Advanced Machine Learning **Fall 2020**

Lecture 6: Linear methods for classification

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

Disclaimer: *These notes are adapted from ETH's Advanced Machine Learning Course and "Pattern Recognition and Machine Learning, Chapter 4, Springer".*

The goal in classification is to take an input vector $\boldsymbol{x}$ and to assign it to one of K discrete classes $\mathcal{C}_k$ where $k = 1, ..., K$.
In the most common scenario, the classes are taken to be disjoint, so that each input is assigned to one and only one class. The input space is thereby divided into *decision regions* whose boundaries are called *decision boundaries* or *decision surfaces*.
In this lecture we consider linear models for classification, by which we mean that the decision surfaces are linear functions of the input vector $\boldsymbol{x}$ and hence are defined by $(D-1)$-dimensional hyperplanes within the D-dimensional input space. Data sets whose classes can be separated exactly by linear decision surfaces are said to be *linearly separable*.

For probabilistic models, in the case of two-class problems,the target variable can be represented as $t \in \{0, 1\}$ such that $t = 1$ represents class $\mathcal{C}_1$ and $t = 0$ represents class $\mathcal{C}_2$.
We can interpret the value of t as the probability that the class is $\mathcal{C}_1$, with the values of probability taking only the extreme values of 0 and 1.
For $K > 2$ classes, it is convenient to use a 1-of-K coding scheme in which $\boldsymbol{t}$ is a vector of length K such that if the class is $\mathcal{C}_j$, then all elements t_k of $\boldsymbol{t}$ are zero except element t_j, which takes the value 1.

Approaches to the classification problem:

1. *discriminant functions* that directly assigns each vector $\boldsymbol{x}$ to a specific class.
2. Modelling of a conditional probability distribution $p(\mathcal{C}_k|\boldsymbol{x})$ in an inference stage, and then subsequently using this distribution to make optimal decisions. There are two different approaches to determining the conditional probabilities $p(\mathcal{C}_k|\boldsymbol{x})$. One technique is to model them directly, for example by representing them as parametric models and then optimizing the parameters using a training set.
   Alternatively, we can adopt a *generative* approach in which we model the class-conditional densities given by $p(\boldsymbol{x}|\mathcal{C}_k)$, together with the prior probabilities $p(\mathcal{C}_k)$ for the classes, and then we compute the required posterior probabilities using Bayes' theorem

$$p(\mathcal{C}_k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\boldsymbol{x})}$$

For classification problems we wish to predict discrete class labels, or more generally posterior probabilities that lie in the range $(0, 1)$. To achieve this, we transform the linear function of $\boldsymbol{w}$ using a non linear *activation function* $f(\cdot)$ so that

$$y(\boldsymbol{x}) = f(\boldsymbol{w}^\intercal \boldsymbol{x} + w_0) \tag{6.1}$$

Its inverse is called *link function*. The decision surfaces correspond to $y(\boldsymbol{x}) = \text{constant}$, so that $\boldsymbol{w}^\intercal \boldsymbol{x} + w_0 = \text{constant}$ and hence the decision surfaces are linear functions of $\boldsymbol{x}$, even if the function $f(\cdot)$ is non linear. For this reason, the class of models described by Eq.6.1 are called *generalized linear models*.

6.1 Discriminant functions

A discriminant is a function that takes an input vector $\boldsymbol{x}$ and assigns it to one of K classes, denoted $\mathcal{C}_k$.

6.1.1 Two classes

Linear discriminants' decision surfaces are hyperplanes and their simplest representation can be obtained by taking a linear function of the input vector so that

$$y(\boldsymbol{x}) = \boldsymbol{w}^\intercal \boldsymbol{x} + w_0$$

where $\boldsymbol{w}$ is called a *weight* vector, and w_0 is a bias (not in the statistical sense). The negative of the bias is sometimes called a *threshold*. An input vector $\boldsymbol{x}$ is assigned to class $\mathcal{C}_1$ if $y(\boldsymbol{x}) \geq 0$ and to class $\mathcal{C}_2$ otherwise. The corresponding decision boundary is defined by the relation $y(\boldsymbol{x}) = 0$, which corresponds to a $(D-1)$-dimensional hyperplane within the D-dimensional input space. Consider two points $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$ both of which lie on the decision surface.
Because $y(\boldsymbol{x}_A) = y(\boldsymbol{x}_B) = 0$, we have $\boldsymbol{w}^\intercal(\boldsymbol{x}_A - \boldsymbol{x}_B) = 0$ and hence the vector $\boldsymbol{w}$ is orthogonal to every vector lying within the decision surface, and so $\boldsymbol{w}$ determines the orientation of the decision surface. Similarly, if $\boldsymbol{x}$ is a point on the decision surface, then $y(\boldsymbol{x}) = 0$, and so the normal distance from the origin to the decision surface is given by

$$\frac{\boldsymbol{w}^\intercal \boldsymbol{x}}{\|\boldsymbol{w}\|} = -\frac{w_o}{\|\boldsymbol{w}\|}$$

We therefore see that the bias parameter w_0 determines the location of the decision surface. Furthermore, we note that the value $y(\boldsymbol{x})$ gives a signed measure of the perpendicular distance r of the point $\boldsymbol{x}$ from the decision surface. To see this consider an arbitrary point $\boldsymbol{x}$ and let $\boldsymbol{x}_\perp$ be its orthogonal projection onto the decision surface so that

$$\boldsymbol{x} = \boldsymbol{x}_\perp + r\frac{\boldsymbol{w}^\intercal}{\|\boldsymbol{w}\|}$$

Multiplying both sides of this result by $\boldsymbol{w}^\intercal$ and adding w_0, and making use of $y(\boldsymbol{x}) = \boldsymbol{w}^\intercal \boldsymbol{x} + w_0$ and $y(\boldsymbol{x}_\perp) = \boldsymbol{w}^\intercal \boldsymbol{x}_\perp + w_0$, we have

$$r = \frac{y(\boldsymbol{x})}{\|\boldsymbol{w}\|}$$

This result is illustrated in Figure 6.1.

As with the linear regression models, it is sometimes convenient to use a more compact notation in which we introduce an additionally dummy 'input' value $x_0 = 1$ and then define $\tilde{\boldsymbol{w}} = (w_0, \boldsymbol{w})$ and $\tilde{\boldsymbol{x}} = (x_0, \boldsymbol{x})$ so that

$$y(\boldsymbol{x}) = \tilde{\boldsymbol{w}}^\intercal \tilde{\boldsymbol{x}}$$

In this case, the decision surface are D-dimensional hyperplanes passing through the origin of the $(D+1)$-dimensional expanded input space.

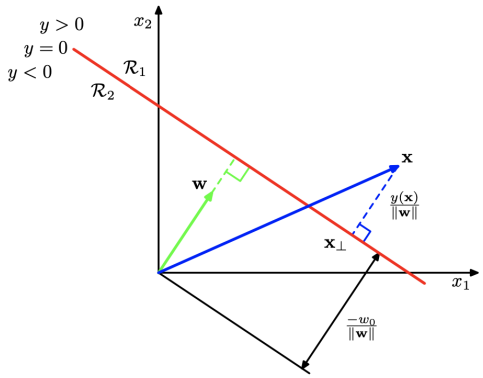

Figure 6.1: Illustration of the geometry of a linear discriminant function in two dimensions. The decision surface, shown in red, is perpendicular to $\boldsymbol{w}$, and its displacement from the origin is controlled by the bias parameter w_0. Also, the signed orthogonal distance of a general point $\boldsymbol{x}$ from the decision surface is given by $y(\boldsymbol{x})/\|\boldsymbol{w}\|$.

6.1.2 Multiple classes

Now consider the extension of linear discriminants to $K > 2$ classes. We might be tempted 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 $\mathcal{C}_k$ from points not in that class. This is known as a *one-versus-the-rest* classifier.

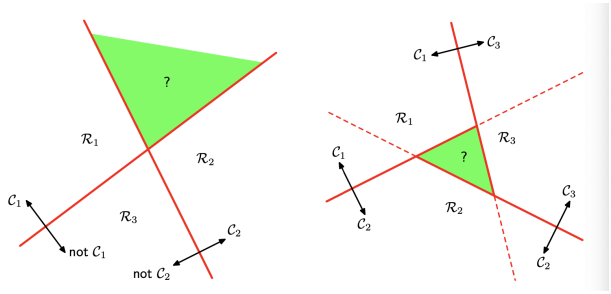

Figure 6.2: Attempting to construct a K class discriminant from a set of two class discriminants leads to ambiguous regions, shown in green. On the left is an example involving the use of two discriminants designed to distinguish points in class $\mathcal{C}_k$ from points not in class $\mathcal{C}_k$. 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 $\mathcal{C}_j$.

The left hand example in Figure 6.2 shows an example involving three classes where this approach leads to regions of input space that are ambiguously classified.

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, as illustrated in the right-hand diagram of Figure 6.2.

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

$$y_k(\boldsymbol{x}) = \boldsymbol{w_k}^\intercal \boldsymbol{x} + w_{k0}$$

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

$$(\boldsymbol{w}_k - \boldsymbol{w}_j)^\intercal \boldsymbol{x} + (w_{k0} - w_{j0}) = 0$$

This has the same form as the decision boundary for the two-class case and so analogous geometrical properties apply.

6.1.3 Fisher's linear discriminant

One way to view a linear classification model is in terms of dimensionality reduction. Consider first the case of two classes, and suppose we take the D-dimensional input vector $\boldsymbol{x}$ and project it down to one dimension using

$$y = \boldsymbol{w}^\intercal \boldsymbol{x} \tag{6.2}$$

If we place a threshold on y and classify $y \geq -w_0$ as class $\mathcal{C}_1$, and otherwise class $\mathcal{C}_2$, then we obtain our standard linear classifier.

In general, the projection onto one dimension leads to a considerable loss of information, and classes that are well separated in the original D-dimensional space may become strongly overlapping in one dimension. However, by adjusting the components of the weight vector $\boldsymbol{w}$, we can select a projection that maximizes the class separation. To begin with, consider a two-class problem in which there are N_1 points of $\mathcal{C}_1$ and N_2 points of class $\mathcal{C}_2$, so that the mean vectors of the two classes are given by

$$\boldsymbol{m}_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} \boldsymbol{x}_n \qquad \boldsymbol{m}_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} \boldsymbol{x}_n$$

The simplest measure of the separation of the classes when projected onto $\boldsymbol{w}$, is the separation of the projected class means. This suggests that we might choose $\boldsymbol{w}$ so as to maximize

$$m_2 - m_1 = \boldsymbol{w}^\intercal (\boldsymbol{m}_2 - \boldsymbol{m}_1)$$

where

$$m_k = \boldsymbol{w}^\intercal \boldsymbol{m}_k \tag{6.3}$$

is the mean of the projected data from class $\mathcal{C}_k$. However, this expression can be arbitrarily large simply by increasing the magnitude of $\boldsymbol{w}$. To solve this problem we could constrain $\boldsymbol{w}$ to have unit length, so that $\sum_i w_i^2 = 1$. Using a Lagrange multiplier to perform the constrained maximization, we then find that $\boldsymbol{w} \propto (\boldsymbol{m}_2 - \boldsymbol{m}_1)$.

There is still a problem with this approach, however, as illustrated in Figure 6.3. This shows two classes that are well separated in the original two- dimensional space (x_1, x_2) but that have considerable overlap when projected onto the line joining their means. This difficulty arises from the strongly non-diagonal covariances of the class distributions. The idea proposed by Fisher is to maximize a function that will give a large separation between the projected class means while also giving a small variance within each class, thereby minimizing the class overlap.

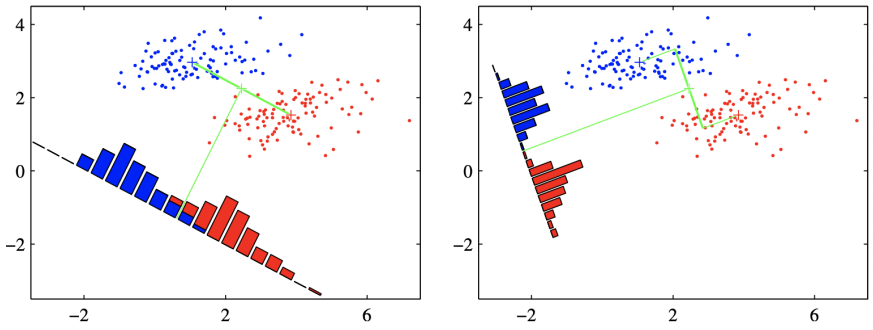

Figure 6.3: The left plot shows samples from two classes (depicted in red and blue) along with the histograms resulting from projection onto the line joining the class means. Note that there is considerable class overlap in the projected space. The right plot shows the corresponding projection based on the Fisher linear discriminant, showing the greatly improved class separation.

The projection formula 6.2 transforms the set of labelled data points in $\boldsymbol{x}$ into a labelled set in the one-dimensional space y. The within-class variance of the transformed data from $\mathcal{C}_k$ is therefore given by

$$s_k^2 = \sum_{n \in \mathcal{C}_k} (y_n - m_k)^2 \tag{6.4}$$

where $y_n = \boldsymbol{w}^\intercal \boldsymbol{x}_n$. We can define the total within-class variance for the whole data set to be simply $s_1^2 + s_2^2$. The fisher criterion is defined to be the ratio of the between-class variance to the within-class variance and is given by

$$J(\boldsymbol{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

We can make the dependence on $\boldsymbol{w}$ explicit by using 6.2, 6.3 and 6.4 to rewrite the Fisher criterion in the form

$$J(\boldsymbol{w}) = \frac{\boldsymbol{w}^\intercal \mathbf{S}_\mathrm{B} \boldsymbol{w}}{\boldsymbol{w}^\intercal \mathbf{S}_\mathrm{W} \boldsymbol{w}} \tag{6.5}$$

where $\mathbf{S}_\mathrm{B}$ is the between-class covariance matrix and is given by

$$\mathbf{S}_\mathrm{B} = (\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^\intercal \tag{6.6}$$

and $\mathbf{S}_\mathrm{W}$ is the total within-class covariance matrix, given by

$$\mathbf{S}_\mathrm{W} = \sum_{n \in \mathcal{C}_1} (\boldsymbol{x}_n - \boldsymbol{m}_1)(\boldsymbol{x}_n - \boldsymbol{m}_1)^\intercal + \sum_{n \in \mathcal{C}_2} (\boldsymbol{x}_n - \boldsymbol{m}_2)(\boldsymbol{x}_n - \boldsymbol{m}_2)^\intercal$$

Differentiating 6.5 with respect to $\boldsymbol{w}$, we find that $J(\boldsymbol{w})$ is maximized when

$$(\boldsymbol{w}^\intercal \mathbf{S}_\mathrm{B} \boldsymbol{w}) \mathbf{S}_\mathrm{W} \boldsymbol{w} = (\boldsymbol{w}^\intercal \mathbf{S}_\mathrm{W} \boldsymbol{w}) \mathbf{S}_\mathrm{B} \boldsymbol{w} \tag{6.7}$$

From 6.6, we see that $\mathbf{S}_\mathrm{B} \boldsymbol{w}$ is always in the direction of $(\boldsymbol{m}_2 - \boldsymbol{m}_1)$. Furthermore we do not care about the magnitude of $\boldsymbol{w}$, only its direction, and so we can drop the scalar factors $(\boldsymbol{w}^\intercal \mathbf{S}_\mathrm{B} \boldsymbol{w})$ and $(\boldsymbol{w}^\intercal \mathbf{S}_\mathrm{W} \boldsymbol{w})$. Multiplying both sides of 6.7 by $\mathbf{S}_\mathrm{W}^{-1}$ we then obtain

$$\boldsymbol{w} \propto \mathbf{S}_\mathrm{W}^{-1} (\boldsymbol{m}_2 - \boldsymbol{m}_1)$$

The projected data can be subsequently used to construct a discriminant, by choosing a threshold y_0 so that we classify a new point as belonging to $\mathcal{C}_1$ if $y(\boldsymbol{x}) \geq y_0$ and classify it as belonging to $\mathcal{C}_2$ otherwise.
We can also model the class-conditional densities $p(y|\mathcal{C}_K)$ using Gaussian distributions and then use MLE to find the parameters of the distributions. Some justification for the Gaussian assumption comes from the central limit theorem by noting that $y = \boldsymbol{w}^\intercal \boldsymbol{x}$ is the sum of a set of random variables.

6.1.4 The Perceptron algorithm

Another example of a linear discriminant model is the perceptron that corresponds to a two-class model in which the input vector $\boldsymbol{x}$ is first transformed using a fixed nonlinear transformation to give a feature vector $\boldsymbol{\phi}(\boldsymbol{x})$, and this is then used to construct a generalized linear model of the form

$$y(\boldsymbol{x}) = f(\boldsymbol{w}^\intercal \boldsymbol{\phi}(\boldsymbol{x})) \tag{6.8}$$

where the non-linear activation function $f(\cdot)$ is given by a step function of the form

$$f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0 \end{cases}$$

The vector $\boldsymbol{\phi}(\boldsymbol{x})$ will typically include bias component $\phi_0(\boldsymbol{x}) = 1$. The target values t are +1 for class $\mathcal{C}_1$ and -1 for class $\mathcal{C}_2$.
The algorithm used to determine the parameters $\boldsymbol{w}$ of the perceptron can most easily be motivated by error function minimization. A natural choice of error function would be the total number of misclassified patterns. However, this does not lead to a simple algorithm because the error is a piecewise constant function of $\boldsymbol{w}$, with discontinuities wherever a change in $\boldsymbol{w}$ causes the decision boundary to move across one of the data points. Methods based on changing $\boldsymbol{w}$ using the gradient of the error function cannot then be applied, because the gradient is zero almost everywhere.

We therefore consider an alternative error function known as the *perceptron criterion*. To derive this, we note that we are seeking a weight vector $\boldsymbol{w}$ such that patterns $\boldsymbol{x}_n$ in class $\mathcal{C}_1$ will have $\boldsymbol{w}^\intercal \boldsymbol{\phi}(\boldsymbol{x}_n) > 0$, whereas patterns $\boldsymbol{x}_n$ in class $\mathcal{C}_2$ have $\boldsymbol{w}^\intercal \boldsymbol{\phi}(\boldsymbol{x}_n) < 0$. Using the $t \in \{-1, +1\}$ target coding scheme it follows that we would like all patterns to satisfy $\boldsymbol{w}^\intercal \boldsymbol{\phi}(\boldsymbol{x}_n) t_n > 0$. The perceptron criterion associates zero error with any pattern that is correctly classified, whereas for a misclassified pattern $\boldsymbol{x}_n$ it tries to minimize the quantity $-\boldsymbol{w}^\intercal \boldsymbol{\phi}(\boldsymbol{x}_n) t_n$. The perceptron criterion is therefore given by

$$E_P(\boldsymbol{w}) = -\sum_{n \in \mathcal{M}} \boldsymbol{w}^\intercal \boldsymbol{\phi}_n t_n$$

where $\mathcal{M}$ denotes the set of all misclassified patterns. The contribution to the error associated with a particular misclassified pattern is a linear function of $\boldsymbol{w}$ in regions of $\boldsymbol{w}$ space where the pattern is misclassified and zero in regions where it is correctly classified. The total error function is therefore piecewise linear.
We now apply the stochastic gradient descent algorithm to this error function. The change in the weight vector $\boldsymbol{w}$ is then given by

$$\boldsymbol{w}^{\tau+1} = \boldsymbol{w}^\tau - \eta \nabla E_P(\boldsymbol{w}) = \boldsymbol{w}^\tau + \eta \boldsymbol{\phi}_n t_n \tag{6.9}$$

where η is the learning rate parameter and τ is an integer that indexes the steps of the algorithm. Because the perceptron function $y(\boldsymbol{x}, \boldsymbol{w})$ is unchanged if we multiply $\boldsymbol{w}$ by a constant, we can set the learning rate η equal to 1 without loss of generality. Note that, as the weight vector evolves during training, the set of patterns that are misclassified will change.

The algorithm consists in cycling through the training patterns, and for each pattern $\boldsymbol{x}_n$, we evaluate the perceptron function 6.8. If the pattern is correctly classified, then the weight vector remains unchanged, whereas if it is incorrectly classified, then for class $\mathcal{C}_1$ we add the vector $\boldsymbol{\phi}(\boldsymbol{x}_n)$ onto the current estimate of the weight vector $\boldsymbol{w}$ while for class $\mathcal{C}_2$ we subtract the vector $\boldsymbol{\phi}(\boldsymbol{x}_n)$ from $\boldsymbol{w}$.
If we consider the effect of a single update in the perceptron learning algorithm, we see that the contribution to the error from a misclassified pattern will be reduced because from 6.9 we have

$$-\boldsymbol{w}^{(\tau+1)\intercal} \boldsymbol{\phi}_n t_n = -\boldsymbol{w}^{(\tau)\intercal} \boldsymbol{\phi}_n t_n - (\boldsymbol{\phi}_n t_n)^\intercal \boldsymbol{\phi}_n t_n < -\boldsymbol{w}^{(\tau)\intercal} \boldsymbol{\phi}_n t_n$$

where we have set $\eta = 1$ and made us of $\|\boldsymbol{\phi}_n t_n\|^2 > 0$.
Of course, this does not imply that the contribution to the error function from the other misclassified patterns will have been reduced. Furthermore, the change in weight vector may have caused some previously correctly classified patterns to become misclassified. Thus the perceptron learning rule is not guaranteed to reduce the total error function at each stage.
However, the perceptron convergence theorem states that if there exists an exact solution (in other words, if the training data set is linearly separable), then the perceptron learning algorithm is guaranteed to find an exact solution in a finite number of steps.
Aside from difficulties with the learning algorithm, the perceptron does not provide probabilistic outputs, nor does it generalize readily to $K > 2$ classes. The most important limitation, however, arises from the fact that it is based on linear combinations of fixed basis functions.

6.2 Probabilistic Generative Models

We turn next to a probabilistic view of classification and show how models with linear decision boundaries arise from simple assumptions about the distribution of the data.
Here we shall adopt a generative approach in which we model the class-conditional densities $p(\boldsymbol{x}|\mathcal{C}_k)$, as well as the class priors $p(\mathcal{C}_k)$, and then use these to compute posterior probabilities $p(\mathcal{C}_k|\boldsymbol{x})$ through Bayes' theorem.
In the case of two classes, the posterior probability for class $\mathcal{C}_1$ can be written as

$$\begin{aligned} p(\mathcal{C}_1|\boldsymbol{x}) &= \frac{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\boldsymbol{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \\ &= \frac{1}{1+\exp(-a)} = \sigma(a) \end{aligned} \tag{6.10}$$

where we have defined

$$a = \ln \frac{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\boldsymbol{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \tag{6.11}$$

and $\sigma(a)$ is the *logistic sigmoid* function. It satisfies the symmetry property $\sigma(-a) = 1 - \sigma(a)$ and its inverse is given by

$$a = \ln\left(\frac{\sigma}{1-\sigma}\right)$$

and is know as the *logit* function. It represents the log of the ratio of probabilities $\ln[p(\mathcal{C}_1|\boldsymbol{x})/p(\mathcal{C}_2|\boldsymbol{x})]$ for the two classes, also know as the *log odds*.

Let us assume that the class-conditional densities are Gaussian and then explore the resulting form for the posterior probabilities. To start with, we shall assume that all classes share the same covariance matrix. Thus the density for class $\mathcal{C}_k$ is given by

$$p(\boldsymbol{x}|\mathcal{C}_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^\intercal \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)\right\}$$

Consider the case of two classes. From 6.10 and 6.11, we have

$$p(\mathcal{C}_1|\boldsymbol{x}) = \sigma(\boldsymbol{w}^\intercal \boldsymbol{x} + w_0)$$

where we have defined

$$\begin{aligned} \boldsymbol{w} &= \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\ w_0 &= -\frac{1}{2}\boldsymbol{\mu}_1^\intercal \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^\intercal \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + \ln\frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)} \end{aligned}$$

We see that the quadratic terms in $\boldsymbol{x}$ from the exponents of the Gaussian densities have cancelled (due to the assumption of common covariance matrices) leading to a linear function of $\boldsymbol{x}$ in the argument of the logistic sigmoid. The resulting decision boundaries correspond to surfaces along which the posterior probabilities $p(\mathcal{C}_k|\boldsymbol{x})$ are constant and so will be given by linear functions of $\boldsymbol{x}$, and therefore the decision boundaries are linear in input space. The prior probabilities $p(\mathcal{C}_k)$ enter only through the bias parameter w_0 so that changes in the priors have the effect of making parallel shifts of the decision boundary and more generally of the parallel contours of constant posterior probability.

If we relax the assumption of a shared covariance matrix and allow each class-conditional density $p(\boldsymbol{x}|\mathcal{C}_k)$ to have its own covariance matrix $\boldsymbol{\Sigma}_k$, then the earlier cancellations will no longer occur, and we will obtain quadratic functions of $\boldsymbol{x}$, giving rise to a *quadratic discriminant*

$$\begin{aligned}\ln \frac{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\boldsymbol{x}|\mathcal{C}_2)p(\mathcal{C}_2)} &= \ln p(\boldsymbol{x}|\mathcal{C}_1) - \ln p(\boldsymbol{x}|\mathcal{C}_2) + \ln \frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)} \\ &= \boldsymbol{x}^\intercal \boldsymbol{W}\boldsymbol{x} + \boldsymbol{w}^\intercal \boldsymbol{x} + w_0\end{aligned}$$

where the *generalized quadratic discriminant* is $p(\mathcal{C}_k|\boldsymbol{x}) = \sigma(\boldsymbol{x}^\intercal \boldsymbol{W}\boldsymbol{x} + \boldsymbol{w}^\intercal \boldsymbol{x} + w_0)$.

6.2.1 MLE Solution

Once we have specified a parametric functional form for the class-conditional densities $p(\boldsymbol{x}|\mathcal{C}_k)$, we can then determine the values of the parameters, together with the prior class probabilities $p(\mathcal{C}_k)$, using maximum likelihood. This requires a data set comprising observations of $\boldsymbol{X}$ along with their corresponding class labels. In the case of two classes, each having a Gaussian class-conditional density with a shared covariance matrix, and suppose we have a data set $\{\boldsymbol{x}_n, t_n\}$ where $n = 1, ..., N$. Here $t_n = 1$ denotes class $\mathcal{C}_1$ and $t_n = 0$ denotes class $\mathcal{C}_2$. We denote the prior class probability $p(\mathcal{C}_1) = \pi$, so that $p(\mathcal{C}_2) = 1 - \pi$. For a data point $\boldsymbol{x}_n$ from class $\mathcal{C}_1$, we have $t_n = 1$ and hence

$$p(\boldsymbol{x}_n, \mathcal{C}_1) = p(\mathcal{C}_1)p(\boldsymbol{x}_n|\mathcal{C}_1) = \pi\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma})$$

Similarly for class $\mathcal{C}_2$, we have $t_n = 0$ and hence

$$p(\boldsymbol{x}_n, \mathcal{C}_2) = p(\mathcal{C}_2)p(\boldsymbol{x}_n|\mathcal{C}_2) = (1 - \pi)\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_2, \boldsymbol{\Sigma})$$

Thus the likelihood function is given by

$$p(\boldsymbol{t}|\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = \prod_{n=1}^{N} [\pi\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma})]^{t_n} [(1 - \pi)\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_2, \boldsymbol{\Sigma})]^{1-t_n}$$

where $\boldsymbol{t} = (t_1, ..., t_N)^\intercal$. As usual, it is convenient to maximize the log of the likelihood function. Consider first the maximization with respect to π. The terms in the log likelihood function that depend on π are

$$\sum_{n=1}^{N} \{(t_n \ln \pi + (1 - t_n) \ln (1 - \pi))\}$$

Setting the derivative with respect to π equal to zero and rearranging, we obtain

$$\pi = \frac{1}{N} \sum_{n=1}^{N} t_n = \frac{N_1}{N} = \frac{N_1}{N_1 + N_2}$$

where N_1 denotes the total number of data points in class $\mathcal{C}_1$, and N_2 denotes the total number of data points in class $\mathcal{C}_2$. Thus the maximum likelihood estimate for π is simply the fraction of points in class $\mathcal{C}_1$

as expected. This result is easily generalized to the multiclass case where again the maximum likelihood estimate of the prior probability associated with class $\mathcal{C}_k$ is given by the fraction of the training set points assigned to that class.
Now consider the maximization with respect to $\boldsymbol{\mu}_1$. Again we can pick out of the log likelihood function those terms that depend on $\boldsymbol{\mu}_1$ giving

$$\sum_{n=1}^{N} t_n \ln \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}) = -\frac{1}{2}\sum_{n=1}^{N} t_n(\boldsymbol{x}_n - \boldsymbol{\mu}_1)^\intercal \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_1) + \text{const}$$

Setting the derivative with respect to $\boldsymbol{\mu}_1$ to zero and rearranging, we obtain

$$\boldsymbol{\mu}_1 = \frac{1}{N_1}\sum_{n=1}^{N} t_n \boldsymbol{x}_n$$

which is simply the mean of all the input vectors $\boldsymbol{x}_n$ assigned to class $\mathcal{C}_1$. By a similar argument, the corresponding result for $\boldsymbol{\mu}_2$ is given by

$$\boldsymbol{\mu}_2 = \frac{1}{N_2}\sum_{n=1}^{N} (1 - t_n)\boldsymbol{x}_n$$

which again is the mean of all the input vectors $\boldsymbol{x}_n$ assigned to class $\mathcal{C}_2$.
Finally, consider the maximum likelihood solution for the shared covariance matrix $\boldsymbol{\Sigma}$. Picking out the terms in the log likelihood function that depend on $\boldsymbol{\Sigma}$, we have

$$\begin{aligned}
&-\frac{1}{2}\sum_{n=1}^{N} t_n \ln|\boldsymbol{\Sigma}| - \frac{1}{2}\sum_{n=1}^{N} t_n(\boldsymbol{x}_n - \boldsymbol{\mu}_1)^\intercal \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_1) \\
&-\frac{1}{2}\sum_{n=1}^{N} (1 - t_n) \ln|\boldsymbol{\Sigma}| - \frac{1}{2}\sum_{n=1}^{N} (1 - t_n)(\boldsymbol{x}_n - \boldsymbol{\mu}_2)^\intercal \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_2) \\
&= -\frac{N}{2}\ln|\boldsymbol{\Sigma}| - \frac{N}{2}\text{Tr}\{\boldsymbol{\Sigma}^{-1}\mathbf{S}\}
\end{aligned}$$

where we have defined

$$\mathbf{S} = \frac{N_1}{N}\mathbf{S1} + \frac{N_2}{N}\mathbf{S2}$$

$$\mathbf{S}_1 = \frac{1}{N_1}\sum_{n\in\mathcal{C}_1} (\boldsymbol{x}_n - \boldsymbol{\mu}_1)(\boldsymbol{x}_n - \boldsymbol{\mu}_1)^\intercal$$

$$\mathbf{S}_2 = \frac{1}{N_2}\sum_{n\in\mathcal{C}_2} (\boldsymbol{x}_n - \boldsymbol{\mu}_2)(\boldsymbol{x}_n - \boldsymbol{\mu}_2)^\intercal$$

Using the standard result for the maximum likelihood solution for a Gaussian distribution, we see that $\boldsymbol{\Sigma} = \mathbf{S}$, which represents a weighted average of the covariance matrices associated with each of the two classes separately.
This result is easily extended to the K class problem to obtain the corresponding maximum likelihood solutions for the parameters in which each class-conditional density is Gaussian with a shared covariance matrix. Note that the approach of fitting Gaussian distributions to the classes is not robust to outliers, because the maximum likelihood estimation of a Gaussian is not robust.