Classification Using KNN

In addition to always assigning a class (provided $K/\#class$ is not an integer), KNN gives estimated probabilities between 0 and 1.

- In the binary case, if all the nearest neighbours have $y_i = 0$, $\widehat{\Pr}(y_0 = 1 \,|\, \boldsymbol{x}_0) = 0$.
- If all the nearest neighbours have $y_i = 1$, $\widehat{\Pr}(y_0 = 1 \,|\, \boldsymbol{x}_0) = 1$.
- Unless K is pretty small, usually $0 < \widehat{\Pr}(y_0 = 1 \,|\, \boldsymbol{x}_0) < 1$.

KNN can work well when there are 3 or more classes. It always yields estimated probabilities between 0 and 1 that add to 1.

It may even work fairly well when p is not very small, provided the observations fall into clusters associated with various outcomes.

The key thing is that, for any $\boldsymbol{x}_0$ of interest, there need to be at least K observations "near" $\boldsymbol{x}_0$.

If the K nearest observations are far from $\boldsymbol{x}_0$, it may not work well.

Linear Regression and Classification

When the output is binary, linear regression is called the **linear probability model**, or **LPM**. It is not very satisfactory.

- It may give estimated probabilities less than 0 or greater than 1.
- This tends to happen for observations with extreme values of some elements of $\boldsymbol{x}_0$.
- When all inputs are dummy variables, there cannot be any extreme values of $\boldsymbol{x}_0$.
- If in addition $\bar{y}$ is not close to 0 or 1, all of the $\hat{y}_i$ are likely to be in the 0-1 interval. So LPM may work OK.

The conditional variance of $u_i = y_i - \boldsymbol{X}_i$ is not constant:

$$\mathrm{E}(y_i - \boldsymbol{X}_i\boldsymbol{\beta})^2 = \Pr(y_i = 1)(1 - \boldsymbol{X}_i\boldsymbol{\beta})^2 + \Pr(y_i = 0)(\boldsymbol{X}_i\boldsymbol{\beta})^2. \tag{1}$$

So we must allow for heteroskedasticity (and often clustering).

Why use linear regression at all?

- It is fast and numerically reliable.
- Coefficient estimates often have the same signs and the same relative magnitudes as ones for logit or probit.

Linear regression is completely inappropriate when there are three or more unordered outcomes (classes).

In Section 4.2, ISLR/ISLP gives an example. Patients come into the emergency room with symptoms that could indicate a stroke, an epileptic seizure, or a drug overdose.

If we code these as 1 for stroke, 2 for seizure, and 3 for overdose, we will almost certainly get very different results than if we code them as 1 for overdose, 2 for stroke, and 3 for seizure.

These outcomes are not ordered, so it makes no sense to use any sort of regression model.

Using regression for classification is only viable if the outcomes are ordered (e.g. 1 for healthy, 2 for unwell, 3 for sick, 4 for really sick).

In addition, the “distance” between each outcome must be about the same, which is probably not true for this example.

However, there are methods designed for ordered outcomes that should work better than linear regression.

- Classical methods are **ordered logit** and **ordered probit**. The former generalizes **logistic regression**, or the **logit model**, or **logit regression**, which we are about to discuss.
- In practice, logit and probit models tend to yield very similar predictions and marginal effects.
- This is also true for ordered logit and ordered probit models, but it is not true for models of three or more unordered outcomes.
- Logit models are a little easier to estimate than probit models, so ISLR/ISLP focuses on them.

Logistic Regression

Logistic regression, a.k.a. the logit model, is based on the relationship

$$\log\left(\frac{p(\boldsymbol{x})}{1-p(\boldsymbol{x})}\right) = \boldsymbol{x}^\top\boldsymbol{\beta}. \tag{2}$$

Equation (2) says that the **log of the odds** is linear in the predictors, that is, the elements of the vector $\boldsymbol{x}$.

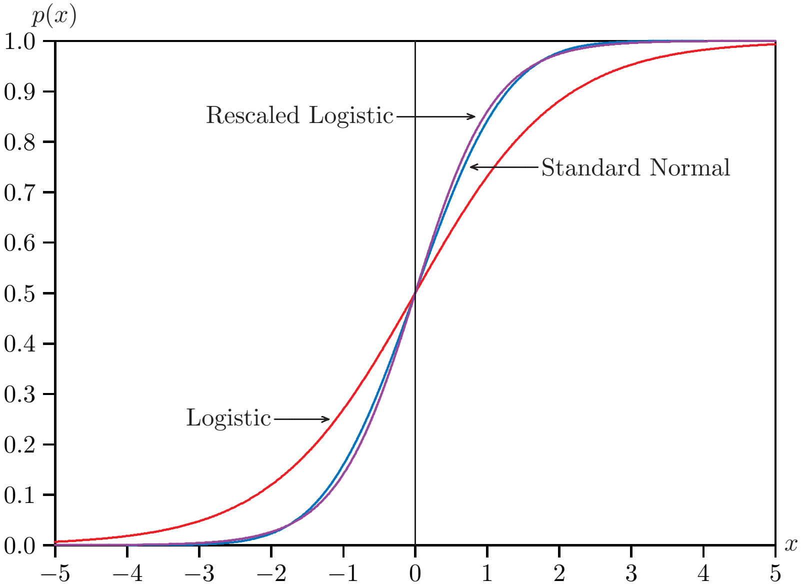
It is easy to solve (2) for $p(\boldsymbol{x})$. We obtain

$$p(\boldsymbol{x}) = \frac{\exp(\boldsymbol{x}^\top\boldsymbol{\beta})}{1+\exp(\boldsymbol{x}^\top\boldsymbol{\beta})} = \frac{1}{1+\exp(-\boldsymbol{x}^\top\boldsymbol{\beta})} = \Lambda(\boldsymbol{x}^\top\boldsymbol{\beta}). \tag{3}$$

The **logistic function** $\Lambda(\cdot)$ has a nice shape. It is monotonically increasing, with $p(-\infty) = 0$, $p(0) = 0.5$, and $P(\infty) = 1$. It looks like a CDF for a symmetric distribution. See Figure 6.1.

For any finite argument, $0 < p(\boldsymbol{x}^\top\boldsymbol{\beta}) < 1$.

Figure 6.1


$p(x)$
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Rescaled Logistic
Standard Normal
Logistic
x
−5
−4
−3
−2
−1
0
1
2
3
4
5

The logit model implies that the derivatives of $p(\boldsymbol{x})$ are not constant:

$$\frac{\partial p(\boldsymbol{x})}{\partial x_j} = \beta_j \Lambda(\boldsymbol{x}^\top \boldsymbol{\beta}) \Lambda(-\boldsymbol{x}^\top \boldsymbol{\beta}). \tag{4}$$

The derivatives are large when $\boldsymbol{x}^\top \boldsymbol{\beta} \approx 0$ and small when $|\boldsymbol{x}^\top \boldsymbol{\beta}|$ is large. $\boldsymbol{x}^\top \boldsymbol{\beta}$ is sometimes called an **index function**.

It is not hard to estimate a logit model. We simply maximize the **loglikelihood function**, which is the log of the joint probability of the observed sample.

If $y_i = 1$, the probability is $\Lambda(\boldsymbol{x}^\top \boldsymbol{\beta})$.

If $y_i = 0$, the probability is $1 - \Lambda(\boldsymbol{x}^\top \boldsymbol{\beta}) = \Lambda(-\boldsymbol{x}^\top \boldsymbol{\beta})$.

Thus the loglikelihood function is

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} \big(y_i \log \Lambda(\boldsymbol{x}_i^\top \boldsymbol{\beta}) + (1 - y_i) \log \Lambda(-\boldsymbol{x}_i^\top \boldsymbol{\beta}) \big). \tag{5}$$

The first term is non-zero for observations with $y_i = 1$, and the second is non-zero for observations with $y_i = 0$.

Another way to write the loglikelihood function is

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} \Big(y_i \, [\boldsymbol{x}_i^\top \boldsymbol{\beta}] - \log \big(1 + \exp(\boldsymbol{x}_i^\top \boldsymbol{\beta})\big) \Big). \tag{6}$$

If there is a constant term β_0, the first-order condition for it is

$$\sum_{i=1}^{n} y_i - \sum_{i=1}^{n} \frac{\exp(\boldsymbol{x}_i^\top \boldsymbol{\beta})}{1 + \exp(\boldsymbol{x}_i^\top \boldsymbol{\beta})} = \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} \Lambda(\boldsymbol{x}_i^\top \boldsymbol{\beta}) = 0. \tag{7}$$

So the sum of the y_i must be equal to the sum of the probabilities that $y = 1$. Thus, at the ML estimates, the expected number of 1s must equal the actual number.

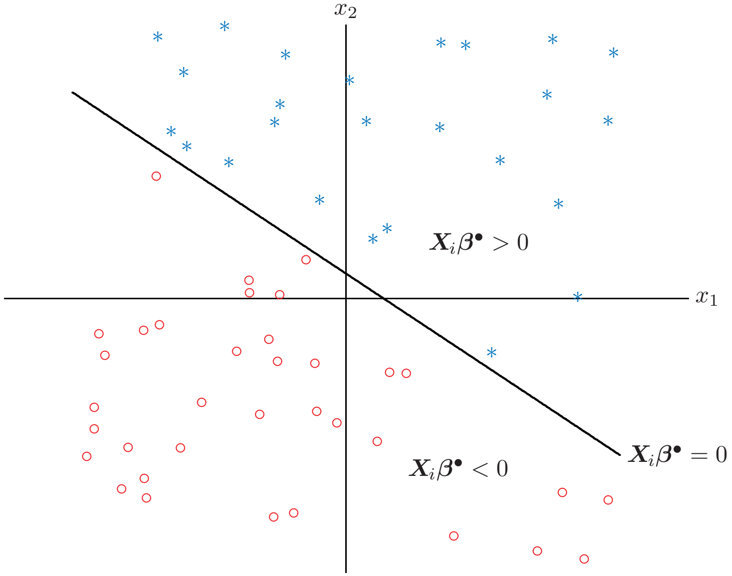
Minus twice the value of $\ell(\hat{\boldsymbol{\beta}})$ is called the **deviance**. Maximizing the loglikelihood is equivalent to minimizing the deviance.

The maximum possible value of $\ell(\boldsymbol{\beta})$ is 0. This occurs when the fitted probabilities are all 0 or 1, and $y_i = p(\boldsymbol{x}_i)$ for all i.

But this can only happen if $||\hat{\boldsymbol{\beta}}|| = \infty$. In such cases, there exists a **perfect classifier**.

- For prediction, a perfect classifier seems great, although perfection on the training sample does not mean perfection on all test samples.
- But for conventional inference about $\boldsymbol{\beta}$, it is terrible.
- When there is a perfect classifier, we know the signs and (approximate) relative magnitudes of the elements of $\boldsymbol{\beta}$, but asymptotic inference does not work.
- A numerical optimization procedure will probably terminate abnormally, perhaps giving an incomprehensible error message.
- A perfect classifier is almost never unique. See Figure 6.2.
- **Regularization** is one way to solve the perfect classifier problem, and other problems as well.

Figure 6.2 (Logit with a separating hyperplane)



An Example of Cross-Validation

In Chapter 5, ISLR/ISLP provides a simulated example to illustrate how cross-validation works.

The data come from Figure 2.13 and involve 200 observations, 100 from each of two classes. There are two inputs.

The DGP is highly nonlinear, leading to the very nonlinear Bayes decision boundary given by the purple dashed line.

The books estimate linear, quadratic, cubic, and quartic logistic regression models, none of which is the true model.

None of these does a great job; see Figure 5.7.

The test error rate is lowest for the cubic model, although it is only a little smaller (at 0.160) than the one for the quartic model (at 0.162).

The linear logit model has a test error rate of 0.201, and the Bayes error rate is 0.133.

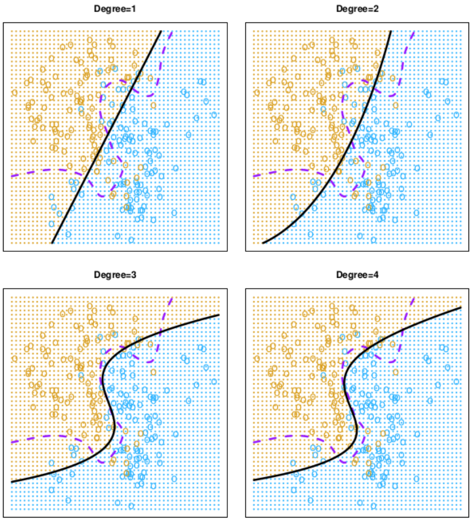


FIGURE 5.7. *Logistic regression fits on the two-dimensional classification data displayed in Figure 2.13. The Bayes decision boundary is represented using a purple dashed line. Estimated decision boundaries from linear, quadratic, cubic and quartic (degrees 1–4) logistic regressions are displayed in black. The test error rates for the four logistic regression fits are respectively* 0.201, 0.197, 0.160, *and* 0.162, *while the Bayes error rate is* 0.133.

The left panel of Figure 5.8 shows training, test, and CV errors as a function of d, the order of the polynomial.

Oddly, the training error rate actually increases as d goes from 4 to 5 to 6, before falling again.

As we saw, the test error is minimized at $d = 3$, but it rises gently for a while after that.

When 10-fold cross-validation is used, the CV error is minimized at $d = 4$.

Figure 5.8 also shows training, test, and CV errors for KNN (apparently based on 10-fold CV not LOOCV).

The CV error is minimized at $K = 8$ (I think) and the test error at $K = 11$.

The minimum test error for KNN is about 0.135, which is noticeably lower than 0.160 for logistic regression with cubic polynomials.

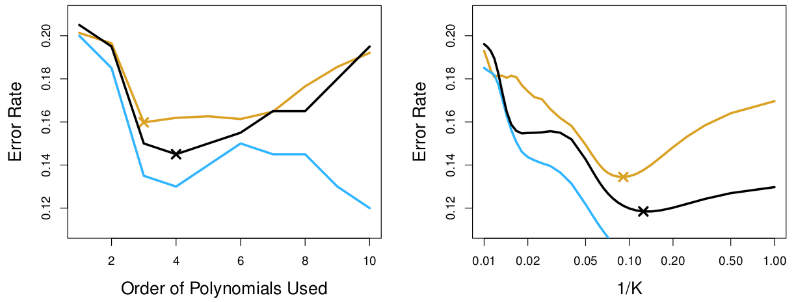


FIGURE 5.8. *Test error (brown), training error (blue), and 10-fold CV error (black) on the two-dimensional classification data displayed in Figure 5.7.* Left: *Logistic regression using polynomial functions of the predictors. The order of the polynomials used is displayed on the x-axis.* Right: *The KNN classifier with different values of K, the number of neighbors used in the KNN classifier.*

Other Binary Classification Methods

The **probit model** is about as popular as the logit model. For it,

$$p(\boldsymbol{x}) = \Phi(\boldsymbol{x}^\top \boldsymbol{\beta}), \tag{8}$$

where $\Phi(\cdot)$ is the cumulative normal distribution function.

Probit and logit models generally yield extremely similar predictions.

Linear discriminant analysis and **quadratic discriminant analysis** use information about the distributions of the $\boldsymbol{x}_i$ conditional on the y_i.

Both assume the $\boldsymbol{x}_i$ are multivariate normal. For LDA, the covariance matrices (but not the means) are the same for all classes. For QDA, both means and covariances are allowed to differ.

LDA leads to linear decision boundaries, QDA to nonlinear ones.

A related approach is **naive Bayes classification**. It drops normality but assumes that the elements of the $\boldsymbol{x}_i$ are independently distributed.