7 *Score Functions for Data Mining Algorithms*

7.1 Introduction

In chapter 6 we focused on different representations and structures that are available to us for fitting models and patterns to data. Now we are ready to consider how we will match these structures to data. Recall that a model or pattern structure is the functional form, with the parameters left "floating." For example, $Y = aX + b$ might be one such model structure, with a and b the parameters. Given a model or pattern structure, we must score different settings of the parameter values with respect to data, so that we can choose a good set (or even "the best"). In the simple linear regression example in chapter 1, we saw how a *least squares* principle could be used to choose between different parameter values. This involved finding the values of the parameters a and b that minimized the sum of squared differences between the predicted values of y (from the model) and the observed (data) values of y. In this case, the *score function* is thus the sum of squared errors between model predictions and actual target measurements. Our goal in this chapter is to broaden the reader's horizon in terms of the score functions that can be used for data mining. We will see that the venerable squared error score function is but one of many, and indeed can be viewed as a special case arising from more general principles.

It is important to bear in mind why we are interested in score functions in the first place. Ultimately the purpose of a score function should be to rank models as a function of how useful the models are to the data miner. Unfortunately in practice it can be quite difficult to measure "utility" in terms of direct practical usefulness to the person building the model. For example, in predicting stock market returns one might use squared error between predictions and actual data as a score function to train one's model. However, if the model is then used in a real financial environment, a host of other factors such as trading costs, risks, diversity, and so forth, come into play to determine the true utility of the model. This illustrates that we often settle for simpler "generic" score functions (such as squared error) that have many desirable well-understood properties and are relatively easy to work with. Of course, one should not take this to an extreme: the score function being used should reflect the overall goals of the data mining task as far as is possible. One should try to avoid the situation, unfortunately all too common in practice, of using a convenient score function (perhaps because it is the default score function in the software package being used) that is completely inappropriate for the task.

Different score functions have different properties, and are useful in different situations. One of the goals of this chapter is to make the reader aware of these differences and of the implications of using one score function rather than another. Just as there are a few fundamental principles underlying model and pattern structures, so there are some basic principles underlying the different score functions. These are outlined in this chapter.

It is useful to make three distinctions at the outset. The first is between score functions for models, and score functions for patterns. The second is between score functions for *predictive* structures, and score functions for *descriptive* structures. And the third is between score functions for models of fixed complexity, and score functions for models of different complexity. These distinctions will be illustrated below.

A minor comment on the terminology used below is in order. In some places we will refer to score functions (such as error) that we clearly wish to *minimize,* whereas in other places we will refer to score functions (such as log-likelihood) that we clearly wish to *maximize.* The general concept is the same in either case, since the negative (or "inverse") of an "error-based" score function can always be maximized, and vice versa.

7.2 Scoring Patterns

Since the whole idea of searching for local patterns in data is relatively recent, there is a far smaller toolbox of techniques available for scoring patterns compared to the plethora of techniques for scoring models. Indeed, there is really no general consensus on how patterns should be scored. This is largely a result of the fact that the usefulness of a pattern lies in the eye of the beholder. One person's noisy outlier may be another person's Nobel Prize. Fundamentally, patterns might be evaluated in terms of how interesting or unexpected they are to the data analyst. But we could only hope to quantify this if somehow we had a precise model of what the user actually knows already. We are all familiar with the experience that the first time we learn something surprising is a lot more informative than the fifth or tenth time we hear the same information again. Thus, the degree to which a pattern is interesting to a person must be a function of their prior knowledge.

In practice, however, we cannot hope (except in simple situations) to be able to model a person's prior knowledge. Faced with a data set, a scientist or a marketing expert would have difficulty in precisely formulating what it is that they already know about the problem. Even subjective Bayesians can have problems choosing priors for complex multiparameter models—and evade them by choosing standard forms for the priors, that are only very simplistic representations of prior knowledge. We have found that, once certain patterns begin to emerge from the data (via visualization, descriptive statistics, or rules found by a data mining algorithm), database owners often say "Ah yes, but of course we knew that already," changing their minds about what they claim to have expected once they have seen the data.

Having said all of this, the fact remains that most techniques currently used in data mining for scoring patterns essentially assume that they are measuring degree of informativeness relative to a *completely uninformed prior model*; that is, it is effectively assumed that the data analyst has no prior information at all about the problem, beyond perhaps a few simple marginal and descriptive statistics. The hope is that this will eliminate the very obvious patterns (by focusing attention on patterns that are different from the known simple ones) and that the user can effectively "post-prune" the remaining patterns found by the algorithm to retain the truly interesting ones. The danger, of course, is that for some data sets and some forms of pattern searches, almost all patterns that are found by the data mining algorithm will essentially be uninteresting to the data analyst.

To illustrate these ideas we choose one simple (but widely used) pattern structure, the *probabilistic rule* (as discussed in chapter 5 under association rules) and explored in detail later in chapter 13. This has the form

$$\text{IF } a \text{ THEN } b \text{ with probability } p$$

where a and b are Boolean propositions (events) defined on a subset of the variables of interest and $p = p(b|a)$. How can we measure how interesting or informative this rule is to an uninformed observer? One simple approach is to assume that the observer already knows the marginal (unconditional) probability for the event b, $p(b)$.

For example, suppose that we are studying a population of data miners. Let b represent the event that a randomly chosen person in this population is a data mining researcher, and let a be the event that such a person has read this book. Suppose we find that $p(b) = 0.25$ and that $p(b|a) = 0.75$; that is, 25% of this population are researchers and 75% of people who have read this book are researchers. This is interesting because it tells us that the proportion of people who undertake research is higher among those who have read the book than it is in this population of data miners in general (and hence, by implication, it is higher than among the people who have not read the book). Note, as an aside, that there are no causal implications to this. It could be that the book inspired a reader to take up research, or that a person involved in research hoped the book would help them.

The types of simple score functions that are used to capture the informativeness of such a rule rely in general on how "far" the posterior probability $p(b|a)$ is (after learning that event a is true), from the prior probability $p(b)$. Thus, for example, one could simply measure absolute distance between the probabilities $|p(b|a) - p(b)|$, or perhaps measure distance on log-odds scale, $\log \frac{p(b|a)}{p(\bar{b}|a)}$ where $\bar{b}$ represents the event that a person is not a researcher.

When we compare *different* patterns, such as $p(b|a)$ and $p(b|c)$, it is also useful to take into account the *coverage* of a pattern—that is, the proportion of the data to which it applies. To continue our example above, let c be the condition that the a randomly chosen data miner is one of the three authors of this book. A second pattern might be "if c then b" ("if a data miner is an author of this book then they are a researcher"), with $p(b|c) = 1$ since the three authors are all researchers. However, the condition c only applies to three data miners, which is a very small fraction of the universe of data miners. On the other hand, (we hope that) the coverage of event a will be much larger; that is, $p(a)$ is significantly greater than $p(c)$. To illustrate, suppose that $p(a) = 0.2$ and $p(c) = 0.003$. Then, although the second pattern is very accurate ($p(b|c) = 1$) it is not particularly useful since it only applies to a very small fraction of the population (0.3%), whereas the first pattern is not as accurate ($p(b|a) = 0.75$) but it has much broader applicability (to 20% of the population). It is easy to develop a variety of measures that augment the score function to take coverage into account. For example, we could multiply the previously defined scores by the probability of the conditioning event; $p(a)|p(b|a) - p(b)| = |p(b, a) - p(b)p(a)|$ that can be interpreted as measuring the difference in probability between an independence assumption and the observed joint probability for the two events a and b. Alternatively, the approach used in association rule mining (chapters 5 and 13) defines a threshold p_t, and only seeks patterns with coverage greater than p_t.

There are numerous other score functions for patterns that have been proposed in the data mining literature. None have gained widespread acceptance or general use, largely because judging the novelty and utility of a pattern is often quite subjective and application-specific. Thus, human interpretation by a domain expert remains the most practical way to evaluate patterns at present (e.g., having a human search through and interpret a set of candidate patterns produced by a data mining algorithm).

7.3 Predictive versus Descriptive Score Functions

We now turn to score functions for models, where there is a much greater selection of useful methods available compared to patterns.

7.3.1 Score Functions for Predictive Models

A convenient place to begin is by considering the distinction between prediction and description. Score functions for predictive problems are relatively straightforward. In a prediction task, our training data comes with a "target" value Y, this being a quantitative variable for regression or a categorical variable for classification, and our data set $D = \{(\mathbf{x}(1), y(1)), \ldots, (\mathbf{x}(n), y(n))\}$ consists of pairs of input vectors and target values. Let $\hat{f}(\mathbf{x}(i); \theta)$ be the prediction generated by the model for individual i, $1 \leq i \leq n$, using parameter values θ. Let $y(i)$ be the actual observed value (or "target") for the ith individual in the training data set.

Clearly our score function should be a function of the difference between the predictions $\hat{f}(\mathbf{x}(i); \theta)$ and the targets $y(i)$. Commonly used score functions include the sum of squared errors,

$$S_{SSE}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left(\hat{f}(\mathbf{x}(i); \theta) - y(i) \right)^2 \tag{7.1}$$

for quantitative Y, and the misclassification rate (or error rate or "zero-one" score function) for categorical Y, namely,

$$S_{0/1}(\theta) = \frac{1}{N} \sum_{i=1}^{N} I \left(\hat{f}(\mathbf{x}(i); \theta), y(i) \right) \tag{7.2}$$

where $I(a, b) = 1$ if a is not equal to b and 0 otherwise. These are the two most widely-used score functions for regression and classification respectively. They are simple to understand and (in the case of squared error at least) often lead to straightforward optimization problems.

However, note that we have made some strong assumptions in how these score functions are defined above. For example, by summing over the individual errors we are assuming that errors for all individuals may be treated equally. This is a very common assumption and generally useful. However, if (for example) we have a data set in which the measurements were taken at different times, we might want to assign higher weight in the score function to predictions on more recent items. Similarly, we might have different subsets of items in the data set where the target values are more reliable in some subsets than others (for example, some quantification of measurement error in a subset). Here we might wish to assign lower weight in the score function to predictions on the items with less reliable measurements.

Furthermore, both are functions *only* of the difference between the predictions and targets—in particular, they do not depend on the *values* of the target $y(i)$. This is something we might want to take account of. For example, if Y were a categorical variable indicating whether or not a person had cancer, we might wish to give more weight to the error of not detecting a true cancer and less weight to errors that correspond to false alarms. For real-valued Y, *squared*-error may not be appropriate—perhaps the quality of the model is more appropriately reflected in *absolute error* (squared-error gives greater weight to extreme differences between the observed and predicted Y values than does absolute error). And, as a third example, in an investment scenario, we might want be more tolerant (from a risk-taking standpoint) of predictions of Y that *underestimate* the true value than we are to predictions that *overestimate,* suggesting that an asymmetric function might be more appropriate.

The basic score functions above are rather simple. Thus, we may need in practice to adjust them to reflect the aims of our data mining project more accurately. Sometimes this is not easy (defining the "real aims" may be difficult, especially in data mining contexts, where problems are often open ended). In other cases, even if one cannot state the aims precisely, one might be able to improve on the basic score function. For example, for the cancer problem, instead of using the zero-one loss function it might be more appropriate to define a score function based on a *cost matrix.* Thus, let $\hat{k}$ be the predicted class, k the true class, and define a matrix of "costs" $c(\hat{k}, k), 1 \leq \hat{k}, k \leq K$ that reflects the severity of classifying a patient with true class k into class $\hat{k}$.

In selecting a score function for a particular predictive data mining task there is always a trade-off between choosing a simple score function (such as sum of squared errors) and a much more complex one. The simpler score function will usually be more convenient to work with computationally and will be easier to define. However, more complex score functions (such as those mentioned above) may reflect better the actual reality of the prediction problem. An important point is that many data mining algorithms (such as tree models, linear regression models, and so forth) can in principle handle fairly general score functions—e.g., an algorithm based on cross-validation can use any well-defined score function. Of course, even though this is true in theory, in practice not all software implementations allow the data miner to define their own application-specific score function.

7.3.2 Score Functions for Descriptive Models

For descriptive models, in which there is no "target" variable to be predicted, it is less clear how to define a score function. A fundamental approach is through the likelihood function, which we introduced in chapter 4, but which we here describe from a slightly different perspective. Let $\hat{p}(\mathbf{x};\theta)$ be the estimated probability of observing a data point at $\mathbf{x}$, as defined by our model $\hat{p}$ with parameters θ, where X is categorical (the extension to continuous variables is straightforward, and $\hat{p}$ would then be a probability density function). If the model is a good one, then it might be expected to place a high probability at those values of X where a data point is observed. Thus $\hat{p}(\mathbf{x})$ itself can be taken as a measure of quality of the model—a score function—at the point $\mathbf{x}$. This is the basic idea of maximum likelihood (chapter 4) once again: better models assign higher probability to observed data. (This is fine actually as long as we can assume that all the models we are considering have equal functional complexity, so that the comparison is "fair"—the case in which we are comparing models of different complexities will be discussed later in this chapter.)

If we assume that the data points have arisen independently, we can define an overall score function for the model by combining these score functions for the individual data points simply by multiplying them together:

$$L(\theta) = \prod_{i=1}^{n} \hat{p}(\mathbf{x}(i);\theta). \tag{7.3}$$

This is again the likelihood function of chapter 4, for a set of data points, that we maximize to find an estimate of θ. As we noted there, it is typically more convenient to work with the log-likelihood. Now the contribution of an individual data point to the overall score function is $\log \hat{p}(\mathbf{x}(i); \theta)$, and the overall function is the sum of these:

$$\log L(\theta) = \sum_{i=1}^{n} \log \hat{p}(\mathbf{x}(i); \theta). \tag{7.4}$$

If we work with the negative of the $\log \hat{p}(\mathbf{x}(i); \theta)$, as is often done, then this function needs to be *minimized*. We define

$$S_L(\theta) = -\log L(\theta) = -\sum_{i=1}^{n} \log \hat{p}(\mathbf{x}(i); \theta). \tag{7.5}$$

Note again the intuitive interpretation: $-\log \hat{p}$ is our error term (it gets larger as $\hat{p}$ gets smaller), and we are summing this over all of our data points. The largest possible value for $\hat{p}$ is 1 (for categorical data) and, hence, $S_L(\theta)$ is lower bounded by 0. Thus, we can think of $S_L(\theta)$ as a type of entropy term that measures how well the parameters θ can compress (or predict) the training data.

A particularly useful feature of the likelihood (or, equivalently, the negative log-likelihood) is that it is very general. It can be defined for any problem in which the model or pattern being examined is expressed in terms of probability functions. For example, one might assume that Y in a predictive model is a perfect linear function of some predictor variable X, as well as extra randomly distributed errors, as discussed in the last section. If one can postulate a parametric form for the probability distribution of these errors, then one can compute the likelihood of the data for any proposed parameters in the model. In fact, as we saw in chapter 4, if the error terms are supposed to be Normally distributed with mean 0 about a deterministic function of X then the likelihood score function is equivalent to the sum of squared errors score function.

Although (negative log-)likelihood is a powerful and useful score function, it too has its limitations. In particular, if a parameterization assigns any data point a probability near 0, the log-likelihood will approach $-\infty$. Thus, the overall error can be dominated by extreme points. If the true probability of that same point is also very small, then the model is being penalized for a prediction in the tails of the density function (very unlikely events), that may have little relation to the practical utility of the model. Conversely, there may be problems (such as predicting the occurrence of rare events) in which it is precisely in the tails of the density that we are most interested in accurate prediction. Thus, while likelihood is based on strong theoretical foundations and is generally useful for scoring probabilistic models, it is important to realize that it may not necessarily reflect the true utility of a model for a particular task. Other score functions for determining the quality of probabilistic predictions are also possible, each with its own particular characteristics. For example we can define the integrated squared error between our estimate $\hat{p}(\mathbf{x};\theta)$ and the true probability $p(\mathbf{x})$, $\int(\hat{p}(\mathbf{x};\theta)-p(\mathbf{x}))^2 d\mathbf{x}$. By completing the square, and ignoring terms not depending on θ, we get a score function of the form $\int \hat{p}(\mathbf{x};\theta)^2 d\mathbf{x} - 2E[\hat{p}(\mathbf{x};\theta)]$, where each term can be empirically approximated to provide an estimate of the true integrated squared error as a function of θ.

For *nonprobabilistic* descriptive models, such as partition-based clustering, it is quite easy to come up with all sorts of score functions based on how well separated the clusters are, how compact they are, and so forth. For example, for simple prototype-based clustering (the k-means model discussed in chapter 9), a simple and widely used score function is the sum of square errors within each cluster

$$S_{KSSE}(\theta) = \sum_{k=1}^{K} e_k, \quad e_k = \sum_{i \in cluster_k} ||\mathbf{x}(i) - \mu_k||^2 \tag{7.6}$$

where θ is the parameter vector for the cluster model, $\theta = \{\mu_1, \ldots, \mu_K\}$, and the μ_ks are the cluster centers. However, it is quite difficult to formulate any score function for cluster models that reflect how close the clusters are to "truth" (if this is regarded as meaningful). The ultimate judgment on the utility of a given clustering depends on how useful the clustering is in the context of the particular application. Does it provide new insight into the data? Does it permit a meaningful categorization of the data? And so on. These are questions that typically can only be answered in the context of a particular problem and cannot be captured by a single score metric. To put it another way, once again the score functions for tasks such as clustering are not necessarily very closely related to the true utility function for the problem. We will return to the issue of score functions for clustering tasks in chapter 9.

To summarize, there are simple "generic" score functions for tasks such as classification, regression, and density estimation, that are all useful in their own right. However, they do have limitations, and it is perhaps best to regard them as starting points from which to generalize to more application-specific score functions.

7.4 Scoring Models with Different Complexities

In the preceding sections we described score functions as minimizing some measure of discrepancy between the observed data and the proposed model. One might expect models that are close to the data (in the sense embodied in the score function) to be "good" models. However, we need to be clear about why we are building the model.

7.4.1 General Concepts in Comparing Models

We can distinguish between two types of situations (as we have in earlier chapters). In one type of situation we are merely trying to build a summarizing descriptive model of a data set that captures its main features. Thus, for example, we might want to summarize the main chemical compounds among the members of a particular family of compounds, where our database contains records for all possible members of this family. In this case, accuracy of the model is paramount—though it will be mediated by considerations of comprehensibility. The best accuracy is given by a model that exactly reproduces the data, or describes the data in some equivalent form, but the whole point of the modeling exercise in this case is to reduce the complexity of the data to something that is more comprehensible. In situations like this, simple goodness of fit of the model to the data will be one part of an overall score measure, with comprehensibility being another part (and this part will be subjective). An example of a general technique in this context is based on data compression and information-theoretic arguments, where our score function is generally decomposed as

$$\begin{aligned} S_I(\theta, M) \quad = \quad & \text{number of bits to describe the data given the model} \\ & \quad + \text{number of bits to describe the model (and parameters)} \end{aligned}$$

where the first term measures the goodness of fit to the data and the second measures the complexity of the model M and its parameters θ. In fact, for the first term ("number of bits to describe the data given the model) we can use $S_L = -\log p(D|\theta, M)$ (negative log-likelihood, log base 2). For the second term ("number of bits to describe the model") we can use $-\log p(\theta, M)$ (this is in effect just taking negative logs of the general Bayesian score function discussed in chapter 4). Intuitively, we can think of $-\log p(\theta, M)$ (the second term) as the communication "cost" in bits to transmit the model structure and its parameters from some hypothetical transmitter to a hypothetical receiver, and S_L (the first term) as the cost of transmitting the portion of the data (the errors) that the model and its parameters do not account for. These two parts will tend to work in opposite directions—a good fit to the data will be achieved by a complicated model, while comprehensibility will be achieved by a simple model. The overall score function trades off what is meant by an acceptable model.

In the other general situation our aim is really to generalize from the available data to new data that could arise. For example, we might want to infer how new customers are likely to behave or infer the likely properties of new sky objects not yet observed. Once again, while goodness of fit to the observed data is clearly a part of what we will mean by a good model, it is not the whole story. In particular, since the data do not represent the whole population (there would be no need for generalization if they did) there will be aspects of the observed data ("noise") that are not characteristic of the entire population and vice versa. A model that provided a very good fit to the observed data would also fit these aspects—and, hence, would not provide the best possible predictions. Once again, we need to modify the simple goodness of fit measure in order to define an overall score function. In particular, we need to modify it by a component that prevents the model from becoming too complex, and fitting all the idiosyncrasies of the observed data.

In both situations, an ideal score function strikes some sort of compromise between how well the model fits the data and the simplicity of the model, although the theoretical basis for the compromise is different. This difference is likely to mean that different score functions are appropriate for the different situations. Since the compromise when the aim is simply to summarize the main features of a data set necessarily involves a subjective component ("what does the data miner regard as an acceptably simple model?"), we will concentrate here on the other situation: our aim is to determine, from the data we have available, which model will perform best on data we have not yet seen.

7.4.2 Bias-Variance Again

Before examining score functions that we might hope will provide a good fit to data as yet unseen, it will be useful to look in more detail at the need to avoid modeling the available data too closely. We discussed bias and variance in the context of estimates of parameters θ in chapter 4 and we discuss it again here in the more general context of score functions.

As we have mentioned in earlier chapters, it is extremely unlikely that one's chosen model structure will be "correct." There are too many features of the real world for us to be able to model them exactly (and there are also deep questions about just what "correct" means). This implies that the chosen model form will provide only an approximation to the "truth." Let us take a predictive model to illustrate. Then, at any given value of X (which we take to be univariate for simplicity—exactly the same argument holds for multivariate X), the model is likely to provide predicted values of Y that are not exactly right. More formally, suppose we draw many different data sets, fit a model of the specified structure (for example, a piecewise local model with given number of components, each of given complexity; a polynomial function of X of given degree; and so on) to each of them, and determine the expected value of the predicted Y at any X. Then this expected predicted value is unlikely to coincide exactly with the true value. That is, the model is likely to provide a biased prediction of the true Y at any given X. (Recall that bias of an estimate was defined in chapter 4 as the difference between the expected value of the estimate and the true value.) Thus, perfect prediction is too much to hope for!

However, we can make the difference between the expected value of the predictions and the unknown true value smaller (indeed, we can make it as small as we like for some classes of models and some situations) by increasing the complexity of the model structure. In the examples above, this means increasing the number of components in the piecewise linear model, or increasing the degree of the polynomial.

At first glance, this looks great—we can obtain a model that is as accurate as we like, in terms of bias, simply by taking a complicated enough model structure. Unfortunately, there is no such thing as a free lunch, and the increased accuracy in terms of bias is only gained at a loss in other terms.

By virtue of the very flexibility of the model structure, its predictions at any fixed X could vary dramatically between different data sets. That is, although the average of the predictions at any given X will be close to the true Y (this is what small bias means), there may be substantial variation between the predictions arising from the different data sets. Since, in practice, we will only ever observe one of these predictions (we really have only one data set to use to estimate the model's parameters) the fact that "on average" things are good will provide little comfort. For all we know we have picked a data set that yields predictions far from the average. There is no way of telling.

There is another way of looking at this. Our very flexible model (with, for example, a large number of piecewise components or a high degree) has led to one that closely follows the data. Since, at any given X, the observed value of Y will be randomly distributed about its mean, our flexible model is also modeling this random component of the observed Y value. That is, the flexible model is *overfitting* the data.

Finally (though, yet again, it is really just another way of looking at the same thing), increasing the complexity of the model structure means increasing the number of parameters to be estimated. Generally, if more parameters are being estimated, then the accuracy of each estimate will decrease (its variance, from data set to data set, will increase).

The complementarity of bias and variance in the above, is termed the *bias-variance trade-off*. We want to choose a model in which neither is too large—but reducing either one tends to increase the other. They can be combined to yield an overall measure of discrepancy between the data and the model to yield the *mean squared error* (MSE). Consider the standard regression setting we have discussed before, where we are assuming that y is a deterministic function of $\mathbf{x}$ (where we now generalize to the vector case) with additive noise, that is, $y = f(\mathbf{x}; \theta) + e$, where e is (for example) Normal with zero-mean. Thus, $\mu_y = E[y|\mathbf{x}]$ represents the true (and unknown) expected value for any data point $\mathbf{x}$ (where here the expectation E is with respect to the noise e), and $\hat{y} = f(\mathbf{x}; \theta)$ is the estimate provided by our model and fitted parameters θ. The MSE at $\mathbf{x}$ is then defined as:

$$\begin{aligned} MSE(\mathbf{x}) &= E\left[\hat{y} - \mu_y\right]^2 \\ &= E\left[\hat{y} - E\left(\hat{y}\right)\right]^2 + E\left[E\left(\hat{y}\right) - \mu_y\right]^2 \end{aligned} \tag{7.7}$$

or *MSE* = *Variance* + *Bias*^{2}. (The expectation E here is taken with respect to $p(D)$, the probability distribution over all possible data sets for some fixed size n). This equation bears close inspection. We are treating our prediction $\hat{y}$ here as a random quantity, where the randomness arises from the random sampling that generated the training data D. Different data sets D would lead to different models and parameters, and different predictions $\hat{y}$. The expectation, E, is over different data sets of the same size n, each randomly chosen from the population in question. The variance term $E\left[\hat{y} - E\left(\hat{y}\right)\right]^2$ tell us how much our estimate $\hat{y}$ will vary across different potential data sets of size n. In other words, it measures the sensitivity of $\hat{y}$ to the particular data set being used to train our model. As an extreme example, if we always picked a constant y_1 as our prediction, without regard to the data at all, then this variance would be zero. At the other extreme, if we have an extremely complex model with many parameters, our predictions $\hat{y}$ may vary greatly depending from one individual training data set to the next.

The bias term $E\left[E\left(\hat{y}\right) - \mu_y\right]$ reflects the systematic error in our prediction—

that is how far away our average prediction is, $E(\hat{y})$, from truth μ_y. If we use a constant y_1 as our prediction, ignoring the data, we may have large bias (that is, this difference may be large). If we use a more complex model, our average prediction may get closer to the truth, but our variance may be quite large. The bias-variance quantifies the tension between simpler models (low variance, but potentially high bias) and more complex ones (potentially low bias but typically high variance).

In practice, of course, we are interested in the average MSE over the entire domain of the function we are estimating, so we might define the expected MSE (with respect to the input distribution $p(\mathbf{x})$) as $\int MSE(\mathbf{x})p(\mathbf{x})d\mathbf{x}$, that again has the same additive decomposition (since expectation is linear).

Note that while we can in principle measure the variance of our predictions $\hat{y}$ (for example, by some form of resampling such as the bootstrap method), the bias will always be unknown since it involves μ_y that is itself unknown (this is after all what we are trying to learn). Thus, the bias-variance decomposition is primarily of theoretical interest since we cannot measure the bias component explicitly, and in turn it does not provide a practical score function combining these two aspects of estimation error. Nonetheless, the practical implications in general are clear: we need to choose a model that is not too inflexible (because its predictions will then have substantial bias) but not too flexible (since then its predictions will have substantial variance). That is, we need a score function that can handle models of different complexities and take into account this compromise, and one that can be implemented in practice. This is the focus of the next section.

We should note that in certain data mining applications, the issue of variance may not be too important, particularly when the models are relatively simple compared to the amount of data being used to fit them. This is because variance is a function of sample size (as we discussed in chapter 4). Increasing the sample size decreases the variance of an estimator. Unfortunately, no general statements can be made about when variance and overfitting will be important issues. It depends on both the sample size of the training data D and the complexity of the model being fit.

7.4.3 Score Functions That Penalize Complexity

How, then, can we choose a suitable compromise between flexibility (so that a reasonable fit to the available data is obtained) and overfitting (in which the model fits chance components in the data)? One way is to choose a score function that encapsulates the compromise. That is, we choose an overall score function that is explicitly composed of two components: a component that measures the goodness of fit of the model to the data, and an extra component that puts a premium on simplicity. This yields an overall score function of the form

$$\text{score(model)} = \text{error(model)} + \text{penalty-function(model)},$$

where we want want to minimize this score. We have discussed several different ways to define the error component of the score in the preceding sections. What might the additional penalty component look like?

In general (though there are subtleties that mean that this is something of a simplification), the complexity of a model M will be related to the number of parameters, d, under consideration. We will adopt the following notation in this context. Consider that there are K different model structures, $M_1, \ldots, M_K$, from which we wish to choose one (ideally the one that predicts best on future data). Model M_k has d_k parameters. We will assume that for each model structure $M_k, 1 \leq k \leq K$, the best fitting parameters $\hat{\theta}_k$ for that model (those that maximize goodness-of-fit to the data) have already been chosen; that is, we have already determined point estimates of these parameters for each of these K model structures and now we wish to choose among these fitted models.

The widely used *Akaike information criterion* or *AIC* is defined as

$$S_{AIC}(M_k) = 2S_L(\hat{\theta}_k; M_k) + 2d_k, \quad 1 \leq k \leq K. \tag{7.8}$$

where S_L is the negative log-likelihood as defined in equation 7.5 and the penalty term is $2d_k$. This can be derived formally using asymptotic arguments.

An alternative, based on Bayesian arguments, also takes into account the sample size, n. This *Bayesian Information Criterion* or *BIC* is defined as

$$S_{BIC}(M_k) = 2S_L(\hat{\theta}_k; M_k) + d_k \log n \tag{7.9}$$

where S_L is again the negative log-likelihood of 7.5. Note the effect of the additive penalty term $d_k \log n$. For fixed n, the penalty term grows linearly in number of parameters d_k, which is quite intuitive. For a fixed number of parameters d_k, the penalty term increases in proportion to $\log n$. Note that this logarithmic growth in n is offset by the potentially linear growth in S_L as a function of n (since it is a sum of n terms). Thus, asymptotically as n gets very large, for relatively small values of d_k, the error term S_L (linear in n) will dominate the penalty term (logarithmic in n). Intuitively, for very large numbers of data points n, we can "trust" the error on the training data and the penalty function term is less relevant. Conversely, for small numbers of data points n, the penalty function term $d_k \log n$ will play a more influential role in model selection.

There are many other penalized score functions with similar additive forms to those above (namely an error-based term plus a penalty term) include the adjusted R^2 and C_p scores for regression, the minimum description length (MDL) method (which is closely related to the MAP score of chapter 4), and Vapnik's structural risk minimization approach (SRM).

Several of these penalty functions can be derived from fairly fundamental theoretical arguments. However, in practice these types of penalty functions are often used under far broader conditions than the assumptions used in the derivation of the theory justify. Nonetheless, since they are easy to compute they are often quite convenient in practice in terms of giving at least a general idea of what the appropriate complexity for a model is, given a particular data set and data mining task.

A different approach is provided by the Bayesian framework of chapter 4. We can try to compute the posterior probability of each model given the data directly, and select the one with the highest posterior probability; that is,

$$\begin{aligned} p(M_k|D) &\propto p(D|M_k)p(M_k) \\ &= \int p(D, \theta_k|M_k)p(M_k)d\theta_k \\ &= \int p(D|\theta_k)p(\theta_k|M_k)d\theta_k \; p(M_k) \end{aligned} \tag{7.10}$$

where the integral represents calculating the expectation of the likelihood of the data over parameter space (also known as *marginal likelihood*), relative to a prior in parameter space $p(\theta_k|M_k)$, and the term $p(M_k)$ is a prior probability for each model. This is clearly quite different from the "point estimate" methods—the Bayesian philosophy is to fully acknowledge uncertainty and, thus, average over our parameters (since we are unsure of their exact values) rather than "picking" point estimates such as $\hat{\theta_k}$. Note that this Bayesian approach implicitly penalizes complexity, since higher dimensional parameter spaces (more complex models) will mean that the probability mass in $p(\theta_k|M_k)$ is spread more thinly than in simpler models.

Of course, in practice explicit integration is often intractable for many parameter spaces and models of interest and Monte Carlo sampling techniques are used. Furthermore, for large data sets, the $p(D|\theta_k)$ function may in fact be quite "peaked" about a single value $\hat{\theta_k}$ (recall the maximum likelihood estimation examples in chapter 4), in which case we can reasonably approximate the Bayesian expression above by the value of the peak plus some estimate of the surrounding volume (for example, a Taylor series type of expansion around the posterior mode of $p(D|\theta)p(\theta)$)—this type of argument can be shown to lead to approximations such as BIC above).

7.4.4 Score Functions Using External Validation

A different strategy for choosing models is sometimes used, not based on adding a penalty term, but instead based on external validation of the model. The basic idea is to (randomly) split the data into two mutually exclusive parts, a *design* part D_d, and a *validation* part D_v. The design part is used to construct the models and estimate the parameters. Then the score function is recalculated using the validation part. These validation scores are used to select models (or patterns). An important point here is that our estimate of the score function for a particular model, say $S(M_k)$, is itself a random variable, where the randomness comes from both the data set being used to train (design) the model and the data set being used to validate it. For example, if our score is some error function between targets and model predictions (such as sum of squared errors), then ideally we would like to have an *unbiased estimate* of the value of this score function on future data, for each model under consideration. In the validation context, since the two data sets are independently and randomly selected, for a given model the validation score provides an unbiased estimate of the score value of that model for new ("out-of-sample") data points. That is, the bias in estimates, that inevitably arises with the design component, is absent from the independent validation estimate. It follows from this (and the linearity of expectation) that the difference between the scores of two models evaluated on a validation set will have an expected value in the direction favoring the better model. Thus, the difference in validation scores can be used to choose between models. Note that we have previously discussed unbiased estimates of parameters θ (chapter 4), unbiased estimates of what we are trying to predict μ_y (earlier in this chapter), and now unbiased estimates of our score function S. The same principles of bias and variance underly all three contexts, and indeed all three contexts are closely interlinked (accuracy in parameter estimates will affect accuracy of our predictions, for example)—it is important, however, to understand the distinction between them.

This general idea of validation has been extended to the notion of *cross-validation*. The splitting into two independent sets is randomly repeated many times, each time estimating a new model (of the given form) from the design part of the data and obtaining an unbiased estimate the out-of-sample performance of each model from the validation component. These unbiased estimates are then averaged to yield an overall estimate. We described the use of cross-validation to choose between CART recursive partitioning models in chapter 5. Cross-validation is popular in practice, largely because it is simple and reasonably robust (in the sense that it relies on relatively few assumptions). However, if the partitioning is repeated m times it does come at a cost of (on the order of) m times the complexity of a method based on just using a single validation set. (There are exceptions in special cases. For example, there is an algorithm for the leave-one-out special case of cross-validation applied to linear discriminant analysis that has the same order of computational complexity as the basic model construction algorithm.)

For small data sets, the process of selecting validation subsets D_v can lead to significant variation across data sets, and thus, the *variance* of the cross-validation score also needs to be monitored in practice to check whether or not the variation may be unreasonably high. Finally, there is a subtlety in cross-validation scoring in that we are averaging over models that have potentially different parameters but the same complexity. It is important that we are actually averaging over essentially the *same* basic model each time. If, for example, the fitting procedure we are using can get trapped at different local maxima in parameter space, on different subsets of training data, it is not clear that it is meaningful to average over the validation scores for these models.

It is true, as stated above, that the estimate of performance obtained from such a process for a given model is unbiased. This is why such methods are very widely used and have been extensively developed for performance assessment (see Further Reading). However, some care needs to be exercised. If the validation measures are subsequently used to choose between models (for example, to choose between models of different complexity), then the validation score of the model that is finally selected will be a biased estimate of this model's performance. To see this, imagine that, purely by chance some model did exceptionally well on a validation set. That is, by the accidental way the validation set happened to have fallen, this model did well. Then this model is likely to be chosen as the "best" model. But clearly, this model will not do so well with new out-of-sample data sets. What this means in practice is that, if an assessment of the likely future performance of a (predictive) model is needed, then this must be based on yet a third data set, the *test set*, about which we shall say more in the next subsection.

7.5 Evaluation of Models and Patterns

Once we have selected a model or pattern, based on its score function, we will often want to know (in a predictive context) how well this model or pattern will perform on new unseen data. For example, what error rate, on future unseen data, would we expect from a predictive classification model we have built using a given training set? We have already referred to this issue when discussing the validation set method of model selection above.

Again we note that if any of the same data that have been used for selecting a model or used for parameter estimation are then also used again for performance evaluation, then the evaluation will be optimistically biased. The model will have been chosen precisely because it does well on this particular data set. This means that the *apparent* or *resubstitution* performance, as the estimate based on reusing the training set is called, will tend to be optimistically biased.

If we are only considering a single model structure, and not using validation to select a model, then we can use subsampling techniques such as validation or cross-validation, splitting the data into training and test sets, to obtain an unbiased estimate of our model's future performance. Again this can be repeated multiple times, and the results averaged. At an extreme, the test set can consist of only one point, so that the process is repeated N times, with an average of the N single scores yielding the final estimate. This principle of leaving out part of the data, so that it can provide an independent test set, has been refined and developed to a great degree of technical depth and sophistication, notably in jackknife and bootstrap methods, as well as the leaving-one-out cross-validation method (all of these are different, though related and sometimes confused). The further reading section below gives pointers to publications containing more details.

The essence of the above is that, to obtain unbiased estimates of likely future performance of a model we must assess its performance using a data set which is independent of the data set used to construct and select the model. This also applies if validation data sets are used. Suppose, for example, we chose between K models by partitioning the data into two subsets, where we fit parameters on the first subset, and select the single "best" model using the model scores on the second (validation) subset . Then, since we will choose that model which does best on the validation data set, the model will be selected so that it fits the idiosyncrasies of this validation data set. In effect, the validation data set is being used as part of the design process and performance as measured on the validation data will be optimistic. This be-

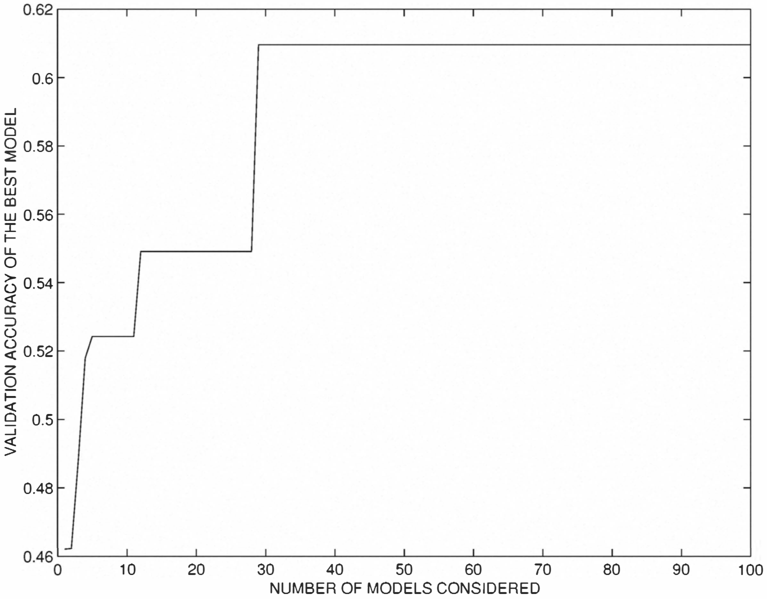

Figure 7.1 Classification accuracy of the best model selected on a validation data set from a set of K models, $1 \leq K \leq 100$, where each model is making random predictions.

comes more severe, the larger is the set of models from which the final model is chosen.

Example 7.1 The problem of optimistic performance on validation data is illustrated by a hypothetical two-class classification problem where we have selected the best of K models using a validation data set of 100 data points. We have taken the two classes to have equal prior probabilities of 0.5 and, to take an extreme situation, have arranged things so that none of the "predictor" variables in our models have any predictive power at all; that is, all the input variables are independent of the class variable Y. This means that each model is in fact generating purely random predictions so that the long-run accuracy on new unseen data for any of the models will be 0.5 (although of course we would not be aware of this fact). Figure 7.1 shows the cross-validation accuracy obtained from a simple simulation of this scenario, where we increased the number of models K being considered from 1 to 100. When we chose from a small number of models (fewer than 10) the proportion of validation set points correctly classified by the best of them is close enough to 0.5. However, by $K = 15$ the "best" model, selected using the validation set, correctly classifies a proportion 0.55 of the validation set points, and by $k = 30$ the best model classifies a proportion 0.61 of the validation set correctly.

The message here is that if one uses a validation set to choose between models, one cannot also use it to provide an estimate of likely future performance. The very fact that one is choosing models which do well on the validation set means that performance estimates on this set are biased as estimates of performance on other unseen data. As we said above, the validation set, being used to choose between models, has really become part of the design process. This means that to obtain unbiased estimates of likely future performance we ideally need access to yet another data set (a "hold-out" set) that has not been used in any way in the estimation or model selection so far. For very large data sets this is usually not a problem, in that data is readily available, but for small data sets it can be problematic since it effectively reduces the data available for training.

7.6 Robust Methods

We have pointed out elsewhere that the notion of a "true" model is nowadays regarded as a weak one. Rather, it is assumed that all models are approximations to whatever is going on in nature, and our aim is to find a model that is close enough for the purpose to hand. In view of this, it would be reassuring if our model did not change too dramatically as the data on which it was based changed. Thus, if a slight shift in value of one data point led to radically different parameter estimates and predictions in a model, one might be wary of using it. Put another way, we would like our models and patterns to be insensitive to small changes in the data. Likewise, the score functions and models may be based on certain assumptions (for example, about underlying probability distributions). Again it would be reassuring if, if such assumptions were relaxed slightly, the fitted model and its parameters and predictions did not change dramatically.

Score functions aimed at achieving these aims have been developed. For example, in a *trimmed* mean a small proportion of the most extreme data points are dropped, and the mean of the remainder used. Now the values of outlying points have no effect on the estimate. The extreme version of this (assuming a univariate distribution with equal numbers being dropped from each tail), arising as a higher and higher proportion is dropped from the tails, is the median—which is well known to be less sensitive to changes in outlying points than is the arithmetic mean. As another example, the Winsorized mean involves reducing the most extreme points to have the same values as the next most extreme points, before computing the usual mean.

Although such modifications can be thought of as robust forms of score functions, it is sometimes easier to describe them (and, indeed think of them) in terms of the algorithms used to compute them.

7.7 Further Reading

Piatetsky-Shapiro (1991), Silberschatz and Tuzhilin (1996), and Bayardo and Agrawal (1999) contain general discussions on score functions for patterns and probabilistic rules.

Hand (1997) discuss score functions for classification problems in great detail. Bishop (1995) discusses score functions in the context of neural networks. Breiman et al. (1984) discuss how general misclassification costs can be used as score functions for tree classifiers. Domingos (1999) provides a flexible methodology for converting any classification algorithm that operates on the assumption of 0-1 classification loss into a more general algorithm that can use any classification cost-matrix.

Devroye (1984) argues for the use of L1 distance measures as score functions for density estimation problems, while Silverman (1986) describes more conventional squared-error (L2) score functions in the same context.

The topics of bias and variance are discussed in a general learning context in the paper by Geman, Bienenstock, and Doursat (1992). Friedman (1997) develops a bias-variance decomposition for classification problems that turns out to have fundamentally different properties to classical squared-error bias-variance.

Linhart and Zucchini (1986) provide an overview of statistical model selection techniques. Chapter 2 of Ripley (1996) provides a comprehensive overview of score functions for model selection in classification and regression. The first general treatment of cross-validation was provided by Stone (1974) while Hjort (1993) outlines more recent ideas on cross-validation and related sampling techniques for model selection. Books on statistical theory (for example Lindsey, 1996) include discussions of penalized model selection in general, including measures such as AIC and BIC. Akaike (1973) introduced the AIC principle and Schwarz (1978) contains the original discussion on BIC. Burnham and Anderson (1998) provide a recent detailed treatment of

BIC and related approaches. Vapnik (1995) contains a detailed account of the structural risk minimization (SRM) approach to model selection and Rissanen (1987) provides a detailed discussion of stochastic complexity, minimum description length (MDL) and related concepts. Lehmann (1986) discusses the more classical statistical approach of comparing two models at a time within a hypothesis-testing framework.

Bernardo and Smith (1994) has a detailed theoretical account of Bayesian approaches to score functions and model selection in general (see also Dawid (1984) and Kass and Raftery (1995)).

Ripley (1996, chapter 2) and Hand (1997) provide detailed discussions of evaluating the performance of classification and regression models. Salzberg (1997) and Dietterich (1998) discuss the specific problem of assessing statistical significance in differences of performance among multiple classification models and algorithms.

Huber (1980) is an important book on robust methods.