1. Introduction

**TABLE 1.1.** Average percentage of words or characters in an email message equal to the indicated word or character. We have chosen the words and characters showing the largest difference between spam and email.

<table>
<thead>
<tr>
<th></th>
<th>george</th>
<th>you</th>
<th>your</th>
<th>hp</th>
<th>free</th>
<th>hpl</th>
<th>!</th>
<th>our</th>
<th>re</th>
<th>edu</th>
<th>remove</th>
</tr>
</thead>
<tbody>
<tr>
<td>spam</td>
<td>0.00</td>
<td>2.26</td>
<td>1.38</td>
<td>0.02</td>
<td>0.52</td>
<td>0.01</td>
<td>0.51</td>
<td>0.13</td>
<td>0.01</td>
<td>0.28</td>
<td></td>
</tr>
<tr>
<td>email</td>
<td>1.27</td>
<td>1.27</td>
<td>0.44</td>
<td>0.90</td>
<td>0.07</td>
<td>0.43</td>
<td>0.11</td>
<td>0.18</td>
<td>0.42</td>
<td>0.29</td>
<td>0.01</td>
</tr>
</tbody>
</table>

measurements for a set of objects (such as people). Using this data we build a prediction model, or learner, which will enable us to predict the outcome for new unseen objects. A good learner is one that accurately predicts such an outcome.

The examples above describe what is called the **supervised learning** problem. It is called “supervised” because of the presence of the outcome variable to guide the learning process. In the **unsupervised learning problem**, we observe only the features and have no measurements of the outcome. Our task is rather to describe how the data are organized or clustered. We devote most of this book to supervised learning; the unsupervised problem is less developed in the literature, and is the focus of Chapter 14.

Here are some examples of real learning problems that are discussed in this book.

**Example 1: Email Spam**

The data for this example consists of information from 4601 email messages, in a study to try to predict whether the email was junk email, or “spam.” The objective was to design an automatic spam detector that could filter out spam before clogging the users’ mailboxes. For all 4601 email messages, the true outcome (email type) email or spam is available, along with the relative frequencies of 57 of the most commonly occurring words and punctuation marks in the email message. This is a supervised learning problem, with the outcome the class variable email/spam. It is also called a **classification** problem.

Table 1.1 lists the words and characters showing the largest average difference between spam and email.

Our learning method has to decide which features to use and how: for example, we might use a rule such as

\[
\text{if } (\% \text{george} < 0.6) \text{ & } (\% \text{you} > 1.5) \text{ then spam \ else email.}
\]

Another form of a rule might be:

\[
\text{if } (0.2 \cdot \% \text{you} - 0.3 \cdot \% \text{george}) > 0 \text{ then spam \ else email.}
\]
2. Overview of Supervised Learning

\[ f(X) = \begin{cases} 0 & \text{if } X_1 < 0 \\ 1 & \text{if } X_1 \geq 0 \end{cases} \]

\[ \text{MSE vs. Dimension} \]

- MSE
- Variance
- Sq. Bias

**FIGURE 2.8.** A simulation example with the same setup as in Figure 2.7. Here the function is constant in all but one dimension: \( F(X) = \frac{1}{2}(X_1 + 1)^3 \). The variance dominates.

unbiased, we find that

\[
\begin{align*}
\text{EPE}(x_0) & = E_{y_0|x_0}E_T(y_0 - \hat{y}_0)^2 \\
& = \text{Var}(y_0|x_0) + E_T[\hat{y}_0 - E_T\hat{y}_0]^2 + [E_T\hat{y}_0 - x_0^T\beta]^2 \\
& = \text{Var}(y_0|x_0) + \text{Var}_T(\hat{y}_0) + \text{Bias}^2(\hat{y}_0) \\
& = \sigma^2 + \text{E}_T x_0^T(X^T X)^{-1}x_0 \sigma^2 + 0^2. \\
& = \frac{\text{trace}[\text{Cov}(X)^{-1}\text{Cov}(x_0)]\sigma^2}{N} + \sigma^2.
\end{align*}
\]

(2.27)

Here we have incurred an additional variance \( \sigma^2 \) in the prediction error, since our target is not deterministic. There is no bias, and the variance depends on \( x_0 \). If \( N \) is large and \( T \) were selected at random, and assuming \( E(X) = 0 \), then \( X^T X \to N\text{Cov}(X) \) and

\[
\begin{align*}
\text{E}_{x_0}\text{EPE}(x_0) & \sim \text{E}_{x_0}x_0^T\text{Cov}(X)^{-1}x_0 \sigma^2/N + \sigma^2 \\
& = \text{trace}[\text{Cov}(X)^{-1}\text{Cov}(x_0)]\sigma^2/N + \sigma^2 \\
& = \sigma^2(p/N) + \sigma^2.
\end{align*}
\]

(2.28)

Here we see that the expected EPE increases linearly as a function of \( p \), with slope \( \sigma^2/N \). If \( N \) is large and/or \( \sigma^2 \) is small, this growth in variance is negligible (0 in the deterministic case). By imposing some heavy restrictions on the class of models being fitted, we have avoided the curse of dimensionality. Some of the technical details in (2.27) and (2.28) are derived in Exercise 2.5.

Figure 2.9 compares 1-nearest neighbor vs. least squares in two situations, both of which have the form \( Y = f(X) + \varepsilon \), \( X \) uniform as before, and \( \varepsilon \sim N(0,1) \). The sample size is \( N = 500 \). For the orange curve, \( f(x) \)
is linear in the first coordinate, for the blue curve, cubic as in Figure 2.8. Shown is the relative EPE of 1-nearest neighbor to least squares, which appears to start at around 2 for the linear case. Least squares is unbiased in this case, and as discussed above the EPE is slightly above \( \sigma^2 = 1 \). The EPE for 1-nearest neighbor is always above 2, since the variance of \( \hat{f}(x_0) \) in this case is at least \( \sigma^2 \), and the ratio increases with dimension as the nearest neighbor strays from the target point. For the cubic case, least squares is biased, which moderates the ratio. Clearly we could manufacture examples where the bias of least squares would dominate the variance, and the 1-nearest neighbor would come out the winner.

By relying on rigid assumptions, the linear model has no bias at all and negligible variance, while the error in 1-nearest neighbor is substantially larger. However, if the assumptions are wrong, all bets are off and the 1-nearest neighbor may dominate. We will see that there is a whole spectrum of models between the rigid linear models and the extremely flexible 1-nearest-neighbor models, each with their own assumptions and biases, which have been proposed specifically to avoid the exponential growth in complexity of functions in high dimensions by drawing heavily on these assumptions.

Before we delve more deeply, let us elaborate a bit on the concept of *statistical models* and see how they fit into the prediction framework.
The quantity \( \text{trace}(S) \) is the effective number of parameters, as defined in Section 7.6. GCV can have a computational advantage in some settings, where the trace of \( S \) can be computed more easily than the individual elements \( S_{ii} \). In smoothing problems, GCV can also alleviate the tendency of cross-validation to undersmooth. The similarity between GCV and AIC can be seen from the approximation 

\[
\frac{1}{(1-x)^2} \approx 1 + 2x
\]

(Exercise 7.7).

7.10.2 The Wrong and Right Way to Do Cross-validation

Consider a classification problem with a large number of predictors, as may arise, for example, in genomic or proteomic applications. A typical strategy for analysis might be as follows:

1. Screen the predictors: find a subset of “good” predictors that show fairly strong (univariate) correlation with the class labels
2. Using just this subset of predictors, build a multivariate classifier.
3. Use cross-validation to estimate the unknown tuning parameters and to estimate the prediction error of the final model.

Is this a correct application of cross-validation? Consider a scenario with \( N = 50 \) samples in two equal-sized classes, and \( p = 5000 \) quantitative predictors (standard Gaussian) that are independent of the class labels. The true (test) error rate of any classifier is 50%. We carried out the above recipe, choosing in step (1) the 100 predictors having highest correlation with the class labels, and then using a 1-nearest neighbor classifier, based on just these 100 predictors, in step (2). Over 50 simulations from this setting, the average CV error rate was 3%. This is far lower than the true error rate of 50%.

What has happened? The problem is that the predictors have an unfair advantage, as they were chosen in step (1) on the basis of all of the samples. Leaving samples out after the variables have been selected does not correctly mimic the application of the classifier to a completely independent test set, since these predictors “have already seen” the left out samples.

Figure 7.10 (top panel) illustrates the problem. We selected the 100 predictors having largest correlation with the class labels over all 50 samples. Then we chose a random set of 10 samples, as we would do in five-fold cross-validation, and computed the correlations of the pre-selected 100 predictors with the class labels over just these 10 samples (top panel). We see that the correlations average about 0.28, rather than 0, as one might expect.

Here is the correct way to carry out cross-validation in this example:

1. Divide the samples into \( K \) cross-validation folds (groups) at random.
2. For each fold \( k = 1, 2, \ldots, K \)
terms of the distances rather than squared distances: the square root on the outside is just a convention. A gradient descent algorithm is used to minimize $S_M$.

A variation on least squares scaling is the so-called Sammon mapping which minimizes

$$S_{Sm}(z_1, z_2, \ldots, z_N) = \sum_{i \neq i'} \frac{(d_{ii'} - \|z_i - z_{i'}\|)^2}{d_{ii'}}. \quad (14.99)$$

Here more emphasis is put on preserving smaller pairwise distances.

In classical scaling, we instead start with similarities $s_{ii'}$: often we use the centered inner product $s_{ii'} = \langle x_i - \bar{x}, x_{i'} - \bar{x} \rangle$. The problem then is to minimize

$$S_C(z_1, z_2, \ldots, z_N) = \sum_{i, i'} (s_{ii'} - \langle z_i - \bar{z}, z_{i'} - \bar{z} \rangle)^2 \quad (14.100)$$

over $z_1, z_2, \ldots, z_N \in \mathbb{R}^k$. This is attractive because there is an explicit solution in terms of eigenvectors: see Exercise 14.11. If we have distances rather than inner-products, we can convert them to centered inner-products if the distances are Euclidean;\(^{14}\) see (18.31) on page 671 in Chapter 18. If the similarities are in fact centered inner-products, classical scaling is exactly equivalent to principal components, an inherently linear dimension-reduction technique. Classical scaling is not equivalent to least squares scaling; the loss functions are different, and the mapping can be nonlinear.

Least squares and classical scaling are referred to as metric scaling methods, in the sense that the actual dissimilarities or similarities are approximated. Shephard–Kruskal nonmetric scaling effectively uses only ranks. Nonmetric scaling seeks to minimize the stress function

$$S_{NM}(z_1, z_2, \ldots, z_N) = \frac{\sum_{i \neq i'} \left( \|z_i - z_{i'}\| - \theta(d_{ii'}) \right)^2}{\sum_{i \neq i'} \|z_i - z_{i'}\|^2} \quad (14.101)$$

over the $z_i$ and an arbitrary increasing function $\theta$. With $\theta$ fixed, we minimize over $z_i$ by gradient descent. With the $z_i$ fixed, the method of isotonic regression is used to find the best monotonic approximation $\theta(d_{ii'})$ to $\|z_i - z_{i'}\|$. These steps are iterated until the solutions stabilize.

Like the self-organizing map and principal surfaces, multidimensional scaling represents high-dimensional data in a low-dimensional coordinate system. Principal surfaces and SOMs go a step further, and approximate the original data by a low-dimensional manifold, parametrized in the low dimensional coordinate system. In a principal surface and SOM, points

---

\(^{14}\)An $N \times N$ distance matrix is Euclidean if the entries represent pairwise Euclidean distances between $N$ points in some dimensional space.
There are variations to this approach that adjust the individual $p$-values to achieve an FWER of at most $\alpha$, with some approaches avoiding the assumption of independence; see, e.g., Dudoit et al. (2002b).

### 18.7.1 The False Discovery Rate

A different approach to multiple testing does not try to control the FWER, but focuses instead on the proportion of falsely significant genes. As we will see, this approach has a strong practical appeal.

Table 18.5 summarizes the theoretical outcomes of $M$ hypothesis tests. Note that the family-wise error rate is $\Pr(V \geq 1)$. Here we instead focus on the false discovery rate

\[ \text{FDR} = \frac{E(V)}{E(R)} \]

(18.43)

In the microarray setting, this is the expected proportion of genes that are incorrectly called significant, among the $R$ genes that are called significant. The expectation is taken over the population from which the data are generated. Benjamini and Hochberg (1995) first proposed the notion of false discovery rate, and gave a testing procedure (Algorithm 18.2) whose FDR is bounded by a user-defined level $\alpha$. The Benjamini–Hochberg (BH) procedure is based on $p$-values; these can be obtained from an asymptotic approximation to the test statistic (e.g., Gaussian), or a permutation distribution, as is done here.

If the hypotheses are independent, Benjamini and Hochberg (1995) show that regardless of how many null hypotheses are true and regardless of the distribution of the $p$-values when the null hypothesis is false, this procedure has the property

\[ \text{FDR} \leq \frac{M_0}{M} \alpha \leq \alpha. \]

(18.45)

For illustration we chose $\alpha = 0.15$. Figure 18.19 shows a plot of the ordered $p$-values $p(j)$, and the line with slope $0.15/12625$. 

### Table 18.5. Possible outcomes from $M$ hypothesis tests. Note that $V$ is the number of false-positive tests; the type-I error rate is $E(V)/M_0$. The type-II error rate is $E(T)/M_1$, and the power is $1 - E(T)/M_1$. 

<table>
<thead>
<tr>
<th></th>
<th>Called Not Significant</th>
<th>Called Significant</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H_0$ True</td>
<td>$U$</td>
<td>$V$</td>
<td>$M_0$</td>
</tr>
<tr>
<td>$H_0$ False</td>
<td>$T$</td>
<td>$S$</td>
<td>$M_1$</td>
</tr>
<tr>
<td>Total</td>
<td>$M - R$</td>
<td>$R$</td>
<td>$M$</td>
</tr>
</tbody>
</table>