Lecture 3: Elementary methods for classification

Alan Lee

Department of Statistics
STATS 769 Lecture 3

July 28, 2015

Outline

Today's agenda

In this lecture we discuss linear methods for classification, (although many of the concepts covered will be applicable to other methods). Some of the ideas should be familiar from other courses, particularly STATS 330/762. Today we will cover

- Classification: general principles;
- Bayes classifier;
- Logistic regression;
- Cross-validation and bootstrap estimates of classification error;
- K-nearest neighbour classifier;

We will use the credit card default data from *Introduction to Statistical Learning* as a running example.

Classification

In today's lecture we consider the case when our output is a categorical variable, indicating which one of a number of classes the observation belongs to e.g. alive/dead, win/draw/loss, 1-5 in a Likert scale in a questionnaire.

Given we have observed inputs $x_1, \ldots, x_k$, how should we classify an observation? For example, given a person's age, can we predict if they suffer from arthritis?

Bayes classifier

Suppose for each age x, we know the conditional probabilities

$$\text{Prob}[\text{ has arthritis } | \text{age } = x],$$

$$\text{Prob}[\text{ does not have arthritis } | \text{age } = x].$$

Then the best classifier (one that minimises the probability of misclassification) is the one that assigns to a individual aged x the class with the highest conditional probability. (In the case of just two classes, as is the case here, this is the same as classifying an individual as having arthritis if

$$\text{Prob}[\text{ has arthritis}|\text{age } = x] > 0.5.$$

Estimating conditional probabilities

The problem with this approach is that we don't usually know these conditional probabilities. However, if we could **estimate** them, we could use the Bayes classifier with the estimated probabilities.

We will look at two ways of doing this: using logistic regression, and K-nearest neighbours.

Logistic regression

We first discuss the case where there are just two categories, which we label 0 and 1. We denote the output as Y, so either $Y = 0$ or $Y = 1$, and the inputs as x. In logistic regression, we model the probabilities as

$$\text{Prob}[Y = 1|x] = \frac{\exp\{b_0 + b_1x_1 + \cdots b_kx_k\}}{1 + \exp\{b_0 + b_1x_1 + \cdots b_1x_k\}},$$

$$\text{Prob}[Y = 0|x] = \frac{1}{1 + \exp\{b_0 + b_1x_1 + \cdots b_1x_1\}},$$

where we have to choose the values of b_0, b_1, b_k, based on a training set.

Odds & log-odds form

Odds form:

$$\frac{\text{Prob}}{1 - \text{Prob}} = \exp(b_0 + b_1 x_1 + \cdots + b_k x_k)$$

Log-odds form:

$$\log \frac{\text{Prob}}{1 - \text{Prob}} = b_0 + b_1 x_1 + \cdots + b_k x_k.$$

Choosing the b's

The b's are chosen by the method of maximum likelihood - this is a statistical technique based on assumptions on how the data was generated. See STATS 330 for more detail. In R, this is done using the function `glm`. See the example below.

Prediction error

In classification problems, the prediction error is measured by the probability of misclassification. This can be estimated from the training set by the proportion of misclassified observations, or from the test set in the same way. We illustrate with an example, from *An Introduction to Statistical Learning*. In the package `ISLR` there is a data set `Defaults` containing data on 10,000 monthly credit card bills. The variables are

income	:	annual income of card holder;
balance	:	the monthly balance;
student	:	student status Yes=student, No=Not a student;
default	:	defaulted on payment? Yes/No

Note: The variable student is categorical, so the effect of being a student is to increase the predicted log-odds by the amount of the student coefficient.

Calculating the *b*'s

```
> library(ISLR)
> data(Default)
> default.logistic = glm(default~ student+balance+income,
+       data=Default, family=binomial)
> coefficients(default.logistic)
  (Intercept)    studentYes       balance        income
-1.086905e+01 -6.467758e-01  5.736505e-03  3.033450e-06
```

Note: The effect of being a student is to decrease the predicted log-odds by 6.467758e-01.

Predicting

To make the predictions, we use the generic `predict` function to calculate the estimated conditional probabilities, and then make a table of predictions versus actuals. We predict a default if the estimated probability is greater than 0.5:

```
> probs = predict(default.logistic, type="response")
> my.table = table(Default$default, probs>0.5)
> my.table
    FALSE TRUE
No   9627   40
Yes   228  105
```

The estimated error is the proportion of individuals on the off-diagonals:

```
> 1-sum(diag(my.table))/sum(my.table)
[1] 0.0268
```

Sensitivity and specificity

The predictor does well when predicting the non-defaults, but poorly when predicting the defaults. We can distinguish between these:

- Sensitivity: probability of predicting a 1 when the case is truly a 1: the "true positive rate"
- Specificity: probability of predicting a 0 when the case is truly a 0: the "true negative rate" (1-specificity is called the "false positive rate")
- Ideally, want both to be close to 1
- Thus, the sensitivity is the probability of correctly identifying a default - i.e. $105/(228+105) = 0.315$, the specificity is $9627/(9627+40)=0.996$.

Sensitivity and specificity

We can trade off the specificity against the sensitivity by moving away from the Bayes rule. For example, if we decide to classify a customer as a default if the probability of default is more than 0.1 (instead of 0.5) we would get

```
> my.table = table(Default$default, probs>0.1)
my.table
      FALSE TRUE
  No   9107  560
  Yes    85  248
> 9107/(9107+560)
[1] 0.942071
> 248/(85+248)
[1] 0.7447447
```

so the sensitivity has gone down a bit (from 0.996 to 0.942) but the specificity has improved from 0.315 to 0.745. The overall prediction error has increased (from 0.0268 to 0.0645).

Bootstrapping and cross-validation

This works exactly as before, except that instead of averaging the squared errors, we average the quantity

$$L(y, \hat{y}) = \begin{cases} 1, & y \neq \hat{y}, \\ 0, & y = \hat{y}, \end{cases} .$$

where y is the actual category and $\hat{y}$ the predicted one. (For numerical outputs we use $L(y, \hat{y}) = (y - \hat{y})^2$)

Example: the default data

```
> library(R330)
> cross.val(default.logistic,nfold=10)
Mean Specificity =  0.9958682
Mean Sensitivity =  0.3174139
Mean Correctly classified =  0.973215
> 1-0.973215
[1] 0.026785

err.boot(default.logistic,B=100)
$err
[1] 0.0268

$Err
[1] 0.026654
```

Example: More than two output categories

Suppose that there are J output categories, $C_1, C_2, \ldots, C_J$. We estimate the conditional probabilities by

$$\text{Prob}[Y \text{ in } C_j|x] = \frac{\exp\{b_{j0} + b_{j1}x_1 + \cdots b_{jk}x_k\}}{1 + \sum_{j=1}^{J-1} \exp\{b_{j0} + b_{j1}x_1 + \cdots b_{jk}x_k\}}$$

for $j = 1, 2, \ldots, (J-1)$ and

$$\text{Prob}[Y \text{ in } C_J|x] = \frac{1}{1 + \sum_{j=1}^{J-1} \exp\{b_{j0} + b_{j1}x_1 + \cdots b_{jk}x_k\}}$$

The coefficients b_{jk} are found using the `nnet` function - see later.

K-nearest neighbours

Here we estimate

$$\text{Prob}[Y \text{ in } C_j | x]$$

by the proportion of the K data points closest to x that are in category C_j.

To define "closest" we need a definition of distance - in the case of numeric variables we use Euclidean distance, in the case of categorical inputs we can use dummy variables - see STATS 330 for details.

We should standardise the variables to prevent one variable dominating the distance.

Example

```
> input.df = Default[,-1]
> # change the varible "student" to a dummy variable
> input.df[,1] = as.numeric(input.df[,1]) - 1
> # scale the inputs
> input.df=scale(input.df)
> predictions = knn(input.df, input.df, Default$default, k=5)
> knn.table = table(Default$default,predictions)
> knn.table
     predictions
        No  Yes
  No  9614   53
  Yes  199  134
> 1-sum(diag(knn.table))/sum(knn.table)
[1] 0.0252
```