Discriminant Functions

Hiroshi Shimodaira*

January-March 2020

In the previous chapter we saw how we can combine a Gaussian probability density function with class prior probabilities using Bayes' theorem to estimate class-conditional posterior probabilities. For each point in the input space we can estimate the posterior probability of each class, assigning that point to the class with the maximum posterior probability. We can view this process as dividing the input space into decision regions, separated by decision boundaries. In the next section we investigate whether the maximum posterior probability rule is indeed the best decision rule (in terms of minimising the number of errors). In the following sections we introduce discriminant functions which define the decision boundaries, and investigate the form of decision functions induced by Gaussian pdfs with different constraints on the covariance matrix. We then discuss a particular type of discriminant functions, *perceptron*, which finds a decision boundary directly from data without knowing probabilistic distributions.

10.1 Decision boundaries

We may assign each point in the input space as a particular class. This divides the input space into *decision regions* $\mathcal{R}_k$, such that a point falling in $\mathcal{R}_k$ is assigned to class C_k. In the general case, a decision region $\mathcal{R}_k$ need not be contiguous, but may consist of several disjoint regions each associated with class C_k. The boundaries between these regions are called *decision boundaries*.

Figure 10.1 shows the decision regions that result from assigning each point to the class with the maximum posterior probability, using the Gaussians estimated for classes A, B and C from the example in the previous chapter.

10.1.1 Placement of decision boundaries

Estimating posterior probabilities for each class results in the input space being divided into decision regions, if each point is classified as the class with the highest posterior probability. But is this an optimal placement of decision boundaries?

Consider a 1-dimensional feature space (x) and two classes C_1 and C_2. A reasonable criterion for the placement of decision boundaries is one that *minimises the probability of misclassification*. To estimate the probability of misclassification we need to consider the two ways that a point can be classified wrongly:

1. assigning x to C_1 when it belongs to C_2 (x is in decision region $\mathcal{R}_1$ when it belongs to class C_2);

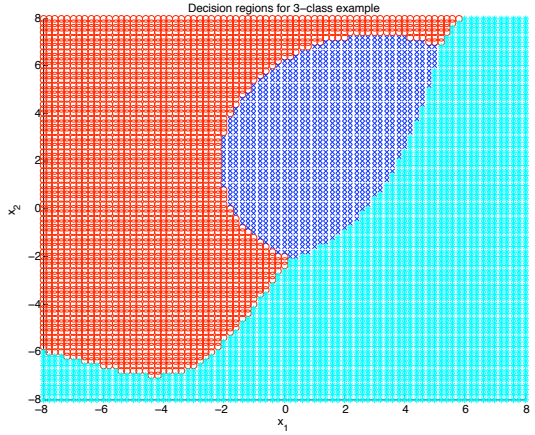


Figure 10.1: Decision regions for the three-class two-dimensional problem from the previous chapter. Class A (red), class B (blue), class C (cyan).

2. assigning x to C_2 when it belongs to C_1 (x is in $\mathcal{R}_2$ when it belongs to C_1).

Thus the probability of the total error may be written as:

$$P(\text{error}) = P(x \in \mathcal{R}_2, C_1) + P(x \in \mathcal{R}_1, C_2)\,.$$

Expanding the terms on the right hand side as conditional probabilities, we may write:

$$P(\text{error}) = P(x \in \mathcal{R}_2 | C_1)\, P(C_1) + P(x \in \mathcal{R}_1 | C_2)\, P(C_2)\,. \tag{10.1}$$

10.1.2 Overlapping Gaussians

Figure 10.2 illustrates two overlapping Gaussian distributions (assuming equal priors). Two possible decision boundaries are illustrated and the two regions of error are coloured.

We can obtain $P(x \in \mathcal{R}_2 | C_1)$ by integrating $p(x|C_1)$ within $\mathcal{R}_2$, and similarly for $P(x \in \mathcal{R}_1 | C_2)$, and thus rewrite Equation (10.1) as:

$$P(\text{error}) = \int_{\mathcal{R}_2} p(x|C_1)\, P(C_1)\, \mathrm{d}x + \int_{\mathcal{R}_1} p(x|C_2)\, P(C_2)\, \mathrm{d}x\,. \tag{10.2}$$

Minimising the probability of misclassification is equivalent to minimising $P(\text{error})$. From Equation (10.2) we can see that this is achieved as follows, for a given x:

- if $p(x|C_1)\, P(C_1) > p(x|C_2)\, P(C_2)$, then point x should be in region $\mathcal{R}_1$;
- if $p(x|C_2)\, P(C_2) > p(x|C_1)\, P(C_1)$, then point x should be in region $\mathcal{R}_2$.

The probability of misclassification is thus minimised by assigning each point to the class with the maximum posterior probability.

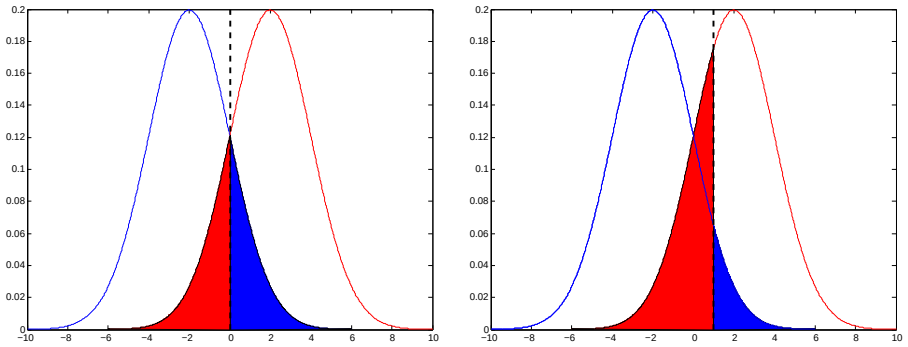


Figure 10.2: Overlapping Gaussian pdfs. Two possible decision boundaries are shown by the dashed line. The decision boundary on the left hand plot is optimal, assuming equal priors. The overall probability of error is given by the area of the shaded regions under the pdfs.

It is possible to extend this justification for a decision rule based on the maximum posterior probability to D-dimensional feature vectors and K classes. In this case consider the probability of a pattern being correctly classified:

$$P(\text{correct}) = \sum_{k=1}^{K} P(\mathbf{x} \in \mathcal{R}_k, C_k) \tag{10.3}$$

$$= \sum_{k=1}^{K} P(\mathbf{x} \in \mathcal{R}_k | C_k)\, P(C_k) \tag{10.4}$$

$$= \sum_{k=1}^{K} \int_{\mathcal{R}_k} p(\mathbf{x}|C_k)\, P(C_k)\, \mathrm{d}x\,. \tag{10.5}$$

This performance measure is maximised by choosing the $\mathcal{R}_k$ such that each $\mathbf{x}$ is assigned to the class k that maximises $p(\mathbf{x}|C_k)\,P(C_k)$. This procedure is equivalent to assigning each $\mathbf{x}$ to the class with the maximum posterior probability.

Thus the maximum posterior probability decision rule is equivalent to minimising the probability of misclassification. However, to obtain this result we assumed both that the class-conditional models are correct, and that the models are well-estimated from the data.

10.2 Discriminant functions

If we have a set of K classes then we may define a set of K *discriminant functions* $y_k(\mathbf{x})$, one for each class. Data point $\mathbf{x}$ is assigned to class C_k if

$$y_k(\mathbf{x}) > y_\ell(\mathbf{x}) \qquad \text{for all } \ell \neq k.$$

In other words: assign $\mathbf{x}$ to the class C_k whose discriminant function $y_k(\mathbf{x})$ is biggest.

This is precisely what we did in the previous chapter when classifying based on the values of the log posterior probability. Thus the log posterior probability of class C_k given a data point $\mathbf{x}$ is a possible

discriminant function:

$$y_k(\mathbf{x}) = \ln P(C_k|\mathbf{x}) = \ln p(\mathbf{x}|C_k) + \ln P(C_k) + \text{const.} \tag{10.6}$$

The posterior probability could also be used as a discriminant function, with the same results: choosing the class with the largest posterior probability is an identical decision rule to choosing the class with the largest log posterior probability.

As discussed above, classifying a point as the class with the largest (log) posterior probability corresponds to the decision rule which minimises the probability of misclassification. In that sense, it forms an optimal discriminant function. A decision boundary occurs at points in the input space where discriminant functions are equal. If the region of input space classified as class C_k and the region classified as class C_ℓ are contiguous, then the decision boundary separating them is given by:

$$y_k(\mathbf{x}) = y_\ell(\mathbf{x})\,.$$

Decision boundaries are not changed by monotonic transformations (such as taking the log) of the discriminant functions.

Formulating a pattern classification problem in terms of discriminant functions is useful since it is possible to estimate the discriminant functions directly from data, without having to estimate probability density functions on the inputs. Direct estimation of the decision boundaries is sometimes referred to as *discriminative* modelling. In contrast, the models that we have considered so far are *generative* models: they could generate new 'fantasy' data by choosing a class label, and then sampling an input from its class-conditional model.

10.3 Discriminant functions for class-conditional Gaussians

What is the form of the discriminant function when using a Gaussian pdf? As before, we take the discriminant function as the log posterior probability:

$$\begin{aligned} y_k(\mathbf{x}) &= \ln P(C_k|\mathbf{x}) = \ln p(\mathbf{x}|C_k) + \ln P(C_k) + \text{const.} \\ &= -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_k| + \ln P(C_k)\,. \end{aligned} \tag{10.7}$$

We have dropped the term $-1/2\ln(2\pi)$, since it is a constant that occurs in the discriminant function for each class. The first term on the right hand side of Equation (10.7) is quadratic in the elements of $\mathbf{x}$ (i.e., if you multiply out the elements, there will be some terms containing x_i^2 or $x_i x_j$).

10.4 Linear discriminants

Let's consider the case in which the Gaussian pdfs for each class all share the same covariance matrix. That is, for all classes C_k, $\boldsymbol{\Sigma_k} = \boldsymbol{\Sigma}$. In this case $\boldsymbol{\Sigma}$ is class-independent (since it is equal for all classes), therefore the term $-1/2\ln|\boldsymbol{\Sigma}|$ may also be dropped from the discriminant function and we have:

$$y_k(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) + \ln P(C_k)\,. \tag{10.8}$$

If we explicitly expand the quadratic matrix-vector expression we obtain the following:

$$y_k(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - \mathbf{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k - \boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\mathbf{x} + \boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k) + \ln P(C_k)\,. \tag{10.9}$$

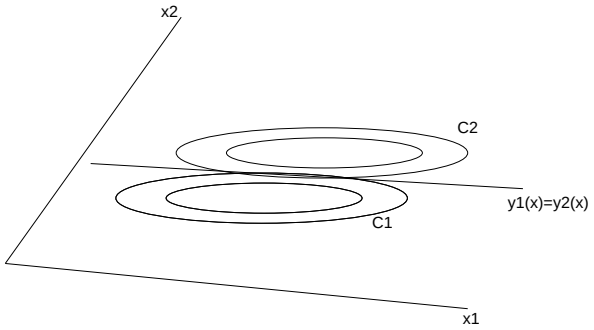


Figure 10.3: Discriminant function for equal covariance Gaussians

The mean $\boldsymbol{\mu_k}$ depends on class C_k, but (as stated before) the covariance matrix is class-independent. Therefore, terms that do not include the mean or the prior probabilities are class independent, and may be dropped. Thus we may drop $\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x}$ from the discriminant.

We can simplify this discriminant function further. It is a fact that for a symmetric matrix $\mathbf{M}$ and vectors $\mathbf{a}$ and $\mathbf{b}$:

$$\mathbf{a}^T\mathbf{M}\mathbf{b} = \mathbf{b}^T\mathbf{M}\mathbf{a}\,.$$

Now since the covariance matrix $\boldsymbol{\Sigma}$ is symmetric, it follows that $\boldsymbol{\Sigma}^{-1}$ is also symmetric.[1] Therefore:

$$\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k = \boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\mathbf{x}\,.$$

We can thus simplify Equation (10.9) as:

$$y_k(\mathbf{x}) = \boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \ln P(C_k)\,. \tag{10.10}$$

This equation has three terms on the right hand side, but only the first depends on $\mathbf{x}$. We can define two new variables $\mathbf{w_k}$ (D-dimension vector) and w_{k0}, which are derived from $\boldsymbol{\mu}_k$, $P(C_k)$, and $\boldsymbol{\Sigma}$:

$$\mathbf{w}_k^T = \boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1} \tag{10.11}$$

$$w_{k0} = -\frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \ln P(C_k) = -\frac{1}{2}\mathbf{w}_k^T\boldsymbol{\mu}_k + \ln P(C_k)\,. \tag{10.12}$$

Substituting Equation (10.11) and Equation (10.12) into Equation (10.10) we obtain:

$$y_k(\mathbf{x}) = \mathbf{w}_k^T\mathbf{x} + w_{k0}\,. \tag{10.13}$$

This is a linear equation in D dimensions. We refer to $\mathbf{w}_k$ as the *weight vector* and w_{k0} as the *bias* for class C_k.

We have thus shown that the discriminant function for a Gaussian which shares the same covariance matrix with the Gaussians pdfs of all the other classes may be written as Equation (10.13). We call such discriminant functions *linear discriminants*: they are linear functions of $\mathbf{x}$. If $\mathbf{x}$ is two-dimensional, the decision boundaries will be straight lines, illustrated in Figure 10.3. In three dimensions the decision boundaries will be planes. In D-dimensions the decision boundaries are called *hyperplanes*.

[1] It also follows that $\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x} \geq 0$ for any $\mathbf{x}$.

10.5 Spherical Gaussians with equal covariance

Let's look at an even more constrained case, where not only do all the classes share a covariance matrix, but that covariance matrix is *spherical*: the off-diagonal terms (covariances) are all zero, and the diagonal terms (variances) are equal for all components. In this case the matrix may be defined by a single number, σ^2, the value of the variances:

$$\boldsymbol{\Sigma} = \sigma^2\mathbf{I}$$

$$\boldsymbol{\Sigma}^{-1} = \frac{1}{\sigma^2}\mathbf{I}$$

where $\mathbf{I}$ is the identity matrix.

Since this is a special case of Gaussians with equal covariance, the discriminant functions are linear, and may be written as Equation (10.13). However, we can get another view of the discriminant functions if we write them as:

$$y_k(\mathbf{x}) = -\frac{\|\mathbf{x} - \boldsymbol{\mu}_k\|^2}{2\sigma^2} + \ln P(C_k)\,. \tag{10.14}$$

If the prior probabilities are equal for all classes, the decision rule simply assigns an unseen vector to the nearest class mean (using the Euclidean distance). In this case the class means may be regarded as class *templates* or *prototypes*.

Exercise: Show that Equation (10.14) is indeed reduced to a linear discriminant.

10.6 Two-class linear discriminants

To get some more insights into linear discriminants, we can look at another special case: two-class problems. Two class problems occur quite often in practice, and they are more straightforward to think about because we are considering a single decision boundary between the two classes.

In the two-class case it is possible to use a single discriminant function: for example one which takes value zero at the decision boundary, negative values for one class and positive values for the other. A suitable discriminant function in this case is the log odds (log ratio of posterior probabilities):

$$\begin{aligned} y(\mathbf{x}) &= \ln\frac{P(C_1|\mathbf{x})}{P(C_2|\mathbf{x})} = \ln\frac{p(\mathbf{x}|C_1)}{p(\mathbf{x}|C_2)} + \ln\frac{P(C_1)}{P(C_2)} \\ &= \ln p(\mathbf{x}|C_1) - \ln p(\mathbf{x}|C_2) + \ln P(C_1) - \ln P(C_2)\,. \end{aligned} \tag{10.15}$$

Feature vector $\mathbf{x}$ is assigned to class C_1 when $y(\mathbf{x}) > 0$; $\mathbf{x}$ is assigned to class C_2 when $y(\mathbf{x}) < 0$. The decision boundary is defined by $y(\mathbf{x}) = 0$.

If the pdf for each class is a Gaussian, and the covariance matrix is shared, then the discriminant function is linear:

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

where $\mathbf{w}$ is a function of the class-dependent means and the class-independent covariance matrix, and the w_0 is a function of the means, the covariance matrix and the prior probabilities.

The decision boundary for the two-class linear discriminant corresponds to a $(D-1)$-dimensional hyperplane in the input space. Let $\mathbf{a}$ and $\mathbf{b}$ be arbitrary two points on the decision boundary. Then:

$$y(\mathbf{a}) = 0 = y(\mathbf{b})\,.$$

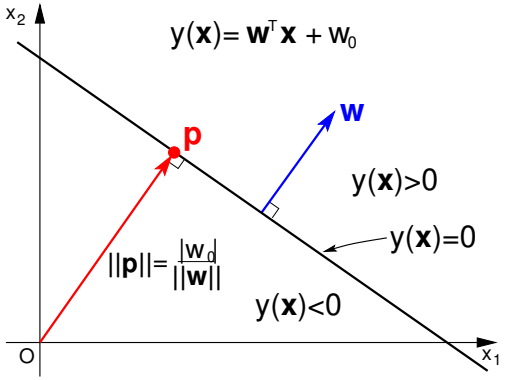


Figure 10.4: Geometry of a two-class linear discriminant $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$.

And since $y(\mathbf{x})$ is a linear discriminant:

$$\mathbf{w}^T\mathbf{a} + w_0 = 0 = \mathbf{w}^T\mathbf{b} + w_0.$$

And a little rearranging gives us:

$$\mathbf{w}^T(\mathbf{a} - \mathbf{b}) = 0\,. \tag{10.16}$$

In three dimensions, Equation (10.16) is the equation of a plane, with $\mathbf{w}$ being the vector normal to the plane. In higher dimensions, this equation describes a *hyperplane*, and $\mathbf{w}$ is normal to any vector lying on the hyperplane. The hyperplane is the decision boundary in this two-class problem.

The distance from the hyperplane to the origin $(0, 0)$ is the length of of the position vector $\mathbf{p}$ that points to the foot of perpendicular from the origin to the hyperplane. Since $y(\mathbf{p}) = \mathbf{w}^T\mathbf{p} + w_0 = 0$, the distance is thus given by:

$$\ell = \|\mathbf{p}\| = \left|\left(\frac{\mathbf{w}}{\|\mathbf{w}\|}\right)^T\mathbf{p}\right| = \frac{|w_0|}{\|\mathbf{w}\|}, \tag{10.17}$$

which is illustrated in Figure 10.4.

10.7 Perceptron

We now consider a two-class linear discriminant function whose output is binary - 0 for Class 0, and 1 for Class 1. This can be achieved by applying a unit step function $g(a)$ to the output of linear discriminant, so that the binary-output discriminant function is defined as

$$y(\mathbf{x}) = g(\mathbf{w}^T\mathbf{x} + w_0), \tag{10.18}$$

where

$$g(a) = \begin{cases} 1, & a \geq 0, \\ 0, & a < 0. \end{cases} \tag{10.19}$$

We see that the output $y(\mathbf{x})$ is 1 if $\mathbf{x}$ belongs to the positive side of the decision boundary (including the boundary), and 0 otherwise.

This type of discriminant function is called '*perceptron*', which was invented by Frank Rosenblatt in late 1950s. The original idea of perceptron was to mimic a funciton of human brain with a machine

that is capable of learning, where the machine adjusts its parameters in a trial and error manner to reduce errors. The Rosenblatt's original perceptron has very limited ability, but it has been extended in various ways, and it forms the basis of modern artificial neural networks. We consider a simple mathematical model of the original perceptron in this section.

10.7.1 Perceptron error correction algorithm

Assume we have got a set of training samples $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, where each sample $\mathbf{x}_i$, $i = 1, \ldots, N$, has a label t_i of a binary value, 0 for Class 0 and 1 for Class 1.

For the sake of simplicity, we introduce new variables (augmented vectors), $\dot{\mathbf{w}} = \begin{pmatrix} w_0 \\ \mathbf{w} \end{pmatrix}$ and $\dot{\mathbf{x}} = \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix}$, so that

$$\mathbf{w}^T\mathbf{x} + w_o = \dot{\mathbf{w}}^T\dot{\mathbf{x}}\,. \tag{10.20}$$

We re-define $y(\dot{\mathbf{x}})$ accordingly such that $y(\dot{\mathbf{x}}) = g(\dot{\mathbf{w}}^T\dot{\mathbf{x}})$. In the perceptron training, we repeat adjusting the weight vector $\dot{\mathbf{w}}$ for each input $\dot{\mathbf{x}}_i$ to reduce misclassification. Let $\dot{\mathbf{w}}^{(s)}$ denote the weight vector after s-th training iteration, where $\dot{\mathbf{w}}^{(0)}$ means the initial weight vector. We train the model with the following algorithm.

Step 1 Set $s = 0$, and initialise the weight vector $\dot{\mathbf{w}}^{(0)}$.

Step 2 For each training sample $\dot{\mathbf{x}}_i$, $i = 1, \ldots, N$, calculate the output $y(\dot{\mathbf{x}}_i)$, and modify the weight vector as follows if $\dot{\mathbf{x}}_i$ is misclassified.

$$\dot{\mathbf{w}}^{(s+1)} = \begin{cases} \dot{\mathbf{w}}^{(s)} + \eta\,\dot{\mathbf{x}}_i, & t_i = 1, \\ \dot{\mathbf{w}}^{(s)} - \eta\,\dot{\mathbf{x}}_i, & t_i = 0, \end{cases} \tag{10.21}$$

where η is a positive constant $(0 < \eta < 1)$ called *learning rate*. The weight vector is not modified if $\dot{\mathbf{x}}_i$ is classified correctly.

Step 3 Increment s by 1, and terminate if $\dot{\mathbf{w}}$ does not change (i.e. convergence), otherwise repeat Step 2.

The Step 2 requires us to check whether $\dot{\mathbf{x}}_i$ is classified correctly or not, which can be simplified to the following formula:

$$\dot{\mathbf{w}}^{(s+1)} = \dot{\mathbf{w}}^{(s)} + \eta\,(t_i - y(\dot{\mathbf{x}}_i))\,\dot{\mathbf{x}}_i\,. \tag{10.22}$$

Does the training algorithm above stop in a finite time? We can prove that the algorithm converges if the data set is '*linearly separable*' - there is a linear decision boundary that separates all the training samples without errors. On the other hand, the algorithm does not converge if the data set is not linearly separable.

10.7.2 Multi-layer Perceptron

Although the original perceptron is just a linear classifier, we can combine more than one perceptron to form complex decision boundaries and regions.

Let's consider a toy example of 2D data shown in Figure 10.5, in which there are two decision boundaries and three disjoint regions - one region shown in a dark colour corresponds to Class 1, and the other two regions shown in white correspond to Class 0. Although each of the decision boundaries

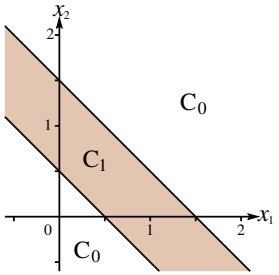


Figure 10.5: An example of a data set that is not linearly separable.

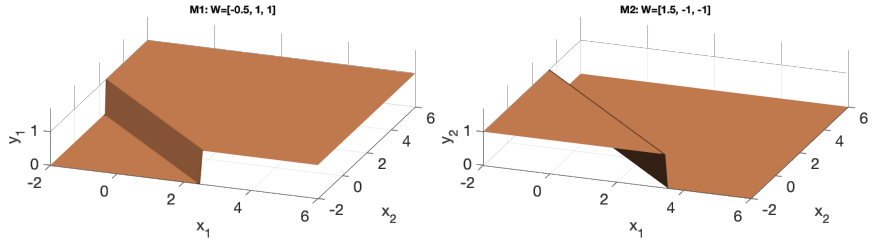
is linear, the data set is not linearly separable, and a single perceptron is unable to have more than one decision boundary.

To tackle this problem, we start with considering two perceptrons, $M_1 : y_1(\dot{\mathbf{x}}) = g(\dot{\mathbf{w}}_1^T\dot{\mathbf{x}})$, and $M_2 : y_2(\dot{\mathbf{x}}) = g(\dot{\mathbf{w}}_2^T\dot{\mathbf{x}})$, each of which is responsible for one of the two decision boundaries. Figure 10.6 illustrates the decision boundaries and regions for M_1 and M_2, in which it can be confirmed that the intersection of the dark regions (where $y_1 = 1$ and $y_2 = 1$) corresponds to Class 1. Figure 10.7 depicts the output y_i of model M_i, $i = 1, 2$ against input (x_1, x_2).

Since the output y_1 and y_2 take binary values, 0 or 1, there are only four possible combinations, $\{(y_1, y_2)\} = \{(0, 0), (0, 1), (1, 0), (1, 1)\}$, among which only the pair $(1, 1)$ corresponds to Class 1, and

(a) M_1 (b) M_2

Figure 10.6: Decision boundaries and regions of M_1 and M_2.



(a) M_1 (b) M_2

Figure 10.7: Output of M_1 and M_2 against input (x_1, x_2).

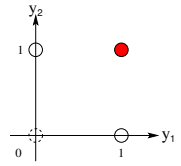


Figure 10.8: Illustration of y_1 - y_2 plane.

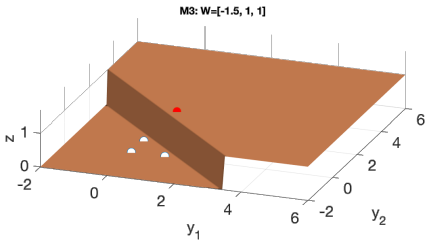


Figure 10.9: Output of M_3.

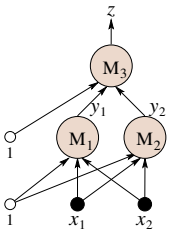


Figure 10.10: Structure of the multi-layer perceptron comprised of M_1, M_2, and M_3.

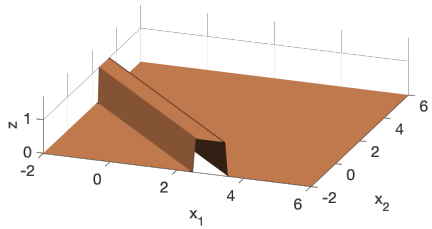


Figure 10.11: Output of the multi-layer perceptron.

$\{(0, 1), (1, 0)\}$ to Class 0.[2] See the plotting on y_1 - y_2 plane in Figure 10.8.

It is easy to see that the point $(1, 1)$ in y_1-y_2 plane can be separated from the other points by a single line, which can be done with another perceptron, say M_3, taking (y_1, y_2) as input, and giving z as output: $z(\dot{\mathbf{y}}) = g(\dot{\mathbf{w}}_3^T\dot{\mathbf{y}})$, where $\dot{\mathbf{y}} = (1, y_1, y_2)^T$. Figure 10.9 depicts the output z of M_3 against input (y_1, y_2), where the three points $(0, 0), (0, 1), (1, 0)$ are shown in white, and the other point $(1, 1)$ in red.

Figure 10.10 depicts the structure of the extended perceptron that consists of three basic perceptron M_1, M_2, and M_3. As you can see, it consists of two layers - the first layer consisting of M_1 and M_2, and the second (output) layer consisting of M_3. Note that the whole model is referred to as *multi-layer perceptron*, and each basic perceptron $(M_i, i = 1, 2, 3)$ is usually referred to as *cell, node* or *unit*.

This example indicates that multi-layer perceptrons can form complex decision boundaries and regions. Unfortunately, the original perceptron training algorithm is not applicable to multi-layer perceptrons, but we can find weight vectors if the decision boundaries are piece-wise linear and the geometry of the boundaries and regions is known. For example, the region of $y_1(\dot{\mathbf{x}}) = 1$ in Figure 10.6a is defined by the following inequality.

$$-0.5 + x_1 + x_2 \geq 0. \tag{10.23}$$

Thus we can use $(-0.5, 1, 1)^T$, or generally $(-0.5c, c, c)^T$, as the weight vector $\dot{\mathbf{w}}_1$ of M_1, where c is an arbitrary positive constant. The weight vectors of M_2 and M_3 can be found in the same manner.

[2] Note that we ignore the point $(0, 0)$, since there is no input (x_1, x_2) that corresponds to that point.

Exercises

1. Considering a two-class classification problem, where each class is modelled with a D-dimensional Gaussian distribution. Derive the formula for the decision boundary, and show that it is quadratic in $\mathbf{x}$.

2. Considering a classification problem of two classes, whose discriminant function takes the form, $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$.

   - Show that the decision boundary is a straight line when $D = 2$.
   - Show that the weight vector $\mathbf{w}$ is a normal vector to the decision boundary.

3. Regarding the perceptron weight update formula of Equation 10.22, show why the learning parameter η needs to be positive.

4. Design a multi-layer perceptron which classifies an input vector (x_1, x_2) into Class 1 if the input lies in the square (including the perimeter) whose four corners are $\{(1,1),(1,2),(2,2),(1,2)\}$, otherwise it classifies the input to Class 0. You should identify the structure of the perceptron and weight vectors.