Stat 427/627 Statistical Machine Learning

In-class Lab 5: Cross-Validation

Contents

Recall the `Auto` data set in the `ISLR2` package. This data frame has 392 observations on the following 9 variables.

- `mpg`: miles per gallon
- `cylinders`: Number of cylinders between 4 and 8
- `displacement`: Engine displacement (cu. inches)
- `horsepower`: Engine horsepower
- `weight`: Vehicle weight (lbs.)
- `acceleration`: Time to accelerate from 0 to 60 mph (sec.)
- `year`: Model year (modulo 100)
- `origin`: Origin of car (1. American, 2. European, 3. Japanese)
- `name`: Vehicle name

```
library(ISLR2)
colnames(Auto)
```

```
## [1] "mpg"          "cylinders"    "displacement" "horsepower"   "weight"
## [6] "acceleration" "year"         "origin"       "name"
```

1 Validation set approach (review)

Randomly split the data set into training vs validation (or testing) set.

```
pct.train <- 0.5  # training percentage can vary, but should be > 0.5
n <- nrow(Auto)
Z <- sample(n, floor(pct.train*n))
auto.train <- Auto[Z, ]
auto.test <- Auto[-Z, ]

mpg.lm <- lm(mpg ~  weight + horsepower + acceleration, data=auto.train)

# Predict Y on the testing/validation set.
pred.test <- predict(mpg.lm, newdata=auto.test)
```

```r
# Prediction MSE
mean((auto.test$mpg - pred.test)^2)
```

```
## [1] 18.11043
```

2 Leave-One-Out Cross-Validation (LOOCV) for linear regression.

There are functions in different R package that can do LOOCV for various algorithms. For examples, `lda()` and `qda()` have a built-in option `CV=TRUE` that returns LOOCV classification. Function `knn.cv()` handles cross-validation for KNN.

We will work with `cv.glm(data, glmfit, cost, K)` in package `boot` for linear regression and logistic regression (later).

```r
# install.package(boot)
library(boot)
```

Be sure to the entire date set for LOOCV. The algorithm will take care of the "split."

- `glm()` can also fit linear regression with `family=gaussian` (default).

```r
mpg.glm <- glm(mpg ~  weight + horsepower + acceleration, family=gaussian, data=Auto)
mpg.glm
```

```
##
## Call:  glm(formula = mpg ~ weight + horsepower + acceleration, family = gaussian,
##     data = Auto)
##
## Coefficients:
##  (Intercept)        weight    horsepower  acceleration
##    45.678293     -0.005789     -0.047496     -0.002066
##
## Degrees of Freedom: 391 Total (i.e. Null);  388 Residual
## Null Deviance:       23820
## Residual Deviance: 6994  AIC: 2252
```

```r
cv.error <- cv.glm(data=Auto, glmfit=mpg.glm)
names(cv.error)
```

```
## [1] "call"  "K"     "delta" "seed"
```

```r
cv.error$delta
```

```
## [1] 18.25595 18.25542
```

`cv.glm()` saves the LOOCV prediction MSE ($\frac{1}{n}\sum(Y_i - \widehat{Y}_{(-i)})^2$) in `delta.delta` consists of 2 numbers:

– `delta[1]` is the estimated prediction error. - `delta[2]` is adjusted for the lost sample size due to cross-validation.

For model comparison purpose, one can use either version of `delta` as long as it is consistent in the analysis.

Consider models with different orders of `horsepower` as the predictor.

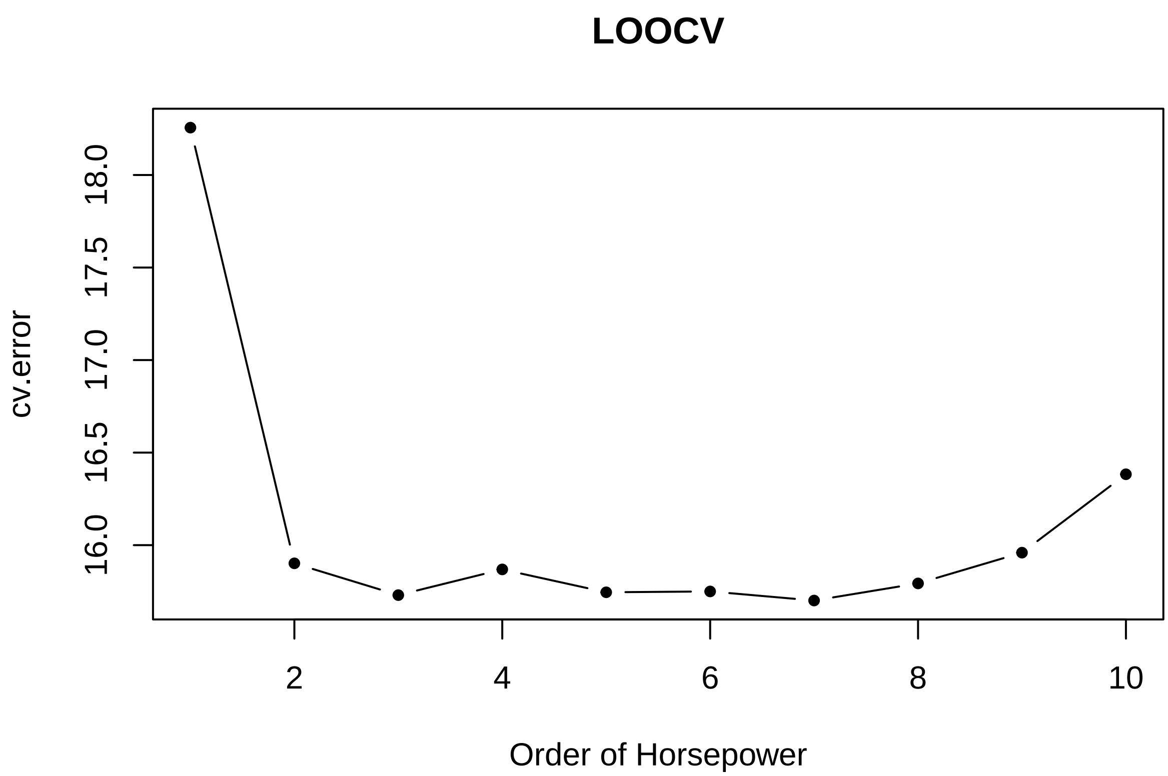
```r
cv.error <- rep(0, 10)
# Use a loop to fit model with `horsepower` in 1, 2, ..., p orders.
for (p in 1:10) {
  glm.fit <- glm( mpg ~ weight + poly(horsepower,p) + acceleration, data=Auto)
```

```r
  cv.error[p] = cv.glm( Auto, glm.fit )$delta[1]
}
```

```r
cv.error
```

```
##  [1] 18.25595 15.90163 15.72995 15.86879 15.74517 15.74989 15.70073 15.79314
##  [9] 15.95933 16.38301
```

```r
plot(cv.error, type="b", pch=20, xlab="Order of Horsepower", main="LOOCV")
```



3 K-fold Cross-Validation for linear regression.

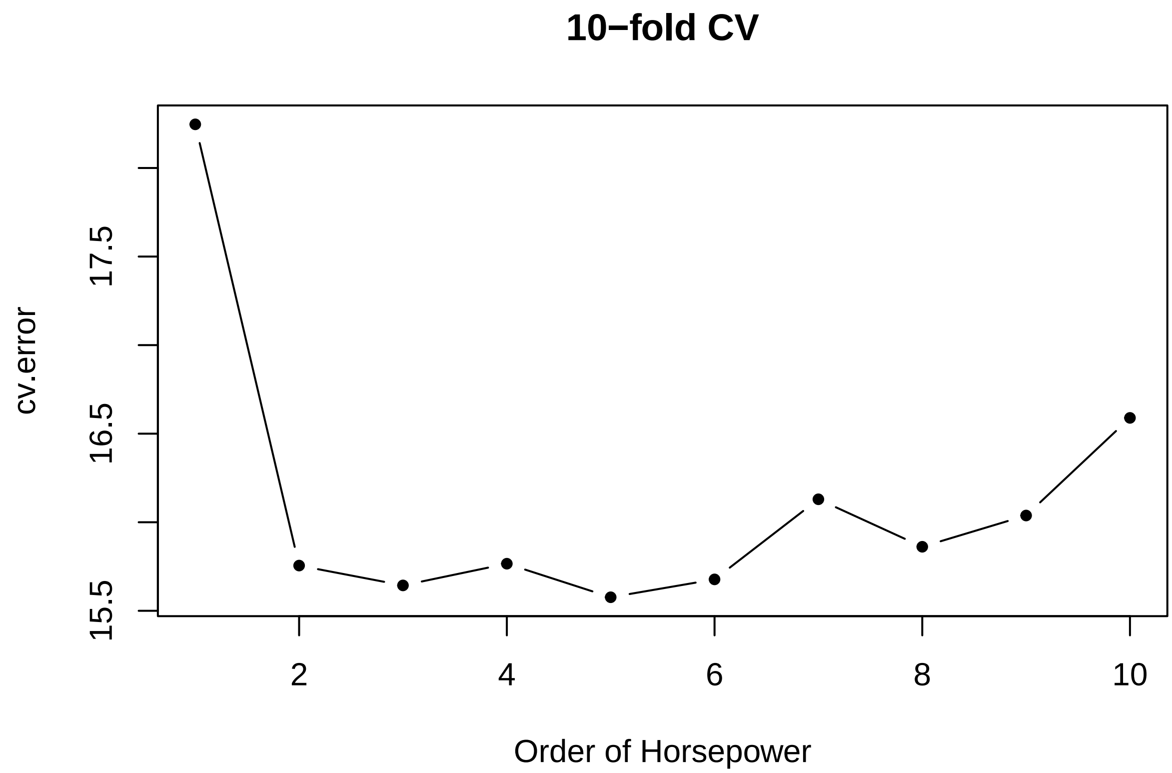
We can set the number of folds in `cv.glm()` using `K=__` in the function. By default, $K = n$ which is LOOCV.

```r
cv.error <- rep(0, 10)
# Use a loop to fit model with `horsepower` in 1, 2, ..., p orders.
for (p in 1:10) {
  glm.fit <- glm( mpg ~ weight + poly(horsepower,p) + acceleration, data=Auto)
  cv.error[p] = cv.glm(Auto, glm.fit, K=10)$delta[1]
}

cv.error
```

```
##  [1] 18.24636 15.75527 15.64321 15.76580 15.57649 15.67733 16.12964 15.86144
##  [9] 16.03796 16.58903
```

```r
plot(cv.error, type="b", pch=20, xlab="Order of Horsepower", main="10-fold CV")
```



Note that K-fold CV values change when you rerun the code, unless the random seed is set. Why?

4 Cross-validation for logistic regression.

Recall that `cv.glm()` computes `delta` to estimate the prediction MSE ($\frac{1}{n}\sum(Y_i - \widehat{Y}_{(-i)})^2$). When the response is 0-1, $\widehat{Y}_{(-i)} = \widehat{\pi}_{(-i)}$, where $\pi = P(Y = 1)$.

However, the correct classification rate and the error rate are more standard measures of classification accuracy. We can force `cv.glm()` to return these measures by introducing a suitable loss function. The prediction error rate will be calcuated and saved in `delta`.

Define a 0-1 loss function for classification. We will use cutoff value 0.5 for now. You can change it if needed.

```
loss <- function(Y, pred.p){
  return( mean( (Y==1 & pred.p < 0.5) | (Y==0 & pred.p >= 0.5) ) )
  }

loss(c(1, 1), c(0.3, 0.6))
```

```
## [1] 0.5
```

Recall the Students' depression data.

```
depr <- read.csv("../Data/depression_data.csv", header=T)
# Remove missing values in Diagnosis
depr <- na.omit(depr)
# Since we'll use logistic regression for this data set. Convert the response as 0-1 or a factor.
```

```r
depr$Diagnosis <- 1*(depr$Diagnosis == 1)
names(depr)
```

```
## [1] "ID"              "Gender"          "Guardian_status"  "Cohesion_score"
## [5] "Depression_score" "Diagnosis"
```

```r
depr.glm1 <- glm(Diagnosis ~ Gender + Guardian_status + Cohesion_score,
                 family = binomial, data=depr)
cv1 <- cv.glm(depr, depr.glm1, cost=loss)  # By default, K=n, LOOCV

# Drop guardian?
depr.glm2 <- glm(Diagnosis ~ Gender + Cohesion_score,
                 family = binomial, data=depr)
cv2 <- cv.glm(depr, depr.glm2, cost=loss)  # By default, K=n, LOOCV

cbind(WithGuardianER=cv1$delta, NoGuardianER=cv2$delta)
```

```
##      WithGuardianER NoGuardianER
## [1,]      0.1615721    0.1572052
## [2,]      0.1615721    0.1572005
```

Which model, with or without `guadian`, performs better based on CV prediction error rate?

5 Cross-validation in `lda()`, `qda()` and KNN.

Read the help file of `lda()`, `qda()` and `knn.cv()` for more details.

Examples of LOOCV in `lda()` and `qda()` can be found in previous R example.

You'll try `knn.cv()` in homework.