HW6

Sandra Villamar and Shobhit Dronamraju

2023-02-28

Problem 1A

Write a function cv.lm(x, y, k) which estimates the prediction error of the linear regression model with y as response using k-fold cross-validation.

```r
cv.lm <- function(x, y, k){

  if(is.null(x)){
    dat = data.frame(y = y)
  } else{
    dat = data.frame(x, y = y)
  }

  n = nrow(dat)
  dat_cv = dat[sample(n),]  # shuffle observations
  if(is.null(x)){ dat_cv = data.frame(y = dat_cv) }
  folds <- cut(seq(1,n),breaks=k,labels=FALSE)  # cut into k folds

  cv_err = rep(NA, k)

  for (i in 1:k) {

    dat_train = dat_cv[folds != i,]
    dat_val = dat_cv[folds == i,]

    if(is.null(x)){
      dat_train = data.frame(y = dat_train)
      dat_val = data.frame(y = dat_val)
      m_train = lm(y ~ 1, data = dat_train)
    } else{
      m_train = lm(y ~ ., data = dat_train)
    }

    pred_val = predict(m_train, newdata = dat_val)
    res_val = dat_val$y - pred_val
    cv_err[i] = sqrt(mean(res_val^2))
  }

  return(mean(cv_err))
}
```

```r
# testing
library(MASS)
data("Boston")

cv.lm(Boston$crim, Boston$medv, 5)
```

```
## [1] 8.440028
```

Problem 1B

Write a function SequentialSelection(x, y, method) which computes the forward selection path for linear regression from 'intercept only' to 'full model' and chooses the model on that path using different criteria specified by method. The function should support these methods:

- method = "AdjR2": Sequentially include the columns of x and choose the model that gives the largest adjusted R2
- method = "AIC": Sequentially include the columns of x and choose the model that gives the smallest AIC.
- method = "CV5": Sequentially include the columns of x and choose the model that gives the smallest 5-fold cross-validation prediction error.

(Instructions were unclear so provided both methods)

```r
# Via Forward Selection

ForwardSelection <- function(x, y, method) {

  x = data.frame(x)
  p <- ncol(x)  # num of features
  n <- nrow(x)  # num of observations

  # null model i.e. best model so far
  model = lm(y ~ 1)
  y_pred <- predict(model)  # prediction
  RSS <- sum((y - y_pred) ^ 2)  # residual sum of squares
  TSS <- sum((y - mean(y)) ^ 2)  # total sum of squares
  k = 0  # number of parameters in the model
  best_adjusted_R2 <- 1 - (RSS / (n - k)) / (TSS / (n - 1))
  best_AIC <- n * log(RSS / n) + 2 * k
  best_cv5 <- cv.lm(NULL, y, 5)

  features <- 1:p  # feature indices
  selected_features <- c()  # selected feature indices

  # Loop over the features
  for (i in 1:p) {

    # store for candidate models:
    candidates <- setdiff(features, selected_features)  # features not chosen yet
    candidate_R2 <- c()  # adjusted R2 values
    candidate_AIC <- c()  # AIC values
    candidate_cv5 <- c()  # cross-validation errors
```

```r
  # Loop over the candidate features
  for (j in candidates) {

    # Select the current feature and the previously selected features
    current_features <- c(selected_features, j)
    model <- lm(y ~ ., data = data.frame(x[, current_features]))
    y_pred <- predict(model)  # prediction

    RSS <- sum((y - y_pred) ^ 2)  # residual sum of squares
    TSS <- sum((y - mean(y)) ^ 2)  # total sum of squares
    k <- length(current_features)  # number of parameters in the model

    # Compute the adjusted R2 value
    R2 <- 1 - (RSS / (n - k)) / (TSS / (n - 1))
    candidate_R2 <- c(candidate_R2, R2)

    # Compute the AIC value
    AIC <- n * log(RSS / n) + 2 * k
    candidate_AIC <- c(candidate_AIC, AIC)

    # Compute the CV error
    cv5 <- cv.lm(x[, current_features], y, 5)
    candidate_cv5 <- c(candidate_cv5, cv5)
  }

  # Choose the best feature based on the specified method &
  # compare to previous model
  if (method == 'AdjR2') {
    if (max(candidate_R2) > best_adjusted_R2) {
      best_adjusted_R2 <- max(candidate_R2)
      best_feature_index <- candidates[which.max(candidate_R2)]
    } else {
      # if didn't improve, return previous model
      return(printFormula(selected_features))
    }
  }
  else if (method == 'AIC') {
    if (min(candidate_AIC) < best_AIC) {
      best_AIC <- min(candidate_AIC)
      best_feature_index <- candidates[which.min(candidate_AIC)]
    } else {
      return(printFormula(selected_features))
    }
  }
  else if (method == 'CV5') {
    if (min(candidate_cv5) < best_cv5) {
      best_cv5 <- min(candidate_cv5)
      best_feature_index <- candidates[which.min(candidate_cv5)]
    } else {
      return(printFormula(selected_features))
    }
  }
  else {
```

```r
      stop("Invalid method specified")
    }

    # Add the best feature to the list of selected features
    selected_features <- c(selected_features, best_feature_index)
  }

  return(printFormula(selected_features))
}

printFormula <- function(selected_features){
  if (length(selected_features) == 0) {
    formula = "y ~ 1"
  }
  else {
    formula = "y ~"
    for (f in selected_features) {
      formula = paste(formula, sprintf("x%i +", f))
    }
    formula = substr(formula, 1, nchar(formula) - 2)  # remove last " + "
  }

  return(formula)
}
```

```r
# testing
ForwardSelection(Boston[1:13], Boston$medv, "AdjR2")
```

```
## [1] "y ~ x13 + x6 + x11 + x8 + x5 + x4 + x12 + x2 + x1 + x9 + x10"
```

```r
ForwardSelection(Boston[1:13], Boston$medv, "AIC")
```

```
## [1] "y ~ x13 + x6 + x11 + x8 + x5 + x4 + x12 + x2 + x1 + x9 + x10"
```

```r
ForwardSelection(Boston[1:13], Boston$medv, "CV5")
```

```
## [1] "y ~ x13 + x6 + x11 + x4 + x12 + x7"
```

```r
# Via Sequential Columns

SequentialSelection <- function(x, y, method) {

  x = data.frame(x)
  p <- ncol(x)  # num of features
  n <- nrow(x)  # num of observations

  # null model i.e. best model so far
  model = lm(y ~ 1)
  y_pred <- predict(model)  # prediction
  RSS <- sum((y - y_pred) ^ 2)  # residual sum of squares
  TSS <- sum((y - mean(y)) ^ 2)  # total sum of squares
```

```r
  k = 0  # number of parameters in the model
  best_adjusted_R2 <- 1 - (RSS / (n - k)) / (TSS / (n - 1))
  best_AIC <- n * log(RSS / n) + 2 * k
  best_cv5 <- cv.lm(NULL, y, 5)
  best_model = model

  features <- 1:p  # feature indices
  selected_features <- c()  # selected feature indices

  # Loop over the features
  for (i in 1:p) {

    current_features = 1:i

    model <- lm(y ~ ., data = data.frame(x[, current_features]))
    y_pred <- predict(model)  # prediction

    RSS <- sum((y - y_pred) ^ 2)  # residual sum of squares
    TSS <- sum((y - mean(y)) ^ 2)  # total sum of squares
    k <- length(current_features)  # number of parameters in the model

    # Compute the adjusted R2 value, AIC, and CV error
    R2 <- 1 - (RSS / (n - k)) / (TSS / (n - 1))
    AIC <- n * log(RSS / n) + 2 * k
    cv5 <- cv.lm(x[, current_features], y, 5)

    # Compare model to previous best model
    if (method == 'AdjR2') {
      if (R2 > best_adjusted_R2) {
        best_adjusted_R2 <- R2
        best_model <- model
      }
    }
    else if (method == 'AIC') {
      if (AIC < best_AIC) {
        best_AIC <- AIC
        best_model <- model
      }
    }
    else if (method == 'CV5') {
      if (cv5 < best_cv5) {
        best_cv5 <- cv5
        best_model <- model
      }
    }
    else {
      stop("Invalid method specified")
    }
  }

  return(best_model)
}
```

```r
# testing
SequentialSelection(Boston[1:13], Boston$medv, "AdjR2")
```

```
##
## Call:
## lm(formula = y ~ ., data = data.frame(x[, current_features]))
##
## Coefficients:
## (Intercept)         crim           zn        indus         chas          nox
##   3.646e+01   -1.080e-01    4.642e-02    2.056e-02    2.687e+00   -1.777e+01
##          rm          age          dis          rad          tax      ptratio
##   3.810e+00    6.922e-04   -1.476e+00    3.060e-01   -1.233e-02   -9.527e-01
##       black        lstat
##   9.312e-03   -5.248e-01
```

```r
SequentialSelection(Boston[1:13], Boston$medv, "AIC")
```

```
##
## Call:
## lm(formula = y ~ ., data = data.frame(x[, current_features]))
##
## Coefficients:
## (Intercept)         crim           zn        indus         chas          nox
##   3.646e+01   -1.080e-01    4.642e-02    2.056e-02    2.687e+00   -1.777e+01
##          rm          age          dis          rad          tax      ptratio
##   3.810e+00    6.922e-04   -1.476e+00    3.060e-01   -1.233e-02   -9.527e-01
##       black        lstat
##   9.312e-03   -5.248e-01
```

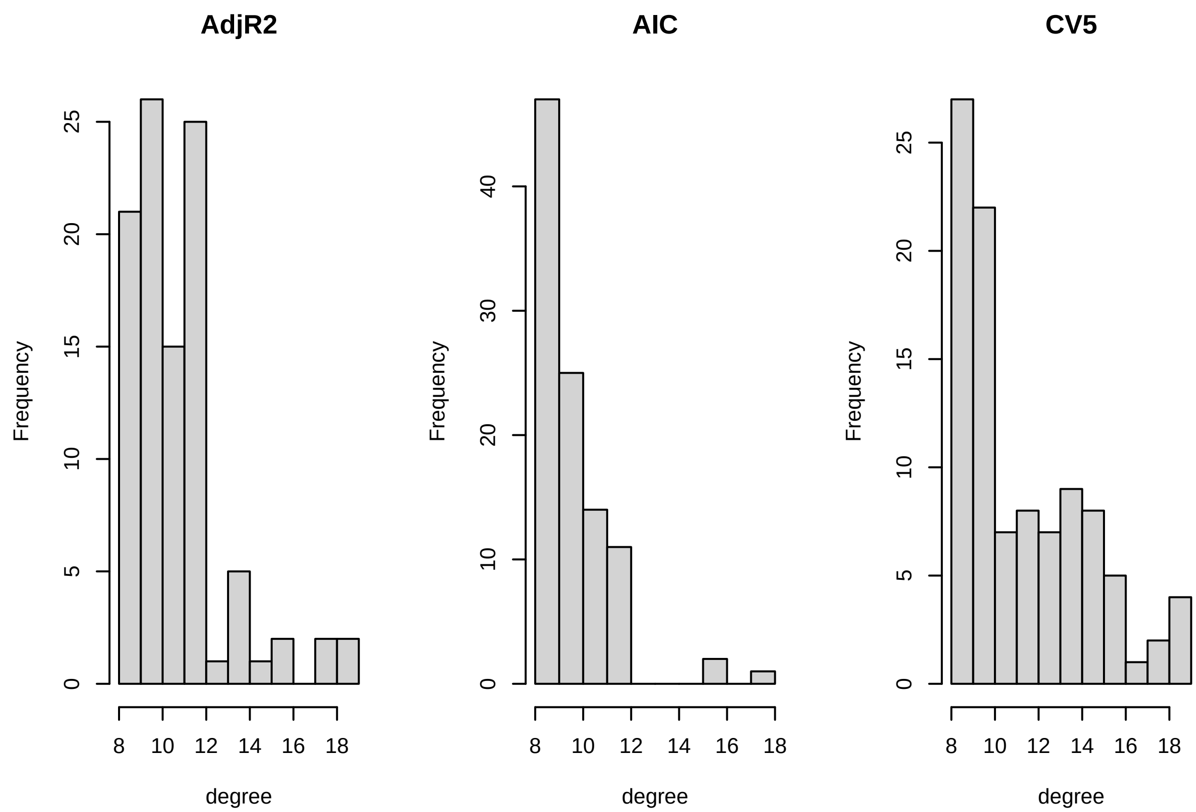
```r
SequentialSelection(Boston[1:13], Boston$medv, "CV5")
```

```
##
## Call:
## lm(formula = y ~ ., data = data.frame(x[, current_features]))
##
## Coefficients:
## (Intercept)         crim           zn        indus         chas          nox
##   3.646e+01   -1.080e-01    4.642e-02    2.056e-02    2.687e+00   -1.777e+01
##          rm          age          dis          rad          tax      ptratio
##   3.810e+00    6.922e-04   -1.476e+00    3.060e-01   -1.233e-02   -9.527e-01
##       black        lstat
##   9.312e-03   -5.248e-01
```

Problem 2

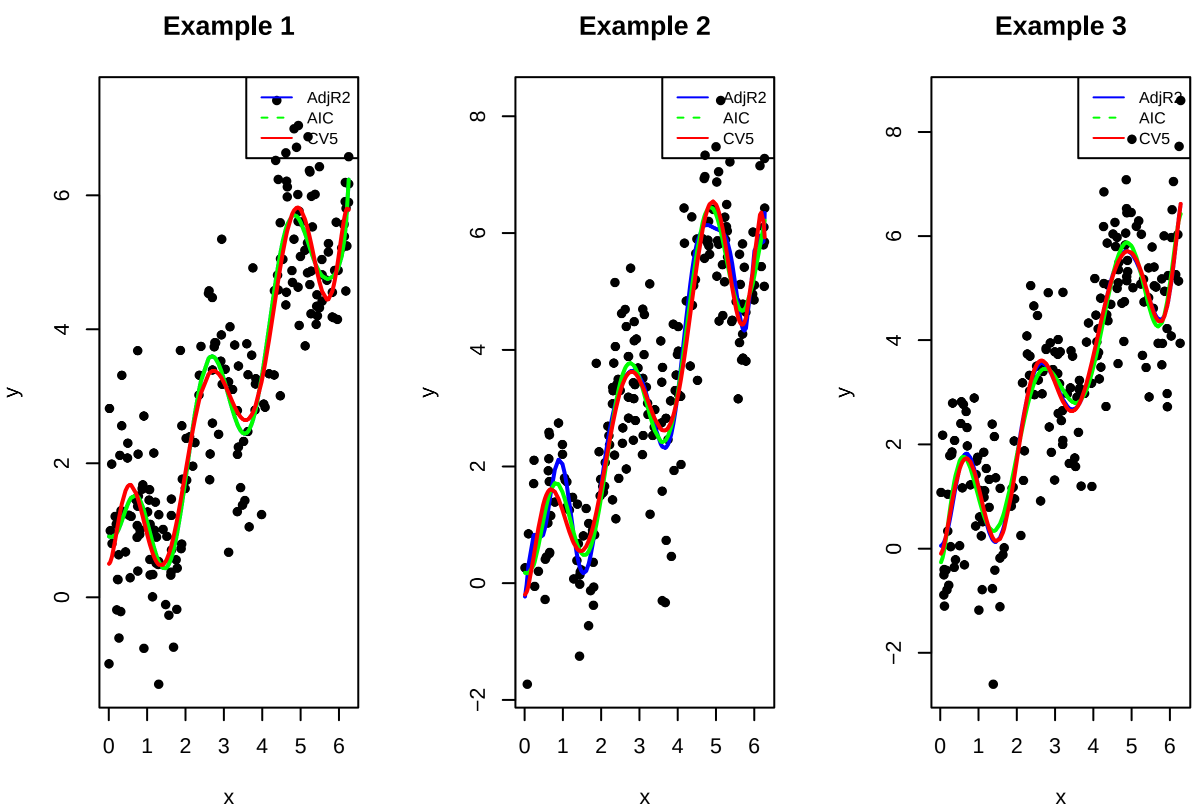
Consider a regression setting where the predictor variable is real valued and the goal is to fit a polynomial model. Specifically, we assume that $x_1, ..., x_n$ are iid uniform in $[0, 2\pi]$ and conditional on these, $y_1, ..., y_n$ are independent, with y_i normal with mean $sin(3x_i) + x_i$ and variance 1. Take $n = 200$ and set the maximum degree at 20. Perform simulations (at least 100 data instances) to compare the choice of degree by the sequential model selection methods in Problem 1. Produce plots of 3 example data instances and their best model fits according to different methods. Produce plots of the distribution of the polynomial degrees chosen by the different methods over all simulated instances. Offer comments on what you observe.

```r
generate_data <- function(n, p){
  # generate matrix X with n rows and p cols corresponding to degrees of 1st col
  xi = runif(n, 0, 2*pi)
  X = poly(xi, degree = p, raw = TRUE)
  y = rnorm(n, mean = sin(3*xi) + xi, sd = 1)
  return(cbind(y, X))
}

n = 200
p = 20
sims = 100
deg_selection_AdjR2 = rep(NA, sims)
deg_selection_AIC = rep(NA, sims)
deg_selection_CV5 = rep(NA, sims)

# record selected degrees for each method's best model over all simulations
for(sim in 1:sims){
  data = generate_data(n, p)
  X = data[,2:p+1]
  y = data[,1]

  model_AdjR2 = SequentialSelection(X, y, "AdjR2")
  deg_selection_AdjR2[sim] = length(model_AdjR2$coefficients) - 1

  model_AIC = SequentialSelection(X, y, "AIC")
  deg_selection_AIC[sim] = length(model_AIC$coefficients) - 1

  model_CV5 = SequentialSelection(X, y, "CV5")
  deg_selection_CV5[sim] = length(model_CV5$coefficients) - 1
}

# histograms of selected degrees
par(mfrow=c(1,3))
hist(deg_selection_AdjR2, xlab="degree", main="AdjR2")
hist(deg_selection_AIC, xlab="degree", main="AIC")
hist(deg_selection_CV5, xlab="degree", main="CV5")
```




```
# example plots
for(i in 1:3){
  data = generate_data(n, p)
  X = data[,2:p+1]
  x = data[,2]
  y = data[,1]

  model_AdjR2 = SequentialSelection(X, y, "AdjR2")
  pred_AdjR2 = predict(model_AdjR2)

  model_AIC = SequentialSelection(X, y, "AIC")
  pred_AIC = predict(model_AIC)

  model_CV5 = SequentialSelection(X, y, "CV5")
  pred_CV5 = predict(model_CV5)

  # plot points
  plot(x = data[,2], y = data[,1], pch=16,
       xlab="x", ylab="y", main=sprintf('Example %i', i))

  # plot 3 polys via different methods
  ix = sort(x, index.return=T)$ix
  colors = c("blue", "green", "red")
  lines(x[ix], pred_AdjR2[ix], col=colors[1], lwd=2)
  lines(x[ix], pred_AIC[ix], col=colors[2], lwd=2)
  lines(x[ix], pred_CV5[ix], col=colors[3], lwd=2)
```

```
  legend("topright", legend=c("AdjR2", "AIC", "CV5"), col=colors,
         lty=1:2, cex=0.8)
}
```



From the histograms, we see that all the evaluation metrics produce the majority of optimal models at degree 8 to 10. Using AIC as an evaluation metric for model selection generates models with lower degrees compared to the other evaluation metrics: adjusted r squared and cross-validation error. In particular, for the cross-validation error, we see that a good portion of optimal models have high degrees. This is probably due to the fact that cross-validation error does not penalize model complexity, while AIC and adjusted r squared do.

The plotted examples all show fitted polynomials via different methods. There is no significant difference between the three evaluation metrics. They all do a fairly good job of fitting the data in this instance.