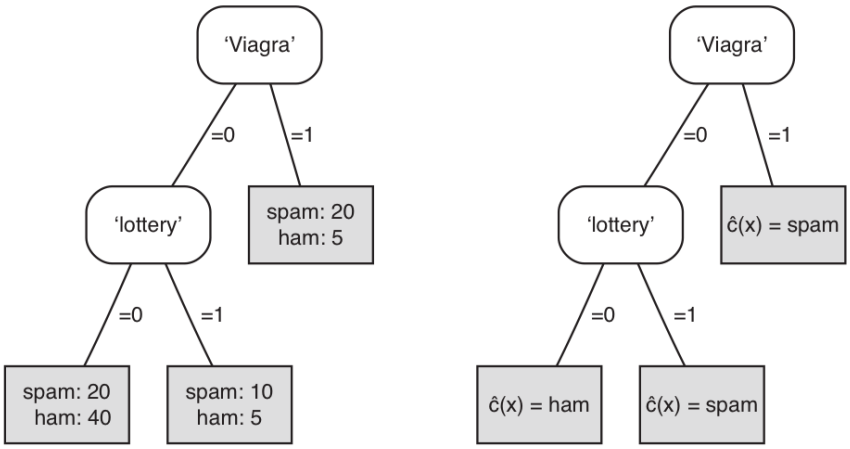

Figure 2.1. (left) A feature tree with training set class distribution in the leaves. **(right)** A decision tree obtained using the majority class decision rule.

a concept). Spam e-mail filtering is a good example of binary classification, in which spam is conventionally taken as the positive class, and ham as the negative class (clearly, positive here doesn't mean 'good'!). Other examples of binary classification include medical diagnosis (the positive class here is having a particular disease) and credit card fraud detection.

The feature tree in Figure 2.1 (left) can be turned into a classifier by labelling each leaf with a class. The simplest way to do this is by assigning the *majority class* in each leaf, resulting in the decision tree in Figure 2.1 (right). The classifier works as follows: if an e-mail contains the word 'Viagra' it is classified as spam (right-most leaf); otherwise, the occurrence of the word 'lottery' decides whether it gets labelled spam or ham.[2] From the numbers in Figure 2.1 we can get an idea how well this classifier does. The left-most leaf correctly predicts 40 ham e-mails but also mislabels 20 spam e-mails that contain neither 'Viagra' nor 'lottery'. The middle leaf correctly classifies 10 spam e-mails but also erroneously labels 5 ham e-mails as spam. The 'Viagra' test correctly picks out 20 spam e-mails but also 5 ham e-mails. Taken together, this means that 30 out of 50 spam e-mails are classified correctly, and 40 out of 50 ham e-mails.

Assessing classification performance

The performance of such classifiers can be summarised by means of a table known as a *contingency table* or *confusion matrix* (Table 2.2 (left)). In this table, each row refers to

[2] If you are keen to know how such a decision tree can be learned from data, you may want to take a sneak preview at Algorithm 5.1 on p.132.

actual classes as recorded in the test set, and each column to classes as predicted by the classifier. So, for instance, the first row states that the test set contains 50 positives, 30 of which were correctly predicted and 20 incorrectly. The last column and the last row give the *marginals* (i.e., column and row sums). Marginals are important because they allow us to assess statistical significance. For instance, the contingency table in Table 2.2 (right) has the same marginals, but the classifier clearly makes a random choice as to which predictions are positive and which are negative – as a result the distribution of actual positives and negatives in either predicted class is the same as the overall distribution (uniform in this case).

	Predicted ⊕	*Predicted* ⊖	
Actual ⊕	**30**	**20**	50
Actual ⊖	**10**	**40**	50
	40	60	100

	⊕	⊖	
⊕	**20**	**30**	50
⊖	**20**	**30**	50
	40	60	100

Table 2.2. (left) A two-class contingency table or confusion matrix depicting the performance of the decision tree in Figure 2.1. Numbers on the descending diagonal indicate correct predictions, while the ascending diagonal concerns prediction errors. **(right)** A contingency table with the same marginals but independent rows and columns.

From a contingency table we can calculate a range of performance indicators. The simplest of these is *accuracy*, which is the proportion of correctly classified test instances. In the notation introduced at the beginning of this chapter, accuracy over a test set Te is defined as

$$acc = \frac{1}{|Te|} \sum_{x \in Te} I[\hat{c}(x) = c(x)] \tag{2.1}$$

Here, the function $I[\cdot]$ denotes the *indicator function*, which is 1 if its argument evaluates to true, and 0 otherwise. In this case it is a convenient way to count the number of test instances that are classified correctly by the classifier (i.e., the estimated class label $\hat{c}(x)$ is equal to the true class label $c(x)$). For example, in Table 2.2 (left) the accuracy of the classifier is 0.70 or 70%, and in Table 2.2 (right) it is 0.50. Alternatively, we can calculate the *error rate* as the proportion of incorrectly classified instances, here 0.30 and 0.50, respectively. Clearly, accuracy and error rate sum to 1.

Test set accuracy can be seen as an *estimate* of the probability that an arbitrary instance $x \in \mathscr{X}$ is classified correctly: more precisely, it estimates the probability

$$P_{\mathscr{X}}(\hat{c}(x) = c(x))$$

(Notice that I write $P_{\mathscr{X}}$ to emphasise that this is a probability distribution over the instance space $\mathscr{X}$; I will often omit subscripts if this is clear from the context.) We

typically only have access to the true classes of a small fraction of the instance space and so an estimate is all we can hope to get. It is therefore important that the test set is as representative as possible. This is usually formalised by the assumption that the occurrence of instances in the world – i.e., how likely or typical a particular e-mail is – is governed by an unknown probability distribution on $\mathscr{X}$, and that the test set Te is generated according to this distribution.

It is often convenient – not to say necessary – to distinguish performance on the classes. To this end, we need some further terminology. Correctly classified positives and negatives are referred to as *true positives* and *true negatives*, respectively. Incorrectly classified positives are, perhaps somewhat confusingly, called *false negatives*; similarly, misclassified negatives are called *false positives*. A good way to think of this is to remember that positive/negative refers to the classifier's prediction, and true/false refers to whether the prediction is correct or not. So, a false positive is something that was incorrectly predicted as positive, and therefore an actual negative (e.g., a ham e-mail misclassified as spam, or a healthy patient misclassified as having the disease in question). In the previous example (Table 2.2 (left)) we have 30 true positives, 20 false negatives, 40 true negatives and 10 false positives.

The *true positive rate* is the proportion of positives correctly classified, and can be defined mathematically as

$$tpr = \frac{\sum_{x \in Te} I[\hat{c}(x) = c(x) = \oplus]}{\sum_{x \in Te} I[c(x) = \oplus]} \tag{2.2}$$

True positive rate is an estimate of the probability that an arbitrary positive is classified correctly, that is, an estimate of $P_{\mathscr{X}}(\hat{c}(x) = \oplus | c(x) = \oplus)$. Analogously, the *true negative rate* is the proportion of negatives correctly classified (see Table 2.3 on p.57 for the mathematical definition), and estimates $P_{\mathscr{X}}(\hat{c}(x) = \ominus | c(x) = \ominus)$. These rates, which are sometimes called *sensitivity* and *specificity*, can be seen as per-class accuracies. In the contingency table, the true positive and negative rates can be calculated by dividing the number on the descending (good) diagonal by the row total. We can also talk about per-class error rates, which is the *false negative rate* for the positives (i.e., the number of misclassified positives or false negatives as a proportion of the total number of positives) and the *false positive rate* for the negatives (sometimes called the *false alarm rate*). These rates can be found by dividing the number on the ascending (bad) diagonal by the row total.

In Table 2.2 (left) we have a true positive rate of 60%, a true negative rate of 80%, a false negative rate of 40% and a false positive rate of 20%. In Table 2.2 (right) we have a true positive rate of 40%, a true negative rate of 60%, a false negative rate of 60% and a false positive rate of 40%. Notice that the accuracy in both cases is the average of the true positive rate and the true negative rate (and the error rate is the average of the false positive rate and the false negative rate). However, this is true only if the test set

contains equal numbers of positives and negatives – in the general case we need to use a *weighted* average, where the weights are the proportions of positives and negatives in the test set.

Example 2.1 (Accuracy as a weighted average). Suppose a classifier's predictions on a test set are as in the following table:

	Predicted ⊕	*Predicted* ⊖	
Actual ⊕	**60**	**15**	75
Actual ⊖	**10**	**15**	25
	70	30	100

From this table, we see that the true positive rate is $tpr = 60/75 = 0.80$ and the true negative rate is $tnr = 15/25 = 0.60$. The overall accuracy is $acc = (60 + 15)/100 = 0.75$, which is no longer the average of true positive and negative rates. However, taking into account the proportion of positives $pos = 0.75$ and the proportion of negatives $neg = 1 - pos = 0.25$, we see that

$$acc = pos \cdot tpr + neg \cdot tnr \tag{2.3}$$

This equation holds in general: if the numbers of positives and negatives are equal, we obtain the unweighted average from the earlier example ($acc = (tpr + tnr)/2$).

Equation 2.3 has a neat intuition: good performance on either class contributes to good classification accuracy, but the more prevalent class contributes more strongly. In order to achieve good accuracy, a classifier should concentrate on the *majority class*, particularly if the class distribution is highly unbalanced. However, it is often the case that the majority class is also the least interesting class. To illustrate, suppose you issue a query to an internet search engine,[3] and suppose that for that particular query there is only one relevant page in every 1 000 web pages. Now consider a 'reluctant' search engine that doesn't return *any* answers – i.e., it classifies every web page as irrelevant to your query. Consequently, it will achieve 0% true positive rate and 100% true negative rate. Because $pos = 1/1000 = 0.1\%$ and $neg = 99.9\%$, the reluctant search engine's accuracy is very high (99.9%). Put differently, if we select a random web page uniformly

[3] An internet search engine can be seen as a binary classifier into the classes relevant and irrelevant, or interesting and not interesting, if we fix the query – not very realistic in practice, but a useful analogy for our purposes.

Measure	*Definition*	*Equal to*	*Estimates*
number of positives	$Pos = \sum_{x \in Te} I[c(x) = \oplus]$		
number of negatives	$Neg = \sum_{x \in Te} I[c(x) = \ominus]$	$\lvert Te \rvert - Pos$	
number of true positives	$TP = \sum_{x \in Te} I[\hat{c}(x) = c(x) = \oplus]$		
number of true negatives	$TN = \sum_{x \in Te} I[\hat{c}(x) = c(x) = \ominus]$		
number of false positives	$FP = \sum_{x \in Te} I[\hat{c}(x) = \oplus, c(x) = \ominus]$	$Neg - TN$	
number of false negatives	$FN = \sum_{x \in Te} I[\hat{c}(x) = \ominus, c(x) = \oplus]$	$Pos - TP$	
proportion of positives	$pos = \frac{1}{\lvert Te \rvert} \sum_{x \in Te} I[c(x) = \oplus]$	$Pos / \lvert Te \rvert$	$P(c(x) = \oplus)$
proportion of negatives	$neg = \frac{1}{\lvert Te \rvert} \sum_{x \in Te} I[c(x) = \ominus]$	$1 - pos$	$P(c(x) = \ominus)$
class ratio	$clr = pos / neg$	Pos / Neg	
(*) accuracy	$acc = \frac{1}{\lvert Te \rvert} \sum_{x \in Te} I[\hat{c}(x) = c(x)]$		$P(\hat{c}(x) = c(x))$
(*) error rate	$err = \frac{1}{\lvert Te \rvert} \sum_{x \in Te} I[\hat{c}(x) \neq c(x)]$	$1 - acc$	$P(\hat{c}(x) \neq c(x))$
true positive rate, sensitivity, recall	$tpr = \frac{\sum_{x \in Te} I[\hat{c}(x) = c(x) = \oplus]}{\sum_{x \in Te} I[c(x) = \oplus]}$	TP / Pos	$P(\hat{c}(x) = \oplus \lvert c(x) = \oplus)$
true negative rate, specificity, negative recall	$tnr = \frac{\sum_{x \in Te} I[\hat{c}(x) = c(x) = \ominus]}{\sum_{x \in Te} I[c(x) = \ominus]}$	TN / Neg	$P(\hat{c}(x) = \ominus \lvert c(x) = \ominus)$
false positive rate, false alarm rate	$fpr = \frac{\sum_{x \in Te} I[\hat{c}(x) = \oplus, c(x) = \ominus]}{\sum_{x \in Te} I[c(x) = \ominus]}$	$FP / Neg = 1 - tnr$	$P(\hat{c}(x) = \oplus \lvert c(x) = \ominus)$
false negative rate	$fnr = \frac{\sum_{x \in Te} I[\hat{c}(x) = \ominus, c(x) = \oplus]}{\sum_{x \in Te} I[c(x) = \oplus]}$	$FN / Pos = 1 - tpr$	$P(\hat{c}(x) = \ominus \lvert c(x) = \oplus)$
precision, confidence	$prec = \frac{\sum_{x \in Te} I[\hat{c}(x) = c(x) = \oplus]}{\sum_{x \in Te} I[\hat{c}(x) = \oplus]}$	$TP / (TP + FP)$	$P(c(x) = \oplus \lvert \hat{c}(x) = \oplus)$

Table 2.3. A summary of different quantities and evaluation measures for classifiers on a test set *Te*. Symbols starting with a capital letter denote absolute frequencies (counts), while lower-case symbols denote relative frequencies or ratios. All except those indicated with (*) are defined only for binary classification. The right-most column specifies the instance space probabilities that these relative frequencies are estimating.

over all web pages, the probability of selecting a positive is only 0.001, and these are the only pages on which the reluctant engine makes an error. However, we are not normally selecting pages from the web uniformly, and hence accuracy is not a meaningful quantity in this context. To be of any use at all, a search engine should achieve a much better true positive rate, which usually comes at the expense of a worse true negative rate (and hence a drop in accuracy).

We conclude from this example that, if the minority class is the class of interest and very small, accuracy and performance on the majority class are not the right quantities to optimise. For this reason, an alternative to true negative rate called *precision* is usually considered in such cases. Precision is a counterpart to true positive rate in the following sense: while true positive rate is the proportion of predicted positives among the actual positives, precision is the proportion of actual positives among the predicted positives. In Example 2.1 the classifier's precision on the test set is 60/70 = 85.7%. In

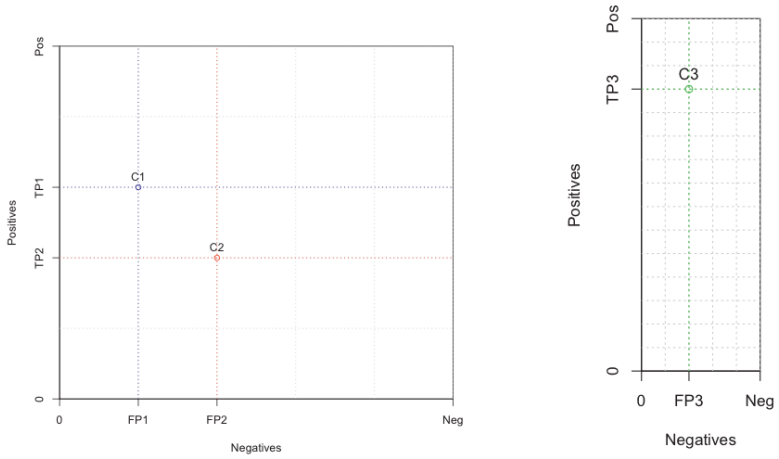

Figure 2.2. (left) A coverage plot depicting the two contingency tables in Table 2.2. The plot is square because the class distribution is uniform. **(right)** Coverage plot for Example 2.1, with a class ratio $clr = 3$.

the reluctant search engine example we have not only 0 true positive rate (which in this context is usually called *recall*) but also 0 precision, which clearly demonstrates the problem with a search engine that doesn't return any answers. Table 2.3 summarises the evaluation measures introduced in this section.

Visualising classification performance

I will now introduce an important tool for visualising the performance of classifiers and other models called a *coverage plot*. If you look at two-class contingency tables such as the ones depicted in Table 2.2, you realise that, even though the table contains nine numbers, only four of those can be chosen freely. For instance, once you've determined the true/false positives/negatives, the marginals are fixed. Or if you know the true positives, true negatives, total number of positives and size of the test set, you can reconstruct all other numbers. Statisticians say that the table has four *degrees of freedom*.[4]

Often we are particularly interested in the following four numbers that completely determine the contingency table: the number of positives *Pos*, the number of negatives *Neg*, the number of true positives *TP* and the number of false positives *FP*. A coverage plot visualises these four numbers by means of a rectangular coordinate system and a point. Imagine a rectangle with height *Pos* and width *Neg*. Imagine furthermore that all positives live on the y-axis of this rectangle, and all negatives on the x-axis. We don't

[4] More generally, a k-class contingency table has $(k+1)^2$ entries and k^2 degrees of freedom.

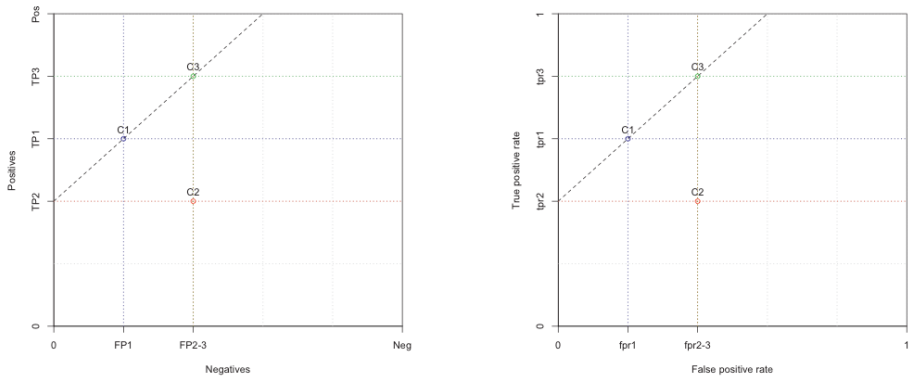

Figure 2.3. (left) C1 and C3 both dominate C2, but neither dominates the other. The diagonal line indicates that C1 and C3 achieve equal accuracy. **(right)** The same plot with normalised axes. We can interpret this plot as a merger of the two coverage plots in Figure 2.2, employing normalisation to deal with the different class distributions. The diagonal line now indicates that C1 and C3 have the same average recall.

really care how positives and negatives are ordered on their respective axes, as long as *positive predictions come before negative predictions*. This gives us enough information to depict the whole contingency table as a single point within the rectangle (Figure 2.2).

Consider the two classifiers marked C1 and C2 in Figure 2.2 (left). One reason why coverage plots are so useful is that we can immediately see that C1 is better than C2. How do we know that? Well, C1 has both more true positives and fewer false positives than C2, and so is better in both respects. Put differently, C1 achieves better performance than C2 on *both* classes. If one classifier outperforms another classifier on all classes, the first one is said to *dominate* the second.[5] However, things are not always that straightforward. Consider a third classifier C3, better than C1 on the positives but worse on the negatives (Figure 2.3 (left)). Although both C1 and C3 dominate C2, neither of them dominates the other. Which one we prefer depends on whether we put more emphasis on the positives or on the negatives.

We can make this a little bit more precise. Notice that the line segment connecting C1 and C3 has a slope of 1. Imagine travelling up that line: whenever we gain a true positive, we also lose a true negative (or gain a false positive, which is the same thing). This doesn't affect the sum of true positives and true negatives, and hence the accuracy is the same wherever we are on the line. It follows that C1 and C3 have the same accuracy. *In a coverage plot, classifiers with the same accuracy are connected by line segments with slope 1*. If true positives and true negatives are equally important, the

[5] This terminology comes from the field of *multi-criterion optimisation*. A dominated solution is one that is not on the *Pareto front*.

choice between C1 and C3 is arbitrary; if true positives are more important we should choose C3, if true negatives are more important we prefer C1.

Now consider Figure 2.3 (right). What I have done here is renormalise the axes by dividing the x-axis by *Neg* and the y-axis by *Pos*, resulting in a plot in the unit square with true positive rate on the y-axis and false positive rate on the x-axis. In this case the original coverage plot was already square ($Pos = Neg$), so the relative position of the classifiers isn't affected by the normalisation. However, since the normalised plot will be square regardless of the shape of the original plot, normalisation is a way to combine differently shaped coverage plots, and thus to combine results on test sets with different class distributions. Suppose you would normalise Figure 2.2 (right): since C3's true and false positive rates are 80% and 40%, respectively (see Example 2.1 on p.56), its position in a normalised plot is exactly the same as the one labelled C3 in Figure 2.3 (right)! In other words, classifiers occupying different points in different coverage spaces (e.g., C3 in Figure 2.2 (right) and C3 in Figure 2.3 (left)) can end up in the same point in a normalised plot.

What is the meaning of the diagonal line connecting C1 and C3 in Figure 2.3 (right)? It can't have the same meaning as in the coverage plot, because in a normalised plot we know the true and false positive rates but not the class distribution, and so we cannot calculate accuracy (refer back to Equation 2.3 on p.56 if you want to remind yourself why). The line is defined by the equation $tpr = fpr + y_0$, where y_0 is the y-intercept (the value of tpr where the line intersects the y-axis) . Now consider the average of the true positive rate and the true negative rate, which we will call *average recall*, denoted *avg-rec*.[6] On a line with slope 1 we have $avg\text{-}rec = (tpr + tnr)/2 = (tpr + 1 - fpr)/2 = (1 + y_0)/2$, which is a constant. *In a normalised coverage plot, line segments with slope 1 connect classifiers with the same average recall.* If recall on the positives and the negatives are equally important, the choice between C1 and C3 is arbitrary; if positive recall is more important we should choose C3, if negative recall is more important we prefer C1.

In the literature, normalised coverage plots are referred to as *ROC plots*, and we will follow that convention from now on.[7] ROC plots are much more common than coverage plots, but both have their specific uses. Broadly speaking, you should use a coverage plot if you explicitly want to take the class distribution into account, for instance when you are working with a single data set. An ROC plot is useful if you want to combine results from different data sets with different class distributions. Clearly, there are many connections between the two. Since an ROC plot is always square, lines of constant average recall (so-called average recall *isometrics*) do not only have

[6] Remember that recall is just a different name for true positive rate; negative recall is then the same as the true negative rate, and average recall is the average of positive recall (or true positive rate) and negative recall (or true negative rate). It is sometimes called *macro-averaged accuracy*.

[7] ROC stands for *receiver operating characteristic*, a term originating from *signal detection theory*.

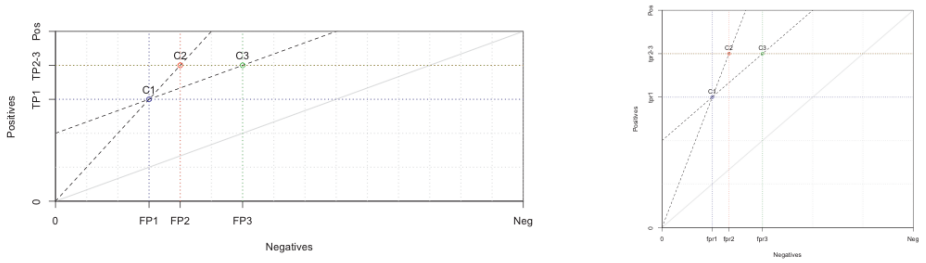

Figure 2.4. (left) In a coverage plot, accuracy isometrics have a slope of 1, and average recall isometrics are parallel to the ascending diagonal. **(right)** In the corresponding ROC plot, average recall isometrics have a slope of 1; the accuracy isometric here has a slope of 3, corresponding to the ratio of negatives to positives in the data set.

a slope of 1 but are parallel to the ascending diagonal. The latter property carries over to coverage plots. To illustrate, in the coverage plot in Figure 2.4, C1 and C2 have the same accuracy (they are connected by a line segment with slope 1), and C1 and C3 have the same average recall (they are connected by a line segment parallel to the diagonal). You can also argue that C2 has both higher accuracy and higher average recall than C3 (why?). In the corresponding ROC plot, the average recall isometric has a slope of 1, and the accuracy isometric's slope is $Neg/Pos = 1/clr$.

2.2 Scoring and ranking

Many classifiers compute scores on which their class predictions are based. For instance, in the Prologue we saw how SpamAssassin calculates a weighted sum from the rules that 'fire' for a particular e-mail. Such scores contain additional information that can be beneficial in a number of ways, which is why we perceive scoring as a task in its own right. Formally, a *scoring classifier* is a mapping $\hat{\mathbf{s}} : \mathcal{X} \to \mathbb{R}^k$, i.e., a mapping from the instance space to a k-vector of real numbers. The boldface notation indicates that a scoring classifier outputs a vector $\hat{\mathbf{s}}(x) = (\hat{s}_1(x), \ldots, \hat{s}_k(x))$ rather than a single number; $\hat{s}_i(x)$ is the score assigned to class C_i for instance x. This score indicates how likely it is that class label C_i applies. If we only have two classes, it usually suffices to consider the score for only one of the classes; in that case, we use $\hat{s}(x)$ to denote the score of the positive class for instance x.

Figure 2.5 demonstrates how a feature tree can be turned into a scoring tree. In order to obtain a score for each leaf, we first calculate the ratio of spam to ham, which is 1/2 for the left leaf, 2 for the middle leaf and 4 for the right leaf. Because it is often more convenient to work with an additive scale, we obtain scores by taking the logarithm of the class ratio (the base of the logarithm is not really important; here we have

2.4 Binary classification and related tasks: Summary and further reading

In this chapter we have looked at binary classification, a ubiquitous task that forms the starting point of a lot of work in machine learning. Although we haven't talked much about learning in this chapter, my philosophy is that you will reach a better understanding of machine learning models and algorithms if you first study the tasks that these models are meant to address.

☞ In Section 2.1 we defined the binary classification task and introduced an important tool to assess performance at such a task, namely the two-by-two contingency table. A wide range of performance indicators are derived from the counts in a contingency table. I introduced the coverage plot, which visualises a contingency table as a rectangle with size *Pos* up and size *Neg* across, and within that rectangle a point with y-coordinate *TP* and x-coordinate *FP*. We can visualise several models evaluated on the same data set by several points, and use the fact that accuracy is constant along line segments with slope 1 to visually rank these classifiers on accuracy. Alternatively, we can normalise the rectangle to be a unit square with true and false positive rate on the axes. In this so-called ROC space, line segments with slope 1 (i.e., those parallel to the ascending diagonal) connect points with the same average recall (sometimes also called macro-accuracy). The use of these kinds of plot in machine learning was pioneered by Provost and Fawcett (2001). Unnormalised coverage plots were introduced by Fürnkranz and Flach (2003).

☞ Section 2.2 considered the more general task of calculating a score for each example (or a vector of scores in the general case of more than two classes). While the scale on which scores are expressed is unspecified, it is customary to put the decision threshold at $\hat{s}(x) = 0$ and let the sign of the score stand for the prediction (positive or negative). Multiplying the score with the true class gives us the margin, which is positive for a correct prediction and negative for an incorrect one. A loss function determines how much negative margins are penalised and positive margins rewarded. The advantage of working with convex and continuously differentiable 'surrogate' loss functions (rather than with 0–1 loss, which is the loss function we ultimately want to optimise) is that this often leads to more tractable optimisation problems.

☞ Alternatively, we can ignore the scale on which scores are measured altogether and only work with their order. Such a ranker is visualised in coverage or ROC space by a piecewise continuous curve. For grouping models the line segments in these curves correspond to instance space segments (e.g., the leaves of a tree

model) whereas for grading models there is a segment for each unique score assigned by the model. The area under the ROC curve gives the ranking accuracy (an estimate of the probability that a random positive is ranked before a random negative) and is known in statistics as the Wilcoxon-Mann-Whitney statistic These curves can be used to find a suitable operating point by translating the operating condition (class and cost distribution) into an isometric in ROC or coverage space. The origins of ROC curves are in signal detection theory (Egan, 1975); accessible introductions can be found in (Fawcett, 2006; Flach, 2010*b*).

☞ In Section 2.3 we looked at scoring models whose scores can be interpreted as estimates of the probability that the instance belongs to a particular class. Such models were pioneered in forecasting theory by Brier (1950) and Murphy and Winkler (1984), among others. We can assess the quality of class probability estimates by comparing them to the 'ideal' probabilities (1 for a positive, 0 for a negative) and taking mean squared error. Since there is no reason why the true probabilities should be categorical this is quite a crude assessment, and decomposing it into calibration loss and refinement loss provides useful additional information. We have also seen a very useful trick for smoothing relative frequency estimates of probabilities by adding pseudo-counts, either uniformly distributed (Laplace correction) or according to a chosen prior (m-estimate). Finally, we have seen how we can use the ROC convex hull to obtain calibrated class probability estimates. The approach has its roots in isotonic regression (Best and Chakravarti, 1990) and was introduced to the machine learning community by Zadrozny and Elkan (2002). Fawcett and Niculescu-Mizil (2007) and Flach and Matsubara (2007) show that the approach is equivalent to calibration by means of the ROC convex hull. (Note that in this chapter we have seen two different uses of the term 'convex': one in relation to loss functions, where convexity means that linear interpolation between any two points on the curve depicting the loss function will never result in a point below the curve; and the other in relation to the ROC convex hull, where it refers to the linearly interpolated boundary of a convex set which envelopes all points in the set.)