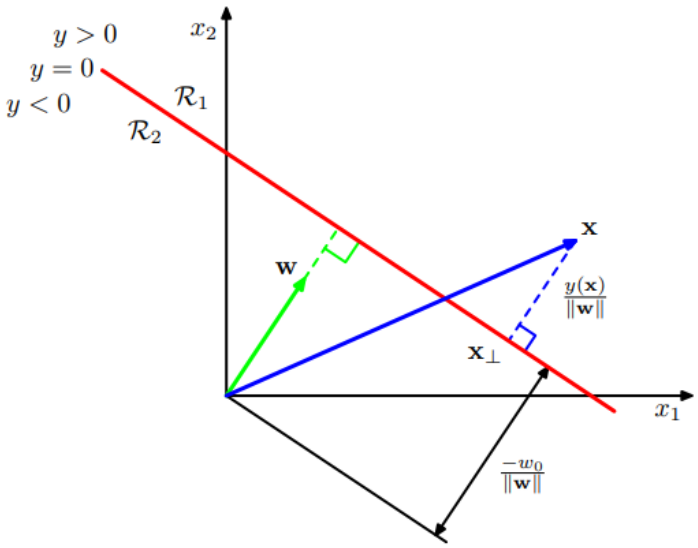


Figure 4.1 Illustration of the geometry of a linear discriminant function in two dimensions. The decision surface, shown in red, is perpendicular to $\mathbf{w}$, and its displacement from the origin is controlled by the bias parameter w_0. Also, the signed orthogonal distance of a general point $\mathbf{x}$ from the decision surface is given by $y(\mathbf{x})/\|\mathbf{w}\|$.

Fisher's Linear Discriminant Analysis (or LDA)

A very popular type of a linear discriminant is the **Fisher's Linear Discriminant**.

- Given two classes, we can compute the mean of each class:

$$\vec{\mathbf{m}}_1 = \frac{1}{N_1} \sum_{n \in C_1} \vec{\mathbf{x}}_\mathbf{n}$$
$$\vec{\mathbf{m}}_\mathbf{2} = \frac{1}{N_2} \sum_{n \in C_2} \vec{\mathbf{x}}_\mathbf{n}$$

We can maximize the separation of the means:

$$m_2 - m_1 = \vec{\mathbf{w}}^T (\vec{\mathbf{m}}_2 - \vec{\mathbf{m}}_1)$$

- $\vec{\mathbf{w}}^T \vec{\mathbf{x}}$ takes a D dimensional data point and projects it down to 1-D with a weight sum of the original features. We want to find a weighting that maximizes the separation of the class means.

- Not only do we want well separated means for each class, but we also want each class to be *compact* to minimize overlap between the classes.

- Consider the *within class variance:*

$$
\begin{aligned}
s_k^2 = \sum_{n \in C_k} (y_n - m_k)^2 &= \sum_{n \in C_k} (\overrightarrow{\mathbf{w}}^T \overrightarrow{\mathbf{x}}_n - m_k)^2 \\
&= \overrightarrow{\mathbf{w}}^T \sum_{n \in C_k} (\overrightarrow{\mathbf{x}}_n - \overrightarrow{\mathbf{m}_\mathbf{k}})(\overrightarrow{\mathbf{x}}_n - \overrightarrow{\mathbf{m}_\mathbf{k}})^T \overrightarrow{\mathbf{w}}
\end{aligned}
$$

to be continued...