Model Validation Techniques

Cross Validation and Bootstrapping

Thursday March 7, 2024

```r
library(alr4)
```

```
## Loading required package: car

## Loading required package: carData

## Loading required package: effects

## lattice theme set by effectsTheme()
## See ?effectsTheme for details.
```

```r
library(MASS)
```

Model assessment with cross-validation

Consider a simulated dataset with a binary response `y` and 5000 predictors measured on $n = 50$ cases, saved as a matrix `x`.

```r
set.seed(1)
y <- c(rep("c1", 25), rep("c2", 25))
x <- matrix(NA, nrow = 50, ncol = 5000)
for (i in 1:5000) {
    x[, i] <- rnorm(50)
}
```

A simple classifier is applied to the simulated dataset, where the classification is performed in two steps:

- Step 1. Feature selection: Select 20 predictors with smallest p-values from two-sample t-tests
- Step 2. Model fitting: Fit a linear discriminant analysis (LDA) model, using only these 20 selected predictors.

We would like to compute the 5-fold cross-validation (CV) estimate of test accuracy rate for the classifier.

1a. To assess the performance of the classifier, one may begin by selecting 20 predictors using the entire dataset and construct a reduced dataset `selected`:

```r
pval <- vector("double", 5000)
for (i in 1:5000) {
    pval[i] <- t.test(x[, i] ~ y)$p.value
}
selected <- data.frame(x[, order(pval)[1:20]])
selected$y <- factor(y)
str(selected)
```

```
## 'data.frame':    50 obs. of  21 variables:
##  $ X1 : num  1.409 0.6 0.196 -0.806 -1.204 ...
##  $ X2 : num  -0.931 -1.49 0.531 0.456 -0.187 ...
##  $ X3 : num  -1.281 1.098 0.954 1.084 0.744 ...
##  $ X4 : num  0.2866 -0.8162 0.3976 -2.5967 0.0561 ...
##  $ X5 : num  1.2054 0.0271 0.0526 0.5043 0.4723 ...
##  $ X6 : num  1.5034 0.405 -0.0691 1.2264 -0.7358 ...
##  $ X7 : num  0.863 -1.2 -2.259 -1.344 -0.209 ...
##  $ X8 : num  0.899 1.066 0.801 0.413 0.255 ...
##  $ X9 : num  0.836 -1.865 -1.427 -0.531 -1.05 ...
##  $ X10: num  0.6107 -0.4738 0.6172 -0.0752 2.0126 ...
##  $ X11: num  -0.418 -0.737 1.149 0.262 -0.89 ...
##  $ X12: num  0.691 0.268 0.146 -0.389 1.719 ...
##  $ X13: num  -0.118 1.256 1.257 1.011 -0.166 ...
##  $ X14: num  1.458 -0.0668 0.1484 0.2871 -0.1924 ...
##  $ X15: num  0.8211 1.3305 0.0962 -0.7422 1.4135 ...
##  $ X16: num  0.433 0.927 2.237 -0.455 -1.039 ...
##  $ X17: num  -0.761 -0.388 -2.351 -0.765 0.462 ...
##  $ X18: num  1.6874 0.4309 0.7248 0.0186 0.9178 ...
##  $ X19: num  0.592 -1.01 -0.546 -1.033 0.272 ...
##  $ X20: num  -0.9743 -0.84 -0.9997 0.0734 -0.2099 ...
##  $ y  : Factor w/ 2 levels "c1","c2": 1 1 1 1 1 1 1 1 1 1 ...
```

Then, based on the `selected` dataset, the 5-fold CV estimate of accuracy rate can be obtained with the following procedure:

1. Partition the `selected` dataset into 5 folds.
2. Each fold is used to calculate the accuracy rate of the LDA model trained using the observations in the remaining 4 folds.
3. Average the 5 accuracy rates.

Implement this procedure to compute the 5-fold CV estimate of accuracy rate. Comment on the result.

Ans:

```r
library(MASS)  # Load the MASS package for LDA

# Function to calculate accuracy rate
accuracy <- function(predicted, actual) {
  mean(predicted == actual)
}

# Function to perform 5-fold cross-validation for test accuracy
cross_validation_test <- function(data, folds) {
  t_accuracies <- numeric(length = folds)

  # Shuffle the data
  data <- data[sample(nrow(data)), ]

  # Create folds
  fold_indices <- cut(seq(1, nrow(data)), breaks = folds, labels = FALSE)

  for (i in 1:folds) {
    # Split data into train and test sets
    test_indices <- which(fold_indices == i)
    test_data <- data[test_indices, ]
    train_data <- data[-test_indices, ]

    # Fit LDA model using train data
    lda_model <- lda(y ~ ., data = train_data)

    # Predict classes on test data
    predicted <- predict(lda_model, newdata = test_data)$class

    # Calculate accuracy rate
    t_accuracies[i] <- accuracy(predicted, test_data$y)
  }

  # Return average accuracy rate
  mean(t_accuracies)
}

# Compute 5-fold CV estimate of test accuracy rate
set.seed(1)  # Set seed for reproducibility
cv_test_accuracy <- cross_validation_test(selected, 5)
```

```r
# Print the result
print(paste("The 5-fold CV estimate of accuracy rate is ", cv_test_accuracy))
```

```
## [1] "The 5-fold CV estimate of accuracy rate is  0.88"
```

Inference: This accuracy is quite high as we are performing feature selection first and then selecting the model. This might include noise.

1b. Compute the 5-fold CV estimate of test accuracy rate for the classifier.

Ans:

```r
library(class)
library(rsample)
library(dslabs)
library(ggplot2)
library(class)
library(MASS)
library(e1071)
```

```
##
## Attaching package: 'e1071'

## The following object is masked from 'package:rsample':
##
##     permutations
```

```r
data_df = as.data.frame(x)
data_df$y = y

set.seed(1)

folds = vfold_cv(data_df, v = 5)
accuracy_rates = vector("numeric", length(folds))
for (i in 1:5) {
  train_data = analysis(folds$splits[[i]])
  val_data = assessment(folds$splits[[i]])

  pvals = vector("double", ncol(train_data) - 1)
  for (j in 1:(ncol(train_data) - 1)) {
    pvals[j] = t.test(train_data[, j] ~ train_data$y)$p.value
  }

  selected_indices = order(pvals)[1:20]
  selected_data = train_data[, c(selected_indices, ncol(train_data))]

  lda_model = lda(y ~ ., data = selected_data)

  predicted = predict(lda_model, newdata = val_data)$class
  accuracy_rates[i] = mean(predicted == val_data$y)
}

avg_accuracy = mean(accuracy_rates)
print(paste("The test accuracy rate for the classifier is:" , avg_accuracy))
```

```
## [1] "The test accuracy rate for the classifier is: 0.42"
```

This code modifies the fold_accuracy function. Within each fold, it performs feature selection using t-tests on the current fold's data. Then, it selects the top 20 features and trains the LDA model on the reduced training data within the fold. The rest of the code for partitioning and calculating CV accuracy remains the same.

This approach provides a more realistic estimate of the classifier's performance, considering both feature selection and model fitting within each fold of the cross-validation process.

The bootstrap

The `alr4::Rateprof` dataset consists professor ratings from the website `RateMyProfessor.com` includes averages of many student ratings for each instructor on five different measures, including `quality`, `helpfulness`, `clarity`, `easiness` of the course, and `raterInterest` in the subject matter. All the ratings were on a five-point scale, so the averages are numbers between 1 and 5.

```r
str(Rateprof)
```

```
## 'data.frame':    366 obs. of  17 variables:
##  $ gender        : Factor w/ 2 levels "female","male": 2 2 2 2 2 2 2 2 2 2 ...
##  $ numYears      : int  7 6 10 11 11 10 7 11 11 7 ...
##  $ numRaters     : int  11 11 43 24 19 15 17 16 12 18 ...
##  $ numCourses    : int  5 5 2 5 7 9 3 3 4 4 ...
##  $ pepper        : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ discipline    : Factor w/ 4 levels "Hum","SocSci",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ dept          : Factor w/ 48 levels "Accounting","Anthropology",..: 17 42 3 17 45 45 45 1
##  $ quality       : num  4.64 4.32 4.79 4.25 4.68 ...
##  $ helpfulness   : num  4.64 4.55 4.72 4.46 4.68 ...
##  $ clarity       : num  4.64 4.09 4.86 4.04 4.68 ...
##  $ easiness      : num  4.82 4.36 4.6 2.79 4.47 ...
##  $ raterInterest : num  3.55 4 3.43 3.18 4.21 ...
##  $ sdQuality     : num  0.552 0.902 0.453 0.933 0.65 ...
##  $ sdHelpfulness  : num  0.674 0.934 0.666 0.932 0.82 ...
##  $ sdClarity      : num  0.505 0.944 0.413 0.999 0.582 ...
##  $ sdEasiness     : num  0.405 0.505 0.541 0.588 0.612 ...
##  $ sdRaterInterest: num  1.128 1.074 1.237 1.332 0.975 ...
```

Consider a subset of `Rateprof`, named `rating` and obtained as follows.

```r
set.seed(1)
rating <- Rateprof[
    sample(1:nrow(Rateprof), size = 30, replace = FALSE),
    c("quality", "helpfulness", "clarity", "easiness", "raterInterest")
]
str(rating)
```

```
## 'data.frame':    30 obs. of  5 variables:
##  $ quality      : num  4.18 4.68 3.2 2.15 3.31 ...
##  $ helpfulness  : num  4.07 4.89 3.17 2.35 3.46 ...
##  $ clarity      : num  4.29 4.47 3.23 1.96 3.15 ...
##  $ easiness     : num  3.14 4.21 3.03 2.12 2.77 ...
##  $ raterInterest: num  4 4.37 3.66 2.88 3.69 ...
```

Based on `rating`, the correlation matrix and its eigenvalues are

```r
(cc <- cor(rating))
```

```
##                 quality helpfulness   clarity  easiness raterInterest
## quality       1.0000000   0.9839995 0.9851903 0.7051664     0.7224085
## helpfulness   0.9839995   1.0000000 0.9389001 0.7472081     0.7195456
## clarity       0.9851903   0.9389001 1.0000000 0.6438475     0.7030109
## easiness      0.7051664   0.7472081 0.6438475 1.0000000     0.5137544
## raterInterest 0.7224085   0.7195456 0.7030109 0.5137544     1.0000000
```

```r
(ev <- eigen(cc)$value)
```

```
## [1] 4.099049e+00 4.925268e-01 3.597779e-01 4.863861e-02 7.828109e-06
```

The "eigenratio" statistic

$$\hat{\theta} = \text{largest eigenvalue/sum of eigenvalues}$$

measures how closely the 5 measures can be predicted by a single linear combination.

```r
max(ev) / sum(ev)
```

```
## [1] 0.8198098
```

We would like to quantify the uncertainty associated with the estimate $\hat{\theta}$.

2a. Using the bootstrap, estimate the standard error of $\hat{\theta}$.

Ans:

```r
library(boot)
```

```
## Warning: package 'boot' was built under R version 4.3.3

##
## Attaching package: 'boot'

## The following object is masked from 'package:car':
##
##     logit
```

```r
theta_hat <- function(data, indices) {
  sampled_data <- data[indices, ]
  ev <- eigen(cor(sampled_data))$values
  max_ev <- max(ev)
  sum_ev <- sum(ev)
  theta_hat <- max_ev / sum_ev
  return(theta_hat)
}

bootstrap_se <- function(data, B = 10000) {
  boot_results <- boot(data, theta_hat, R = B)
  se <- sd(boot_results$t)
  return(se)
}

theta_se <- bootstrap_se(rating)

print(paste("Standard Error of theta_hat:", theta_se))
```

```
## [1] "Standard Error of theta_hat: 0.0387224602224526"
```

2b. Find a 95% confidence interval with the bootstrap. **Hint:** Use the `quantile()` function.

Ans:

```r
# Bootstrap function to compute confidence interval
bootstrap_ci <- function(data, B = 1000, alpha = 0.05) {
  theta_hat_vals <- replicate(B, theta_hat(data[sample(nrow(data), replace = TRUE), ]))
  ci <- quantile(theta_hat_vals, c(alpha/2, 1 - alpha/2))
  return(ci)
}

# Compute 95% confidence interval for theta_hat using bootstrap
ci <- bootstrap_ci(rating)
print(paste("The 95% confidence interval for theta hat is:"))
```

```
## [1] "The 95% confidence interval for theta hat is:"
```

```r
ci
```

```
##      2.5%     97.5%
## 0.7406040 0.8883542
```