2 *Supervised Learning*

We discuss supervised learning starting from the simplest case, which is learning a class from its positive and negative examples. We generalize and discuss the case of multiple classes, then regression, where the outputs are continuous.

2.1 Learning a Class from Examples

LET US say we want to learn the *class*, C, of a "family car." We have a set of examples of cars, and we have a group of people that we survey to whom we show these cars. The people look at the cars and label them; the cars that they believe are family cars are *positive examples*, and the other cars are *negative examples*. Class learning is finding a description that is shared by all the positive examples and none of the negative examples. Doing this, we can make a prediction: Given a car that we have not seen before, by checking with the description learned, we will be able to say whether it is a family car or not. Or we can do knowledge extraction: This study may be sponsored by a car company, and the aim may be to understand what people expect from a family car.

POSITIVE EXAMPLES

NEGATIVE EXAMPLES

After some discussions with experts in the field, let us say that we reach the conclusion that among all features a car may have, the features that separate a family car from other type of cars are the price and engine power. These two attributes are the *inputs* to the class recognizer. Note that when we decide on this particular *input representation*, we are ignoring various other attributes as irrelevant. Though one may think of other attributes such as seating capacity and color that might be important for distinguishing among car types, we will consider only price and engine power to keep this example simple.

INPUT REPRESENTATION

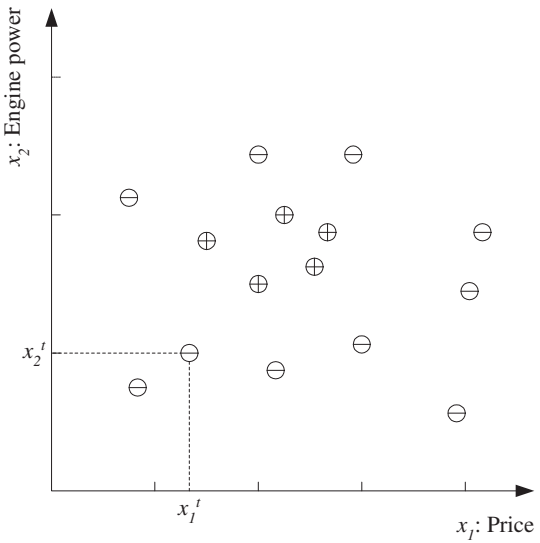

Figure 2.1 Training set for the class of a "family car." Each data point corresponds to one example car, and the coordinates of the point indicate the price and engine power of that car. '+' denotes a positive example of the class (a family car), and '−' denotes a negative example (not a family car); it is another type of car.

Let us denote price as the first input attribute x_1 (e.g., in U.S. dollars) and engine power as the second attribute x_2 (e.g., engine volume in cubic centimeters). Thus we represent each car using two numeric values

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{2.1}$$

and its label denotes its type

$$r = \begin{cases} 1 & \text{if } \mathbf{x} \text{ is a positive example} \\ 0 & \text{if } \mathbf{x} \text{ is a negative example} \end{cases} \tag{2.2}$$

Each car is represented by such an ordered pair $(\mathbf{x}, r)$ and the training set contains N such examples

$$\mathcal{X} = \{\mathbf{x}^t, r^t\}_{t=1}^N \tag{2.3}$$

where t indexes different examples in the set; it does not represent time or any such order.

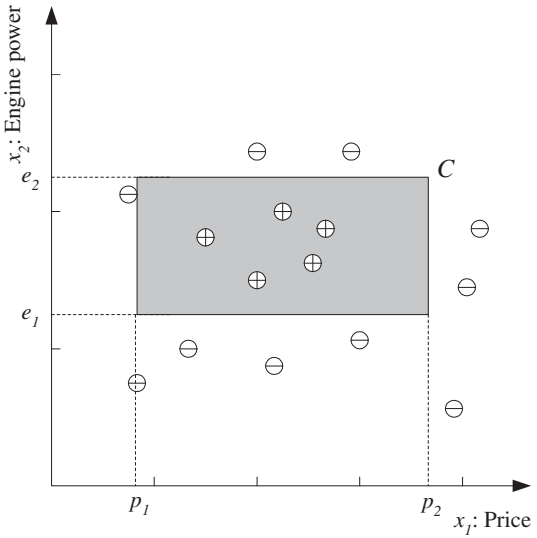

Figure 2.2 Example of a hypothesis class. The class of family car is a rectangle in the price-engine power space.

Our training data can now be plotted in the two-dimensional (x_1, x_2) space where each instance t is a data point at coordinates (x_1^t, x_2^t) and its type, namely, positive versus negative, is given by r^t (see figure 2.1).

After further discussions with the expert and the analysis of the data, we may have reason to believe that for a car to be a family car, its price and engine power should be in a certain range

$$(p_1 \leq \text{price} \leq p_2) \text{ AND } (e_1 \leq \text{engine power} \leq e_2) \tag{2.4}$$

for suitable values of p_1, p_2, e_1, and e_2. Equation 2.4 thus assumes C to be a rectangle in the price-engine power space (see figure 2.2).

HYPOTHESIS CLASS

Equation 2.4 fixes $\mathcal{H}$, the *hypothesis class* from which we believe C is drawn, namely, the set of rectangles. The learning algorithm then finds the particular *hypothesis*, $h \in \mathcal{H}$, specified by a particular quadruple of $(p_1^h, p_2^h, e_1^h, e_2^h)$, to approximate C as closely as possible.

HYPOTHESIS

Though the expert defines this hypothesis class, the values of the parameters are not known; that is, though we choose $\mathcal{H}$, we do not know

which particular $h \in \mathcal{H}$ is equal, or closest, to C. But once we restrict our attention to this hypothesis class, learning the class reduces to the easier problem of finding the four parameters that define h.

The aim is to find $h \in \mathcal{H}$ that is as similar as possible to C. Let us say the hypothesis h makes a prediction for an instance $\boldsymbol{x}$ such that

$$h(\boldsymbol{x}) = \begin{cases} 1 & \text{if } h \text{ classifies } \boldsymbol{x} \text{ as a positive example} \\ 0 & \text{if } h \text{ classifies } \boldsymbol{x} \text{ as a negative example} \end{cases} \tag{2.5}$$

In real life we do not know $C(\boldsymbol{x})$, so we cannot evaluate how well $h(\boldsymbol{x})$ matches $C(\boldsymbol{x})$. What we have is the training set $\mathcal{X}$, which is a small subset of the set of all possible $\boldsymbol{x}$. The *empirical error* is the proportion of training instances where *predictions* of h do not match the *required values* given in $\mathcal{X}$. The error of hypothesis h given the training set $\mathcal{X}$ is

EMPIRICAL ERROR

$$E(h|\mathcal{X}) = \sum_{t=1}^{N} 1(h(\boldsymbol{x}^t) \neq r^t) \tag{2.6}$$

where $1(a \neq b)$ is 1 if $a \neq b$ and is 0 if $a = b$ (see figure 2.3).

In our example, the hypothesis class $\mathcal{H}$ is the set of all possible rectangles. Each quadruple $(p_1^h, p_2^h, e_1^h, e_2^h)$ defines one hypothesis, h, from $\mathcal{H}$, and we need to choose the best one, or in other words, we need to find the values of these four parameters given the training set, to include all the positive examples and none of the negative examples. Note that if x_1 and x_2 are real-valued, there are infinitely many such h for which this is satisfied, namely, for which the error, E, is 0, but given a future example somewhere close to the boundary between positive and negative examples, different candidate hypotheses may make different predictions. This is the problem of *generalization*—that is, how well our hypothesis will correctly classify future examples that are not part of the training set.

GENERALIZATION

MOST SPECIFIC HYPOTHESIS

One possibility is to find the *most specific hypothesis*, S, that is the tightest rectangle that includes all the positive examples and none of the negative examples (see figure 2.4). This gives us one hypothesis, $h = S$, as our induced class. Note that the actual class C may be larger than S but is never smaller. The *most general hypothesis*, G, is the largest rectangle we can draw that includes all the positive examples and none of the negative examples (figure 2.4). Any $h \in \mathcal{H}$ between S and G is a valid hypothesis with no error, said to be *consistent* with the training set, and such h make up the *version space*. Given another training set, S, G, version space, the parameters and thus the learned hypothesis, h, can be different.

MOST GENERAL HYPOTHESIS

VERSION SPACE

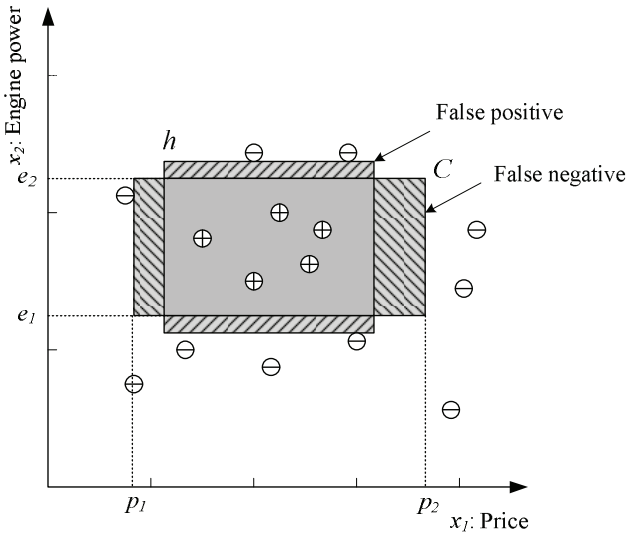

Figure 2.3 C is the actual class and h is our induced hypothesis. The point where C is 1 but h is 0 is a false negative, and the point where C is 0 but h is 1 is a false positive. Other points—namely, true positives and true negatives—are correctly classified.

Actually, depending on $\mathcal{X}$ and $\mathcal{H}$, there may be several S_i and G_j which respectively make up the S-set and the G-set. Every member of the S-set is consistent with all the instances, and there are no consistent hypotheses that are more specific. Similarly, every member of the G-set is consistent with all the instances, and there are no consistent hypotheses that are more general. These two make up the boundary sets and any hypothesis between them is consistent and is part of the version space. There is an algorithm called candidate elimination that incrementally updates the S- and G-sets as it sees training instances one by one; see Mitchell 1997. We assume $\mathcal{X}$ is large enough that there is a unique S and G.

Given $\mathcal{X}$, we can find S, or G, or any h from the version space and use it as our hypothesis, h. It seems intuitive to choose h halfway between S and G; this is to increase the *margin*, which is the distance between the boundary and the instances closest to it (see figure 2.5). For our error function to have a minimum at h with the maximum margin, we should

MARGIN

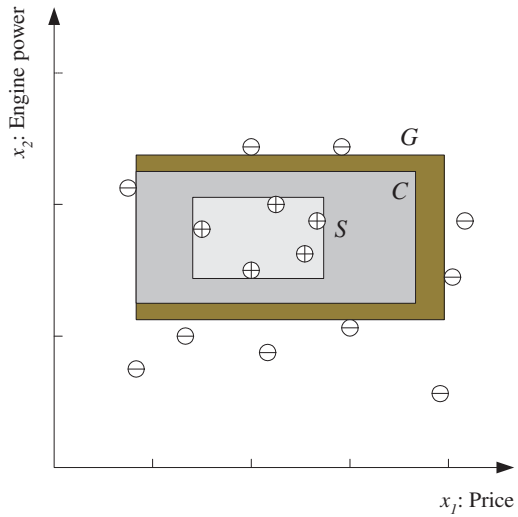

Figure 2.4 S is the most specific and G is the most general hypothesis.

use an error (loss) function which not only checks whether an instance is on the correct side of the boundary but also how far away it is. That is, instead of $h(\boldsymbol{x})$ that returns 0/1, we need to have a hypothesis that returns a value which carries a measure of the distance to the boundary and we need to have a loss function which uses it, different from $1(\cdot)$ that checks for equality.

In some applications, a wrong decision may be very costly and in such a case, we can say that any instance that falls in between S and G is a case of *doubt*, which we cannot label with certainty due to lack of data. In such a case, the system *rejects* the instance and defers the decision to a human expert.

DOUBT

Here, we assume that $\mathcal{H}$ includes C; that is, there exists $h \in \mathcal{H}$, such that $E(h|\mathcal{X})$ is 0. Given a hypothesis class $\mathcal{H}$, it may be the case that we cannot learn C; that is, there exists no $h \in \mathcal{H}$ for which the error is 0. Thus, in any application, we need to make sure that $\mathcal{H}$ is flexible enough, or has enough "capacity," to learn C.

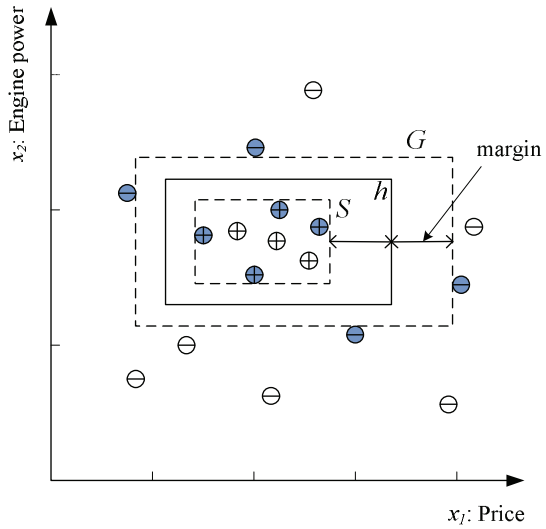

Figure 2.5 We choose the hypothesis with the largest margin, for best separation. The shaded instances are those that define (or support) the margin; other instances can be removed without affecting h.

2.2 Vapnik-Chervonenkis Dimension

Let us say we have a dataset containing N points. These N points can be labeled in 2^N ways as positive and negative. Therefore, 2^N different learning problems can be defined by N data points. If for any of these problems, we can find a hypothesis $h \in \mathcal{H}$ that separates the positive examples from the negative, then we say $\mathcal{H}$ *shatters* N points. That is, any learning problem definable by N examples can be learned with no error by a hypothesis drawn from $\mathcal{H}$. The maximum number of points that can be shattered by $\mathcal{H}$ is called the *Vapnik-Chervonenkis* (VC) *dimension* of $\mathcal{H}$, is denoted as $VC(\mathcal{H})$, and measures the *capacity* of $\mathcal{H}$.

VC DIMENSION

In figure 2.6, we see that an axis-aligned rectangle can shatter four points in two dimensions. Then $VC(\mathcal{H})$, when $\mathcal{H}$ is the hypothesis class of axis-aligned rectangles in two dimensions, is four. In calculating the VC dimension, it is enough that we find four points that can be shattered; it is not necessary that we be able to shatter *any* four points in two di-

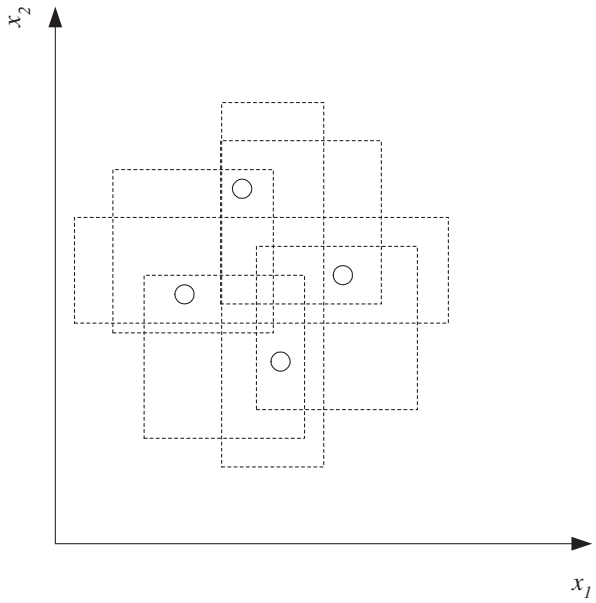

Figure 2.6 An axis-aligned rectangle can shatter four points. Only rectangles covering two points are shown.

mensions. For example, four points placed on a line cannot be shattered by rectangles. However, we cannot place five points in two dimensions *anywhere* such that a rectangle can separate the positive and negative examples for all possible labelings.

VC dimension may seem pessimistic. It tells us that using a rectangle as our hypothesis class, we can learn only datasets containing four points and not more. A learning algorithm that can learn datasets of four points is not very useful. However, this is because the VC dimension is independent of the probability distribution from which instances are drawn. In real life, the world is smoothly changing, instances close by most of the time have the same labels, and we need not worry about *all possible labelings*. There are a lot of datasets containing many more data points than four that are learnable by our hypothesis class (figure 2.1). So even hypothesis classes with small VC dimensions are applicable and are preferred over those with large VC dimensions, for example, a lookup table that has infinite VC dimension.

2.3 Probably Approximately Correct Learning

Using the tightest rectangle, S, as our hypothesis, we would like to find how many examples we need. We would like our hypothesis to be approximately correct, namely, that the error probability be bounded by some value. We also would like to be confident in our hypothesis in that we want to know that our hypothesis will be correct most of the time (if not always); so we want to be probably correct as well (by a probability we can specify).

PAC LEARNING

In *probably approximately correct* (PAC) *learning*, given a class, C, and examples drawn from some unknown but fixed probability distribution, $p(x)$, we want to find the number of examples, N, such that with probability at least $1-\delta$, the hypothesis h has error at most ϵ, for arbitrary $\delta \leq 1/2$ and $\epsilon > 0$

$$P\{C\Delta h \leq \epsilon\} \geq 1-\delta$$

where $C\Delta h$ is the region of difference between C and h.

In our case, because S is the tightest possible rectangle, the error region between C and $h = S$ is the sum of four rectangular strips (see figure 2.7). We would like to make sure that the probability of a positive example falling in here (and causing an error) is at most ϵ. For any of these strips, if we can guarantee that the probability is upper bounded by $\epsilon/4$, the error is at most $4(\epsilon/4) = \epsilon$. Note that we count the overlaps in the corners twice, and the total actual error in this case is less than $4(\epsilon/4)$. The probability that a randomly drawn example misses this strip is $1-\epsilon/4$. The probability that all N independent draws miss the strip is $(1-\epsilon/4)^N$, and the probability that all N independent draws miss any of the four strips is at most $4(1-\epsilon/4)^N$, which we would like to be at most δ. We have the inequality

$$(1-x) \leq \exp[-x]$$

So if we choose N and δ such that we have

$$4\exp[-\epsilon N/4] \leq \delta$$

we can also write $4(1-\epsilon/4)^N \leq \delta$. Dividing both sides by 4, taking (natural) log and rearranging terms, we have

$$N \geq (4/\epsilon)\log(4/\delta) \tag{2.7}$$

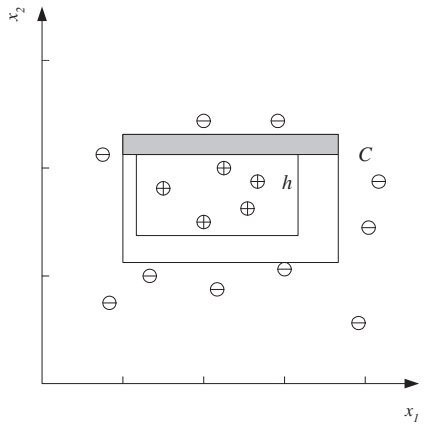

Figure 2.7 The difference between h and C is the sum of four rectangular strips, one of which is shaded.

Therefore, provided that we take at least $(4/\epsilon)\log(4/\delta)$ independent examples from C and use the tightest rectangle as our hypothesis h, with *confidence probability* at least $1-\delta$, a given point will be misclassified with *error probability* at most ϵ. We can have arbitrary large confidence by decreasing δ and arbitrary small error by decreasing ϵ, and we see in equation 2.7 that the number of examples is a slowly growing function of $1/\epsilon$ and $1/\delta$, linear and logarithmic, respectively.

2.4 Noise

NOISE *Noise* is any unwanted anomaly in the data and due to noise, the class may be more difficult to learn and zero error may be infeasible with a simple hypothesis class (see figure 2.8). There are several interpretations of noise:

- There may be imprecision in recording the input attributes, which may shift the data points in the input space.
- There may be errors in labeling the data points, which may relabel

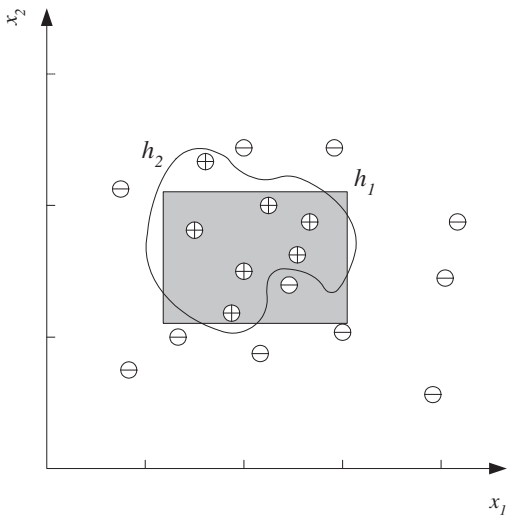

Figure 2.8 When there is noise, there is not a simple boundary between the positive and negative instances, and zero misclassification error may not be possible with a simple hypothesis. A rectangle is a simple hypothesis with four parameters defining the corners. An arbitrary closed form can be drawn by piecewise functions with a larger number of control points.

positive instances as negative and vice versa. This is sometimes called *teacher noise*.

- There may be additional attributes, which we have not taken into account, that affect the label of an instance. Such attributes may be *hidden* or *latent* in that they may be unobservable. The effect of these neglected attributes is thus modeled as a random component and is included in "noise."

As can be seen in figure 2.8, when there is noise, there is not a simple boundary between the positive and negative instances and to separate them, one needs a complicated hypothesis that corresponds to a hypothesis class with larger capacity. A rectangle can be defined by four numbers, but to define a more complicated shape one needs a more complex model with a much larger number of parameters. With a complex model,

one can make a perfect fit to the data and attain zero error; see the wiggly shape in figure 2.8. Another possibility is to keep the model simple and allow some error; see the rectangle in figure 2.8.

Using the simple rectangle (unless its training error is much bigger) makes more sense because of the following:

1. It is a simple model to use. It is easy to check whether a point is inside or outside a rectangle and we can easily check, for a future data instance, whether it is a positive or a negative instance.

2. It is a simple model to train and has fewer parameters. It is easier to find the corner values of a rectangle than the control points of an arbitrary shape. With a small training set when the training instances differ a little bit, we expect the simpler model to change less than a complex model: A simple model is thus said to have less *variance*. On the other hand, a too simple model assumes more, is more rigid, and may fail if indeed the underlying class is not that simple: A simpler model has more *bias*. Finding the optimal model corresponds to minimizing both the bias and the variance.

3. It is a simple model to explain. A rectangle simply corresponds to defining intervals on the two attributes. By learning a simple model, we can extract information from the raw data given in the training set.

4. If indeed there is mislabeling or noise in input and the actual class is really a simple model like the rectangle, then the simple rectangle, because it has less variance and is less affected by single instances, will be a better discriminator than the wiggly shape, although the simple one may make slightly more errors on the training set. Given comparable empirical error, we say that a simple (but not too simple) model would generalize better than a complex model. This principle is known as *Occam's razor*, which states that *simpler explanations are more plausible* and any unnecessary complexity should be shaved off.

OCCAM'S RAZOR

2.5 Learning Multiple Classes

In our example of learning a family car, we have positive examples belonging to the class family car and the negative examples belonging to all other cars. This is a *two-class* problem. In the general case, we have K

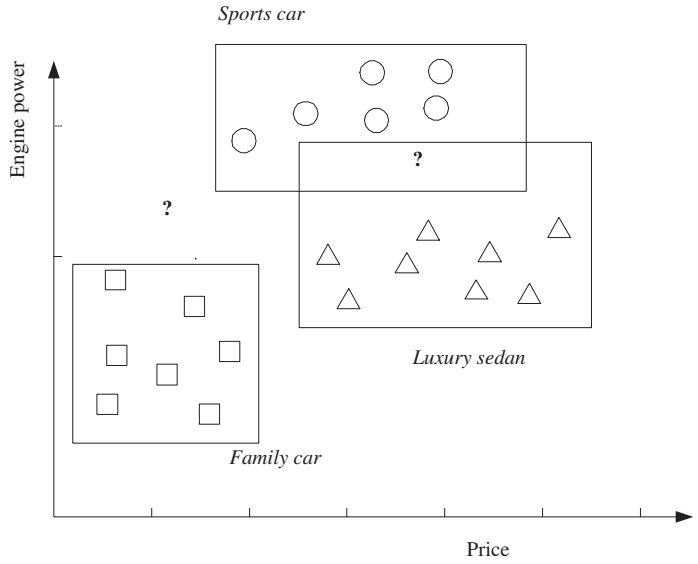

Figure 2.9 There are three classes: family car, sports car, and luxury sedan. There are three hypotheses induced, each one covering the instances of one class and leaving outside the instances of the other two classes. '?' are reject regions where no, or more than one, class is chosen.

classes denoted as $C_i, i = 1, \ldots, K$, and an input instance belongs to one and exactly one of them. The training set is now of the form

$$\mathcal{X} = \{\boldsymbol{x}^t, \boldsymbol{r}^t\}_{t=1}^N$$

where $\boldsymbol{r}$ has K dimensions and

$$r_i^t = \begin{cases} 1 & \text{if } \boldsymbol{x}^t \in C_i \\ 0 & \text{if } \boldsymbol{x}^t \in C_j, j \neq i \end{cases} \tag{2.8}$$

An example is given in figure 2.9 with instances from three classes: family car, sports car, and luxury sedan.

In machine learning for classification, we would like to learn the boundary separating the instances of one class from the instances of all other classes. Thus we view a K-class classification problem as K two-class problems. The training examples belonging to C_i are the positive instances of hypothesis h_i and the examples of all other classes are the

negative instances of h_i. Thus in a K-class problem, we have K hypotheses to learn such that

$$h_i(\boldsymbol{x}^t) = \begin{cases} 1 & \text{if } \boldsymbol{x}^t \in C_i \\ 0 & \text{if } \boldsymbol{x}^t \in C_j, j \neq i \end{cases} \tag{2.9}$$

The total empirical error takes a sum over the predictions for all classes over all instances:

$$E(\{h_i\}_{i=1}^K | \mathcal{X}) = \sum_{t=1}^{N} \sum_{i=1}^{K} 1(h_i(\boldsymbol{x}^t) \neq r_i^t) \tag{2.10}$$

For a given $\boldsymbol{x}$, ideally only one of $h_i(\boldsymbol{x}), i = 1, \ldots, K$ is 1 and we can choose a class. But when no, or two or more, $h_i(\boldsymbol{x})$ is 1, we cannot choose a class, and this is the case of *doubt* and the classifier *rejects* such cases.

REJECT

In our example of learning a family car, we used only one hypothesis and only modeled the positive examples. Any negative example outside is not a family car. Alternatively, sometimes we may prefer to build two hypotheses, one for the positive and the other for the negative instances. This assumes a structure also for the negative instances that can be covered by another hypothesis. Separating family cars from sports cars is such a problem; each class has a structure of its own. The advantage is that if the input is a luxury sedan, we can have both hypotheses decide negative and reject the input.

If in a dataset, we expect to have all classes with similar distribution—shapes in the input space—then the same hypothesis class can be used for all classes. For example, in a handwritten digit recognition dataset, we would expect all digits to have similar distributions. But in a medical diagnosis dataset, for example, where we have two classes for sick and healthy people, we may have completely different distributions for the two classes; there may be multiple ways for a person to be sick, reflected differently in the inputs: All healthy people are alike; each sick person is sick in his or her own way.

2.6 Regression

In classification, given an input, the output that is generated is Boolean; it is a yes/no answer. When the output is a numeric value, what we would like to learn is not a class, $C(\boldsymbol{x}) \in \{0, 1\}$, but is a numeric function. In

machine learning, the function is not known but we have a training set of examples drawn from it

$$\mathcal{X} = \{\boldsymbol{x}^t, r^t\}_{t=1}^N$$

INTERPOLATION

where $r^t \in \Re$. If there is no noise, the task is *interpolation*. We would like to find the function $f(x)$ that passes through these points such that we have

$$r^t = f(\boldsymbol{x}^t)$$

In *polynomial interpolation*, given N points, we find the $(N-1)$st degree polynomial that we can use to predict the output for any $\boldsymbol{x}$. This is called *extrapolation* if $\boldsymbol{x}$ is outside of the range of $\boldsymbol{x}^t$ in the training set. In time-series prediction, for example, we have data up to the present and we want to predict the value for the future. In *regression*, there is noise added to the output of the unknown function

EXTRAPOLATION

REGRESSION

$$r^t = f(\boldsymbol{x}^t) + \epsilon \tag{2.11}$$

where $f(\boldsymbol{x}) \in \Re$ is the unknown function and ϵ is random noise. The explanation for noise is that there are extra *hidden* variables that we cannot observe

$$r^t = f^*(\boldsymbol{x}^t, \boldsymbol{z}^t) \tag{2.12}$$

where $\boldsymbol{z}^t$ denote those hidden variables. We would like to approximate the output by our model $g(\boldsymbol{x})$. The empirical error on the training set $\mathcal{X}$ is

$$E(g|\mathcal{X}) = \frac{1}{N}\sum_{t=1}^{N}[r^t - g(\boldsymbol{x}^t)]^2 \tag{2.13}$$

Because r and $g(\boldsymbol{x})$ are numeric quantities, for example, $\in \Re$, there is an ordering defined on their values and we can define a *distance* between values, as the square of the difference, which gives us more information than equal/not equal, as used in classification. The square of the difference is one error (loss) function that can be used; another is the absolute value of the difference. We will see other examples in the coming chapters.

Our aim is to find $g(\cdot)$ that minimizes the empirical error. Again our approach is the same; we assume a hypothesis class for $g(\cdot)$ with a small set of parameters. If we assume that $g(\boldsymbol{x})$ is linear, we have

$$g(\boldsymbol{x}) = w_1 x_1 + \cdots + w_d x_d + w_0 = \sum_{j=1}^{d} w_j x_j + w_0 \tag{2.14}$$

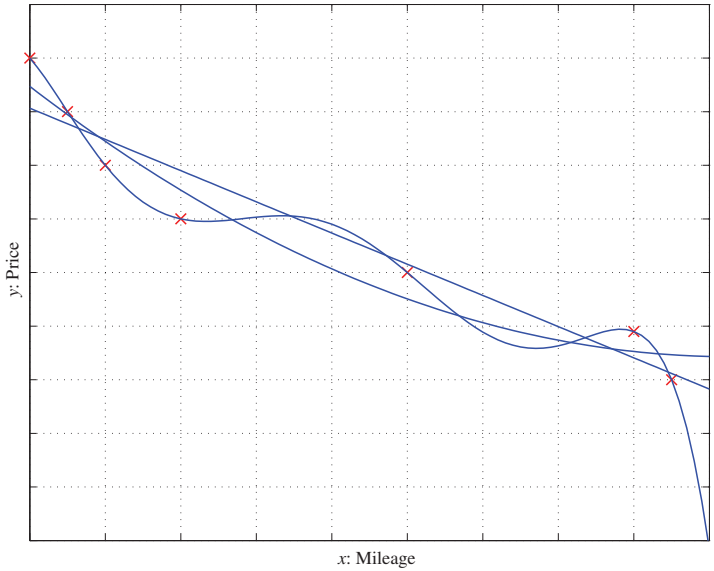

Figure 2.10 Linear, second-order, and sixth-order polynomials are fitted to the same set of points. The highest order gives a perfect fit, but given this much data it is very unlikely that the real curve is so shaped. The second order seems better than the linear fit in capturing the trend in the training data.

Let us now go back to our example in section 1.2.3 where we estimated the price of a used car. There we used a single input linear model

$$g(x) = w_1 x + w_0 \tag{2.15}$$

where w_1 and w_0 are the parameters to learn from data. The w_1 and w_0 values should minimize

$$E(w_1, w_0|\mathcal{X}) = \frac{1}{N}\sum_{t=1}^{N}[r^t - (w_1 x^t + w_0)]^2 \tag{2.16}$$

Its minimum point can be calculated by taking the partial derivatives of E with respect to w_1 and w_0, setting them equal to 0, and solving for the two unknowns:

$$\begin{aligned} w_1 &= \frac{\sum_t x^t r^t - \overline{x}\overline{r}N}{\sum_t (x^t)^2 - N\overline{x}^2} \\ w_0 &= \overline{r} - w_1\overline{x} \end{aligned} \tag{2.17}$$

Table 2.1 With two inputs, there are four possible cases and sixteen possible Boolean functions

x_1	x_2	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_9	h_{10}	h_{11}	h_{12}	h_{13}	h_{14}	h_{15}	h_{16}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

where $\overline{x} = \sum_t x^t / N$ and $\overline{r} = \sum_t r^t / N$. The line found is shown in figure 1.2.

If the linear model is too simple, it is too constrained and incurs a large approximation error, and in such a case, the output may be taken as a higher-order function of the input—for example, quadratic

$$g(x) = w_2 x^2 + w_1 x + w_0 \tag{2.18}$$

where similarly we have an analytical solution for the parameters. When the order of the polynomial is increased, the error on the training data decreases. But a high-order polynomial follows individual examples closely, instead of capturing the general trend; see the sixth-order polynomial in figure 2.10. This implies that Occam's razor also applies in the case of regression and we should be careful when fine-tuning the model complexity to match it with the complexity of the function underlying the data.

2.7 Model Selection and Generalization

Let us start with the case of learning a Boolean function from examples. In a Boolean function, all inputs and the output are binary. There are 2^d possible ways to write d binary values and therefore, with d inputs, the training set has at most 2^d examples. As shown in table 2.1, each of these can be labeled as 0 or 1, and therefore, there are 2^{2^d} possible Boolean functions of d inputs.

Each distinct training example removes half the hypotheses, namely, those whose guesses are wrong. For example, let us say we have $x_1 = 0$, $x_2 = 1$ and the output is 0; this removes $h_5, h_6, h_7, h_8, h_{13}, h_{14}, h_{15}, h_{16}$. This is one way to interpret learning: We start with all possible hypotheses and as we see more training examples, we remove those hypotheses

that are not consistent with the training data. In the case of a Boolean function, to end up with a single hypothesis we need to see *all* 2^d training examples. If the training set we are given contains only a small subset of all possible instances, as it generally does—that is, if we know what the output should be for only a small percentage of the cases—the solution is not unique. After seeing N example cases, there remain 2^{2^d-N} possible functions. This is an example of an *ill-posed problem* where the data by itself is not sufficient to find a unique solution.

ILL-POSED PROBLEM

The same problem also exists in other learning applications, in classification, and in regression. As we see more training examples, we know more about the underlying function, and we carve out more hypotheses that are inconsistent from the hypothesis class, but we still are left with many consistent hypotheses.

So because learning is ill-posed, and data by itself is not sufficient to find the solution, we should make some extra assumptions to have a unique solution with the data we have. The set of assumptions we make to have learning possible is called the *inductive bias* of the learning algorithm. One way we introduce inductive bias is when we assume a hypothesis class $\mathcal{H}$. In learning the class of family cars, there are infinitely many ways of separating the positive examples from the negative examples. Assuming the shape of a rectangle is one inductive bias, and then the rectangle with the largest margin for example, is another inductive bias. In linear regression, assuming a linear function is an inductive bias, and among all lines, choosing the one that minimizes squared error is another inductive bias.

INDUCTIVE BIAS

But we know that each hypothesis class has a certain capacity and can learn only certain functions. The class of functions that can be learned can be extended by using a hypothesis class with larger capacity, containing more complex hypotheses. For example, the hypothesis class that is a union of two rectangles has higher capacity, but its hypotheses are more complex. Similarly in regression, as we increase the order of the polynomial, the capacity and complexity increase. The question now is to decide where to stop.

Thus learning is not possible without inductive bias, and now the question is how to choose the right bias. This is called *model selection*, which is choosing between possible $\mathcal{H}$. In answering this question, we should remember that the aim of machine learning is rarely to replicate the training data but the prediction for new cases. That is we would like to be able to generate the right output for an input instance outside the training set,

MODEL SELECTION

one for which the correct output is not given in the training set. How well a model trained on the training set predicts the right output for new instances is called *generalization*.

GENERALIZATION

For best generalization, we should match the complexity of the hypothesis class $\mathcal{H}$ with the complexity of the function underlying the data. If $\mathcal{H}$ is less complex than the function, we have *underfitting*, for example, when trying to fit a line to data sampled from a third-order polynomial. In such a case, as we increase the complexity, the training error decreases. But if we have $\mathcal{H}$ that is too complex, the data is not enough to constrain it and we may end up with a bad hypothesis, $h \in \mathcal{H}$, for example, when fitting two rectangles to data sampled from one rectangle. Or if there is noise, an overcomplex hypothesis may learn not only the underlying function but also the noise in the data and may make a bad fit, for example, when fitting a sixth-order polynomial to noisy data sampled from a third-order polynomial. This is called *overfitting*. In such a case, having more training data helps but only up to a certain point. Given a training set and $\mathcal{H}$, we can find $h \in \mathcal{H}$ that has the minimum training error but if $\mathcal{H}$ is not chosen well, no matter which $h \in \mathcal{H}$ we pick, we will not have good generalization.

UNDERFITTING

OVERFITTING

TRIPLE TRADE-OFF

We can summarize our discussion citing the *triple trade-off* (Dietterich 2003). In all learning algorithms that are trained from example data, there is a trade-off between three factors:

- the complexity of the hypothesis we fit to data, namely, the capacity of the hypothesis class,
- the amount of training data, and
- the generalization error on new examples.

As the amount of training data increases, the generalization error decreases. As the complexity of the model class $\mathcal{H}$ increases, the generalization error decreases first and then starts to increase. The generalization error of an overcomplex $\mathcal{H}$ can be kept in check by increasing the amount of training data but only up to a point. If the data is sampled from a line and if we are fitting a higher-order polynomial, the fit will be constrained to lie close to the line if there is training data in the vicinity; where it has not been trained, a high-order polynomial may behave erratically.

We can measure the generalization ability of a hypothesis, namely, the quality of its inductive bias, if we have access to data outside the training

set. We simulate this by dividing the dataset we have into two parts. We use one part for training (i.e., to fit a hypothesis), and the remaining part is called the *validation set* and is used to test the generalization ability. That is, given a set of possible hypothesis classes $\mathcal{H}_i$, for each we fit the best $h_i \in \mathcal{H}_i$ on the training set. Then, assuming large enough training and validation sets, the hypothesis that is the most accurate on the validation set is the best one (the one that has the best inductive bias). This process is called *cross-validation*. So, for example, to find the right order in polynomial regression, given a number of candidate polynomials of different orders where polynomials of different orders correspond to $\mathcal{H}_i$, for each order, we find the coefficients on the training set, calculate their errors on the validation set, and take the one that has the least validation error as the best polynomial.

VALIDATION SET

CROSS-VALIDATION

Note that if we then need to report the error to give an idea about the expected error of our best model, we should not use the validation error. We have used the validation set to choose the best model, and it has effectively become a part of the training set. We need a third set, a *test set*, sometimes also called the *publication set*, containing examples not used in training or validation. An analogy from our lives is when we are taking a course: the example problems that the instructor solves in class while teaching a subject form the training set; exam questions are the validation set; and the problems we solve in our later, professional life are the test set.

TEST SET

We cannot keep on using the same training/validation split either, because after having been used once, the validation set effectively becomes part of training data. This will be like an instructor who uses the same exam questions every year; a smart student will figure out not to bother with the lectures and will only memorize the answers to those questions.

We should always remember that the training data we use is a random sample, that is, for the same application, if we collect data once more, we will get a slightly different dataset, the fitted h will be slightly different and will have a slightly different validation error. Or if we have a fixed set which we divide for training, validation, and test, we will have different errors depending on how we do the division. These slight differences in error will allow us to estimate how large differences should be to be considered *significant* and not due to chance. That is, in choosing between two hypothesis classes $\mathcal{H}_i$ and $\mathcal{H}_j$, we will use them both multiple times on a number of training and validation sets and check if the difference between average errors of h_i and h_j is larger than the average difference

between multiple h_i. In chapter 19, we discuss how to design machine learning experiments using limited data to best answer our questions—for example, which is the best hypothesis class?—and how to analyze the results of these experiments so that we can achieve statistically significant conclusions minimally affected by random chance.

2.8 Dimensions of a Supervised Machine Learning Algorithm

Let us now recapitulate and generalize. We have a sample

$$\mathcal{X} = \{x^t, r^t\}_{t=1}^N \tag{2.19}$$

INDEPENDENT AND IDENTICALLY DISTRIBUTED (IID)

The sample is *independent and identically distributed* (*iid*); the ordering is not important and all instances are drawn from the same joint distribution $p(x, r)$. t indexes one of the N instances, x^t is the arbitrary dimensional input, and r^t is the associated desired output. r^t is 0/1 for two-class learning, is a K-dimensional binary vector (where exactly one of the dimensions is 1 and all others 0) for $(K > 2)$-class classification, and is a real value in regression.

The aim is to build a good and useful approximation to r^t using the model $g(x^t|\theta)$. In doing this, there are three decisions we must make:

1. *Model* we use in learning, denoted as

   $$g(x|\theta)$$

   where $g(\cdot)$ is the model, x is the input, and θ are the parameters.

   $g(\cdot)$ defines the hypothesis class $\mathcal{H}$, and a particular value of θ instantiates one hypothesis $h \in \mathcal{H}$. For example, in class learning, we have taken a rectangle as our model whose four coordinates make up θ; in linear regression, the model is the linear function of the input whose slope and intercept are the parameters learned from the data. The model (inductive bias), or $\mathcal{H}$, is fixed by the machine learning system designer based on his or her knowledge of the application and the hypothesis h is chosen (parameters are tuned) by a learning algorithm using the training set, sampled from $p(x, r)$.

2. *Loss function*, $L(\cdot)$, to compute the difference between the desired output, r^t, and our approximation to it, $g(x^t|\theta)$, given the current value

of the parameters, θ. The *approximation error*, or *loss*, is the sum of losses over the individual instances

$$E(\theta|\mathcal{X}) = \sum_t L(r^t, g(x^t|\theta)) \tag{2.20}$$

In class learning where outputs are 0/1, $L(\cdot)$ checks for equality or not; in regression, because the output is a numeric value, we have ordering information for distance and one possibility is to use the square of the difference.

3. *Optimization procedure* to find θ^* that minimizes the total error

$$\theta^* = \arg\min_\theta E(\theta|\mathcal{X}) \tag{2.21}$$

where arg min returns the argument that minimizes. In polynomial regression, we can solve analytically for the optimum, but this is not always the case. With other models and error functions, the complexity of the optimization problem becomes important. We are especially interested in whether it has a single minimum corresponding to a globally optimal solution, or whether there are multiple minima corresponding to locally optimal solutions.

For this setting to work well, the following conditions should be satisfied: First, the hypothesis class of $g(\cdot)$ should be large enough, that is, have enough capacity, to include the unknown function that generated the data that is represented in $\mathcal{X}$ in a noisy form. Second, there should be enough training data to allow us to pinpoint the correct (or a good enough) hypothesis from the hypothesis class. Third, we should have a good optimization method that finds the correct hypothesis given the training data.

Different machine learning algorithms differ either in the models they assume (their hypothesis class/inductive bias), the loss measures they employ, or the optimization procedure they use. We will see many examples in the coming chapters.

2.9 Notes

Mitchell proposed version spaces and the candidate elimination algorithm to incrementally build S and G as instances are given one by one;

see Mitchell 1997 for a recent review. The rectangle-learning is from exercise 2.4 of Mitchell 1997. Hirsh (1990) discusses how version spaces can handle the case when instances are perturbed by small amount of noise.

In one of the earliest works on machine learning, Winston (1975) proposed the idea of a "near miss." A near miss is a negative example that is very much like a positive example. In our terminology, we see that a near miss would be an instance that falls in the gray area between S and G, an instance which would affect the margin, and would hence be more useful for learning, than an ordinary positive or negative example. The instances that are close to the boundary are the ones that define it (or support it); those which are inside and are surrounded by many instances with the same label can be removed without affecting the boundary.

Related to this idea is *active learning* where the learning algorithm can generate instances itself and ask for them to be labeled, instead of passively being given them (Angluin 1988) (see exercise 5).

VC dimension was proposed by Vapnik and Chervonenkis in the early 1970s. A recent source is Vapnik 1995 where he writes, "Nothing is more practical than a good theory" (p. x), which is as true in machine learning as in any other branch of science. You should not rush to the computer; you can save yourself from hours of useless programming by some thinking, a notebook, and a pencil—you may also need an eraser.

The PAC model was proposed by Valiant (1984). The PAC analysis of learning a rectangle is from Blumer et al. 1989. A good textbook on computational learning theory covering PAC learning and VC dimension is Kearns and Vazirani 1994.

The definition of the optimization problem solved for model fitting has been getting very important in recent years. Once quite content with local descent methods that converge to the nearest good solution starting from some random initial state, nowadays we are, for example, interested in showing that the problem is convex—there is a single, global solution (Boyd and Vandenberghe 2004). As dataset sizes grow and models get more complex, we are also, for example, interested in how fast the optimization procedure converges to a solution.

2.10 Exercises

1. Let us say our hypothesis class is a circle instead of a rectangle. What are the parameters? How can the parameters of a circle hypothesis be calculated in

such a case? What if it is an ellipse? Why does it make more sense to use an ellipse instead of a circle?

SOLUTION: In the case of a circle, the parameters are the center and the radius (see figure 2.11). We then need to find S and G where S is the tightest circle that includes all the positive examples and G is the largest circle that includes all the positive examples and no negative example; any circle between them is a consistent hypothesis.

It makes more sense to use an ellipse because the two axes need not have the same scale and an ellipse has two separate parameters for the widths in the two axes rather than a single radius. Actually, price and engine power are positively correlated; the price of a car tends to increase as its engine power increases, and hence it makes more sense to use an oblique ellipse—we will see such models in chapter 5.

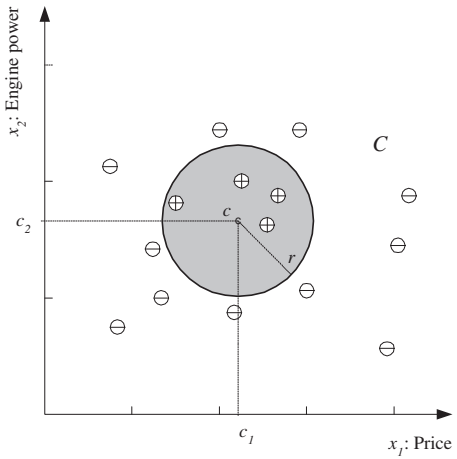

Figure 2.11 Hypothesis class is a circle with two parameters, the coordinates of its center and its radius.

2. Imagine our hypothesis is not one rectangle but a union of two (or $m > 1$) rectangles. What is the advantage of such a hypothesis class? Show that any class can be represented by such a hypothesis class with large enough m.

   SOLUTION: In the case when there is a single rectangle, all the positive instances should form one single group; with two rectangles, for example (see figure 2.12), the positive instances can form two, possibly disjoint clusters in the input space. Note that each rectangle corresponds to a conjunction on the two input attributes, and having multiple rectangles corresponds to a disjunction. Any logical formula can be written as a disjunction of conjunctions.

In the worst case ($m = N$), we have a separate rectangle for each positive instance.

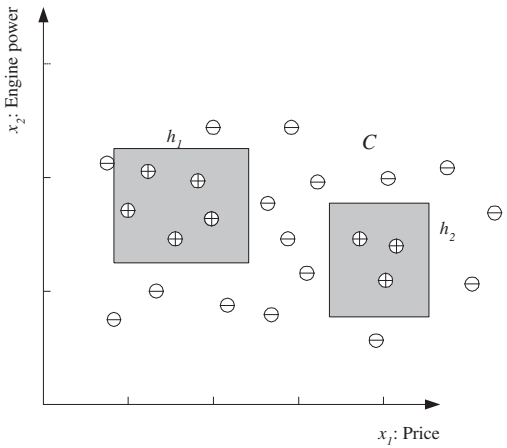

Figure 2.12 Hypothesis class is a union of two rectangles.

3. In many applications, wrong decisions—namely, false positives and false negatives—have a monetary cost, and these two costs may be different. What is the relationship between the positioning of h between S and G and the relative costs of these?

   SOLUTION: We can see that S makes no false positives, but only false negatives; similarly, G makes no false negatives, only false positives. So if false positives and false negatives are equally bad, we want our h to be halfway; if false positives are costlier, we want h to be closer to S; if false negatives are costlier, h should be closer to G.

4. The complexity of most learning algorithms is a function of the training set. Can you propose a filtering algorithm that finds redundant instances?

   SOLUTION: The instances that affect the hypothesis are those that are in the vicinity of instances with a different label. A positive instance that is surrounded on all sides by many positive instances is not needed, nor is a negative instance surrounded by many negative instances. We discuss such *neighbor-based* methods in chapter 8.

5. If we have a supervisor who can provide us with the label for any $\boldsymbol{x}$, where should we choose $\boldsymbol{x}$ to learn with fewer queries?

   SOLUTION: The region of ambiguity is between S and G. It would be best to be given queries there, so that we can make this region of doubt smaller. If a

given instance there turns out to be positive, this means we can make S larger up to that instance; if it is negative, this means we can shrink G down until there.

6. In equation 2.13, we summed up the squares of the differences between the actual value and the estimated value. This error function is the one most frequently used, but it is one of several possible error functions. Because it sums up the squares of the differences, it is not robust to outliers. What would be a better error function to implement *robust regression*?

7. Derive equation 2.17.

8. Assume our hypothesis class is the set of lines, and we use a line to separate the positive and negative examples, instead of bounding the positive examples as in a rectangle, leaving the negatives outside (see figure 2.13). Show that the VC dimension of a line is 3.

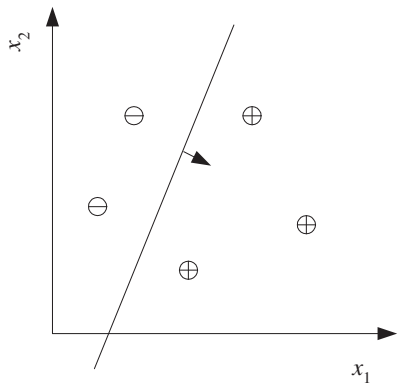

Figure 2.13 A line separating positive and negative instances.

9. Show that the VC dimension of the triangle hypothesis class is 7 in two dimensions. (Hint: For best separation, it is best to place the seven points equidistant on a circle.)

10. Assume as in exercise 8 that our hypothesis class is the set of lines. Write down an error function that not only minimizes the number of misclassifications but also maximizes the margin.

11. One source of noise is error in the labels. Can you propose a method to find data points that are highly likely to be mislabeled?

2.11 References

Angluin, D. 1988. "Queries and Concept Learning." *Machine Learning* 2:319–342.

Blumer, A., A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. 1989. "Learnability and the Vapnik-Chervonenkis Dimension." *Journal of the ACM* 36:929–965.

Boyd, S., and L. Vandenberge. 2004. *Convex Optimization*. Cambridge, UK: Cambridge University Press.

Dietterich, T. G. 2003. "Machine Learning." In *Nature Encyclopedia of Cognitive Science*. London: Macmillan.

Hirsh, H. 1990. *Incremental Version Space Merging: A General Framework for Concept Learning*. Boston: Kluwer.

Kearns, M. J., and U. V. Vazirani. 1994. *An Introduction to Computational Learning Theory*. Cambridge, MA: MIT Press.

Mitchell, T. 1997. *Machine Learning*. New York: McGraw-Hill.

Valiant, L. 1984. "A Theory of the Learnable." *Communications of the ACM* 27:1134–1142.

Vapnik, V. N. 1995. *The Nature of Statistical Learning Theory*. New York: Springer.

Winston, P. H. 1975. "Learning Structural Descriptions from Examples." In *The Psychology of Computer Vision*, ed. P. H. Winston, 157–209. New York: McGraw-Hill.

teriors, $P(C_i|x)$, and make our decision. We then discuss *regression* where the estimated density is $p(y|x)$. In this chapter, x is one-dimensional and thus the densities are univariate. We generalize to the multivariate case in chapter 5.

4.2 Maximum Likelihood Estimation

Let us say we have an independent and identically distributed (iid) sample $\mathcal{X} = \{x^t\}_{t=1}^N$. We assume that x^t are instances drawn from some known probability density family, $p(x|\theta)$, defined up to parameters, θ:

$$x^t \sim p(x|\theta)$$

LIKELIHOOD

We want to find θ that makes sampling x^t from $p(x|\theta)$ as likely as possible. Because x^t are independent, the *likelihood* of parameter θ given sample $\mathcal{X}$ is the product of the likelihoods of the individual points:

$$l(\theta|\mathcal{X}) \equiv p(\mathcal{X}|\theta) = \prod_{t=1}^{N} p(x^t|\theta) \tag{4.1}$$

MAXIMUM LIKELIHOOD ESTIMATION

In *maximum likelihood estimation*, we are interested in finding θ that makes $\mathcal{X}$ the most likely to be drawn. We thus search for θ that maximizes the likelihood, which we denote by $l(\theta|\mathcal{X})$. We can maximize the log of the likelihood without changing the value where it takes its maximum. $\log(\cdot)$ converts the product into a sum and leads to further computational simplification when certain densities are assumed, for example, containing exponents. The *log likelihood* is defined as

LOG LIKELIHOOD

$$\mathcal{L}(\theta|\mathcal{X}) \equiv \log l(\theta|\mathcal{X}) = \sum_{t=1}^{N} \log p(x^t|\theta) \tag{4.2}$$

Let us now see some distributions that arise in the applications we are interested in. If we have a two-class problem, the distribution we use is *Bernoulli*. When there are $K > 2$ classes, its generalization is the *multinomial*. *Gaussian* (*normal*) density is the one most frequently used for modeling class-conditional input densities with numeric input. For these three distributions, we discuss the maximum likelihood estimators (MLE) of their parameters.

4.2.1 Bernoulli Density

In a Bernoulli distribution, there are two outcomes: An event occurs or it does not; for example, an instance is a positive example of the class, or it is not. The event occurs and the Bernoulli random variable X takes the value 1 with probability p, and the nonoccurrence of the event has probability $1 - p$ and this is denoted by X taking the value 0. This is written as

$$P(x) = p^x(1-p)^{1-x}, x \in \{0,1\} \tag{4.3}$$

The expected value and variance can be calculated as

$$\begin{aligned} E[X] &= \sum_x xp(x) = 1 \cdot p + 0 \cdot (1-p) = p \\ \text{Var}(X) &= \sum_x (x - E[X])^2 p(x) = p(1-p) \end{aligned}$$

p is the only parameter and given an iid sample $\mathcal{X} = \{x^t\}_{t=1}^N$, where $x^t \in \{0,1\}$, we want to calculate its estimator, $\hat{p}$. The log likelihood is

$$\begin{aligned} \mathcal{L}(p|\mathcal{X}) &= \log \prod_{t=1}^N p^{(x^t)}(1-p)^{(1-x^t)} \\ &= \sum_t x^t \log p + \left(N - \sum_t x^t\right) \log(1-p) \end{aligned}$$

$\hat{p}$ that maximizes the log likelihood can be found by solving for $d\mathcal{L}/dp = 0$. The hat (circumflex) denotes that it is an estimate.

$$\hat{p} = \frac{\sum_t x^t}{N} \tag{4.4}$$

The estimate for p is the ratio of the number of occurrences of the event to the number of experiments. Remembering that if X is Bernoulli with p, $E[X] = p$, and, as expected, the maximum likelihood estimator of the mean is the sample average.

Note that the estimate is a function of the sample and is another random variable; we can talk about the distribution of $\hat{p}_i$ given different $\mathcal{X}_i$ sampled from the same $p(x)$. For example, the variance of the distribution of $\hat{p}_i$ is expected to decrease as N increases; as the samples get bigger, they (and hence their averages) get more similar.

4.2.2 Multinomial Density

Consider the generalization of Bernoulli where instead of two states, the outcome of a random event is one of K mutually exclusive and exhaustive states, for example, classes, each of which has a probability of occurring p_i with $\sum_{i=1}^{K} p_i = 1$. Let $x_1, x_2, \ldots, x_K$ are the indicator variables where x_i is 1 if the outcome is state i and 0 otherwise.

$$P(x_1, x_2, \ldots, x_K) = \prod_{i=1}^{K} p_i^{x_i} \tag{4.5}$$

Let us say we do N such independent experiments with outcomes $\mathcal{X} = \{\boldsymbol{x}^t\}_{t=1}^{N}$ where

$$x_i^t = \begin{cases} 1 & \text{if experiment } t \text{ chooses state } i \\ 0 & \text{otherwise} \end{cases}$$

with $\sum_i x_i^t = 1$. The MLE of p_i is

$$\hat{p}_i = \frac{\sum_t x_i^t}{N} \tag{4.6}$$

The estimate for the probability of state i is the ratio of experiments with outcome of state i to the total number of experiments. There are two ways one can get this: If x_i are 0/1, then they can be thought of as K separate Bernoulli experiments. Or, one can explicitly write the log likelihood and find p_i that maximize it (subject to the condition that $\sum_i p_i = 1$).

4.2.3 Gaussian (Normal) Density

X is Gaussian (normal) distributed with mean $E[X] \equiv \mu$ and variance $\text{Var}(X) \equiv \sigma^2$, denoted as $\mathcal{N}(\mu, \sigma^2)$, if its density function is

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right], -\infty < x < \infty \tag{4.7}$$

Given a sample $\mathcal{X} = \{x^t\}_{t=1}^{N}$ with $x^t \sim \mathcal{N}(\mu, \sigma^2)$, the log likelihood is

$$\mathcal{L}(\mu, \sigma | \mathcal{X}) = -\frac{N}{2}\log(2\pi) - N\log\sigma - \frac{\sum_t (x^t - \mu)^2}{2\sigma^2}$$

The MLE that we find by taking the partial derivatives of the log likelihood and setting them equal to 0 are

$$\begin{aligned} m &= \frac{\sum_t x^t}{N} \\ s^2 &= \frac{\sum_t (x^t - m)^2}{N} \end{aligned} \tag{4.8}$$

We follow the usual convention and use Greek letters for the population parameters and Roman letters for their estimates from the sample. Sometimes, the hat is also used to denote the estimator, for example, $\hat{\mu}$.

4.3 Evaluating an Estimator: Bias and Variance

Let $\mathcal{X}$ be a sample from a population specified up to a parameter θ, and let $d = d(\mathcal{X})$ be an estimator of θ. To evaluate the quality of this estimator, we can measure how much it is different from θ, that is, $(d(\mathcal{X}) - \theta)^2$. But since it is a random variable (it depends on the sample), we need to average this over possible $\mathcal{X}$ and consider $r(d, \theta)$, the *mean square error* of the estimator d defined as

MEAN SQUARE ERROR

$$r(d, \theta) = E[(d(\mathcal{X}) - \theta)^2] \tag{4.9}$$

BIAS

The *bias* of an estimator is given as

$$b_\theta(d) = E[d(\mathcal{X})] - \theta \tag{4.10}$$

UNBIASED ESTIMATOR

If $b_\theta(d) = 0$ for all θ values, then we say that d is an *unbiased estimator* of θ. For example, with x^t drawn from some density with mean μ, the sample average, m, is an unbiased estimator of the mean, μ, because

$$E[m] = E\left[\frac{\sum_t x^t}{N}\right] = \frac{1}{N}\sum_t E[x^t] = \frac{N\mu}{N} = \mu$$

This means that though on a particular sample, m may be different from μ, if we take many such samples, $\mathcal{X}_i$, and estimate many $m_i = m(\mathcal{X}_i)$, *their* average will get close to μ as the number of such samples increases. m is also a *consistent* estimator, that is, $\text{Var}(m) \to 0$ as $N \to \infty$.

$$\text{Var}(m) = \text{Var}\left(\frac{\sum_t x^t}{N}\right) = \frac{1}{N^2}\sum_t \text{Var}(x^t) = \frac{N\sigma^2}{N^2} = \frac{\sigma^2}{N}$$

As N, the number of points in the sample, gets larger, m deviates less from μ. Let us now check, s^2, the MLE of σ^2:

$$\begin{aligned} s^2 &= \frac{\sum_t (x^t - m)^2}{N} = \frac{\sum_t (x^t)^2 - Nm^2}{N} \\ E[s^2] &= \frac{\sum_t E[(x^t)^2] - N \cdot E[m^2]}{N} \end{aligned}$$

Given that $\text{Var}(X) = E[X^2] - E[X]^2$, we get $E[X^2] = \text{Var}(X) + E[X]^2$, and we can write

$$E[(x^t)^2] = \sigma^2 + \mu^2 \text{ and } E[m^2] = \sigma^2/N + \mu^2$$

Then, plugging these in, we get

$$E[s^2] = \frac{N(\sigma^2 + \mu^2) - N(\sigma^2/N + \mu^2)}{N} = \left(\frac{N-1}{N}\right)\sigma^2 \neq \sigma^2$$

which shows that s^2 is a biased estimator of σ^2. $(N/(N-1))s^2$ is an unbiased estimator. However when N is large, the difference is negligible. This is an example of an *asymptotically unbiased estimator* whose bias goes to 0 as N goes to infinity.

The mean square error can be rewritten as follows—d is short for $d(\mathcal{X})$:

$$\begin{aligned} r(d,\theta) &= E\left[(d-\theta)^2\right] \\ &= E\left[(d - E[d] + E[d] - \theta)^2\right] \\ &= E\left[(d-E[d])^2 + (E[d]-\theta)^2 + 2(E[d]-\theta)(d-E[d])\right] \\ &= E\left[(d-E[d])^2\right] + E\left[(E[d]-\theta)^2\right] + 2E\left[(E[d]-\theta)(d-E[d])\right] \\ &= E\left[(d-E[d])^2\right] + (E[d]-\theta)^2 + 2(E[d]-\theta)E[d-E[d]] \\ &= \underbrace{E\left[(d-E[d])^2\right]}_{variance} + \underbrace{(E[d]-\theta)^2}_{bias^2} \end{aligned} \tag{4.11}$$

The two equalities follow because $E[d]$ is a constant and therefore $E[d] - \theta$ also is a constant, and because $E[d - E[d]] = E[d] - E[d] = 0$. In equation 4.11, the first term is the *variance* that measures how much, on average, d_i vary around the expected value (going from one dataset to another), and the second term is the *bias* that measures how much the expected value varies from the correct value θ (figure 4.1). We then write error as the sum of these two terms, the variance and the square of the bias:

VARIANCE

$$r(d,\theta) = \mathrm{Var}(d) + (b_\theta(d))^2 \tag{4.12}$$

4.4 The Bayes' Estimator

Sometimes, before looking at a sample, we (or experts of the application) may have some *prior* information on the possible value range that a parameter, θ, may take. This information is quite useful and should be used, especially when the sample is small. The prior information does not tell us exactly what the parameter value is (otherwise we would not

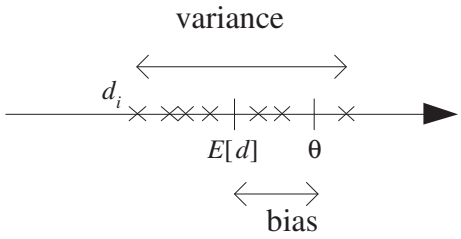

Figure 4.1 θ is the parameter to be estimated. d_i are several estimates (denoted by '$\times$') over different samples $\mathcal{X}_i$. Bias is the difference between the expected value of d and θ. Variance is how much d_i are scattered around the expected value. We would like both to be small.

need the sample), and we model this uncertainty by viewing θ as a random variable and by defining a prior density for it, $p(\theta)$. For example, let us say we are told that θ is approximately normal and with 90 percent confidence, θ lies between 5 and 9, symmetrically around 7. Then we can write $p(\theta)$ to be normal with mean 7 and because

$$\begin{aligned} P\{-1.64 < \frac{\theta - \mu}{\sigma} < 1.64\} &= 0.9 \\ P\{\mu - 1.64\sigma < \theta < \mu + 1.64\sigma\} &= 0.9 \end{aligned}$$

we take $1.64\sigma = 2$ and use $\sigma = 2/1.64$. We can thus assume $p(\theta) \sim \mathcal{N}(7, (2/1.64)^2)$.

PRIOR DENSITY

The *prior density*, $p(\theta)$, tells us the likely values that θ may take *before* looking at the sample. We combine this with what the sample data tells us, namely, the likelihood density, $p(\mathcal{X}|\theta)$, using Bayes' rule, and get the

POSTERIOR DENSITY

posterior density of θ, which tells us the likely θ values *after* looking at the sample:

$$p(\theta|\mathcal{X}) = \frac{p(\mathcal{X}|\theta)p(\theta)}{p(\mathcal{X})} = \frac{p(\mathcal{X}|\theta)p(\theta)}{\int p(\mathcal{X}|\theta')p(\theta')d\theta'} \tag{4.13}$$

For estimating the density at x, we have

$$\begin{aligned} p(x|\mathcal{X}) &= \int p(x, \theta|\mathcal{X})d\theta \\ &= \int p(x|\theta, \mathcal{X})p(\theta|\mathcal{X})d\theta \\ &= \int p(x|\theta)p(\theta|\mathcal{X})d\theta \end{aligned}$$

$p(x|\theta, \mathcal{X}) = p(x|\theta)$ because once we know θ, the sufficient statistics, we know everything about the distribution. Thus we are taking an average over predictions using all values of θ, weighted by their probabilities. If we are doing a prediction in the form, $y = g(x|\theta)$, as in regression, then we have

$$y = \int g(x|\theta)p(\theta|\mathcal{X})d\theta$$

Evaluating the integrals may be quite difficult, except in cases where the posterior has a nice form. When the full integration is not feasible, we reduce it to a single point. If we can assume that $p(\theta|\mathcal{X})$ has a narrow peak around its mode, then using the *maximum a posteriori* (MAP) *estimate* will make the calculation easier:

MAXIMUM A POSTERIORI ESTIMATE

$$\theta_{MAP} = \arg\max_{\theta} p(\theta|\mathcal{X}) \tag{4.14}$$

thus replacing a whole density with a single point, getting rid of the integral and using as

$$\begin{aligned} p(x|\mathcal{X}) &= p(x|\theta_{MAP}) \\ y_{MAP} &= g(x|\theta_{MAP}) \end{aligned}$$

If we have no prior reason to favor some values of θ, then the prior density is flat and the posterior will have the same form as the likelihood, $p(\mathcal{X}|\theta)$, and the MAP estimate will be equivalent to the maximum likelihood estimate (section 4.2) where we have

$$\theta_{ML} = \arg\max_{\theta} p(\mathcal{X}|\theta) \tag{4.15}$$

BAYES' ESTIMATOR

Another possibility is the *Bayes' estimator*, which is defined as the expected value of the posterior density

$$\theta_{Bayes} = E[\theta|\mathcal{X}] = \int \theta p(\theta|\mathcal{X})d\theta \tag{4.16}$$

The reason for taking the expected value is that the best estimate of a random variable is its mean. Let us say θ is the variable we want to predict with $E[\theta] = \mu$. It can be shown that if c, a constant value, is our estimate of θ, then

$$\begin{aligned} E[(\theta - c)^2] &= E[(\theta - \mu + \mu - c)^2] \\ &= E[(\theta - \mu)^2] + (\mu - c)^2 \end{aligned} \tag{4.17}$$

which is minimum if c is taken as μ. In the case of a normal density, the mode is the expected value and if $p(\theta|\mathcal{X})$ is normal, then $\theta_{Bayes} = \theta_{MAP}$.

As an example, let us suppose $x^t \sim \mathcal{N}(\theta, \sigma^2)$ and $\theta \sim \mathcal{N}(\mu_0, \sigma_0^2)$, where μ_0, σ_0^2, and σ^2 are known:

$$\begin{aligned} p(\mathcal{X}|\theta) &= \frac{1}{(2\pi)^{N/2}\sigma^N} \exp\left[-\frac{\sum_t (x^t-\theta)^2}{2\sigma^2}\right] \\ p(\theta) &= \frac{1}{\sqrt{2\pi}\sigma_0} \exp\left[-\frac{(\theta-\mu_0)^2}{2\sigma_0^2}\right] \end{aligned}$$

It can be shown that $p(\theta|\mathcal{X})$ is normal with

$$E[\theta|\mathcal{X}] = \frac{N/\sigma^2}{N/\sigma^2 + 1/\sigma_0^2} m + \frac{1/\sigma_0^2}{N/\sigma^2 + 1/\sigma_0^2} \mu_0 \tag{4.18}$$

Thus the Bayes' estimator is a weighted average of the prior mean μ_0 and the sample mean m, with weights being inversely proportional to their variances. As the sample size N increases, the Bayes' estimator gets closer to the sample average, using more the information provided by the sample. When σ_0^2 is small, that is, when we have little prior uncertainty regarding the correct value of θ, or when N is small, our prior guess μ_0 has a higher effect.

Note that both MAP and Bayes' estimators reduce the whole posterior density to a single point and lose information unless the posterior is unimodal and makes a narrow peak around these points. With computation getting cheaper, we can use a Monte Carlo approach that generates samples from the posterior density (Andrieu et al. 2003). There also are approximation methods one can use to evaluate the full integral. We are going to discuss Bayesian estimation in more detail in chapter 16.

4.5 Parametric Classification

We saw in chapter 3 that using the Bayes' rule, we can write the posterior probability of class C_i as

$$P(C_i|x) = \frac{p(x|C_i)P(C_i)}{p(x)} = \frac{p(x|C_i)P(C_i)}{\sum_{k=1}^K p(x|C_k)P(C_k)} \tag{4.19}$$

and use the discriminant function

$$g_i(x) = p(x|C_i)P(C_i)$$

or equivalently

$$g_i(x) = \log p(x|C_i) + \log P(C_i) \tag{4.20}$$

If we can assume that $p(x|C_i)$ are Gaussian

$$p(x|C_i) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left[-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right] \tag{4.21}$$

equation 4.20 becomes

$$g_i(x) = -\frac{1}{2}\log 2\pi - \log \sigma_i - \frac{(x-\mu_i)^2}{2\sigma_i^2} + \log P(C_i) \tag{4.22}$$

Let us see an example: Assume we are a car company selling K different cars, and for simplicity, let us say that the sole factor that affects a customer's choice is his or her yearly income, which we denote by x. Then $P(C_i)$ is the proportion of customers who buy car type i. If the yearly income distributions of such customers can be approximated with a Gaussian, then $p(x|C_i)$, the probability that a customer who bought car type i has income x, can be taken $\mathcal{N}(\mu_i, \sigma_i^2)$, where μ_i is the mean income of such customers and σ_i^2 is their income variance.

When we do not know $P(C_i)$ and $p(x|C_i)$, we estimate them from a sample and plug in their estimates to get the estimate for the discriminant function. We are given a sample

$$\mathcal{X} = \{x^t, \boldsymbol{r}^t\}_{t=1}^N \tag{4.23}$$

where $x \in \Re$ is one-dimensional and $\boldsymbol{r} \in \{0,1\}^K$ such that

$$r_i^t = \begin{cases} 1 & \text{if } \boldsymbol{x}^t \in C_i \\ 0 & \text{if } \boldsymbol{x}^t \in C_k, k \neq i \end{cases} \tag{4.24}$$

For each class separately, the estimates for the means and variances are (relying on equation 4.8)

$$m_i = \frac{\sum_t x^t r_i^t}{\sum_t r_i^t} \tag{4.25}$$

$$s_i^2 = \frac{\sum_t (x^t - m_i)^2 r_i^t}{\sum_t r_i^t} \tag{4.26}$$

and the estimates for the priors are (relying on equation 4.6)

$$\hat{P}(C_i) = \frac{\sum_t r_i^t}{N} \tag{4.27}$$

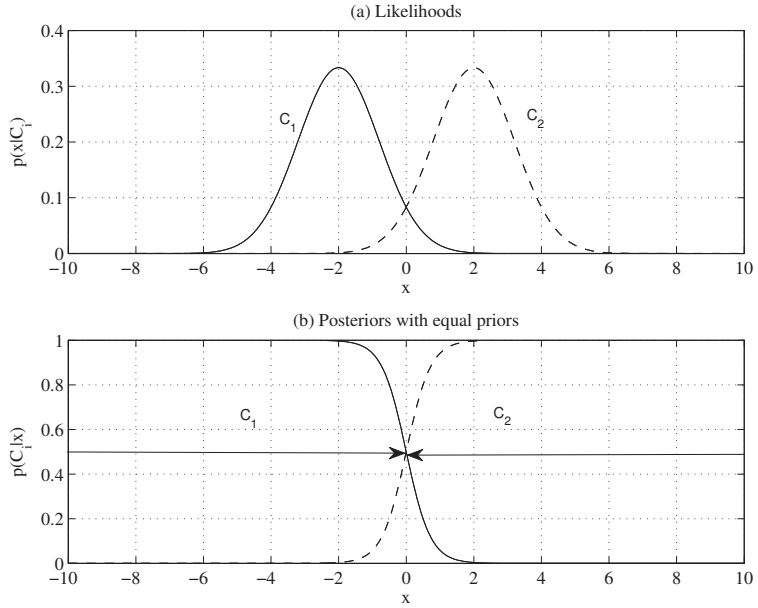

Figure 4.2 (a) Likelihood functions and (b) posteriors with equal priors for two classes when the input is one-dimensional. Variances are equal and the posteriors intersect at one point, which is the threshold of decision.

Plugging these estimates into equation 4.22, we get

$$g_i(x) = -\frac{1}{2}\log 2\pi - \log s_i - \frac{(x-m_i)^2}{2s_i^2} + \log \hat{P}(C_i) \tag{4.28}$$

The first term is a constant and can be dropped because it is common in all $g_i(x)$. If the priors are equal, the last term can also be dropped. If we can further assume that variances are equal, we can write

$$g_i(x) = -(x-m_i)^2 \tag{4.29}$$

and thus we assign x to the class with the nearest mean:

Choose C_i if $|x-m_i| = \min_k |x-m_k|$

With two adjacent classes, the midpoint between the two means is the threshold of decision (see figure 4.2).

$$g_1(x) = g_2(x)$$

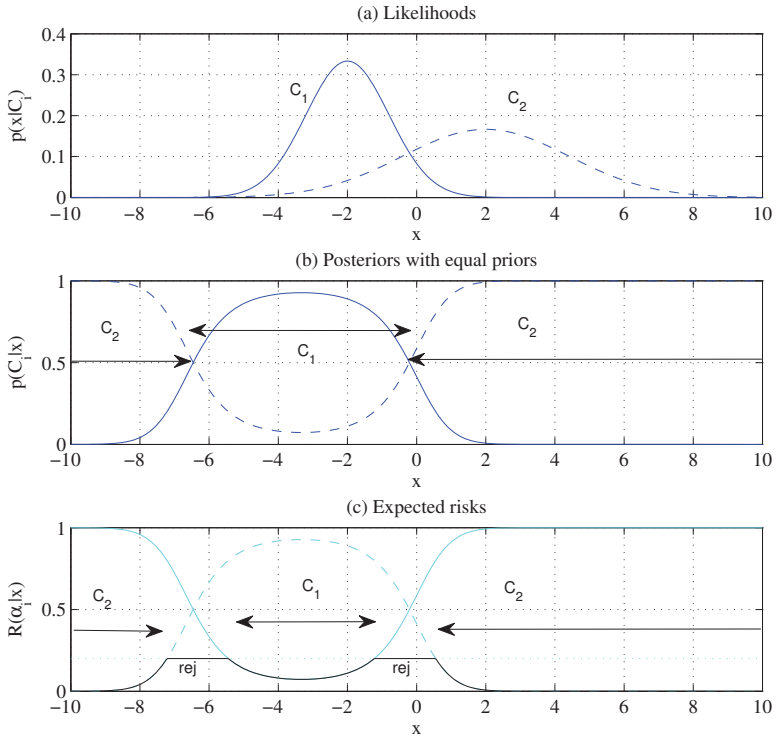

Figure 4.3 (a) Likelihood functions and (b) posteriors with equal priors for two classes when the input is one-dimensional. Variances are unequal and the posteriors intersect at two points. In (c), the expected risks are shown for the two classes and for reject with $\lambda = 0.2$ (section 3.3).

$$\begin{aligned}(x - m_1)^2 &= (x - m_2)^2 \\ x &= \frac{m_1 + m_2}{2}\end{aligned}$$

When the variances are different, there are two thresholds (see figure 4.3), which can be calculated easily (exercise 4). If the priors are different, this has the effect of moving the threshold of decision toward the mean of the less likely class.

Here we use the maximum likelihood estimators for the parameters but if we have some prior information about them, for example, for the means, we can use a Bayesian estimate of $p(x|C_i)$ with prior on μ_i.

One note of caution is necessary here: When x is continuous, we should not immediately rush to use Gaussian densities for $p(x|C_i)$. The classification algorithm—that is, the threshold points—will be wrong if the densities are not Gaussian. In statistical literature, tests exist to check for normality, and such a test should be used before assuming normality. In the case of one-dimensional data, the easiest test is to plot the histogram and to check visually whether the density is bell-shaped, namely, unimodal and symmetric around the center.

This is the *likelihood-based approach* to classification where we use data to estimate the densities separately, calculate posterior densities using Bayes' rule, and then get the discriminant. In later chapters, we discuss the *discriminant-based approach* where we bypass the estimation of densities and directly estimate the discriminants.

4.6 Regression

In regression, we would like to write the numeric output, called the *dependent variable*, as a function of the input, called the *independent variable*. We assume that the numeric output is the sum of a deterministic function of the input and random noise:

$$r = f(x) + \epsilon$$

where $f(x)$ is the unknown function, which we would like to approximate by our estimator, $g(x|\theta)$, defined up to a set of parameters θ. If we assume that ϵ is zero mean Gaussian with constant variance σ^2, namely, $\epsilon \sim \mathcal{N}(0, \sigma^2)$, and placing our estimator $g(\cdot)$ in place of the unknown function $f(\cdot)$, we have (figure 4.4)

$$p(r|x) \sim \mathcal{N}(g(x|\theta), \sigma^2) \tag{4.30}$$

We again use maximum likelihood to learn the parameters θ. The pairs (x^t, r^t) in the training set are drawn from an unknown joint probability density $p(x, r)$, which we can write as

$$p(x, r) = p(r|x)p(x)$$

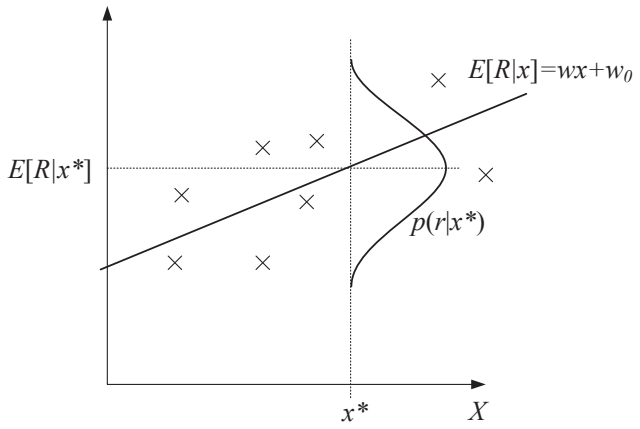

Figure 4.4 Regression assumes 0 mean Gaussian noise added to the model; here, the model is linear.

$p(r|x)$ is the probability of the output given the input, and $p(x)$ is the input density. Given an iid sample $\mathcal{X} = \{x^t, r^t\}_{t=1}^N$, the log likelihood is

$$\begin{aligned}\mathcal{L}(\theta|\mathcal{X}) &= \log \prod_{t=1}^{N} p(x^t, r^t) \\ &= \log \prod_{t=1}^{N} p(r^t|x^t) + \log \prod_{t=1}^{N} p(x^t)\end{aligned}$$

We can ignore the second term since it does not depend on our estimator, and we have

$$\begin{aligned}\mathcal{L}(\theta|\mathcal{X}) &= \log \prod_{t=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{[r^t - g(x^t|\theta)]^2}{2\sigma^2}\right] \\ &= \log\left(\frac{1}{\sqrt{2\pi}\sigma}\right)^N \exp\left[-\frac{1}{2\sigma^2}\sum_{t=1}^{N}[r^t - g(x^t|\theta)]^2\right] \\ &= -N\log(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2}\sum_{t=1}^{N}[r^t - g(x^t|\theta)]^2\end{aligned} \tag{4.31}$$

The first term is independent of the parameters θ and can be dropped, as can the factor $1/\sigma^2$. Maximizing this is equivalent to minimizing

$$E(\theta|\mathcal{X}) = \frac{1}{2}\sum_{t=1}^{N}[r^t - g(x^t|\theta)]^2 \tag{4.32}$$

LEAST SQUARES ESTIMATE

which is the most frequently used error function, and θ that minimize it are called the *least squares estimates*. This is a transformation frequently done in statistics: When the likelihood l contains exponents, instead of maximizing l, we define an *error function*, $E = -\log l$, and minimize it.

LINEAR REGRESSION

In *linear regression*, we have a linear model

$$g(x^t|w_1, w_0) = w_1 x^t + w_0$$

and taking the derivative of the sum of squared errors (equation 4.32) with respect to w_1 and w_0, we have two equations in two unknowns

$$\begin{aligned} \sum_t r^t &= N w_0 + w_1 \sum_t x^t \\ \sum_t r^t x^t &= w_0 \sum_t x_t + w_1 \sum_t (x^t)^2 \end{aligned}$$

which can be written in vector-matrix form as $\mathbf{A}\boldsymbol{w} = \boldsymbol{y}$ where

$$\mathbf{A} = \begin{bmatrix} N & \sum_t x^t \\ \sum_t x^t & \sum_t (x^t)^2 \end{bmatrix}, \quad \boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} \sum_t r^t \\ \sum_t r^t x^t \end{bmatrix}$$

and can be solved as $\boldsymbol{w} = \mathbf{A}^{-1}\boldsymbol{y}$.

POLYNOMIAL REGRESSION

In the general case of *polynomial regression*, the model is a polynomial in x of order k

$$g(x^t|w_k, \ldots, w_2, w_1, w_0) = w_k (x^t)^k + \cdots + w_2 (x^t)^2 + w_1 x^t + w_0$$

The model is still linear with respect to the parameters and taking the derivatives, we get $k+1$ equations in $k+1$ unknowns, which can be written in vector matrix form $\mathbf{A}\boldsymbol{w} = \boldsymbol{y}$ where we have

$$\mathbf{A} = \begin{bmatrix} N & \sum_t x^t & \sum_t (x^t)^2 & \cdots & \sum_t (x^t)^k \\ \sum_t x^t & \sum_t (x^t)^2 & \sum_t (x^t)^3 & \cdots & \sum_t (x^t)^{k+1} \\ \vdots & & & & \\ \sum_t (x^t)^k & \sum_t (x^t)^{k+1} & \sum_t (x^t)^{k+2} & \cdots & \sum_t (x^t)^{2k} \end{bmatrix}$$

$$\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_k \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} \sum_t r^t \\ \sum_t r^t x^t \\ \sum_t r^t (x^t)^2 \\ \vdots \\ \sum_t r^t (x^t)^k \end{bmatrix}$$

We can write $\mathbf{A} = \mathbf{D}^T\mathbf{D}$ and $\boldsymbol{y} = \mathbf{D}^T\boldsymbol{r}$ where

$$\mathbf{D} = \begin{bmatrix} 1 & x^1 & (x^1)^2 & \cdots & (x^1)^k \\ 1 & x^2 & (x^2)^2 & \cdots & (x^2)^k \\ \vdots & & & & \\ 1 & x^N & (x^N)^2 & \cdots & (x^N)^k \end{bmatrix}, \boldsymbol{r} = \begin{bmatrix} r^1 \\ r^2 \\ \vdots \\ r^N \end{bmatrix}$$

and we can then solve for the parameters as

$$\boldsymbol{w} = (\mathbf{D}^T\mathbf{D})^{-1}\mathbf{D}^T\boldsymbol{r} \tag{4.33}$$

Assuming Gaussian distributed error and maximizing likelihood corresponds to minimizing the sum of squared errors. Another measure is the *relative square error* (RSE):

RELATIVE SQUARE ERROR

$$E_{RSE} = \frac{\sum_t [r^t - g(x^t|\theta)]^2}{\sum_t (r^t - \overline{r})^2} \tag{4.34}$$

If E_{RSE} is close to 1, then our prediction is as good as predicting by the average; as it gets closer to 0, we have better fit. If E_{RSE} is close to 1, this means that using a model based on input x does not work better than using the average which would be our estimator if there were no x; if E_{RSE} is close to 0, input x helps.

COEFFICIENT OF DETERMINATION

A measure to check the goodness of fit by regression is the *coefficient of determination* that is

$$R^2 = 1 - E_{RSE}$$

and for regression to be considered useful, we require R^2 to be close to 1.

Remember that for best generalization, we should adjust the complexity of our learner model to the complexity of the data. In polynomial regression, the complexity parameter is the order of the fitted polynomial, and therefore we need to find a way to choose the best order that minimizes the generalization error, that is, tune the complexity of the model to best fit the complexity of the function inherent in the data.

4.7 Tuning Model Complexity: Bias/Variance Dilemma

Let us say that a sample $\mathcal{X} = \{x^t, r^t\}$ is drawn from some unknown joint probability density $p(x, r)$. Using this sample, we construct our estimate

$g(\cdot)$. The expected square error (over the joint density) at x can be written as (using equation 4.17)

$$E[(r-g(x))^2|x] = \underbrace{E[(r-E[r|x])^2|x]}_{noise} + \underbrace{(E[r|x]-g(x))^2}_{squared\ error} \tag{4.35}$$

The first term on the right is the variance of r given x; it does not depend on $g(\cdot)$ or $\mathcal{X}$. It is the variance of noise added, σ^2. This is the part of error that can never be removed, no matter what estimator we use. The second term quantifies how much $g(x)$ deviates from the regression function, $E[r|x]$. This does depend on the estimator and the training set. It may be the case that for one sample, $g(x)$ may be a very good fit; and for some other sample, it may make a bad fit. To quantify how well an estimator $g(\cdot)$ is, we average over possible datasets.

The expected value (average over samples $\mathcal{X}$, all of size N and drawn from the same joint density $p(r,x)$) is (using equation 4.11)

$$E_{\mathcal{X}}[(E[r|x]-g(x))^2|x] = \underbrace{(E[r|x]-E_{\mathcal{X}}[g(x)])^2}_{bias} + \underbrace{E_{\mathcal{X}}[(g(x)-E_{\mathcal{X}}[g(x)])^2]}_{variance} \tag{4.36}$$

As we discussed earlier, bias measures how much $g(x)$ is wrong disregarding the effect of varying samples, and variance measures how much $g(x)$ fluctuate around the expected value, $E[g(x)]$, as the sample varies. We want both to be small.

Let us see a didactic example: To estimate the bias and the variance, we generate a number of datasets $\mathcal{X}_i = \{x_i^t, r_i^t\}, i = 1,\ldots,M$, from some known $f(\cdot)$ with added noise, use each dataset to form an estimator $g_i(\cdot)$, and calculate bias and variance. Note that in real life, we cannot do this because we do not know $f(\cdot)$ or the parameters of the added noise. Then $E[g(x)]$ is estimated by the average over $g_i(\cdot)$:

$$\bar{g}(x) = \frac{1}{M}\sum_{i=1}^{M} g_i(x)$$

Estimated bias and variance are

$$\begin{aligned}\text{bias}^2(g) &= \frac{1}{N}\sum_t [\bar{g}(x^t) - f(x^t)]^2\\ \text{variance}(g) &= \frac{1}{NM}\sum_t\sum_i [g_i(x^t) - \bar{g}(x^t)]^2\end{aligned}$$

Let us see some models of different complexity: The simplest is a constant fit

$$g_i(x) = 2$$

This has no variance because we do not use the data and all $g_i(x)$ are the same. But the bias is high, unless of course $f(x)$ is close to 2 for all x. If we take the average of r^t in the sample

$$g_i(x) = \sum_t r_i^t / N$$

instead of the constant 2, this decreases the bias because we would expect the average in general to be a better estimate. But this increases the variance because the different samples $\mathcal{X}_i$ would have different average values. Normally in this case the decrease in bias would be larger than the increase in variance, and error would decrease.

In the context of polynomial regression, an example is given in figure 4.5. As the order of the polynomial increases, small changes in the dataset cause a greater change in the fitted polynomials; thus variance increases. But a complex model on the average allows a better fit to the underlying function; thus bias decreases (see figure 4.6). This is called the *bias/variance dilemma* and is true for any machine learning system and not only for polynomial regression (Geman, Bienenstock, and Doursat 1992). To decrease bias, the model should be flexible, at the risk of having high variance. If the variance is kept low, we may not be able to make a good fit to data and have high bias. The optimal model is the one that has the best trade-off between the bias and the variance.

BIAS/VARIANCE DILEMMA

If there is bias, this indicates that our model class does not contain the solution; this is *underfitting*. If there is variance, the model class is too general and also learns the noise; this is *overfitting*. If $g(\cdot)$ is of the same hypothesis class with $f(\cdot)$, for example, a polynomial of the same order, we have an unbiased estimator, and estimated bias decreases as the number of models increases. This shows the error-reducing effect of choosing the right model (which we called *inductive bias* in chapter 2—the two uses of "bias" are different but not unrelated). As for variance, it also depends on the size of the training set; the variability due to sample decreases as the sample size increases. To sum up, to get a small value of error, we should have the proper inductive bias (to get small bias in the statistical sense) and have a large enough dataset so that the variability of the model can be constrained with the data.

UNDERFITTING
OVERFITTING

Note that when the variance is large, bias is low: This indicates that $\overline{g}(x)$ is a good estimator. So to get a small value of error, we can take a large number of high-variance models and use their average as our estimator. We discuss such approaches for model combination in chapter 17.

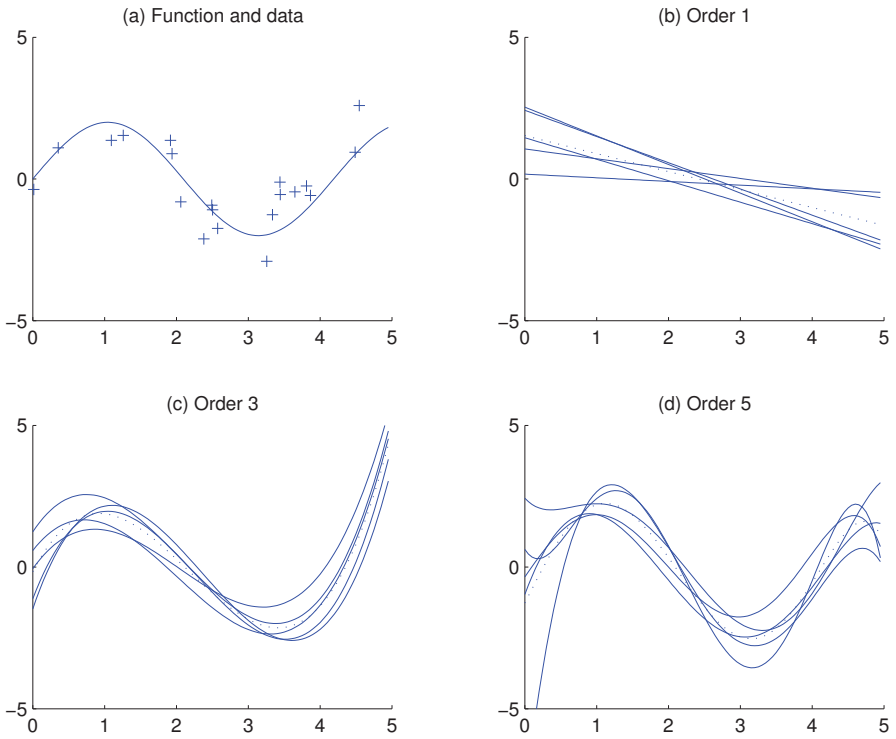

Figure 4.5 (a) Function, $f(x) = 2\sin(1.5x)$, and one noisy ($\mathcal{N}(0,1)$) dataset sampled from the function. Five samples are taken, each containing twenty instances. (b), (c), (d) are five polynomial fits, namely, $g_i(\cdot)$, of order 1, 3, and 5. For each case, dotted line is the average of the five fits, namely, $\overline{g}(\cdot)$.

4.8 Model Selection Procedures

There are a number of procedures we can use to fine-tune model complexity.

CROSS-VALIDATION

In practice, the method we use to find the optimal complexity is *cross-validation*. We cannot calculate the bias and variance for a model, but we can calculate the total error. Given a dataset, we divide it into two parts as training and validation sets, train candidate models of different complexities on the training set and test their error on the validation set left out during training. As the model complexity increases, training error keeps decreasing. The error on the validation set however decreases up to

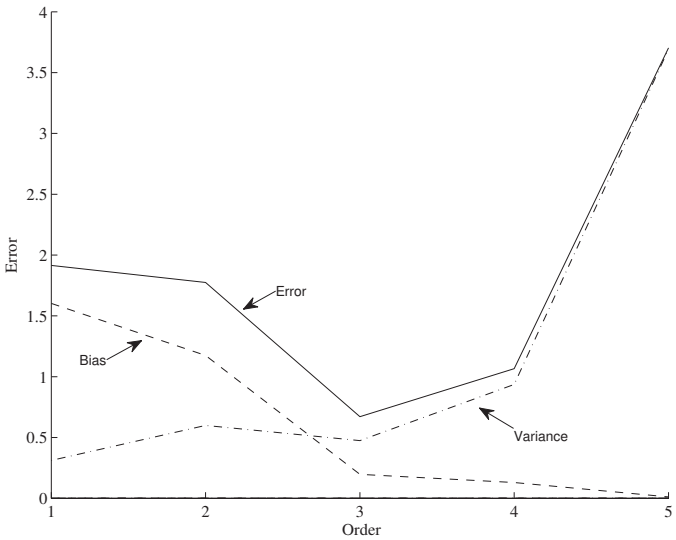

Figure 4.6 In the same setting as that of figure 4.5, using one hundred models instead of five, bias, variance, and error for polynomials of order 1 to 5. Order 1 has the smallest variance. Order 5 has the smallest bias. As the order is increased, bias decreases but variance increases. Order 3 has the minimum error.

a certain level of complexity, then stops decreasing or does not decrease further significantly, or even increases if there is noise in the data. This "elbow" corresponds to the optimal complexity level (see figure 4.7).

In real life, we cannot calculate the bias and hence the error as we do in figure 4.6; the validation error in figure 4.7 is an estimate of that except that it also contains the variance of the noise: Even if we have the right model where there is no bias and large enough data that variance is negligible, there may still be nonzero validation error. Note that the validation error of figure 4.7 is not as V-shaped as the error of figure 4.6 because the former uses more training data and we know that we can constrain variance with more data. Indeed we see in figure 4.5d that even the fifth-order polynomial behaves like a third-order where there is data—note that at the two extremes where there are fewer data points, it is not as accurate.

REGULARIZATION Another approach that is used frequently is *regularization* (Breiman

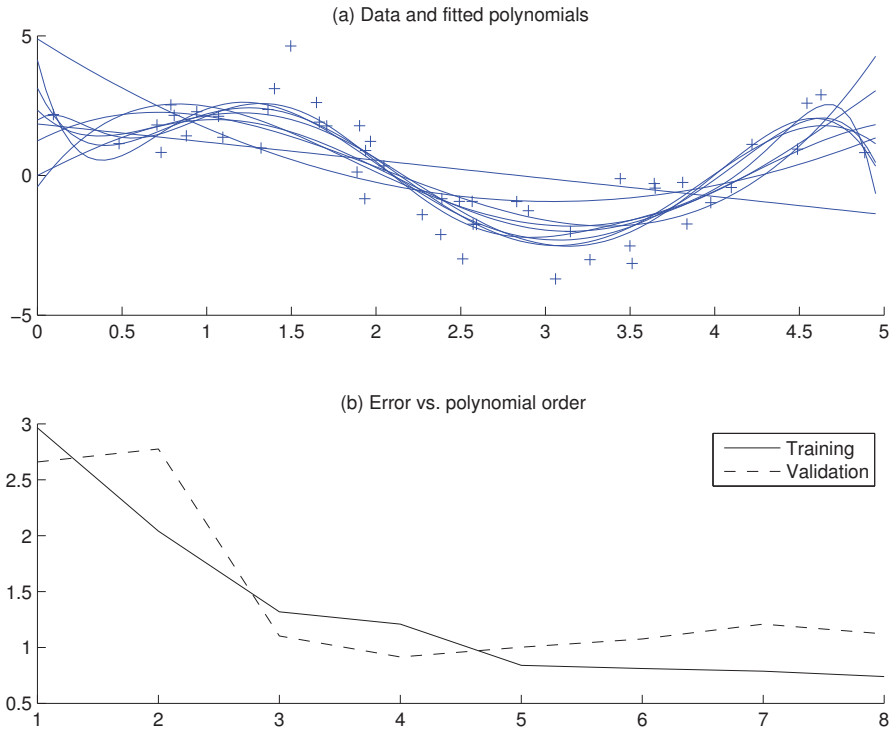

Figure 4.7 In the same setting as that of figure 4.5, training and validation sets (each containing 50 instances) are generated. (a) Training data and fitted polynomials of order from 1 to 8. (b) Training and validation errors as a function of the polynomial order. The "elbow" is at 3.

1998). In this approach, we write an *augmented error function*

$$E' = \text{error on data} + \lambda \cdot \text{model complexity} \tag{4.37}$$

This has a second term that penalizes complex models with large variance, where λ gives the weight of this penalty. When we minimize the augmented error function instead of the error on data only, we penalize complex models and thus decrease variance. If λ is taken too large, only very simple models are allowed and we risk introducing bias. λ is optimized using cross-validation.

Another way we can view equation 4.37 is by regarding E' as the error on new test data. The first term on the right is the training error and the

second is an *optimism* term estimating the discrepancy between training and test error (Hastie, Tibshirani, and Friedman 2011). Methods such as *Akaike's information criterion* (AIC) and *Bayesian information criterion* (BIC) work by estimating this optimism and adding it to the training error to estimate test error, without any need for validation. The magnitude of this optimism term increases linearly with d, the number of inputs (here, it is $k+1$), and decreases as N, training set size, increases; it also increases with σ^2, the variance of the noise added (which we can estimate from the error of a low-bias model). For models that are not linear, d should be replaced with the "effective" number of parameters.

AIC
BIC

STRUCTURAL RISK MINIMIZATION

Structural risk minimization (SRM) (Vapnik 1995) uses a set of models ordered in terms of their complexities. An example is polynomials of increasing order. The complexity is generally given by the number of free parameters. VC dimension is another measure of model complexity. In equation 4.37, we can have a set of decreasing λ_i to get a set of models ordered in increasing complexity. Model selection by SRM then corresponds to finding the model simplest in terms of order and best in terms of empirical error on the data.

MINIMUM DESCRIPTION LENGTH

Minimum description length (MDL) (Rissanen 1978; Grünwald 2007) is based on an information theoretic measure. *Kolmogorov complexity* of a dataset is defined as the shortest description of the data. If the data is simple, it has a short complexity; for example, if it is a sequence of '0's, we can just write '0' and the length of the sequence. If the data is completely random, we cannot have any description of the data shorter than the data itself. If a model is appropriate for the data, then it has a good fit to the data, and instead of the data, we can send/store the model description. Out of all the models that describe the data, we want to have the simplest model so that it lends itself to the shortest description. So we again have a trade-off between how simple the model is and how well it explains the data.

BAYESIAN MODEL SELECTION

Bayesian model selection is used when we have some prior knowledge about the appropriate class of approximating functions. This prior knowledge is defined as a prior distribution over models, $p(\text{model})$. Given the data and assuming a model, we can calculate $p(\text{model}|\text{data})$ using Bayes' rule:

$$p(\text{model}|\text{data}) = \frac{p(\text{data}|\text{model})p(\text{model})}{p(\text{data})} \tag{4.38}$$

$p(\text{model}|\text{data})$ is the posterior probability of the model given our prior

subjective knowledge about models, namely, $p(\text{model})$, and the objective support provided by the data, namely, $p(\text{data}|\text{model})$. We can then choose the model with the highest posterior probability, or take an average over all models weighted by their posterior probabilities. We will talk about the Bayesian approach in detail in chapter 16.

If we take the log of equation 4.38, we get

$$\log p(\text{model}|\text{data}) = \log p(\text{data}|\text{model}) + \log p(\text{model}) - c \tag{4.39}$$

which has the form of equation 4.37; the log likelihood of the data is the training error and the log of the prior is the penalty term. For example, if we have a regression model and use the prior $p(\boldsymbol{w}) \sim \mathcal{N}(0, 1/\lambda)$, we minimize

$$E = \sum_t [r^t - g(x^t|\boldsymbol{w})]^2 + \lambda \sum_i w_i^2 \tag{4.40}$$

That is, we look for w_i that both decrease error and are also as close as possible to 0, and the reason we want them close to 0 is because the fitted polynomial will be smoother. As the polynomial order increases, to get a better fit to the data, the function will go up and down, which will mean coefficients moving away from 0 (see figure 4.8); when we add this penalty, we force a flatter, smoother fit. How much we penalize depends on λ, which is the inverse of the variance of the prior—that is, how much we expect the weights a priori to be away from 0. In other words, having such a prior is equivalent to forcing parameters to be close to 0. We discuss this in greater detail in chapter 16.

That is, when the prior is chosen such that we give higher probabilities to simpler models (following Occam's razor), the Bayesian approach, regularization, SRM, and MDL are equivalent. Cross-validation is different from all other methods for model selection in that it makes no prior assumption about the model or parameters. If there is a large enough validation dataset, it is the best approach. The other models become useful when the data sample is small.

4.9 Notes

A good source on the basics of maximum likelihood and Bayesian estimation is Ross 1987. Many pattern recognition textbooks discuss classification with parametric models (e.g., MacLachlan 1992; Devroye, Györfi, and

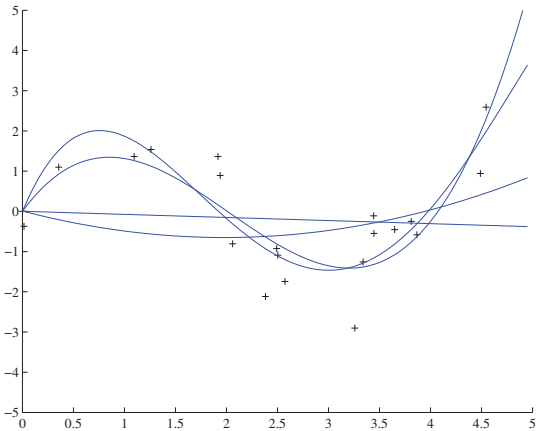

Figure 4.8 In the same setting as that of figure 4.5, polynomials of order 1 to 4 are fitted. The magnitude of coefficients increase as the order of the polynomial increases. They are as follows: $1 : [-0.0769, 0.0016]^T$, $2 : [0.1682, -0.6657, 0.0080]^T$, $3 : [0.4238, -2.5778, 3.4675, -0.0002]^T$, $4 : [-0.1093, 1.4356, -5.5007, 6.0454, -0.0019]^T$.

Lugosi 1996; Webb and Copsey 2011; Duda, Hart, and Stork 2001). Tests for checking univariate normality can be found in Rencher 1995.

Geman, Bienenstock, and Doursat (1992) discuss bias and variance decomposition for several learning models, which we discuss in later chapters. Bias/variance decomposition is for sum of squared loss and is for regression; such a nice additive splitting of error into bias, variance and noise is not possible for 0/1 loss, because in classification, there is error only if we accidentally move to the other side of the boundary. For a two-class problem, if the correct posterior is 0.7 and if our estimate is 0.8, there is no error; we have error only if our estimate is less than 0.5. Various researchers proposed different definitions of bias and variance for classification; see Friedman 1997 for a review.

4.10 Exercises

1. Write the code that generates a Bernoulli sample with given parameter p, and the code that calculates $\hat{p}$ from the sample.
2. Write the log likelihood for a multinomial sample and show equation 4.6.

3. Write the code that generates a normal sample with given μ and σ, and the code that calculates m and s from the sample. Do the same using the Bayes' estimator assuming a prior distribution for μ.
4. Given two normal distributions $p(x|C_1) \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $p(x|C_2) \sim \mathcal{N}(\mu_2, \sigma_2^2)$ and $P(C_1)$ and $P(C_2)$, calculate the Bayes' discriminant points analytically.
   SOLUTION: Given that

$$\begin{aligned} p(x|C_1) &\sim \mathcal{N}(\mu_1, \sigma_1^2) = \frac{1}{\sqrt{2\pi}\sigma_1} \exp\left[-\frac{(x-\mu_1)^2}{2\sigma_1^2}\right] \\ p(x|C_2) &\sim \mathcal{N}(\mu_2, \sigma_2^2) \end{aligned}$$

   we would like to find x that satisfy $P(C_1|x) = P(C_2|x)$, or

$$\begin{aligned} p(x|C_1)P(C_1) &= p(x|C_2)P(C_2) \\ \log p(x|C_1) + \log P(C_1) &= \log p(x|C_2) + \log P(C_2) \\ -\frac{1}{2}\log 2\pi - \log\sigma_1 - \frac{(x-\mu_1)^2}{2\sigma_1^2} + \log P(C_1) &= \cdots \\ -\log\sigma_1 - \frac{1}{2\sigma_1^2}\left(x^2 - 2x\mu_1 + \mu_1^2\right) + \log P(C_1) &= \cdots \\ \left(\frac{1}{2\sigma_2^2} - \frac{1}{2\sigma_1^2}\right)x^2 + \left(\frac{\mu_1}{\sigma_1^2} - \frac{\mu_2}{\sigma_2^2}\right)x + & \\ \left(\frac{\mu_2^2}{2\sigma_2^2} - \frac{\mu_1^2}{2\sigma_1^2}\right) + \log\frac{\sigma_2}{\sigma_1} + \log\frac{P(C_1)}{P(C_2)} &= 0 \end{aligned}$$

   This is of the form $ax^2 + bx + c = 0$ and the two roots are

$$x_1, x_2 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

   Note that if the variances are equal, the quadratic terms vanishes and there is one root, that is, the two posteriors intersect at a single x value.
5. What is the likelihood ratio

$$\frac{p(x|C_1)}{p(x|C_2)}$$

   in the case of Gaussian densities?
   SOLUTION:

$$\frac{p(x|C_1)}{p(x|C_2)} = \frac{\frac{1}{\sqrt{2\pi}\sigma_1}\exp\left[-\frac{(x-\mu_1)^2}{2\sigma_1^2}\right]}{\frac{1}{\sqrt{2\pi}\sigma_2}\exp\left[-\frac{(x-\mu_2)^2}{2\sigma_2^2}\right]}$$

   If we have $\sigma_1^2 = \sigma_2^2 = \sigma^2$, we can simplify

$$\frac{p(x|C_1)}{p(x|C_2)} = \exp\left[-\frac{(x-\mu_1)^2}{2\sigma^2} + \frac{(x-\mu_2)^2}{2\sigma^2}\right]$$

$$= \exp\left[\frac{(\mu_1-\mu_2)}{\sigma^2}x + \frac{(\mu_2^2-\mu_1^2)}{2\sigma^2}\right]$$
$$= \exp(wx + w_0)$$

for $w = (\mu_1 - \mu_2)/\sigma^2$ and $w_0 = (\mu_2^2 - \mu_1^2)/2\sigma^2$.

6. For a two-class problem, generate normal samples for two classes with different variances, then use parametric classification to estimate the discriminant points. Compare these with the theoretical values.
7. Assume a linear model and then add 0-mean Gaussian noise to generate a sample. Divide your sample into two as training and validation sets. Use linear regression using the training half. Compute error on the validation set. Do the same for polynomials of degrees 2 and 3 as well.
8. When the training set is small, the contribution of variance to error may be more than that of bias and in such a case, we may prefer a simple model even though we know that it is too simple for the task. Can you give an example?
9. Let us say, given the samples $\mathcal{X}_i = \{x_i^t, r_i^t\}$, we define $g_i(x) = r_i^1$, namely, our estimate for any x is the r value of the first instance in the (unordered) dataset $\mathcal{X}_i$. What can you say about its bias and variance, as compared with $g_i(x) = 2$ and $g_i(x) = \sum_t r_i^t/N$? What if the sample is ordered, so that $g_i(x) = \min_t r_i^t$?
10. In equation 4.40, what is the effect of changing λ on bias and variance?

    SOLUTION: λ controls smoothness: If it is large, we may smooth too much and decrease variance at the expense of an increase in bias; if it is small, bias may be small but variance will be high.

4.11 References

Andrieu, C., N. de Freitas, A. Doucet, and M. I. Jordan. 2003. "An Introduction to MCMC for Machine Learning." *Machine Learning* 50:5–43.

Breiman, L. 1998. "Bias-Variance, Regularization, Instability and Stabilization." In *Neural Networks and Machine Learning*, ed. C. M. Bishop, 27–56. Berlin: Springer.

Devroye, L., L. Györfi, and G. Lugosi. 1996. *A Probabilistic Theory of Pattern Recognition*. New York: Springer.

Duda, R. O., P. E. Hart, and D. G. Stork. 2001. *Pattern Classification*, 2nd ed. New York: Wiley.

Friedman, J. H. 1997. "On Bias, Variance, 0/1-Loss and the Curse of Dimensionality." *Data Mining and Knowledge Discovery* 1:55–77.

Geman, S., E. Bienenstock, and R. Doursat. 1992. "Neural Networks and the Bias/Variance Dilemma." *Neural Computation* 4:1–58.

Grünwald, P. D. 2007. *The Minimum Description Length Principle.* Cambridge, MA: MIT Press.

Hastie, T., R. Tibshirani, and J. Friedman. 2011. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction,* 2nd ed. New York: Springer.

McLachlan, G. J. 1992. *Discriminant Analysis and Statistical Pattern Recognition.* New York: Wiley.

Rencher, A. C. 1995. *Methods of Multivariate Analysis.* New York: Wiley.

Rissanen, J. 1978. "Modeling by Shortest Data Description." *Automatica* 14:465–471.

Ross, S. M. 1987. *Introduction to Probability and Statistics for Engineers and Scientists.* New York: Wiley.

Vapnik, V. 1995. *The Nature of Statistical Learning Theory.* New York: Springer.

Webb, A., and K. D. Copsey. 2011. *Statistical Pattern Recognition,* 3rd ed. New York: Wiley.

5 *Multivariate Methods*

In chapter 4, we discussed the parametric approach to classification and regression. Now, we generalize this to the multivariate case where we have multiple inputs and where the output, which is class code or continuous output, is a function of these multiple inputs. These inputs may be discrete or numeric. We will see how such functions can be learned from a labeled multivariate sample and also how the complexity of the learner can be fine-tuned to the data at hand.

5.1 Multivariate Data

IN MANY APPLICATIONS, several measurements are made on each individual or event generating an observation vector. The sample may be viewed as a *data matrix*

$$\mathbf{X} = \begin{bmatrix} X_1^1 & X_2^1 & \cdots & X_d^1 \\ X_1^2 & X_2^2 & \cdots & X_d^2 \\ \vdots & & & \\ X_1^N & X_2^N & \cdots & X_d^N \end{bmatrix}$$

where the d columns correspond to d *variables* denoting the result of measurements made on an individual or event. These are also called *inputs*, *features*, or *attributes*. The N rows correspond to independent and identically distributed *observations*, *examples*, or *instances* on N individuals or events.

INPUT
FEATURE
ATTRIBUTE
OBSERVATION
EXAMPLE
INSTANCE

For example, in deciding on a loan application, an observation vector is the information associated with a customer and is composed of age, marital status, yearly income, and so forth, and we have N such past

customers. These measurements may be of different scales, for example, age in years and yearly income in monetary units. Some like age may be numeric, and some like marital status may be discrete.

Typically these variables are correlated. If they are not, there is no need for a multivariate analysis. Our aim may be *simplification*, that is, summarizing this large body of data by means of relatively few parameters. Or our aim may be *exploratory*, and we may be interested in generating hypotheses about data. In some applications, we are interested in predicting the value of one variable from the values of other variables. If the predicted variable is discrete, this is multivariate classification, and if it is numeric, this is a multivariate regression problem.

5.2 Parameter Estimation

MEAN VECTOR

The *mean vector* $\boldsymbol{\mu}$ is defined such that each of its elements is the mean of one column of $\mathbf{X}$:

$$E[\boldsymbol{x}] = \boldsymbol{\mu} = [\mu_1, \ldots, \mu_d]^T \tag{5.1}$$

The variance of X_i is denoted as σ_i^2, and the covariance of two variables X_i and X_j is defined as

$$\sigma_{ij} \equiv \text{Cov}(X_i, X_j) = E[(X_i - \mu_i)(X_j - \mu_j)] = E[X_i X_j] - \mu_i \mu_j \tag{5.2}$$

with $\sigma_{ij} = \sigma_{ji}$, and when $i = j$, $\sigma_{ii} = \sigma_i^2$. With d variables, there are d variances and $d(d-1)/2$ covariances, which are generally represented as

COVARIANCE MATRIX

a $d \times d$ matrix, named the *covariance matrix*, denoted as $\boldsymbol{\Sigma}$, whose (i, j)th element is σ_{ij}:

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1d} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2d} \\ \vdots & & & \\ \sigma_{d1} & \sigma_{d2} & \cdots & \sigma_d^2 \end{bmatrix}$$

The diagonal terms are the variances, the off-diagonal terms are the covariances, and the matrix is symmetric. In vector-matrix notation

$$\boldsymbol{\Sigma} \equiv \text{Cov}(\boldsymbol{X}) = E[(\boldsymbol{X} - \boldsymbol{\mu})(\boldsymbol{X} - \boldsymbol{\mu})^T] = E[\boldsymbol{X}\boldsymbol{X}^T] - \boldsymbol{\mu}\boldsymbol{\mu}^T \tag{5.3}$$

If two variables are related in a linear way, then the covariance will be positive or negative depending on whether the relationship has a positive

or negative slope. But the size of the relationship is difficult to interpret because it depends on the units in which the two variables are measured.

CORRELATION

The *correlation* between variables X_i and X_j is a statistic normalized between -1 and $+1$, defined as

$$\text{Corr}(X_i, X_j) \equiv \rho_{ij} = \frac{\sigma_{ij}}{\sigma_i \sigma_j} \tag{5.4}$$

If two variables are independent, then their covariance, and hence their correlation, is 0. However, the converse is not true: The variables may be dependent (in a nonlinear way), and their correlation may be 0.

SAMPLE MEAN

Given a multivariate sample, estimates for these parameters can be calculated: The maximum likelihood estimator for the mean is the *sample mean*, $\boldsymbol{m}$. Its ith dimension is the average of the ith column of $\mathbf{X}$:

$$\boldsymbol{m} = \frac{\sum_{t=1}^{N} \boldsymbol{x}^t}{N} \text{ with } m_i = \frac{\sum_{t=1}^{N} x_i^t}{N}, i = 1, \ldots, d \tag{5.5}$$

SAMPLE COVARIANCE

The estimator of $\boldsymbol{\Sigma}$ is $\mathbf{S}$, the *sample covariance* matrix, with entries

$$s_i^2 = \frac{\sum_{t=1}^{N} (x_i^t - m_i)^2}{N} \tag{5.6}$$

$$s_{ij} = \frac{\sum_{t=1}^{N} (x_i^t - m_i)(x_j^t - m_j)}{N} \tag{5.7}$$

These are biased estimates, but if in an application the estimates vary significantly depending on whether we divide by N or $N - 1$, we are in serious trouble anyway.

SAMPLE CORRELATION

The *sample correlation* coefficients are

$$r_{ij} = \frac{s_{ij}}{s_i s_j} \tag{5.8}$$

and the sample correlation matrix $\mathbf{R}$ contains r_{ij}.

5.3 Estimation of Missing Values

Frequently, values of certain variables may be missing in observations. The best strategy is to discard those observations all together, but generally we do not have large enough samples to be able to afford this and we do not want to lose data as the non-missing entries do contain information. We try to fill in the missing entries by estimating them. This is called *imputation*.

IMPUTATION

In *mean imputation*, for a numeric variable, we substitute the mean (average) of the available data for that variable in the sample. For a discrete variable, we fill in with the most likely value, that is, the value most often seen in the data.

In *imputation by regression*, we try to predict the value of a missing variable from other variables whose values are known for that case. Depending on the type of the missing variable, we define a separate regression or classification problem that we train by the data points for which such values are known. If many different variables are missing, we take the means as the initial estimates and the procedure is iterated until predicted values stabilize. If the variables are not highly correlated, the regression approach is equivalent to mean imputation.

Depending on the context, however, sometimes the fact that a certain attribute value is missing may be important. For example, in a credit card application, if the applicant does not declare his or her telephone number, that may be a critical piece of information. In such cases, this is represented as a separate value to indicate that the value is missing and is used as such.

5.4 Multivariate Normal Distribution

In the multivariate case where $\boldsymbol{x}$ is d-dimensional and normal distributed, we have

$$p(\boldsymbol{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right] \tag{5.9}$$

and we write $\boldsymbol{x} \sim \mathcal{N}_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ where $\boldsymbol{\mu}$ is the mean vector and $\boldsymbol{\Sigma}$ is the covariance matrix (see figure 5.1). Just as

$$\frac{(x-\mu)^2}{\sigma^2} = (x-\mu)(\sigma^2)^{-1}(x-\mu)$$

is the squared distance from x to μ in standard deviation units, normalizing for different variances, in the multivariate case the *Mahalanobis distance* is used:

MAHALANOBIS DISTANCE

$$(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) \tag{5.10}$$

$(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) = c^2$ is the d-dimensional hyperellipsoid centered at $\boldsymbol{\mu}$, and its shape and orientation are defined by $\boldsymbol{\Sigma}$. Because of the use of the inverse of $\boldsymbol{\Sigma}$, if a variable has a larger variance than another, it receives

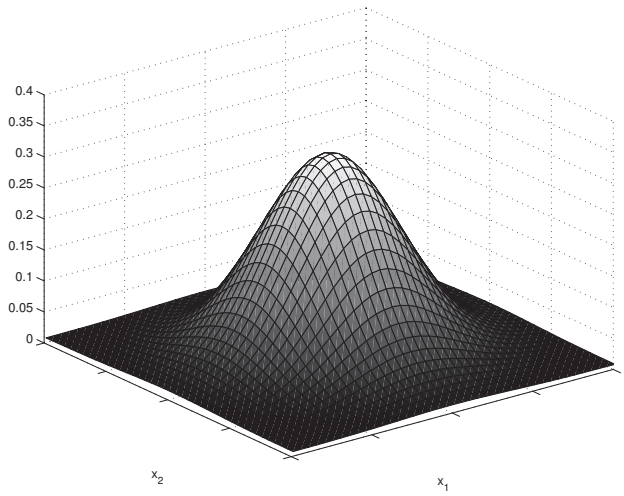

Figure 5.1 Bivariate normal distribution.

less weight in the Mahalanobis distance. Similarly, two highly correlated variables do not contribute as much as two less correlated variables. The use of the inverse of the covariance matrix thus has the effect of standardizing all variables to unit variance and eliminating correlations.

Let us consider the bivariate case where $d = 2$ for visualization purposes (see figure 5.2). When the variables are independent, the major axes of the density are parallel to the input axes. The density becomes an ellipse if the variances are different. The density rotates depending on the sign of the covariance (correlation). The mean vector is $\boldsymbol{\mu}^T = [\mu_1, \mu_2]$, and the covariance matrix is usually expressed as

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}$$

The joint bivariate density can be expressed in the form (see exercise 1)

$$p(x_1, x_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \exp\left[-\frac{1}{2(1-\rho^2)}\left(z_1^2 - 2\rho z_1 z_2 + z_2^2\right)\right] \tag{5.11}$$

where $z_i = (x_i - \mu_i)/\sigma_i, i = 1, 2$, are standardized variables; this is called *z-normalization*. Remember that

Z-NORMALIZATION

$$z_1^2 + 2\rho z_1 z_2 + z_2^2 = \text{constant}$$

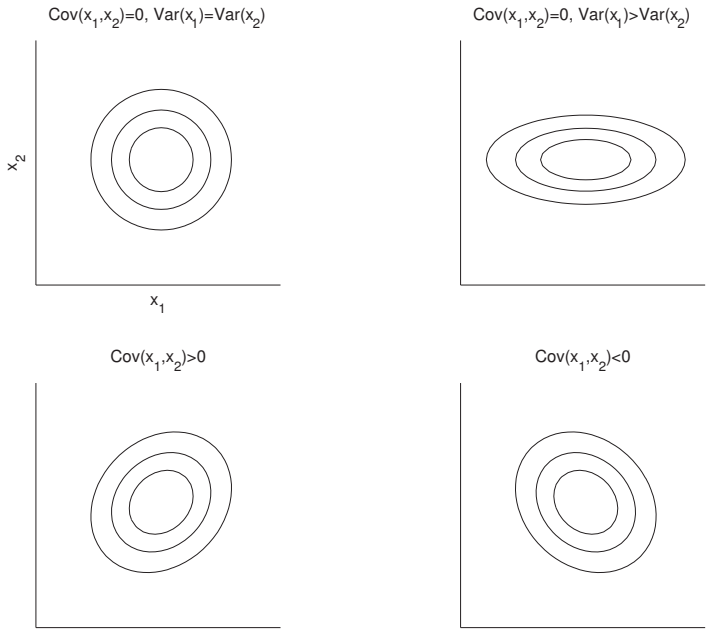

Figure 5.2 Isoprobability contour plot of the bivariate normal distribution. Its center is given by the mean, and its shape and orientation depend on the covariance matrix.

for $|\rho| < 1$, is the equation of an ellipse. When $\rho > 0$, the major axis of the ellipse has a positive slope and if $\rho < 0$, the major axis has a negative slope.

In the expanded Mahalanobis distance of equation 5.11, each variable is normalized to have unit variance, and there is the cross-term that corrects for the correlation between the two variables.

The density depends on five parameters: the two means, the two variances, and the correlation. $\boldsymbol{\Sigma}$ is nonsingular, and hence positive definite, provided that variances are nonzero and $|\rho| < 1$. If ρ is $+1$ or -1, the two variables are linearly related, the observations are effectively one-dimensional, and one of the two variables can be disposed of. If $\rho = 0$, then the two variables are independent, the cross-term disappears, and we get a product of two univariate densities.

In the multivariate case, a small value of $|\boldsymbol{\Sigma}|$ indicates samples are close to $\boldsymbol{\mu}$, just as in the univariate case where a small value of σ^2 indicates

samples are close to μ. Small $|\Sigma|$ may also indicate that there is high correlation between variables. Σ is a symmetric positive definite matrix; this is the multivariate way of saying that $\text{Var}(X) > 0$. If not so, Σ is singular and its determinant is 0. This is either due to linear dependence between the dimensions or because one of the dimensions has variance 0. In such a case, dimensionality should be reduced to a get a positive definite matrix; we discuss methods for this in chapter 6.

If $\boldsymbol{x} \sim \mathcal{N}_d(\boldsymbol{\mu}, \Sigma)$, then each dimension of $\boldsymbol{x}$ is univariate normal. (The converse is not true: Each X_i may be univariate normal and $\boldsymbol{X}$ may not be multivariate normal.) Actually any $k < d$ subset of the variables is k-variate normal.

A special, *naive* case is where the components of $\boldsymbol{x}$ are independent and $\text{Cov}(X_i, X_j) = 0$, for $i \neq j$, and $\text{Var}(X_i) = \sigma_i^2$, $\forall i$. Then the covariance matrix is diagonal and the joint density is the product of the individual univariate densities:

$$p(\boldsymbol{x}) = \prod_{i=1}^{d} p_i(x_i) = \frac{1}{(2\pi)^{d/2} \prod_{i=1}^{d} \sigma_i} \exp\left[-\frac{1}{2}\sum_{i=1}^{d}\left(\frac{x_i - \mu_i}{\sigma_i}\right)^2\right] \tag{5.12}$$

Now let us see another property we make use of in later chapters. Let us say $\boldsymbol{x} \sim \mathcal{N}_d(\boldsymbol{\mu}, \Sigma)$ and $\boldsymbol{w} \in \Re^d$, then

$$\boldsymbol{w}^T\boldsymbol{x} = w_1x_1 + w_2x_2 + \cdots + w_dx_d \sim \mathcal{N}(\boldsymbol{w}^T\boldsymbol{\mu}, \boldsymbol{w}^T\Sigma\boldsymbol{w})$$

given that

$$\begin{aligned} E[\boldsymbol{w}^T\boldsymbol{x}] &= \boldsymbol{w}^T E[\boldsymbol{x}] = \boldsymbol{w}^T\boldsymbol{\mu} && (5.13)\\ \text{Var}(\boldsymbol{w}^T\boldsymbol{x}) &= E[(\boldsymbol{w}^T\boldsymbol{x} - \boldsymbol{w}^T\boldsymbol{\mu})^2] = E[(\boldsymbol{w}^T\boldsymbol{x} - \boldsymbol{w}^T\boldsymbol{\mu})(\boldsymbol{w}^T\boldsymbol{x} - \boldsymbol{w}^T\boldsymbol{\mu})] \\ &= E[\boldsymbol{w}^T(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^T\boldsymbol{w}] = \boldsymbol{w}^T E[(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^T]\boldsymbol{w} \\ &= \boldsymbol{w}^T\Sigma\boldsymbol{w} && (5.14) \end{aligned}$$

That is, the projection of a d-dimensional normal on the vector $\boldsymbol{w}$ is univariate normal. In the general case, if $\mathbf{W}$ is a $d \times k$ matrix with rank $k < d$, then the k-dimensional $\mathbf{W}^T\boldsymbol{x}$ is k-variate normal:

$$\mathbf{W}^T\boldsymbol{x} \sim \mathcal{N}_k(\mathbf{W}^T\boldsymbol{\mu}, \mathbf{W}^T\Sigma\mathbf{W}) \tag{5.15}$$

That is, if we project a d-dimensional normal distribution to a space that is k-dimensional, then it projects to a k-dimensional normal.

5.5 Multivariate Classification

When $\boldsymbol{x} \in \Re^d$, if the class-conditional densities, $p(\boldsymbol{x}|C_i)$, are taken as normal density, $\mathcal{N}_d(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, we have

$$p(\boldsymbol{x}|C_i) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_i|^{1/2}} \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i)\right] \tag{5.16}$$

The main reason for this is its analytical simplicity (Duda, Hart, and Stork 2001). Besides, the normal density is a model for many naturally occurring phenomena in that examples of most classes can be seen as mildly changed versions of a single prototype, $\boldsymbol{\mu}_i$, and the covariance matrix, $\boldsymbol{\Sigma}_i$, denotes the amount of noise in each variable and the correlations of these noise sources. While real data may not often be exactly multivariate normal, it is a useful approximation. In addition to its mathematical tractability, the model is robust to departures from normality as is shown in many works (e.g., McLachlan 1992). However, one clear requirement is that the sample of a class should form a single group; if there are multiple groups, one should use a mixture model (chapter 7).

Let us say we want to predict the type of a car that a customer would be interested in. Different cars are the classes and $\boldsymbol{x}$ are observable data of customers, for example, age and income. $\boldsymbol{\mu}_i$ is the vector of mean age and income of customers who buy car type i and $\boldsymbol{\Sigma}_i$ is their covariance matrix: σ_{i1}^2 and σ_{i2}^2 are the age and income variances, and σ_{i12} is the covariance of age and income in the group of customers who buy car type i.

When we define the discriminant function as

$$g_i(\boldsymbol{x}) = \log p(\boldsymbol{x}|C_i) + \log P(C_i)$$

and assuming $p(\boldsymbol{x}|C_i) \sim \mathcal{N}_d(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, we have

$$g_i(\boldsymbol{x}) = -\frac{d}{2}\log 2\pi - \frac{1}{2}\log|\boldsymbol{\Sigma}_i| - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i) + \log P(C_i) \tag{5.17}$$

Given a training sample for $K \geq 2$ classes, $\mathcal{X} = \{\boldsymbol{x}^t, \boldsymbol{r}^t\}$, where $r_i^t = 1$ if $\boldsymbol{x}^t \in C_i$ and 0 otherwise, estimates for the means and covariances are found using maximum likelihood separately for each class:

$$\begin{aligned} \hat{P}(C_i) &= \frac{\sum_t r_i^t}{N} \\ \boldsymbol{m}_i &= \frac{\sum_t r_i^t \boldsymbol{x}^t}{\sum_t r_i^t} \\ \mathbf{S}_i &= \frac{\sum_t r_i^t (\boldsymbol{x}^t - \boldsymbol{m}_i)(\boldsymbol{x}^t - \boldsymbol{m}_i)^T}{\sum_t r_i^t} \end{aligned} \tag{5.18}$$

These are then plugged into the discriminant function to get the estimates for the discriminants. Ignoring the first constant term, we have

$$g_i(\boldsymbol{x}) = -\frac{1}{2}\log|\mathbf{S}_i| - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{m}_i)^T\mathbf{S}_i^{-1}(\boldsymbol{x}-\boldsymbol{m}_i) + \log\hat{P}(C_i) \tag{5.19}$$

Expanding this, we get

$$g_i(\boldsymbol{x}) = -\frac{1}{2}\log|\mathbf{S}_i| - \frac{1}{2}\left(\boldsymbol{x}^T\mathbf{S}_i^{-1}\boldsymbol{x} - 2\boldsymbol{x}^T\mathbf{S}_i^{-1}\boldsymbol{m}_i + \boldsymbol{m}_i^T\mathbf{S}_i^{-1}\boldsymbol{m}_i\right) + \log\hat{P}(C_i)$$

QUADRATIC DISCRIMINANT

which defines a *quadratic discriminant* (see figure 5.3) that can also be written as

$$g_i(\boldsymbol{x}) = \boldsymbol{x}^T\mathbf{W}_i\boldsymbol{x} + \boldsymbol{w}_i^T\boldsymbol{x} + w_{i0} \tag{5.20}$$

where

$$\begin{aligned}\mathbf{W}_i &= -\frac{1}{2}\mathbf{S}_i^{-1}\\ \boldsymbol{w}_i &= \mathbf{S}_i^{-1}\boldsymbol{m}_i\\ w_{i0} &= -\frac{1}{2}\boldsymbol{m}_i^T\mathbf{S}_i^{-1}\boldsymbol{m}_i - \frac{1}{2}\log|\mathbf{S}_i| + \log\hat{P}(C_i)\end{aligned}$$

The number of parameters to be estimated are $K \cdot d$ for the means and $K \cdot d(d+1)/2$ for the covariance matrices. When d is large and samples are small, $\mathbf{S}_i$ may be singular and inverses may not exist. Or, $|\mathbf{S}_i|$ may be nonzero but too small, in which case it will be unstable; small changes in $\mathbf{S}_i$ will cause large changes in $\mathbf{S}_i^{-1}$. For the estimates to be reliable on small samples, one may want to decrease dimensionality, d, by redesigning the feature extractor and select a subset of the features or somehow combine existing features. We discuss such methods in chapter 6.

Another possibility is to pool the data and estimate a common covariance matrix for all classes:

$$\mathbf{S} = \sum_i \hat{P}(C_i)\mathbf{S}_i \tag{5.21}$$

In this case of equal covariance matrices, equation 5.19 reduces to

$$g_i(\boldsymbol{x}) = -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{m}_i)^T\mathbf{S}^{-1}(\boldsymbol{x}-\boldsymbol{m}_i) + \log\hat{P}(C_i) \tag{5.22}$$

The number of parameters is $K \cdot d$ for the means and $d(d+1)/2$ for the shared covariance matrix. If the priors are equal, the optimal decision rule is to assign input to the class whose mean's Mahalanobis distance to the input is the smallest. As before, unequal priors shift the boundary

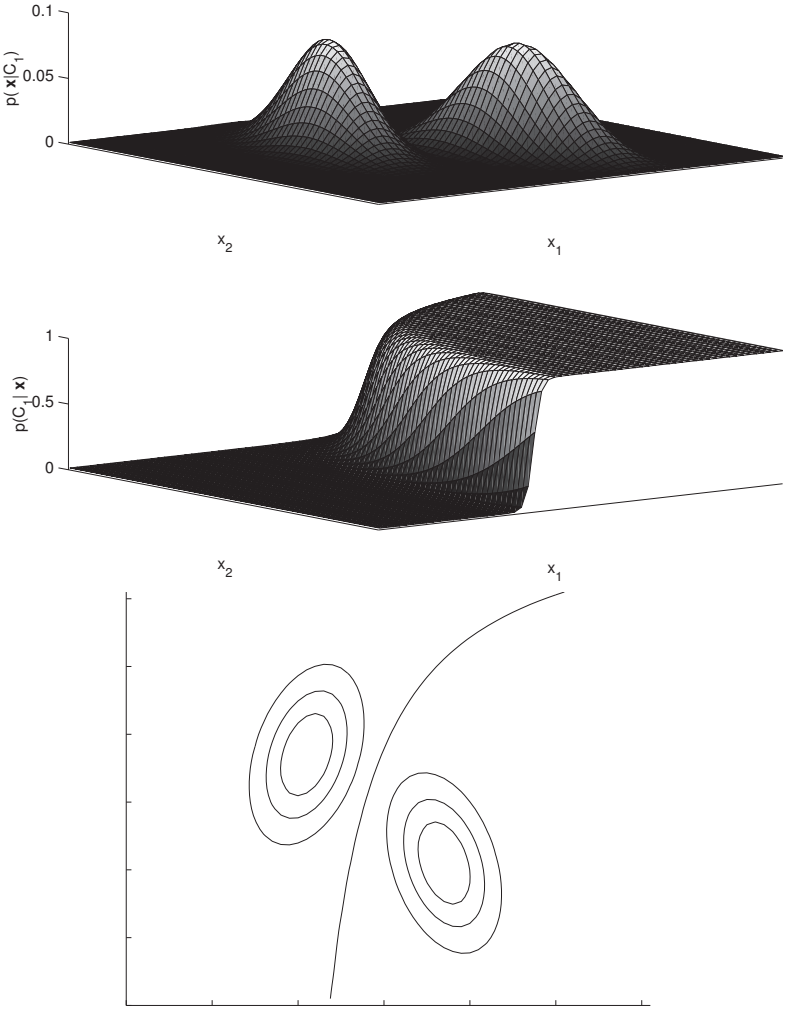

Figure 5.3 Classes have different covariance matrices. Likelihood densities and the posterior probability for one of the classes (top). Class distributions are indicated by isoprobability contours and the discriminant is drawn (bottom).

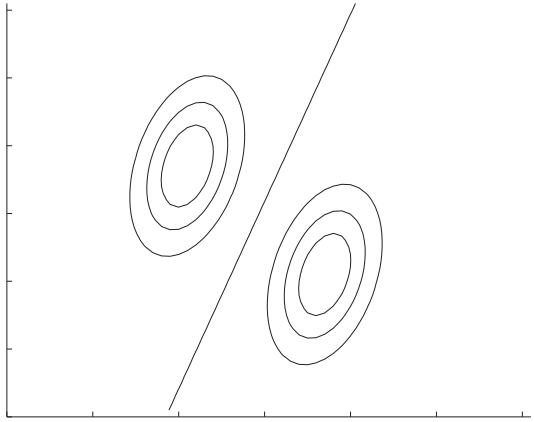

Figure 5.4 Covariances may be arbitary but shared by both classes.

toward the less likely class. Note that in this case, the quadratic term $\boldsymbol{x}^T\mathbf{S}^{-1}\boldsymbol{x}$ cancels since it is common in all discriminants, and the decision boundaries are linear, leading to a *linear discriminant* (figure 5.4) that can be written as

LINEAR DISCRIMINANT

$$g_i(\boldsymbol{x}) = \boldsymbol{w}_i^T\boldsymbol{x} + w_{i0} \tag{5.23}$$

where

$$\begin{aligned} \boldsymbol{w}_i &= \mathbf{S}^{-1}\boldsymbol{m}_i \\ w_{i0} &= -\frac{1}{2}\boldsymbol{m}_i^T\mathbf{S}^{-1}\boldsymbol{m}_i + \log\hat{P}(C_i) \end{aligned}$$

Decision regions of such a linear classifier are convex; namely, when two points are chosen arbitrarily in one decision region and are connected by a straight line, all the points on the line will lie in the region.

Further simplication may be possible by assuming all off-diagonals of the covariance matrix to be 0, thus assuming independent variables. This is the *naive Bayes' classifier* where $p(x_j|C_i)$ are univariate Gaussian. $\mathbf{S}$ and its inverse are diagonal, and we get

NAIVE BAYES' CLASSIFIER

$$g_i(\boldsymbol{x}) = -\frac{1}{2}\sum_{j=1}^{d}\left(\frac{x_j^t - m_{ij}}{s_j}\right)^2 + \log\hat{P}(C_i) \tag{5.24}$$

The term $(x_j^t - m_{ij})/s_j$ has the effect of normalization and measures the distance in terms of standard deviation units. Geometrically speaking,

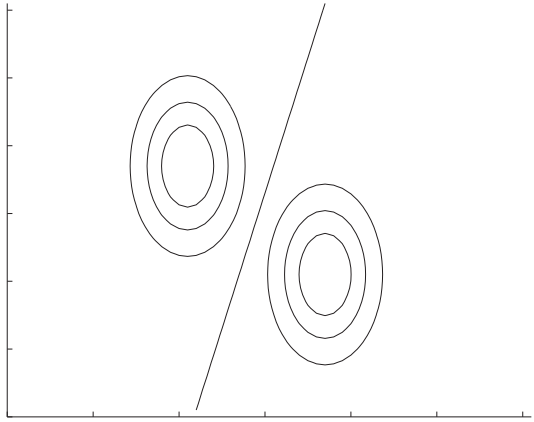

Figure 5.5 All classes have equal, diagonal covariance matrices, but variances are not equal.

classes are hyperellipsoidal and, because the covariances are zero, are axis-aligned (see figure 5.5). The number of parameters is $K \cdot d$ for the means and d for the variances. Thus the complexity of $\mathbf{S}$ is reduced from $\mathcal{O}(d^2)$ to $\mathcal{O}(d)$.

EUCLIDEAN DISTANCE

Simplifying even further, if we assume all variances to be equal, the Mahalanobis distance reduces to *Euclidean distance*. Geometrically, the distribution is shaped spherically, centered around the mean vector $\boldsymbol{m}_i$ (see figure 5.6). Then $|\mathbf{S}| = s^{2d}$ and $\mathbf{S}^{-1} = (1/s^2)\mathbf{I}$. The number of parameters in this case is $K \cdot d$ for the means and 1 for s^2.

$$g_i(\boldsymbol{x}) = -\frac{\|\boldsymbol{x} - \boldsymbol{m}_i\|^2}{2s^2} + \log \hat{P}(C_i) = -\frac{1}{2s^2}\sum_{j=1}^{d}(x_j^t - m_{ij})^2 + \log \hat{P}(C_i) \tag{5.25}$$

NEAREST MEAN CLASSIFIER

TEMPLATE MATCHING

If the priors are equal, we have $g_i(\boldsymbol{x}) = -\|\boldsymbol{x} - \boldsymbol{m}_i\|^2$. This is named the *nearest mean classifier* because it assigns the input to the class of the nearest mean. If each mean is thought of as the ideal prototype or template for the class, this is a *template matching* procedure. This can be expanded as

$$\begin{aligned} g_i(\boldsymbol{x}) &= -\|\boldsymbol{x} - \boldsymbol{m}_i\|^2 = -(\boldsymbol{x} - \boldsymbol{m}_i)^T(\boldsymbol{x} - \boldsymbol{m}_i) \\ &= -(\boldsymbol{x}^T\boldsymbol{x} - 2\boldsymbol{m}_i^T\boldsymbol{x} + \boldsymbol{m}_i^T\boldsymbol{m}_i) \end{aligned} \tag{5.26}$$

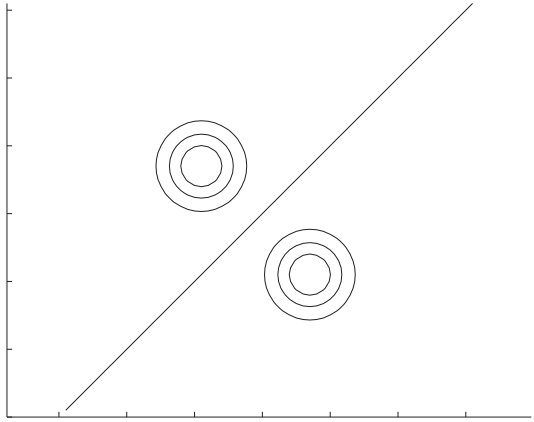

Figure 5.6 All classes have equal, diagonal covariance matrices of equal variances on both dimensions.

The first term, $\boldsymbol{x}^T\boldsymbol{x}$, is shared in all $g_i(\boldsymbol{x})$ and can be dropped, and we can write the discriminant function as

$$g_i(\boldsymbol{x}) = \boldsymbol{w}_i^T\boldsymbol{x} + w_{i0} \tag{5.27}$$

where $\boldsymbol{w}_i = \boldsymbol{m}_i$ and $w_{i0} = -(1/2)\|\boldsymbol{m}_i\|^2$. If all $\boldsymbol{m}_i$ have similar norms, then this term can also be ignored and we can use

$$g_i(\boldsymbol{x}) = \boldsymbol{m}_i^T\boldsymbol{x} \tag{5.28}$$

When the norms of $\boldsymbol{m}_i$ are comparable, dot product can also be used as the similarity measure instead of the (negative) Euclidean distance.

We can actually think of finding the best discriminant function as the task of finding the best distance function. This can be seen as another approach to classification: Instead of learning the discriminant functions, $g_i(\boldsymbol{x})$, we want to learn the suitable distance function $\mathcal{D}(\boldsymbol{x}_1, \boldsymbol{x}_2)$, such that for any $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3$, where $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ belong to the same class, and $\boldsymbol{x}_1$ and $\boldsymbol{x}_3$ belong to two different classes, we would like to have

$$\mathcal{D}(\boldsymbol{x}_1, \boldsymbol{x}_2) < \mathcal{D}(\boldsymbol{x}_1, \boldsymbol{x}_3)$$

Table 5.1 Reducing variance through simplifying assumptions

Assumption	Covariance matrix	No. of parameters
Shared, Hyperspheric	$\mathbf{S}_i = \mathbf{S} = s^2\mathbf{I}$	1
Shared, Axis-aligned	$\mathbf{S}_i = \mathbf{S}$, with $s_{ij} = 0$	d
Shared, Hyperellipsoidal	$\mathbf{S}_i = \mathbf{S}$	$d(d+1)/2$
Different, Hyperellipsoidal	$\mathbf{S}_i$	$K \cdot (d(d+1)/2)$

5.6 Tuning Complexity

In table 5.1, we see how the number of parameters of the covariance matrix may be reduced, trading off the comfort of a simple model with generality. This is another example of bias/variance dilemma. When we make simplifying assumptions about the covariance matrices and decrease the number of parameters to be estimated, we risk introducing bias (see figure 5.7). On the other hand, if no such assumption is made and the matrices are arbitrary, the quadratic discriminant may have large variance on small datasets. The ideal case depends on the complexity of the problem represented by the data at hand and the amount of data we have. When we have a small dataset, even if the covariance matrices are different, it may be better to assume a shared covariance matrix; a single covariance matrix has fewer parameters and it can be estimated using more data, that is, instances of all classes. This corresponds to using *linear discriminants*, which is very frequently used in classification and which we discuss in more detail in chapter 10.

Note that when we use Euclidean distance to measure similarity, we are assuming that all variables have the same variance and that they are independent. In many cases, this does not hold; for example, age and yearly income are in different units, and are dependent in many contexts. In such a case, the inputs may be separately z-normalized in a preprocessing stage (to have zero mean and unit variance), and then Euclidean distance can be used. On the other hand, sometimes even if the variables are dependent, it may be better to assume that they are independent and to use the naive Bayes' classifier, if we do not have enough data to calculate the dependency accurately.

REGULARIZED DISCRIMINANT ANALYSIS

Friedman (1989) proposed a method that combines all these as special cases, named *regularized discriminant analysis* (RDA). We remember

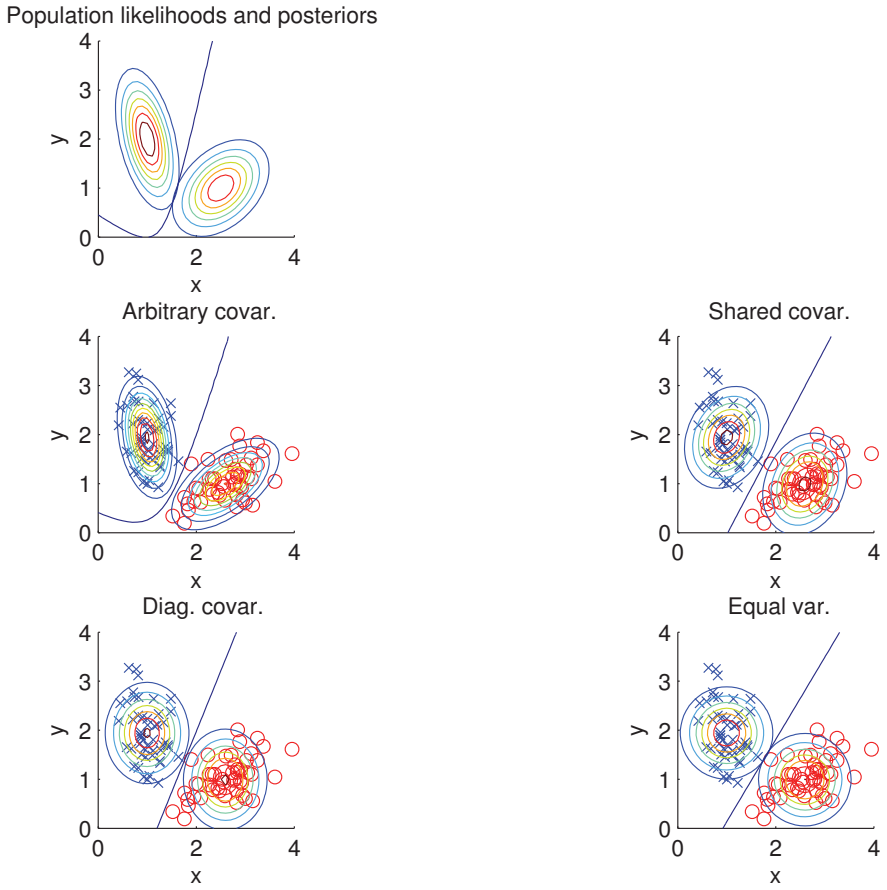

Figure 5.7 Different cases of the covariance matrices fitted to the same data lead to different boundaries.

that regularization corresponds to approaches where one starts with high variance and constrains toward lower variance, at the risk of increasing bias. In the case of parametric classification with Gaussian densities, the covariance matrices can be written as a weighted average of the three special cases:

$$\mathbf{S}_i' = \alpha\sigma^2\mathbf{I} + \beta\mathbf{S} + (1-\alpha-\beta)\mathbf{S}_i \tag{5.29}$$

When $\alpha = \beta = 0$, this leads to a quadratic classifier. When $\alpha = 0$ and

$\beta = 1$, the covariance matrices are shared, and we get linear classifiers. When $\alpha = 1$ and $\beta = 0$, the covariance matrices are diagonal with σ^2 on the diagonals, and we get the nearest mean classifier. In between these extremes, we get a whole variety of classifiers where α, β are optimized by cross-validation.

Another approach to regularization, when the dataset is small, is one that uses a Bayesian approach by defining priors on $\boldsymbol{\mu}_i$ and $\mathbf{S}_i$ or that uses cross-validation to choose the best of the four cases given in table 5.1.

5.7 Discrete Features

In some applications, we have discrete attributes taking one of n different values. For example, an attribute may be color $\in$ {red, blue, green, black}, or another may be pixel $\in$ {on, off}. Let us say x_j are binary (Bernoulli) where

$$p_{ij} \equiv p(x_j = 1|C_i)$$

If x_j are independent binary variables, we have

$$p(\boldsymbol{x}|C_i) = \prod_{j=1}^{d} p_{ij}^{x_j} (1 - p_{ij})^{(1-x_j)}$$

This is another example of the naive Bayes' classifier where $p(x_j|C_i)$ are Bernoulli. The discriminant function is

$$\begin{aligned} g_i(\boldsymbol{x}) &= \log p(\boldsymbol{x}|C_i) + \log P(C_i) \\ &= \sum_j \left[x_j \log p_{ij} + (1 - x_j) \log(1 - p_{ij}) \right] + \log P(C_i) \end{aligned} \tag{5.30}$$

which is linear. The estimator for p_{ij} is

$$\hat{p}_{ij} = \frac{\sum_t x_j^t r_i^t}{\sum_t r_i^t} \tag{5.31}$$

DOCUMENT CATEGORIZATION

BAG OF WORDS

This approach is used in *document categorization*, an example of which is classifying news reports into various categories, such as, politics, sports, fashion, and so forth. In the *bag of words* representation, we choose a priori d words that we believe give information regarding the class (Manning and Schütze 1999). For example, in news classification, words such as "missile," "athlete," and "couture" are useful, rather than ambiguous words such as "model," or even "runway." In this representation, each

text is a d-dimensional binary vector where x_j is 1 if word j occurs in the document and is 0 otherwise. Note that this representation loses all ordering information of words, and hence the name *bag* of words.

After training, $\hat{p}_{ij}$ estimates the probability that word j occurs in document type i. Words whose probabilities are similar for different classes do not convey much information; for them to be useful, we would want the probability to be high for one class (or few) and low for all others; we are going to talk about this type of *feature selection* in chapter 6. Another example application of document categorization is *spam filtering* where there are two classes of emails as spam and legitimate. In bioinformatics, too, inputs are generally sequences of discrete items, whether base-pairs or amino acids.

SPAM FILTERING

In the general case, instead of binary features, let us say we have the multinomial x_j chosen from the set $\{v_1, v_2, \ldots, v_{n_j}\}$. We define new 0/1 dummy variables as

$$z_{jk}^t = \begin{cases} 1 & \text{if } x_j^t = v_k \\ 0 & \text{otherwise} \end{cases}$$

Let p_{ijk} denote the probability that x_j belonging to C_i takes value v_k:

$$p_{ijk} \equiv p(z_{jk} = 1 | C_i) = p(x_j = v_k | C_i)$$

If the attributes are independent, we have

$$p(\boldsymbol{x}|C_i) = \prod_{j=1}^{d} \prod_{k=1}^{n_j} p_{ijk}^{z_{jk}} \tag{5.32}$$

The discriminant function is then

$$g_i(\boldsymbol{x}) = \sum_j \sum_k z_{jk} \log p_{ijk} + \log P(C_i) \tag{5.33}$$

The maximum likelihood estimator for p_{ijk} is

$$\hat{p}_{ijk} = \frac{\sum_t z_{jk}^t r_i^t}{\sum_t r_i^t} \tag{5.34}$$

which can be plugged into equation 5.33 to give us the discriminant.

5.8 Multivariate Regression

MULTIVARIATE LINEAR REGRESSION

In *multivariate linear regression*, the numeric output r is assumed to be

written as a linear function, that is, a weighted sum, of several input variables, $x_1, \ldots, x_d$, and noise. Actually in statistical literature, this is called *multiple* regression; statisticians use the term *multivariate* when there are multiple outputs. The multivariate linear model is

$$r^t = g(\boldsymbol{x}^t | w_0, w_1, \ldots, w_d) + \epsilon = w_0 + w_1 x_1^t + w_2 x_2^t + \cdots + w_d x_d^t + \epsilon \tag{5.35}$$

As in the univariate case, we assume ϵ to be normal with mean 0 and constant variance, and maximizing the likelihood is equivalent to minimizing the sum of squared errors:

$$E(w_0, w_1, \ldots, w_d | \mathcal{X}) = \frac{1}{2} \sum_t (r^t - w_0 - w_1 x_1^t - w_2 x_2^t - \cdots - w_d x_d^t)^2 \tag{5.36}$$

Taking the derivative with respect to the parameters, $w_j, j = 0, \ldots, d$, we get these *normal equations*:

$$\begin{aligned}
\sum_t r^t &= N w_0 + w_1 \sum_t x_1^t + w_2 \sum_t x_2^t + \cdots + w_d \sum_t x_d^t \\
\sum_t x_1^t r^t &= w_0 \sum_t x_1^t + w_1 \sum_t (x_1^t)^2 + w_2 \sum_t x_1^t x_2^t + \cdots + w_d \sum_t x_1^t x_d^t \\
\sum_t x_2^t r^t &= w_0 \sum_t x_2^t + w_1 \sum_t x_1^t x_2^t + w_2 \sum_t (x_2^t)^2 + \cdots + w_d \sum_t x_2^t x_d^t \\
&\vdots \\
\sum_t x_d^t r^t &= w_0 \sum_t x_d^t + w_1 \sum_t x_d^t x_1^t + w_2 \sum_t x_d^t x_2^t + \cdots + w_d \sum_t (x_d^t)^2
\end{aligned} \tag{5.37}$$

Let us define the following vectors and matrix:

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^1 & x_2^1 & \cdots & x_d^1 \\ 1 & x_1^2 & x_2^2 & \cdots & x_d^2 \\ \vdots & & & & \\ 1 & x_1^N & x_2^N & \cdots & x_d^N \end{bmatrix}, \boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d, \end{bmatrix}, \boldsymbol{r} = \begin{bmatrix} r^1 \\ r^2 \\ \vdots \\ r^N \end{bmatrix}$$

Then the normal equations can be written as

$$\mathbf{X}^T \mathbf{X} \boldsymbol{w} = \mathbf{X}^T \boldsymbol{r} \tag{5.38}$$

and we can solve for the parameters as

$$\boldsymbol{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \boldsymbol{r} \tag{5.39}$$

This method is the same as we used for polynomial regression using one input. The two problems are the same if we define the variables as

MULTIVARIATE POLYNOMIAL REGRESSION

$x_1 = x, x_2 = x^2, \ldots, x_k = x^k$. This also gives us a hint as to how we can do *multivariate polynomial regression* if necessary (exercise 7), but unless d is small, in multivariate regression, we rarely use polynomials of an order higher than linear.

Actually using higher-order terms of inputs as additional inputs is only one possibility; we can define any nonlinear function of the original inputs using *basis functions*. For example, we can define new inputs $x_2 = \sin(x), x_3 = \exp(x^2)$ if we believe that such a transformation is useful. Then, using a linear model in this new augmented space will correspond to a nonlinear model in the original space. The same calculation will still be valid; we need only replace $\mathbf{X}$ with the data matrix after the basis functions are applied. As we will see later under various guises (e.g., multilayer perceptrons, support vector machines, Gaussian processes), this type of generalizing the linear model is frequently used.

One advantage of linear models is that after the regression, looking at the $w_j, j = 1, \ldots, d$, values, we can extract knowledge: First, by looking at the signs of w_j, we can see whether x_j have a positive or negative effect on the output. Second, if all x_j are in the same range, by looking at the absolute values of w_j, we can get an idea about how important a feature is, rank the features in terms of their importances, and even remove the features whose w_j are close to 0.

When there are multiple outputs, this can equivalently be defined as a set of independent single-output regression problems.

5.9 Notes

A good review text on linear algebra is Strang 2006. Harville 1997 is another excellent book that looks at matrix algebra from a statistical point of view.

One inconvenience with multivariate data is that when the number of dimensions is large, one cannot do a visual analysis. There are methods proposed in the statistical literature for displaying multivariate data; a review is given in Rencher 1995. One possibility is to plot variables two by two as bivariate scatter plots: If the data is multivariate normal, then the plot of any two variables should be roughly linear; this can be used as a visual test of multivariate normality. Another possibility that we discuss in chapter 6 is to project them to one or two dimensions and display there.

Most work on pattern recognition is done assuming multivariate normal densities. Sometimes such a discriminant is even called the Bayes' optimal classifier, but this is generally wrong; it is only optimal if the densities are indeed multivariate normal and if we have enough data to calculate the correct parameters from the data. Rencher 1995 discusses tests for assessing multivariate normality as well as tests for checking for equal covariance matrices. McLachlan 1992 discusses classification with multivariate normals and compares linear and quadratic discriminants.

One obvious restriction of multivariate normals is that it does not allow for data where some features are discrete. A variable with n possible values can be converted into n dummy 0/1 variables, but this increases dimensionality. One can do a dimensionality reduction in this n-dimensional space by a method explained in chapter 6 and thereby not increase dimensionality. Parametric classification for such cases of mixed features is discussed in detail in McLachlan 1992.

5.10 Exercises

1. Show equation 5.11.

   SOLUTION: Given that

   $$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}$$

   we have

   $$\begin{aligned} |\boldsymbol{\Sigma}| &= \sigma_1^2\sigma_2^2 - \rho^2\sigma_1^2\sigma_2^2 = \sigma_1^2\sigma_2^2(1-\rho^2) \\ |\boldsymbol{\Sigma}|^{1/2} &= \sigma_1\sigma_2\sqrt{1-\rho^2} \\ \boldsymbol{\Sigma}^{-1} &= \frac{1}{\sigma_1^2\sigma_2^2(1-\rho^2)} \begin{bmatrix} \sigma_2^2 & -\rho\sigma_1\sigma_2 \\ -\rho\sigma_1\sigma_2 & \sigma_1^2 \end{bmatrix} \end{aligned}$$

   and $(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})$ can be expanded as

   $$[x_1-\mu_1 \; x_2-\mu_2] \begin{bmatrix} \frac{\sigma_2^2}{\sigma_1^2\sigma_2^2(1-\rho^2)} & -\frac{\rho\sigma_1\sigma_2}{\sigma_1^2\sigma_2^2(1-\rho^2)} \\ -\frac{\rho\sigma_1\sigma_2}{\sigma_1^2\sigma_2^2(1-\rho^2)} & \frac{\sigma_1^2}{\sigma_1^2\sigma_2^2(1-\rho^2)} \end{bmatrix} \begin{bmatrix} x_1-\mu_1 \\ x_2-\mu_2 \end{bmatrix}$$

   $$= \frac{1}{1-\rho^2}\left[\left(\frac{x_1-\mu_1}{\sigma_1}\right)^2 - 2\rho\left(\frac{x_1-\mu_1}{\sigma_1}\right)\left(\frac{x_2-\mu_2}{\sigma_2}\right) + \left(\frac{x_2-\mu_2}{\sigma_2}\right)^2\right]$$

2. Generate a sample from a multivariate normal density $\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$, calculate $\boldsymbol{m}$ and $\mathbf{S}$, and compare them with $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. Check how your estimates change as the sample size changes.

3. Generate samples from two multivariate normal densities $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), i = 1, 2$, and calculate the Bayes' optimal discriminant for the four cases in table 5.1.
4. For a two-class problem, for the four cases of Gaussian densities in table 5.1, derive
   $$\log \frac{P(C_1|\boldsymbol{x})}{P(C_2|\boldsymbol{x})}$$
5. Another possibility using Gaussian densities is to have them all diagonal but allow them to be different. Derive the discriminant for this case.
6. Let us say in two dimensions, we have two classes with exactly the same mean. What type of boundaries can be defined?
7. Let us say we have two variables x_1 and x_2 and we want to make a quadratic fit using them, namely,
   $$f(x_1, x_2) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2 + w_4 (x_1)^2 + w_5 (x_2)^2$$
   How can we find $w_i, i = 0, \ldots, 5$, given a sample of $\mathcal{X} = \{x_1^t, x_2^t, r^t\}$?
   SOLUTION: We write the fit as
   $$f(x_1, x_2) = w_0 + w_1 z_1 + w_2 z_2 + w_3 z_3 + w_4 z_4 + w_5 z_5$$
   where $z_1 = x_1$, $z_2 = x_2$, $z_3 = x_1 x_2$, $z_4 = (x_1)^2$, and $z_5 = (x_2)^2$. We can then use linear regression to learn $w_i, i = 0, \ldots, 5$. The linear fit in the five-dimensional $(z_1, z_2, z_3, z_4, z_5)$ space corresponds to a quadratic fit in the two-dimensional (x_1, x_2) space. We discuss such generalized linear models in more detail (and other nonlinear basis functions) in chapter 10.
8. In regression we saw that fitting a quadratic is equivalent to fitting a linear model with an extra input corresponding to the square of the input. Can we also do this in classification?
   SOLUTION: Yes. We can define new, auxiliary variables corresponding to powers and cross-product terms and then use a linear model. For example, just as in exercise 7, we can define $z_1 = x_1$, $z_2 = x_2$, $z_3 = x_1 x_2$, $z_4 = (x_1)^2$, and $z_5 = (x_2)^2$ and then use a linear model to learn $w_i, i = 0, \ldots, 5$. The linear discriminant in the five-dimensional $(z_1, z_2, z_3, z_4, z_5)$ space corresponds to a quadratic discriminant in the two-dimensional (x_1, x_2) space.
9. In document clustering, ambiguity of words can be decreased by taking the context into account, for example, by considering pairs of words, as in "cocktail party" vs. "party elections." Discuss how this can be implemented.

5.11 References

Duda, R. O., P. E. Hart, and D. G. Stork. 2001. *Pattern Classification*, 2nd ed. New York: Wiley.

Friedman, J. H. 1989. "Regularized Discriminant Analysis." *Journal of American Statistical Association* 84:165–175.

Harville, D. A. 1997. *Matrix Algebra from a Statistician's Perspective*. New York: Springer.

Manning, C. D., and H. Schütze. 1999. *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press.

McLachlan, G. J. 1992. *Discriminant Analysis and Statistical Pattern Recognition*. New York: Wiley.

Rencher, A. C. 1995. *Methods of Multivariate Analysis*. New York: Wiley.

Strang, G. 2006. *Linear Algebra and its Applications*, 4th ed. Boston: Cengage Learning.

LOGIT
LOG ODDS

$\log y/(1-y)$ is known as the *logit* transformation or *log odds* of y. In the case of two normal classes sharing a common covariance matrix, the log odds is linear:

$$\begin{aligned}\text{logit}(P(C_1|\boldsymbol{x})) &= \log\frac{P(C_1|\boldsymbol{x})}{1-P(C_1|\boldsymbol{x})} = \log\frac{P(C_1|\boldsymbol{x})}{P(C_2|\boldsymbol{x})}\\ &= \log\frac{p(\boldsymbol{x}|C_1)}{p(\boldsymbol{x}|C_2)} + \log\frac{P(C_1)}{P(C_2)}\\ &= \log\frac{(2\pi)^{-d/2}|\boldsymbol{\Sigma}|^{-1/2}\exp[-(1/2)(\boldsymbol{x}-\boldsymbol{\mu}_1)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1)]}{(2\pi)^{-d/2}|\boldsymbol{\Sigma}|^{-1/2}\exp[-(1/2)(\boldsymbol{x}-\boldsymbol{\mu}_2)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2)]} + \log\frac{P(C_1)}{P(C_2)}\\ &= \boldsymbol{w}^T\boldsymbol{x} + w_0 \qquad (10.13)\end{aligned}$$

where

$$\begin{aligned}\boldsymbol{w} &= \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)\\ w_0 &= -\frac{1}{2}(\boldsymbol{\mu}_1+\boldsymbol{\mu}_2)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2) + \log\frac{P(C_1)}{P(C_2)} \qquad (10.14)\end{aligned}$$

The inverse of logit

$$\log\frac{P(C_1|\boldsymbol{x})}{1-P(C_1|\boldsymbol{x})} = \boldsymbol{w}^T\boldsymbol{x} + w_0$$

LOGISTIC
SIGMOID

is the *logistic* function, also called the *sigmoid* function (see figure 10.5):

$$P(C_1|\boldsymbol{x}) = \text{sigmoid}(\boldsymbol{w}^T\boldsymbol{x} + w_0) = \frac{1}{1+\exp[-(\boldsymbol{w}^T\boldsymbol{x}+w_0)]} \qquad (10.15)$$

During training, we estimate $\boldsymbol{m}_1, \boldsymbol{m}_2, \mathbf{S}$ and plug these estimates in equation 10.14 to calculate the discriminant parameters. During testing, given $\boldsymbol{x}$, we can either

1. calculate $g(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_0$ and choose C_1 if $g(\boldsymbol{x}) > 0$, or
2. calculate $y = \text{sigmoid}(\boldsymbol{w}^T\boldsymbol{x} + w_0)$ and choose C_1 if $y > 0.5$,

because $\text{sigmoid}(0) = 0.5$. In this latter case, sigmoid transforms the discriminant value to a posterior probability. This is valid when there are two classes and one discriminant; we see in section 10.7 how we can estimate posterior probabilities for $K > 2$.

10.6 Gradient Descent

In likelihood-based classification, the parameters were the sufficient statistics of $p(\boldsymbol{x}|C_i)$ and $P(C_i)$, and the method we used to estimate the parameters is maximum likelihood. In the discriminant-based approach,

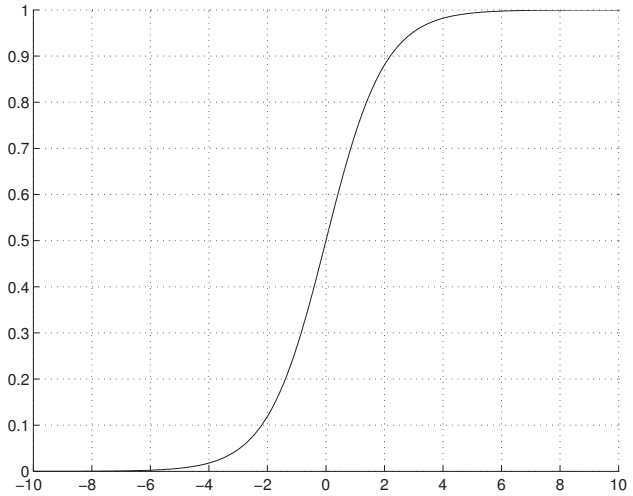

Figure 10.5 The logistic, or sigmoid, function.

the parameters are those of the discriminants, and they are optimized to minimize the classification error on the training set. When $\boldsymbol{w}$ denotes the set of parameters and $E(\boldsymbol{w}|\mathcal{X})$ is the error with parameters $\boldsymbol{w}$ on the given training set $\mathcal{X}$, we look for

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} E(\boldsymbol{w}|\mathcal{X})$$

In many cases, some of which we will see shortly, there is no analytical solution and we need to resort to iterative optimization methods, the most commonly employed being that of *gradient descent*. When $E(\boldsymbol{w})$ is a differentiable function of a vector of variables, we have the *gradient vector* composed of the partial derivatives

GRADIENT DESCENT
GRADIENT VECTOR

$$\nabla_w E = \left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots, \frac{\partial E}{\partial w_d}\right]^T$$

and the *gradient descent* procedure to minimize E starts from a random $\boldsymbol{w}$, and at each step, updates $\boldsymbol{w}$, in the opposite direction of the gradient

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}, \forall i \tag{10.16}$$

$$w_i = w_i + \Delta w_i \tag{10.17}$$

where η is called the *stepsize*, or *learning factor*, and determines how much to move in that direction. Gradient ascent is used to maximize a

function and goes in the direction of the gradient. When we get to a minimum (or maximum), the derivative is 0 and the procedure terminates. This indicates that the procedure finds the nearest minimum that can be a local minimum, and there is no guarantee of finding the global minimum unless the function has only one minimum. The use of a good value for η is also critical; if it is too small, the convergence may be too slow, and a large value may cause oscillations and even divergence.

Throughout this book, we use gradient methods that are simple and quite effective. We keep in mind, however, that once a suitable model and an error function is defined, the optimization of the model parameters to minimize the error function can be done by using one of many possible techniques. There are second-order methods and conjugate gradient that converge faster, at the expense of more memory and computation. More costly methods like simulated annealing and genetic algorithms allow a more thorough search of the parameter space and do not depend as much on the initial point.

10.7 Logistic Discrimination

10.7.1 Two Classes

LOGISTIC DISCRIMINATION

In *logistic discrimination*, we do not model the class-conditional densities, $p(\boldsymbol{x}|C_i)$, but rather their ratio. Let us again start with two classes and assume that the log likelihood ratio is linear:

$$\log \frac{p(\boldsymbol{x}|C_1)}{p(\boldsymbol{x}|C_2)} = \boldsymbol{w}^T\boldsymbol{x} + w_0^o \tag{10.18}$$

This indeed holds when the class-conditional densities are normal (equation 10.13). But logistic discrimination has a wider scope of applicability; for example, $\boldsymbol{x}$ may be composed of discrete attributes or may be a mixture of continuous and discrete attributes.

Using Bayes' rule, we have

$$\begin{aligned} \text{logit}(P(C_1|\boldsymbol{x})) &= \log \frac{P(C_1|\boldsymbol{x})}{1 - P(C_1|\boldsymbol{x})} \\ &= \log \frac{p(\boldsymbol{x}|C_1)}{p(\boldsymbol{x}|C_2)} + \log \frac{P(C_1)}{P(C_2)} \\ &= \boldsymbol{w}^T\boldsymbol{x} + w_0 \end{aligned} \tag{10.19}$$