Chapter 14
Machine Learning

Consider a robot that recognizes and grasps yellow objects (Fig. 14.1). It can use a color camera to identify yellow objects, but the objects will appear different in different environments, such as in sunlight, in a dark room, or in a showroom. Furthermore, it is hard to precisely define what "yellow" means: what is the boundary between yellow and lemon-yellow or between yellow and orange? Rather than write detailed instructions for the robot, we would prefer that the robot learn color recognition as it is performing the task, so that it could adapt to the environment where the task takes place. Specifically, we want to design a *classification algorithm* that can be *trained* to perform the task without supplying all the details in advance.

Classifications algorithms are a central topic in *machine learning*, a field of computer science and statistics that develops computations to recognize patterns and to predict outcomes without explicit programming. These algorithms extract *rules* from the raw data acquired by the system during a training period. The rules are subsequently used to classify a new object and then to take the appropriate action according to the class of the object. For the color-recognition task, we train the robot by presenting it with objects of different colors and telling the robot which objects are yellow and which are not. The machine learning algorithm generates a rule for color classification. When presented with new objects, it uses the rule to decide which objects are yellow and which are not.

The previous chapter presented artificial neural networks which perform a form of machine learning based upon *reinforcement*. In this chapter, we discuss statistical techniques based upon *supervised learning*: during the training period we tell the robot the precise answers, such as whether an object is yellow or not. Section 14.1 introduces the statistical techniques by developing an algorithm for distinguishing between objects of two colors. We present a technique for machine learning called *linear discriminant analysis (LDA)*. Sections 14.2 and 14.3 present LDA in the same context of distinguishing between two colors. LDA is based on the assumption that

M. Ben-Ari and F. Mondada, *Elements of Robotics*,
https://doi.org/10.1007/978-3-319-62533-1_14

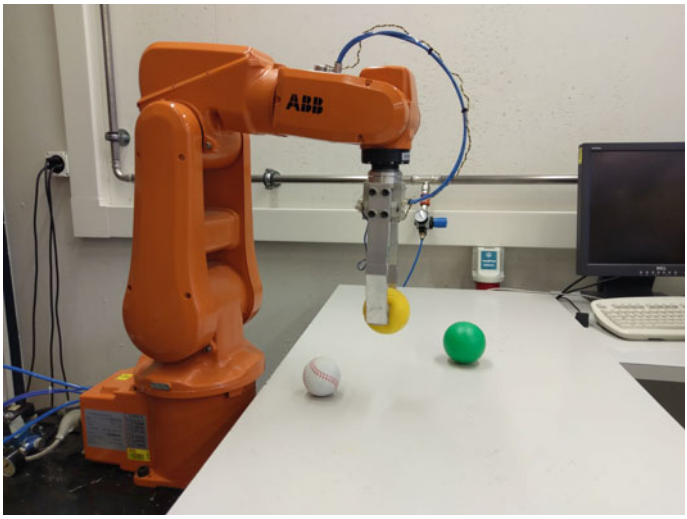

Fig. 14.1 Robotic arm sorting colored balls

the data has specific statistical properties; if these do not hold, *perceptrons* can be used for classification as described in Sect. 14.4.

This chapter assumes that you are familiar with the concepts of *mean*, *variance* and *covariance*. Tutorials on these concepts appear in Appendices B.3 and B.4.

14.1 Distinguishing Between Two Colors

We start with the problem of distinguishing yellow balls from non-yellow balls. To simplify the task, we modify the problem to one of distinguishing dark gray areas from light gray areas that are printed on paper taped to the floor (Fig. 14.2). The robot uses two ground sensors that sample the reflected light as the robot moves over the two areas.

Figure 14.3 shows a plot of the values returned by sampling the two sensors.[1] The robot takes about 70 s to move from left to right, sampling the reflected light once per second. It is easy to see that the data shows significant variation, which is probably due to noise in the sensors and uneven printing. Even more problematic is the variability in the results returned by the two sensors. How can the robot learn to distinguish between the shades of gray given the variability in the samples and the sensors? We want to automatically create a rule to distinguish between the two shades of gray.

[1] The data used in this chapter are real data taken from an experiment with the Thymio robot.

Fig. 14.2 Distinguishing two shades of *gray*

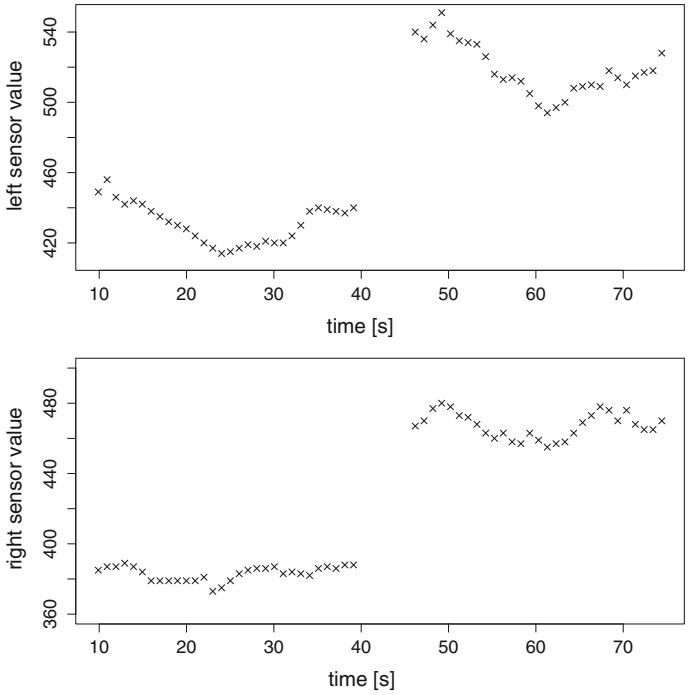

Fig. 14.3 Plots of reflected light versus time for the left sensor (*top*) and the right sensor (*bottom*)

14.1.1 A Discriminant Based on the Means

By examining the plots in Fig. 14.3, it is easy to see which samples are from the dark gray area and which are from the light gray area. For the left sensor, the values of the light gray area are in the range 500–550, while the values of the dark gray area are in the range 410–460. For the right sensor, the ranges are 460–480 and 380–400. For the left sensor, a threshold of 480 would clearly distinguish between light and dark gray, while for the right sensor a threshold of 440 would clearly distinguish between light and dark gray. But how can these optimal values be chosen automatically and how can we reconcile the thresholds of the two sensors?

Let us first focus on the left sensor. We are looking for a *discriminant*, a value that distinguishes samples from the two colors. Consider the values max_{dark}, the

maximum value returned by sampling dark gray, and min_{light}, the minimum value returned by sampling light gray. Under the reasonable assumption that $max_{dark} < min_{light}$, any value x such that $max_{dark} < x < min_{light}$ can distinguish between the two shades of gray. The midpoint between the two values would seem to offer the most robust discriminant.

From Fig. 14.3 we see that $max_{dark} \approx 460$ occurs at about 10 s and $min_{light} \approx 500$ occurs at about 60 s, so we choose their average 480 as the discriminant. While this is correct for this particular data set, in general it is not a good idea to use the maximum and minimum values because they could be *outliers*: extreme values resulting from unusual circumstances, such as a hole in the paper which would incorrectly return a very high value in the dark gray area.

A better solution is to use *all* the data and the simplest function of all the data is the *mean* of the values. Let μ_{dark} denote the mean of the dark gray samples and μ_{light} the mean of the light gray samples. A good discriminant Δ is the midpoint between the two means:

$$\Delta = \frac{\mu_{dark} + \mu_{light}}{2} .$$

For the data in Fig. 14.3 the means for the left sensor and the discriminant are[2]:

$$\mu_{dark}^{left} = 431, \quad \mu_{light}^{left} = 519, \quad \Delta^{left} = \frac{431 + 519}{2} = 475 .$$

A similar computation gives the discriminant for the right sensor:

$$\Delta^{right} = 425 .$$

In order to obtain optimal recognition, we want an algorithm that is able to automatically decide which of the two discriminants is better. This is a preliminary stepping stone to the method described in Sect. 14.2, where a discriminant is computed by combining data from both sensors.

Intuitively, the greater the difference between the means of the light and dark areas:

$$\left| \mu_{dark}^{left} - \mu_{light}^{left} \right| , \quad \left| \mu_{dark}^{right} - \mu_{light}^{right} \right| ,$$

the easier it will be to place a discriminant between the two classes. The difference between the means of the left sensor (88) is a bit greater than the difference between the means of the right sensor (84). This leads us to choose the discriminant (475) computed from the means of the left sensor. However, from the plot in Fig. 14.4 it appears that this might not be the best choice because of the large variation in the samples of the left sensor.

[2] In this chapter, values will be rounded to the nearest integer.

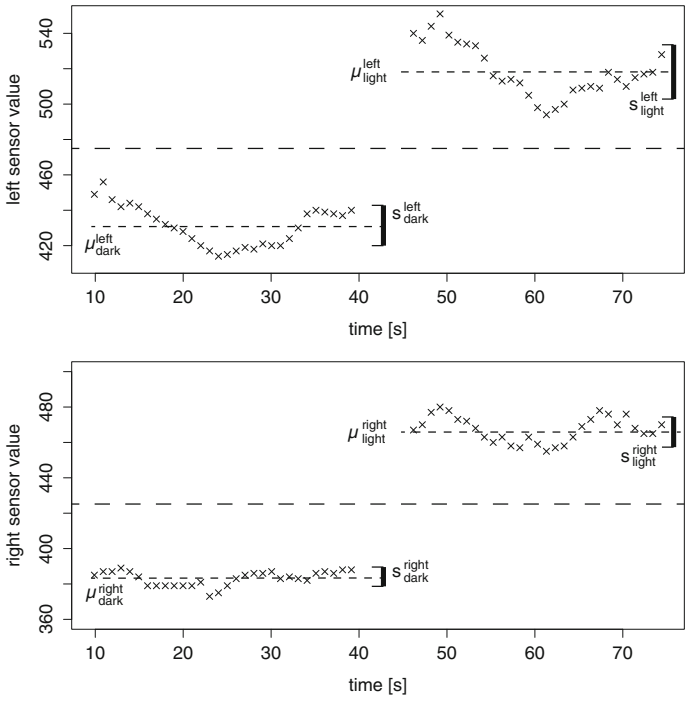

Fig. 14.4 Figure 14.3 with means (*short dashes*), variances (*brackets*), discriminants (*long dashes*)

14.1.2 A Discriminant Based on the Means and Variances

A better discriminant can be obtained if we consider not only the difference of the means but also the spread of the sample values around the mean. This is called the *variance* of the samples. The variance s^2 of a set of values $\{x_1, x_2, \ldots, x_{n-1}, x_n\}$ is[3]:

$$s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \mu)^2 ,$$

where μ is the mean of the values in the set.

The variance computes the average of the distances of each sample from the mean of the samples. The distances are squared because a sample can be greater than or less than the mean, but we want a positive distance that shows how far the sample is from the mean.

The brackets in Fig. 14.4 show the four variances for the sets of samples of the light and dark areas for the left and right sensors. The difference between the left

[3] Appendix B.3 explains why $n - 1$ is used instead of n.

means is somewhat greater than the difference between the right means:

$$\left| \mu_{dark}^{left} - \mu_{light}^{left} \right| > \left| \mu_{dark}^{right} - \mu_{light}^{right} \right| ,$$

but the variances of the right sensor are much smaller than the corresponding variances of the left sensor:

$$\left(s_{dark}^{right}\right)^2 \ll \left(s_{dark}^{left}\right)^2 , \quad \left(s_{light}^{right}\right)^2 \ll \left(s_{light}^{left}\right)^2 .$$

The use of the variances enables better classification of the two sets, since a sensor with less variance is more stable and this facilitates classification.

A good discriminant can be obtained by combining information from the means and the variances. The quality of a discriminant J_k, for $k = left, right$, is given by:

$$J_k = \frac{\left(\mu_{dark}^k - \mu_{light}^k\right)^2}{\left(s_{dark}^k\right)^2 + \left(s_{light}^k\right)^2} . \tag{14.1}$$

To maximize J, the numerator—the distance between the means—should be large, and the denominator—the variances of the samples—should be small.

Table 14.1 displays the computations for the data set from Fig. 14.4. The quality criterion J for the right sensor is much larger than the one for the left sensor. It follows that the midpoint between the means of the right sensor:

$$\Delta^{right} = \frac{383 + 467}{2} = 425$$

is a better discriminant than the midpoint of the means of the left sensor that would be chosen by considering only the differences of the means $|\mu_{dark} - \mu_{light}|$, which is slightly larger for the left sensor than for the right sensor.

Table 14.1 The difference of the means and the quality criteria J

	Left		Right	
	Dark	Light	Dark	Light
μ	431	519	383	467
s^2	11	15	4	7
$\lvert\mu_{dark} - \mu_{light}\rvert$		88		84
J		22		104

14.1.3 Algorithm for Learning to Distinguish Colors

These computations are done by the robot itself, so the choice of the better discriminant and the better sensor is automatic. The details of the computation are given Algorithms 14.1 and 14.2.[4]

There are two classes C_1, C_2 and two sensors. During the learning phase, the robot samples areas of the two gray levels independently and then computes the criterion of quality J. The sampling and computation are done for each sensor, either one after another or simultaneously. After the learning phase, the robot uses midpoint of the means with the best J value for recognition of gray levels.

Algorithm 14.1: Distinguishing classes (learning phase)

float $\mathbf{X_1}, \mathbf{X_2}$	// Sets of samples
float μ_1, μ_2	// Means of C_1, C_2
float s_1, s_2	// Variances of C_1, C_2
float $\mu[2]$	// Means of μ_1, μ_2
float $J[2]$	// Criteria of quality
integer k	// Index of max($J[1], J[2]$)

1: for sensor i=1, 2
2: Collect a set of samples $\mathbf{X_1}$ from C_1
3: Collect a set of samples $\mathbf{X_2}$ from C_2
4: Compute means μ_1 of $\mathbf{X_1}$ and μ_2 of $\mathbf{X_2}$
5: Compute variances s_1 of $\mathbf{X_1}$ and s_2 of $\mathbf{X_2}$
6: Compute the mean $\mu[i] = \dfrac{\mu_1 + \mu_2}{2}$
7: Compute the criterion $J[i]$ from Eq. 14.1
8: $k \leftarrow$ index of max($J[1], J[2]$)
9: Output $\mu[k]$

Algorithm 14.2: Distinguishing classes (recognition phase)

float $\mu \leftarrow$ input $\mu[k]$ from the learning phase	
float x	

1: loop
2: $x \leftarrow$ get new sample
3: if $x < \mu$
4: assign x to class C_1
5: else
6: assign x to class C_2

[4] Boldface variables represent vectors or matrices.

Activity 14.1: Robotic chameleon

- Construct an environment as shown in Fig. 14.2. Print two pieces of paper with different uniform gray levels and tape them to the floor.
- Write a program that causes the robot to move at a constant speed over the area of one color and sample the reflected light periodically. Repeat for the other color.
- Plot the data, compute the means and the discriminant.
- Implement a program that classifies the measurements of the sensor. When the robot classifies a measurement it displays which color is recognized like a chameleon (or gives other feedback if changing color cannot be done).
- Apply the same method with a second sensor and compare the separability of the classes using the criterion J.
- Repeat the exercise with two very close levels of gray. What do you observe?

14.2 Linear Discriminant Analysis

In the previous section we classified samples of two levels of gray based on the measurements of one sensor out of two; the sensor was chosen automatically based on a quality criterion. This approach is simple but not optimal. Instead of choosing a discriminant based on one sensor, we can achieve better recognition by combining samples from both sensors. One method is called *linear discriminant analysis (LDA)* and is based upon pioneering work in 1936 by the statistician Ronald A. Fisher.

14.2.1 Motivation

To understand the advantages of combining samples from two sensors, suppose that we need to classify objects of two colors: *electric violet (ev)* and *cadmium red (cr)*. Electric violet is composed largely of blue with a bit of red, while cadmium red is composed largely of red with a bit of blue. Two sensors are used: one measures the level of red and the other measures the level of blue. For a set of samples, we can compute their means μ_j^k and variances $(s_j^k)^2$, for $j = ev, cr$, $k = blue, red$.

The left plot in Fig. 14.5 shows samples of the electric violet objects contained within a dashed ellipse at the upper left and samples of the cadmium red objects contained within a dashed ellipse at the lower right. The centers of the ellipses are the means because the samples are symmetrically distributed with respect to the ellipses. The y-coordinate of μ_{ev} is the mean of the samples of the electric violet objects by the blue sensor and its x-coordinate is the mean of the samples of these

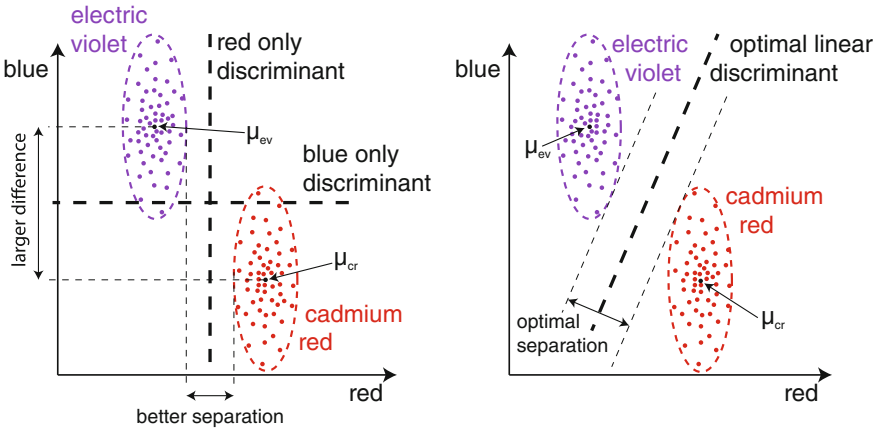

Fig. 14.5 An optimal linear discriminant for distinguishing two colors

objects by the red sensor. The means of the sensors when sampling the cadmium red objects are similarly displayed. It is easy to see that the electric violet objects have more blue (they are higher up the y-axis), while the cadmium red objects have more red (they are further out on the x-axis).

From the diagram we see that there is a larger difference between the means for the blue sensor than between the means for the red sensor. At first glance, it appears that using the blue sensor only would give a better discriminant. However, this is not true: the dashed lines show that the red-only discriminant completely distinguishes between electric violet and cadmium red, while the blue-only discriminant falsely classifies some electric violet samples as cadmium red (some samples are below the line) and falsely classifies some cadmium red samples as electric violet (some samples are above the line).

The reason for this unintuitive result is that the blue sensor returns values that are widely spread out (have a large variance), while the red sensor returns values that are narrowly spread out (have a small variance), and we saw in Sect. 14.1 that classification is better if the variance is small. The right plot in Fig. 14.5 shows that by constructing a discriminant from both sensors it is possible to better separate the electric violet objects from the cadmium red objects. The discriminant is still linear (a straight line) but its slope is no longer parallel to one of the axes. This line is computed using the variances as well as the means. The method is called linear discriminant analysis because the discriminant is linear.

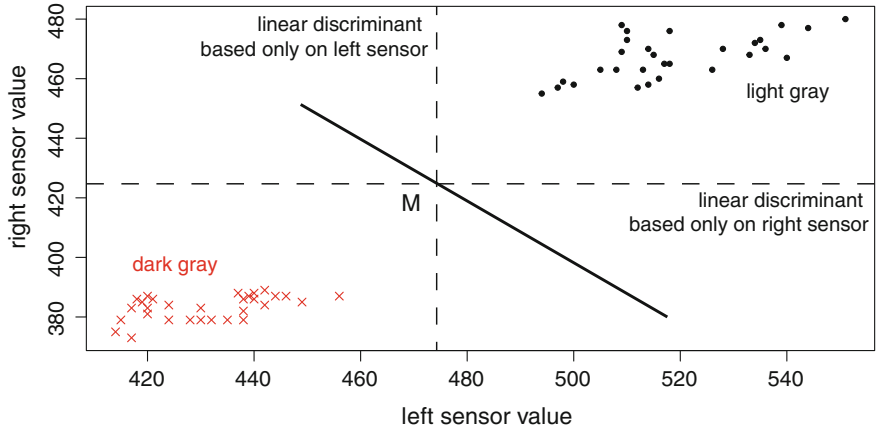

Fig. 14.6 x–y plot of *gray* levels with single-sensor discriminants and an optimal discriminant

14.2.2 The Linear Discriminant

Figure 14.5 is an x–y plot of data sampled from two sensors. Each value is plotted at the point whose x-coordinate is the value returned by the red sensor and whose y-coordinate is the value returned by the blue sensor. Similarly, Fig. 14.6 is an x–y plot of the data from Figs. 14.3 and 14.4 that were collected as the robot moved over two gray areas. In those graphs, the sensor values were plotted as function of time, but time has no role in classification except to the link samples that were measured at the same time by the two sensors.

In Fig. 14.6, classification based only on the left sensor corresponds to the vertical dashed line, while classification based only on the right sensor corresponds to the horizontal dashed line. Both the horizontal and vertical separation lines are not optimal. Suppose that classification based on the left sensor (the vertical line) is used and consider a sample for which the left sensor returns 470 and the right sensor returns 460. The sample will be classified as dark gray even though the classification as light gray is better. Intuitively, it is clear that the solid diagonal line in the graph is a far more accurate discriminant than either of the two discriminants based on a single sensor.

How can such a linear discriminant be defined mathematically so that it can be discovered automatically?[5] The general equation for a line in the plane is $y = mx+a$, where m is the slope and a is the intersect of the line and the y-axis when $x = 0$. Another equation for a line is:

$$w_1 x_1 + w_2 x_2 = c\,, \tag{14.2}$$

[5]The following presentation is abstract and will be easier to understand if read together with the numerical example in Sect. 14.2.5.

where x_1 is the horizontal axis for the left sensor values, x_2 is the vertical axis for the right sensor values. c is a constant and w_1, w_2 are coefficients of the variables.

It is convenient to represent the coefficients as a column vector:

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}.$$

In this representation the vector $\mathbf{w}$ is normal to the discriminant line and therefore defines its slope, while the constant c enables the discriminant to be any line of that slope, that is, any one of the infinite number of parallel lines with a given slope. Once the slope of $\mathbf{w}$ is determined, c is obtained by entering a given point (x_1,x_2) into Eq. 14.2.

Linear discriminant analysis automatically defines the vector $\mathbf{w}$ and constant c that generates an optimal discriminant line between the data sets of the two classes. The first step is to choose a point on the discriminant line. Through that point there are an infinite number of lines and we have to choose the line whose slope gives the optimal discriminant. Finally, the value c can be computed from the slope and the chosen point. The following subsections describe each of these steps in detail.

14.2.3 Choosing a Point for the Linear Discriminant

How can we choose a point? LDA is based upon the *assumption* that the values of both classes have the same distribution. Informally, when looking at an x–y plot, both sets of points should have similar size and shape. Although the distributions will almost certainly not be exactly the same (say a Gaussian distribution) because they result from measurements in the real world, since both sensors are subject to the same types of variability (sensor noise, uneven floor surface) it is likely that they will be similar.

If the two distributions are similar, the means of the samples of each sensor will be more-or-less in the same place with respect to the distributions. The average of the means for each sensor will be equidistant from corresponding points in the data sets. The discriminant is chosen to be some line that passes through the point M (Fig. 14.6) whose coordinates are:

$$\left(\frac{\mu_{light}^{left} + \mu_{dark}^{left}}{2}, \frac{\mu_{light}^{right} + \mu_{dark}^{right}}{2} \right).$$

14.2.4 Choosing a Slope for the Linear Discriminant

Once we have chosen the point M on the discriminant line, the next step is to choose the slope of the line. From Fig. 14.6, we see that there are infinitely many lines

through the point M that would distinguish between the two sets of samples. Which is the best line based on the statistical properties of our data?

In Sect. 14.1 we looked for a way to decide between two discriminants, where each discriminant was a line parallel to the y-axis at the midpoint between the means of the values returned by a sensor (Fig. 14.4). The decision was based on the quality criterion J_k (Eq. 14.1, repeated here for convenience):

$$J_k = \frac{\left(\mu^k_{dark} - \mu^k_{light}\right)^2}{\left(s^k_{dark}\right)^2 + \left(s^k_{light}\right)^2}, \tag{14.3}$$

where $k = left, right$. The discriminant with the larger value of J_k was chosen. To maximize J_k, the numerator—the distance between the means—should be large, and the denominator—the variances of the samples—should be small.

Now, we no longer want to compute a quality criterion based on the values returned by each sensor separately, but instead to compute a criterion from all the values returned by both sensors. Figure 14.7 shows an x–y plot of the values of two sensors, where the values of the two classes are represented by red squares and blue circles. Clearly, there is significant overlap along the x or y axes separately and it is impossible to find lines parallel to the axes that can distinguish between the groups. However, if we project the groups of measurements onto the line defined by a vector:

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix},$$

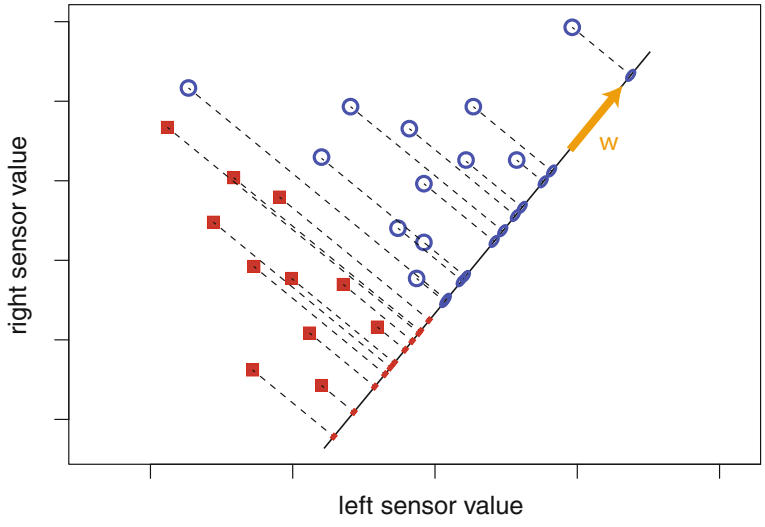

Fig. 14.7 Projection of the samples of two classes onto a line defined by **w**

the two groups can be distinguished by the separator line defined by Eq. 14.2:

$$w_1 x_1 + w_2 x_2 = c\,,$$

where c is defined by a point on the projection line situated between the red and blue points. In analogy with Eq. 14.3, we need to define a quality criterion $J(\mathbf{w})$, such that larger values of $J(\mathbf{w})$ give better discriminant lines. Then, we need to find the value of $\mathbf{w}$ that maximizes $J(\mathbf{w})$ by differentiating this function, setting the derivative to zero and solving for $\mathbf{w}$.

The definition of $J(\mathbf{w})$ is based on the means and variances of the two classes but is too complex for this book. We give without proof that the value of $\mathbf{w}$ that maximizes $J(\mathbf{w})$ is:

$$\mathbf{w} = \mathbf{S}^{-1}\left(\boldsymbol{\mu}_{light} - \boldsymbol{\mu}_{dark}\right), \tag{14.4}$$

where:

$$\boldsymbol{\mu}_{light} = \begin{bmatrix} \mu_{light}^{left} \\ \mu_{light}^{right} \end{bmatrix}, \quad \boldsymbol{\mu}_{dark} = \begin{bmatrix} \mu_{dark}^{left} \\ \mu_{dark}^{right} \end{bmatrix}$$

are the mean vectors of the two classes and $\mathbf{S}^{-1}$ is the inverse of the average of the covariance matrices of the two classes[6]:

$$\mathbf{S} = \frac{1}{2}\left(\begin{bmatrix} s^2\left(\mathbf{x}_{light}^{left}\right) & cov\left(\mathbf{x}_{light}^{left}, \mathbf{x}_{light}^{right}\right) \\ cov\left(\mathbf{x}_{light}^{right}, \mathbf{x}_{light}^{left}\right) & s^2\left(\mathbf{x}_{light}^{right}\right) \end{bmatrix} + \begin{bmatrix} s^2\left(\mathbf{x}_{dark}^{left}\right) & cov\left(\mathbf{x}_{dark}^{left}, \mathbf{x}_{dark}^{right}\right) \\ cov\left(\mathbf{x}_{dark}^{right}, \mathbf{x}_{dark}^{left}\right) & s^2\left(\mathbf{x}_{dark}^{right}\right) \end{bmatrix}\right).$$

Compared with the single-sensor J_k, the means are two-dimensional vectors because we have one mean for each of the sensors. The sum of variances becomes the covariance matrix, which takes into account both the individual variances of the two sensors and the covariances that express how the two sensors are related.

When the values of M and $\mathbf{w}$ have been computed, all that remains is to compute the constant c to fully define the discriminant line. This completes the learning phase of the LDA algorithm. In the recognition phase, the robot uses the line defined by $\mathbf{w}$ and c for classifying new samples.

The computation is formalized in Algorithms 14.3 and 14.4, where we want to distinguish between two classes C_1, C_2 using two sensors. Compare this algorithm with Algorithm 14.1: the two sets of samples $\mathbf{X_1}$, $\mathbf{X_2}$ and the means $\boldsymbol{\mu_1}$, $\boldsymbol{\mu_2}$ and variances $\mathbf{s_1}$, $\mathbf{s_2}$ are vectors with two elements, one element for each of the sensors.

[6] See Appendix B.4 for the definition of covariance and Sect. 14.2.5 for a numerical example that will clarify the computations.

Algorithm 14.3: Linear discriminant analysis (learning phase)	
float array[n_1,2] $\mathbf{X_1}$	// Sets of samples of first area
float array[n_2,2] $\mathbf{X_2}$	// Sets of samples of second area
float array[2] $\boldsymbol{\mu_1}, \boldsymbol{\mu_2}$	// Means of C_1, C_2
float array[2] $\boldsymbol{\mu}$	// Mean of the means
float array[2] $\mathbf{s_1}, \mathbf{s_2}$	// Variances of C_1, C_2
float cov_1, cov_2	// Covariances of C_1, C_2
float array[2] $\mathbf{S_{inv}}$	// Inverse of average
float c	// Constant of the line
1: Collect a set of samples $\mathbf{X_1}$ from C_1	
2: Collect a set of samples $\mathbf{X_2}$ from C_2	
3: Compute means $\boldsymbol{\mu_1}$ of $\mathbf{X_1}$ and $\boldsymbol{\mu_2}$ of $\mathbf{X_2}$	
4: $\boldsymbol{\mu} \leftarrow (\boldsymbol{\mu_1} + \boldsymbol{\mu_2})/2$	
5: Compute variances $\mathbf{s_1}$ of $\mathbf{X_1}$ and $\mathbf{s_2}$ of $\mathbf{X_2}$	
6: Compute covariances cov_1, cov_2 of $\mathbf{X_1}$ and $\mathbf{X_2}$	
7: Compute $\mathbf{S_{inv}}$ of covariance matrix	
8: Compute $\mathbf{w}$ from Eq. 14.4	
9: Compute point M from $\boldsymbol{\mu}$	
10: Compute c from M and $\mathbf{w}$	
11: Output $\mathbf{w}$ and c	

Algorithm 14.4: Linear discriminant analysis (recognition phase)
float $\mathbf{w} \leftarrow$ input from the learning phase
float $c \leftarrow$ input from the learning phase
float $\mathbf{x}$
1: loop
2: $\mathbf{x} \leftarrow$ new sample
3: if $\mathbf{x} \cdot \mathbf{w} < c$
4: assign $\mathbf{x}$ to class C_1
5: else
6: assign $\mathbf{x}$ to class C_2

14.2.5 Computation of a Linear Discriminant: Numerical Example

Figure 14.8 is a plot of the samples from two ground sensors measured as a robot moves over two very similar gray areas, one that is 83.6% black and the other 85% black. The levels are so close that the human eye cannot distinguish between them. Can the linear discriminant computed by Algorithm 14.3 do better?

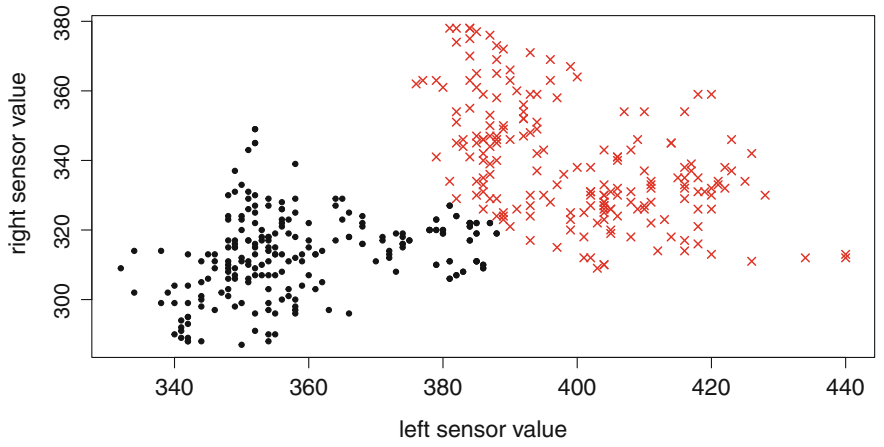

Fig. 14.8 Similar *gray* areas (*black dots* from the darker area, *red x'es* from the lighter area)

Class C_1 is the light gray area and class C_2 is the dark gray area. The elements of the sets of vectors $\mathbf{X_1}[n_1]$, $\mathbf{X_2}[n_2]$ are vectors:

$$\mathbf{x} = \begin{bmatrix} x^{left} \\ x^{right} \end{bmatrix}$$

of samples measured by the left and right sensors, where $n_1 = 192$ and $n_2 = 205$.

First we compute the means of the data of Fig. 14.8:

$$\boldsymbol{\mu_1} = \frac{1}{192}\left(\begin{bmatrix} 389 \\ 324 \end{bmatrix} + \begin{bmatrix} 390 \\ 323 \end{bmatrix} + \ldots + \begin{bmatrix} 389 \\ 373 \end{bmatrix}\right) \approx \begin{bmatrix} 400 \\ 339 \end{bmatrix}$$

$$\boldsymbol{\mu_2} = \frac{1}{205}\left(\begin{bmatrix} 358 \\ 297 \end{bmatrix} + \begin{bmatrix} 358 \\ 296 \end{bmatrix} + \ldots + \begin{bmatrix} 352 \\ 327 \end{bmatrix}\right) \approx \begin{bmatrix} 357 \\ 312 \end{bmatrix}.$$

More samples were taken from the second area (205) than from the first area (192), but that is not important for computing the means. As expected, the means $\boldsymbol{\mu_1}$ of the samples from the light gray area are slightly higher (more reflected light) than the means $\boldsymbol{\mu_2}$ of the samples from the dark gray areas. However, the left sensor consistently measures higher values than the right sensor. Figure 14.9 shows the data from Fig. 14.8 with thin dashed lines indicating the means. There are two lines for each area: the horizontal line for the right sensor and the vertical line for the left sensor.

The covariance matrix is composed from the variances of the two individual sensors and their covariance. For $i = 1, 2$:

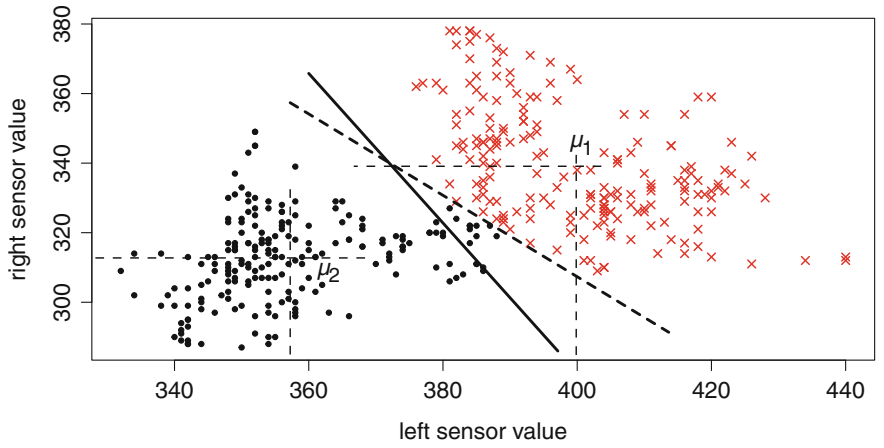

Fig. 14.9 Means for each class (*thin dashed lines*), LDA discriminant line (*solid line*), discriminant line that fully distinguishes the classes (*thick dashed line*)

$$\mathbf{S}_i = \begin{bmatrix} s^2\left(X_i^{left}\right) & cov\left(X_i^{left}, X_i^{right}\right) \\ cov\left(X_i^{right}, X_i^{left}\right) & s^2\left(X_i^{right}\right) \end{bmatrix}.$$

For the data from Fig. 14.8, the variances of the samples of the light gray area are:

$$s^2(X_1^{left}) = \frac{1}{191}\left((389-400)^2 + (390-400)^2 + \cdots + (389-400)^2\right) \approx 187$$

$$s^2(X_1^{right}) = \frac{1}{191}\left((324-339)^2 + (323-339)^2 + \ldots + (373-339)^2\right) \approx 286\,.$$

Equation B.4 from Appendix B.4 is used to compute the covariance. Since covariance is symmetric, the value need be computed only once.

$$cov(X_1^{left}, X_1^{right}) = \frac{1}{191}\left((389-400)(324-339) + \ldots + (389-400)(373-339)\right)$$
$$\approx -118\,.$$

Putting the results together and performing a similar computation for X_2 gives the covariance matrices:

$$\mathbf{S}_1 = \begin{bmatrix} 187 & -118 \\ -118 & 286 \end{bmatrix} \qquad \mathbf{S}_2 = \begin{bmatrix} 161 & 44 \\ 44 & 147 \end{bmatrix}.$$

The next step is to compute the mean of the covariance matrices:

$$\mu_{\mathbf{S}} = \frac{1}{2}\left(\begin{bmatrix} 187 & -118 \\ -118 & 286 \end{bmatrix} + \begin{bmatrix} 161 & 44 \\ 44 & 147 \end{bmatrix}\right) = \begin{bmatrix} 174 & -38 \\ -37 & 216 \end{bmatrix},$$

and to find its inverse[7]:

$$\mathbf{S}^{-1} = \begin{bmatrix} 0.006 & 0.001 \\ 0.001 & 0.005 \end{bmatrix}.$$

We can now use Eq. 14.4 to compute $\mathbf{w}$:

$$\mathbf{w} = \begin{bmatrix} 0.006 & 0.001 \\ 0.001 & 0.005 \end{bmatrix} \cdot \left(\begin{bmatrix} 400 \\ 339 \end{bmatrix} - \begin{bmatrix} 357 \\ 313 \end{bmatrix}\right) = \begin{bmatrix} 0.28 \\ 0.17 \end{bmatrix}.$$

The vector $\mathbf{w}$ gives the direction of the projection line which is perpendicular to discriminant line. We now use Eq. 14.2, repeated here:

$$w_1 x_1 + w_2 x_2 = c$$

to compute the constant c, assuming we know the coordinate (x_1, x_2) of some point. But we specified that the midpoint between the means must be on the discriminant line. Its coordinates are:

$$\boldsymbol{\mu} = \frac{1}{2}(\boldsymbol{\mu_1} + \boldsymbol{\mu_2}) = \frac{1}{2}\left(\begin{bmatrix} 400 \\ 339 \end{bmatrix} + \begin{bmatrix} 357 \\ 313 \end{bmatrix}\right) \approx \begin{bmatrix} 379 \\ 326 \end{bmatrix}.$$

Therefore:

$$c = 0.28 \cdot 379 + 0.17 \cdot 326 \approx 162\,,$$

and the equation of the discriminant line is:

$$0.28x_1 + 0.17x_2 = 162\,.$$

The discriminant line is shown as a solid line in Fig. 14.9. Given a new sample (a, b), compare the value of $0.28a + 0.17b$ to 162: If it is larger, the sample is classified as belonging to class C_1 and if it is is smaller, the sample is classified as belong to class C_2.

[7] See Appendix B.5.

14.2.6 Comparing the Quality of the Discriminants

If we compare the linear discriminant found above with the two simple discriminants based upon the means of a single sensor, we see a clear improvement. Because of the overlap between the classes in a single direction, the simple discriminant for the right sensor correctly classifies only 84.1% of the samples, while the simple discriminant for the left sensor is somewhat better, classifying 93.7% of samples correctly. The linear discriminant found using LDA is better, correctly classifying 97.5% of the samples.

It might be surprising that there are discriminant lines that can correctly classify *all* of the samples! One such discriminant is shown by the thick dashed line in Fig. 14.9. Why didn't LDA find this discriminant? LDA assumes both classes have a similar distribution (spread of values) around the mean and the LDA discriminant is optimal under this assumption. For our data, some points in the second class are far from the mean and thus the distributions of the two classes are slightly different. It is hard to say if these samples are outliers, perhaps caused by problem when printing the gray areas on paper. In that case, it is certainly possible that subsequent sampling of the two areas would result in distributions that are similar to each other, leading to the correct classification by the LDA discriminant.

14.2.7 Activities for LDA

Activities for LDA are collected in this section.

Activity 14.2: Robotic chameleon with LDA

- Construct an environment as shown in Fig. 14.2 but with two gray levels very similar to each other.
- Write a program that causes the robot to move at a constant speed over the area of one color and sample the reflected light periodically. Repeat for the other color.
- Plot the data.
- Compute the averages, the covariance matrices and the discriminant.
- Implement a program that classifies measurements of the sensor. When the robot classifies a measurement it displays which color is recognized (or gives other feedback if changing color cannot be done).

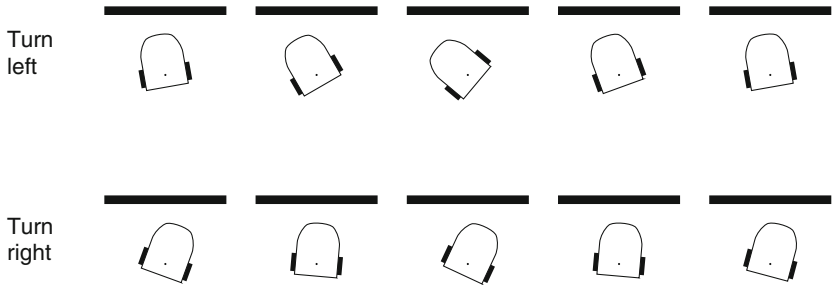

Fig. 14.10 Training for obstacle avoidance

Activity 14.3: Obstacle avoidance with two sensors

- Figure 14.10 shows a robot approaching a wall. The upper part of the diagram shows various situations where the robot detects the wall with its right sensors; therefore, it should turn left to move around the wall. Similarly, in the lower part of the diagram the robot should turn right.
- Write a program that stores the sensor values from both the right and left sensors when a button is pressed. The program also stores the identity of which button was pressed; this represents the class we are looking for when doing obstacle avoidance.
- Train the robot: Place the robot next to a wall and run the program. Touch the left button if the robot should turn left or the right button if the robot should turn right. Repeat many times.
- Plot the samples from the two sensors on an x–y graph and group them by class: turn right or left to avoid of the wall. You should obtain a graph similar to the one in Fig. 14.11.
- Draw a discriminant line separating the two classes.
- How successful is your discriminant line? What percentage of the samples can it successfully classify?
- Compute the optimal discriminant using LDA. How successful is it? Do the assumptions of LDA hold?

Activity 14.4: Following an object

- Write a program that causes the robot to follow an object. The robot moves forward if it detects the object in front; it moves backwards if it is too close to the object. The robot turns right if the object is to its right and the robot turns left if the object is to its left.

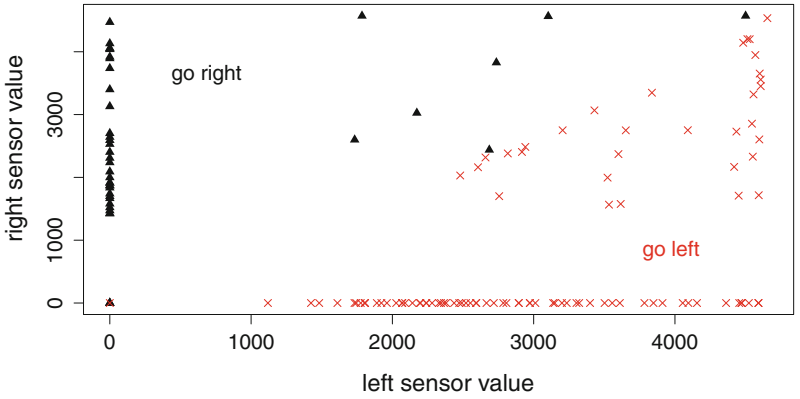

Fig. 14.11 Obstacle avoidance data from the class "go left" (*red x'es*) and the class "go right" (*black triangles*)

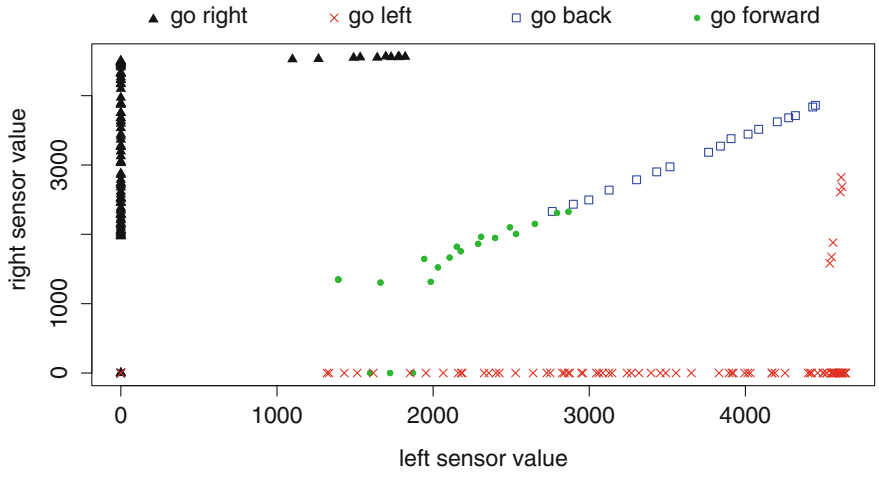

Fig. 14.12 Data acquired in a learning phase for following an object

- Use two sensors so we can visualize the data on an x–y plot.
- Acquire and plot the data as in Activity 14.3. The plot should be similar to the one shown in Fig. 14.12.
- Explain the classifications in Fig. 14.12. What is the problem with classifying a sample as going forwards or going backwards? Why do the samples for going forwards and backwards have different values for the left and right sensors?
- Suggest an algorithm for classifying the four situations. Could you use a combination of linear separators?

14.3 Generalization of the Linear Discriminant

In this section we point out some ways in which LDA can be extended and improved.

First, we can have more sensors. The mathematics becomes more complex because with n sensors, the vectors will have n elements and the covariance matrix will have $n \times n$ elements, requiring more computing power and more memory. Instead of a discriminant line, the discriminant will be an $n - 1$ dimension hyperplane. Classification with multiple sensors is used with electroencephalography (EEG) signals from the brain in order to control a robot by thought alone.

Activity 14.4 demonstrated another generalization: classification into more than two classes. Discriminants are used to classify between each pair of classes. Suppose you have three classes C_1, C_2, and C_3, and discriminants Δ_{12}, Δ_{13}, Δ_{23}. If a new sample is classified in class C_2 by Δ_{12}, in class C_1 by Δ_{12}, and in class C_2 by Δ_{23}, the final classification will be into class C_2 because more discriminants assign the sample to that class.

A third generalization is to use a higher order curve instead of a straight line, for example, a quadratic function. A higher order discriminant can separate classes whose data sets are not simple clusters of samples.

14.4 Perceptrons

LDA can distinguish between classes only under the assumption that the samples have similar distributions in the classes. In this section, we present another approach to classification using *perceptrons* which are related to neural networks (Chap. 13). There we showed how learning rules can generate specified behaviors linking sensors and motors; here we show how they can be used to classify data into classes.

14.4.1 Detecting a Slope

Consider a robot exploring difficult terrain. It is important that the robot identify steep slopes so it won't fall over, but it is difficult to specify in advance all dangerous situations since these depend on characteristics such as the geometry of the ground and its properties (wet/dry, sand/mud). Instead, we wish to train the robot to adapt its behavior in different environments.

To simplify the problem, assume that the robot can move just forwards and backwards, and that it has accelerometers on two axes *relative to the body of the robot*: one measures acceleration forwards and backwards, and the other measures acceleration upwards and downwards. A robot that is stationary on a level surface will measure zero acceleration forwards and backwards, and an downwards acceleration of 9.8 m/sec^2 due to gravity. Gravitational acceleration is relatively strong compared

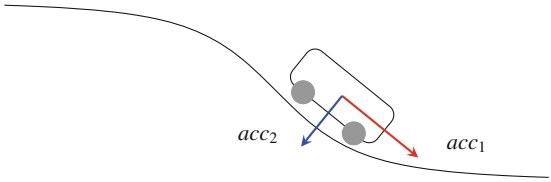

Fig. 14.13 A robot with accelerometers moving in difficult terrain

with the acceleration of a slow-moving robot, so the relative values of the accelerometer along the two axes will give a good indication of its attitude.

Figure 14.13 shows a robot moving forwards on a slope. The values returned by both accelerometers are similar, $acc_1 \approx acc_2$, so we can infer that the robot is on a slope. If $acc_2 \gg acc_1$, we would infer that the robot is on level ground because the up/down accelerometer measures the full force of gravity while the front/back accelerometer measures only the very small acceleration of the moving robot. The task is to distinguish between a safe position of the robot and one in which the slope starts to become too steep so that the robot is in danger of falling.

Figure 14.14 shows data acquired during a training session as the robot moves down a slope. When the operator of the training session determines that the robot is stable (class C_1), she initiates a measurement indicated by a red $\times$; when she determines that the robot is in a dangerous situation (class C_2), she initiates a measurement indicated by a black triangle. The challenge is to find a way of distinguishing samples of the two classes.

The dashed lines in the figure show the means for the two data sets. It is clear that they do not help us classify the data because of the large overlap between the two sets of samples. Furthermore, LDA is not appropriate because there is no similarity in

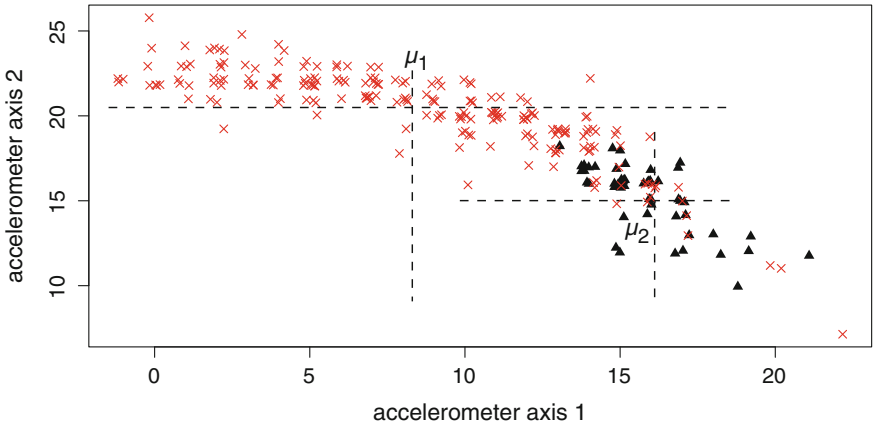

Fig. 14.14 Detecting a dangerous slope using data from the accelerometers

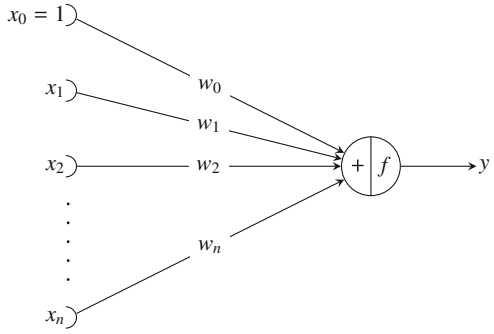

Fig. 14.15 A perceptron

the distributions: samples when the robot is stable appear in many parts of the plot, while samples from dangerous situations are concentrated in a small area around their mean.

14.4.2 Classification with Perceptrons

A *perceptron* is an artificial neuron with a specific structure (Fig. 14.15). It has a summation unit with inputs $\{x_1, \ldots, x_n\}$ and each input x_i is multiplied by a factor w_i before summation. An additional input x_0 has the constant value 1 for setting a bias independent of the inputs. The output of the perceptron is obtained by applying a function f to the result of the addition.

When used as a classifier, the inputs to the perceptron are the values returned by the sensors for a sample to be classified and the output will be one of two values that indicate the class to which the sample is assigned. Usually, the output function f is just the sign of the weighted sum:

$$y = sign\left(\sum_{i=0}^{n} w_i\, x_i\right) = \pm 1\,, \tag{14.5}$$

where one class corresponds to $+1$ and the other to -1.

The data are normalized so that all inputs are in the same range, usually $-1 \leq x_i \leq +1$. The data in Fig. 14.14 can be normalized by dividing each value by 30.

Given a set of input values $\{x_0 = 1, x_1, \ldots, x_n\}$ of a sample, the object of a training session is to find a set of weights $\{w_0, w_1, \ldots, w_n\}$ so that the output will be the value ± 1 that assigns the sample to the correct class.

If the sample is close to the border between two classes, the weighted sum will be close to zero. Therefore, a perceptron is also a linear classifier: to distinguish between outputs of ± 1, the discriminant dividing the two classes is defined by the

set of weights giving an output of zero:

$$\sum_{i=0}^{n} w_i \, x_i = 0 \,,$$

or

$$w_0 + w_1 x_1 + \cdots + w_n x_n = 0 \,. \tag{14.6}$$

The presentation of LDA for a two-dimensional problem ($n = 2$) led to Eq. 14.2, which is the same as Eq. 14.6 when $c = -w_0$. The difference between the two approaches is in the way the weights are obtained: in LDA statistics are used while for perceptrons they result from an iterative learning process.

14.4.3 Learning by a Perceptron

The iterative search for values of the weights $\{w_0, w_1, \ldots, w_n\}$ starts by setting them to a small value such as 0.1. During the learning phase, a set of samples is presented to the perceptron, together with the expected output (the class) for each element of the set. The set of samples must be constructed randomly and include elements from all the classes; furthermore, the elements from a single class must also be chosen randomly. This is to prevent the learning algorithm from generating a discriminant that is optimal in one specific situation, and to ensure that the process converges rapidly to an overall optimal discriminant, rather than spending too much time optimizing for specific cases.

The adjustment of the weights is computed as follows:

$$w_i(t+1) = w_i(t) + \eta \, x_i \, y \,, \quad 0 \leq i \leq n \,. \tag{14.7}$$

This is essentially the Hebbian rule for ANNs (Sect. 13.5.2). $w_i(t)$ and $w_i(t+1)$ are the i'th weights before and after the correction, η defines the learning rate, x_i is the normalized input, and y is the desired output. Since the sign function is applied to the sum of the weighted inputs, y is 1 or -1, except on the rare occasions where the sum is exactly zero.

Equation 14.7 corrects the weights by adding or subtracting a value that is proportional to the input, where the coefficient of proportionality is the learning rate. A small value for the learning rate means that the corrections to the weights will be in small increments, while a high learning rate will cause the corrections to the weights to be in larger increments. Once learning is completed, the weights are used to classify subsequent samples.

Algorithms 14.5 and 14.6 are a formal description of classification by perceptrons. The constant N is the size of the set of samples for the learning phase, while n is the number of sensor values returned for each sample.

Algorithm 14.5: Classification by a perceptron (learning phase)

```
float array[N,n] X                          // Set of samples
float array[n + 1] w ← [0.1, 0.1, ...]      // Weights
float array[n] x                            // Random sample
integer c                                   // Class of the random sample
integer y                                   // Output of the perceptron
1: loop until learning terminated
2:     x ← random element of X
3:     c ← class to which x belongs
4:     y ← output according to Eq. 14.5
5:     if y does not correspond to class c
6:         adjust w_i according to Eq. 14.7
7: Output w
```

Algorithm 14.6: Classification by a perceptron (recognition phase)

```
float w ← weights from the learning phase
float x
integer y
1: loop
2:     x ← new sample
3:     y ← output of perceptron for x, w
4:     if y = 1
5:         assign x to class C_1
6:     else if y = −1
7:         assign x to class C_2
```

When should the learning phase be terminated? One could specify an arbitrary value, for example: terminate the learning phase when 98% of the samples are classified correctly. However, it may not be possible to achieve this level. A better method is to terminate the learning phase when the magnitudes of the corrections to the weights become small.

14.4.4 Numerical Example

We return to the robot that is learning to avoid dangerous slopes and apply the learning algorithm to the data in Fig. 14.14. The perceptron has three inputs: x_0 which always set to 1, x_1 for the data from the front/back accelerometer, and x_2 for the data from the up/down accelerometer. The data is normalized by dividing each sample by 30 so that values will be between 0 and 1. We specify that an output of 1 corresponds to class C_1 (stable) and an output of -1 corresponds to class C_2 (dangerous).

Select a random sample from the input data, for example, a sample in class C_1 whose sensor values are $x_1 = 14$ and $x_2 = 18$. The normalized input is $x_1 = 14/30 = 0.47$ and $x_2 = 18/30 = 0.6$. The output of the perceptron with initial weights 0.1 is:

$$\begin{aligned} y &= sign(w_0 \times 1 + w_1 x_1 + w_2 x_2) \\ &= sign(0.1 \times 1 + 0.1 \times 0.47 + 0.1 \times 0.6) \\ &= sign(0.207) \\ &= 1\,. \end{aligned}$$

This output is correct so the weights need not be corrected. Now choose a random sample in class C_2 whose sensor values are $x_1 = 17$ and $x_2 = 15$. The normalized input is $x_1 = 17/30 = 0.57$ and $x_2 = 15/30 = 0.5$. The output of the perceptron is:

$$\begin{aligned} y &= sign(w_0 \times 1 + w_1 x_1 + w_2 x_2) \\ &= sign(0.1 \times 1 + 0.1 \times 0.57 + 0.1 \times 0.5) \\ &= sign(0.207) \\ &= 1\,. \end{aligned}$$

This output is not correct: the sample is from class C_2 which corresponds to -1. The weights are now adjusted using Eq. 14.7 with a learning rate $\eta = 0.1$:

$$\begin{aligned} w_0(t+1) &= w_0(t) + \eta\, x_0 y &&= 0.1 + 0.1 \times 1 \times -1 &&= 0 \\ w_1(t+1) &= w_1(t) + \eta\, x_1 y &&= 0.1 + 0.1 \times 0.57 \times -1 &&= 0.043 \\ w_2(t+1) &= w_2(t) + \eta\, x_2 y &&= 0.1 + 0.1 \times 0.5 \times -1 &&= 0.05\,. \end{aligned}$$

These will be the new weights for the next iteration. If we continue for 2000 iterations, the weights evolve as shown in Fig. 14.16. At the end of the learning process, the weights are:

$$w_0 = -0.1, \quad w_1 = -0.39, \quad w_2 = 0.53\,.$$

These weights can now be used by the recognition phase of the classification Algorithm 14.6. The discriminant line built by the perceptron (Eq. 14.6) is:

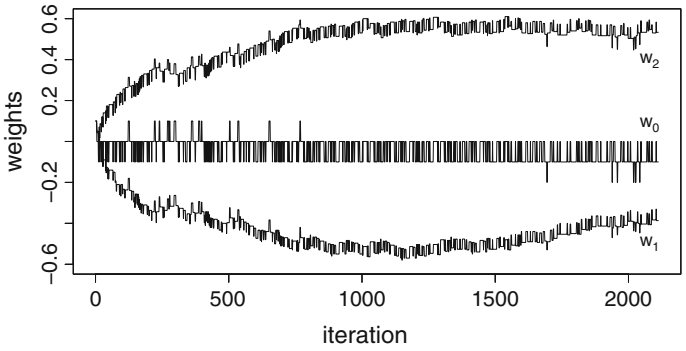

Fig. 14.16 Evolution of the weights for learning by a perceptron

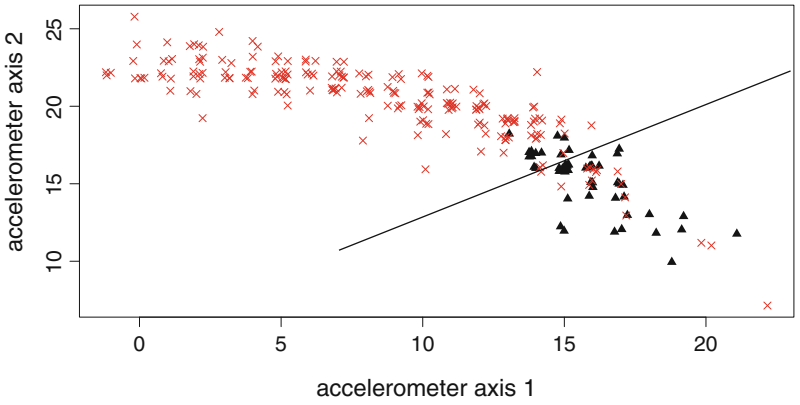

Fig. 14.17 Discriminant line computed from the perceptron weights

$$-0.1 - 0.39x_1 + 0.53x_2 = 0\,.$$

The coordinates of this line are the normalized values, but they can be transformed back into the raw values obtained from the accelerometers. The line is shown in Fig. 14.17 and considering the large overlap of the classes, it does a reasonably good job of distinguishing between them.

14.4.5 Tuning the Parameters of the Perceptron

The performance of a perceptron is determined by the number of iterations and the learning rate. Figure 14.16 shows that there is a strong variation in the weights at the beginning, but the weights stabilize as the number of iterations increases. Thus it is relatively simple to monitor the weights and terminate the computation when the weights stabilize.

This evolution of the weights depends strongly on the learning rate. Increasing the learning rate speeds the variation at the beginning, but strong corrections are not beneficial when the weights begin to stabilize. From Fig. 14.16, it is clear that even at the end of the run, there are significant variations in the weights which oscillate around the optimal value. This suggests that we reduce the learning rate to reduce the oscillations, but doing so will slow down the convergence to the optimal weights at the beginning of the learning phase.

The solution is to use a variable learning rate that is not constant. It should start out large to encourage rapid convergence to the optimal values, and then become smaller to reduce oscillations. For example, we can start with the learning rate of 0.1 and then decrease it continually using the equation:

$$\eta\,(t+1) = \eta\,(t) \times 0.997\,.$$

Figure 14.18 shows the evolution of the weights when this variable learning rate is used. The exponential decrease of η is also plotted in the figure. A comparison of Figs. 14.16 and 14.18 clearly shows the superiority of the variable learning rate and this improvement is obtained with very little additional computation.

Activity 14.5: Learning by a perceptron

- Take a set of measurements of the accelerometers on your robot on various slopes and plot the data. For each sample you will have to decide if the robot is in danger of falling off the slope.
- Classify the data using a perceptron. What discriminant line do you find?
- Use a perceptron to classify the gray areas using the data of Activity 14.2. What discriminant do you find? Compare the discriminant found by the perceptron to the discriminant found by LDA.

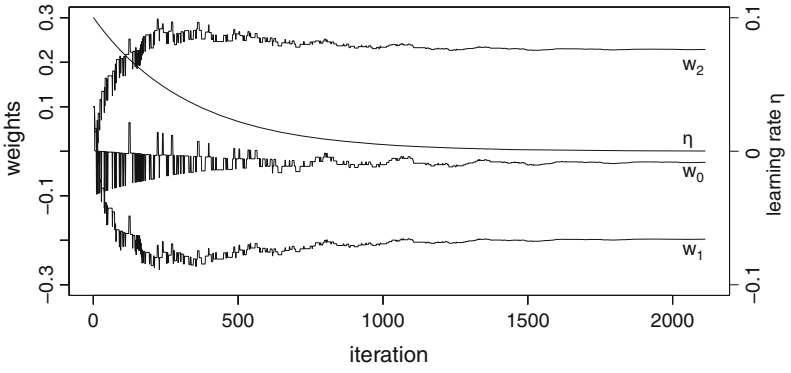

Fig. 14.18 Evolution of the weights for learning by a perceptron with a variable learning rate

14.5 Summary

Samples of two classes can be distinguished using their means alone or using both the means and the variances. Linear discriminant analysis is a method for classification that is based on computing the covariances between the samples of the classes. LDA performs well only when the distributions of the samples of the classes are similar. When this assumption does not hold, perceptrons can be used. For optimum performance, the learning rate of a perceptron must be adjusted, if possible dynamically during the learning phase.

14.6 Further Reading

A detailed mathematical presentation of linear discriminant analysis can be found in Izenman [2, Chap. 8]. Textbooks on machine learning techniques are [1, 3].

References

1. Harrington, P.: Machine Learning in Action, vol. 5. Manning, Greenwich (2012)
2. Izenman, A.J.: Modern Multivariate Statistical Techniques. Springer, Berlin (2008)
3. Kubat, M.: An Introduction to Machine Learning. Springer, Berlin (2015)