Introduction to Classification

Many applications of statistical learning involve **classification**.

- Here y_i can take on a small number of values, say J of them, indexed by j.
- In the binary case, with $J = 2$, statisticians and econometricians usually let $j = 0$ or $j = 1$. Then $\mathrm{E}(y_i) = \Pr(y_i = 1)$.
- However, computer scientists prefer $j = -1$ and $j = 1$.
- When $J > 2$, we can use $j = 0, 1, 2, \ldots, J - 1$ or $j = 1, 2, 3, \ldots, J$.

Classification is related to modeling $\Pr(y_i = j)$, the probability that $y_i = j$, but they are not the same thing.

By itself, a model for $\Pr(y_i = j)$ is not a classifier. We need to add a rule that maps from estimated probabilities to classes.

The obvious one is to classify a point x_0 as belonging to class j when $\Pr(y_0 = j) > \Pr(y_0 = l)$ for all $l \neq j$.

But this rule will yield odd results when any outcome is unlikely conditional on whatever we observe.

- Suppose we have a sample of people who get some disease, where 95% of patients survive.
- Unless a patient has characteristics that make the probability of survival less than 0.5, a good classifier will simply put them into the “patient survives” class.
- But we probably care whether $\Pr(y_i = 1)$ is 0.01 or 0.49!

In cases like this, it is conditional probabilities that we care about, not simply classification.

Some methods, including KNN, naturally provide estimates of conditional probabilities for an observation with features $\boldsymbol{x}_0$.

But others merely assign a class to such an observation.

Evaluating model performance for classification is not the same as evaluating it for estimating conditional probabilities.

A bad measure of classification performance is the **training error rate**

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{I}(y_i \neq \hat{y}_i). \tag{1}$$

Here $\hat{y}_i$ is the class assigned to observation i, presumably on the basis of predictors $\boldsymbol{x}_i$.

Thus (1) is just the fraction of misclassified observations.

The **test error rate** is a more useful measure. It is

$$\frac{1}{n_T}\sum_{i=1}^{n_T}\mathbb{I}(y_{0i} \neq \hat{y}_{0i}), \tag{2}$$

where the y_{0i} belong to the test sample with n_T observations. The $\hat{y}_{0i}$ are computed using observed predictors $\boldsymbol{x}_{0i}$, but only the y_i are used, together with the $\boldsymbol{x}_{0i}$, to obtain the $\hat{y}_{0i}$.

In an ideal world, we would know

$$\Pr(y_0 = j \mid \boldsymbol{x}_0). \tag{3}$$

If so, we could use the **Bayes classifier**, which simply sets $\hat{y}_0$ equal to whichever j maximizes (3).

This is the classification equivalent of using the true $f(\boldsymbol{x}_0)$ for regression. But if the conditional probability of some outcome (like dying) is always small, that outcome will never be chosen.

Since it is the best we can possibly do, the errors made by the Bayes classifier are irreducible.

The Bayes error rate is

$$1 - \mathrm{E}\big(\max_j \Pr(y_0 = j \mid \boldsymbol{x}_0)\big). \tag{4}$$

This is the true irreducible error rate. It relies on the fact that the probabilities are correct. It is never greater than $(J-1)/J$.

The expectation in (4) implicitly takes expectations over all possible values of $\boldsymbol{x}_0$. It is infeasible unless we generated the data and thus know the probabilities.

But with a large sample, a good classification method may come close to the Bayes error rate.

- When $\max_j \Pr(y_0 = j \,|\, \boldsymbol{x}_0)$ is very close to 1, the Bayes error is small. If one probability ≈ 1, then the others are ≈ 0.
- When $\max_j \Pr(y_0 = j \,|\, \boldsymbol{x}_0)$ is $1/J$, the Bayes error is as large as it can be. This happens if all probabilities are the same.

The irreducible error rate can differ greatly across datasets, depending on whether probabilities are close to 1 for the actual outcomes.

In the binary case ($J = 2$), the irreducible error rate is maximized when the conditional probabilities of both outcomes equal 0.5 for all observations. In this case, $\boldsymbol{x}_0$ contains no useful information.

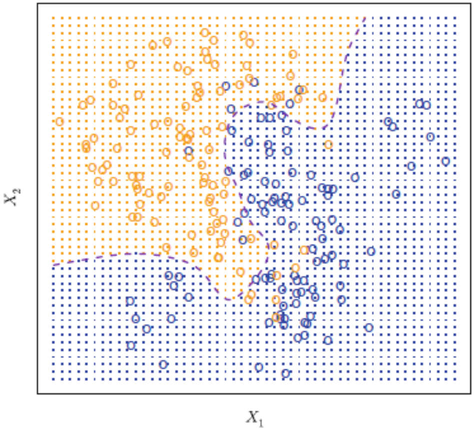

FIGURE 2.13. *A simulated data set consisting of* 100 *observations in each of two groups, indicated in blue and in orange. The purple dashed line represents the Bayes decision boundary. The orange background grid indicates the region in which a test observation will be assigned to the orange class, and the blue background grid indicates the region in which a test observation will be assigned to the blue class.*

It is very easy to use KNN for classification.

If $\mathcal{N}_0$ denotes the K observations nearest to $\boldsymbol{x}_0$, then

$$\widehat{\Pr}(y_0 = j \mid \boldsymbol{x}_0) = \frac{1}{k} \sum_{i \in \mathcal{N}_0} \mathbb{I}(y_i = j). \tag{5}$$

In order to classify each test observation, we then use the Bayes classifier based on $\widehat{\Pr}(y_0 = j \mid \boldsymbol{x}_0)$.

See Figures 2.15 and 2.16.

- For $K = 10$, we get good results.
- For $K = 1$ and $K = 100$, we get very bad results.

It is important to choose K well!

When K is too small, the training error rate will be very misleading.

It will be less misleading when K is too big. See Figure 2.17.

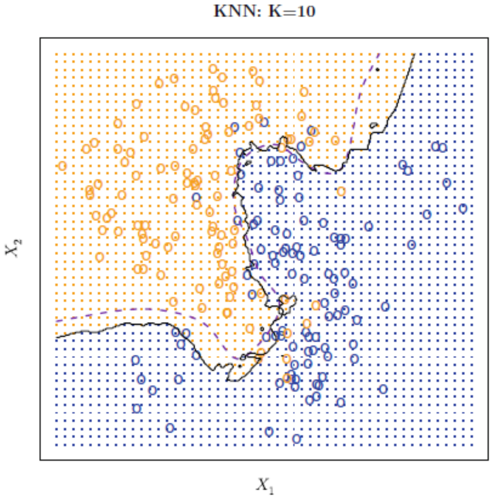

FIGURE 2.15. *The black curve indicates the KNN decision boundary on the data from Figure 2.13, using $K = 10$. The Bayes decision boundary is shown as a purple dashed line. The KNN and Bayes decision boundaries are very similar.*

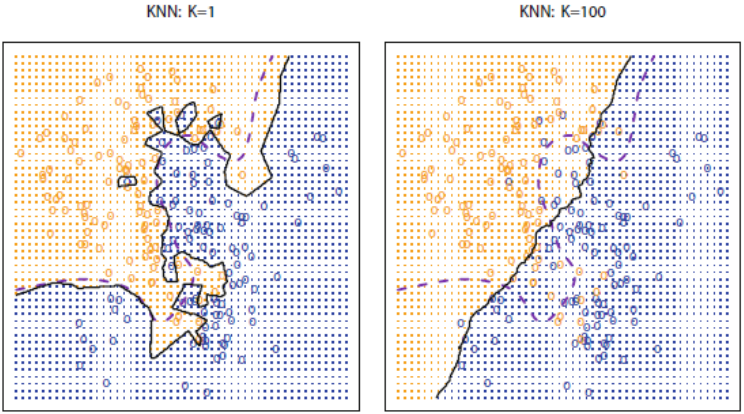

FIGURE 2.16. *A comparison of the KNN decision boundaries (solid black curves) obtained using* $K = 1$ *and* $K = 100$ *on the data from Figure 2.13. With* $K = 1$*, the decision boundary is overly flexible, while with* $K = 100$ *it is not sufficiently flexible. The Bayes decision boundary is shown as a purple dashed line.*

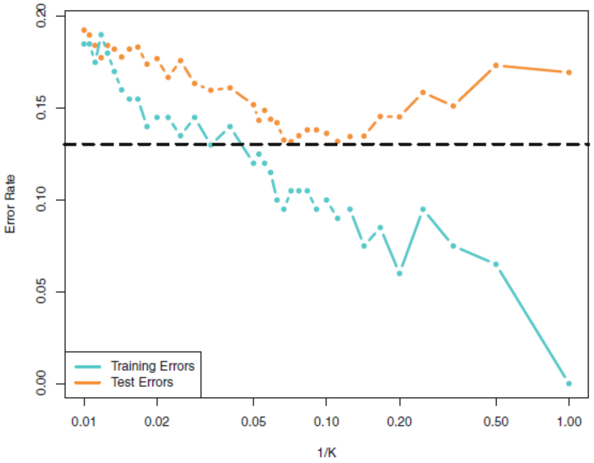

FIGURE 2.17. *The KNN training error rate (blue, 200 observations) and test error rate (orange, 5,000 observations) on the data from Figure 2.13, as the level of flexibility (assessed using $1/K$) increases, or equivalently as the number of neighbors K decreases. The black dashed line indicates the Bayes error rate. The jumpiness of the curves is due to the small size of the training data set.*