Bias-variance trade-off and cross validation – Computer exercises

6.1 Cross validation in \(k\)-NN

In this exercise we will return to the Biopsy data set also used in Exercise 4.1 (Lesson 4). We will try to determine suitable value of \(k\) in \(k\)-NN for this data. For simplicity, we will only consider the three attributes in columns \(V3\), \(V4\) and \(V5\) in this problem.

(a) Consider all data as training data. Investigate how the training error varies with different values of \(k\) (hint: use a for-loop). Which \(k\) gives the best result? Is it a good choice of \(k\)?

(b) Split the data randomly into a training and validation set, and see how well you perform on the validation set. (Previously, we have used the terminology "training" and "test" set. If the other set (not the training set) is used to make design decisions, such as choosing \(k\), it is really not a test set, but rather a "validation" set. Hence the terminology.) Which \(k\) gives the best result?

(c) Perform (b) 10 times for different validation sets and average the result. Which \(k\) gives the best result?

(d) Perform 10-fold cross-validation by first randomly permute the data set, divide the data set into 10 equally sized parts and loop through them by taking one part as validation set and the rest as training set each time (as is described in Section 5.1.3 in ISL). Which \(k\) gives the best result?

6.2 Cross validation for model choice

In this problem we will consider the data sets Pima.tr and Pima.te in the MASS library. Your task is to do as good prediction as possible for the test set Pima.te, but you are only allowed to look at the true output in Pima.te once (like in the real life, where you design and implement a method, and then hand it over to the ultimate test, namely the user). Hence, you will have to use Pima.tr for both deciding which model to use and training the model.

The data set describes the prevalence of diabetes in women at least 21 years old of Pima Indian heritage, living near Phoenix, Arizona, USA. The data set describes, for each individual, whether she has diabetes or not, her age, the diabetes pedigree function (a summary of the diabetes history in her family), BMI, skin thickness, blood pressure, plasma glucose concentration and number of pregnancies.

(a) Load the library and familiarize yourself with Pima.tr

(b) See how well you can fit the Pima.tr with logistic regression, LDA, QDA and \(k\)-NN (\(k = 2\)). The output is whether an individual has diabetes or not, and the input the remaining variables. What error rate does each method have? Is it a good indicator of which method is preferable?

(c) Instead of (a), perform 10-fold cross-validation by first randomly permute Pima.tr and divide it in 10 parts. Then, in a loop with one of the 10 parts held out as validation data, fit logistic regression, LDA, QDA and \(k\)-NN (\(k = 2\)) to the training data and evaluate the performance on the validation data. Plot your results in a box plot with the error rates (akin to Figure 4.10 and 4.11 in ISL). Feel free to play around with the choice of inputs and other settings to improve the performance. Which method does this suggest us to use?

(d) Now, decide which method to choose and train it on the entire data set Pima.tr and predict Pima.te. How well do you perform?

(e) Now, since we are in a simulated environment, we can cheat and break the rule that we were only allowed to look at the true output in Pima.te once. That is, explore how well the other methods do when you train them on Pima.tr and predict Pima.te. Did you make the "right" choice in (d)?

February 4, 2019
6.3 **Implementing problem 5.3** Verify your theoretical findings from problem 5.3 by repeating the experiment $N$ times and approximating all expected values with sums. Let $\sigma^2 = 1$.

(a) Generate training data ($n = 1$), estimate $\beta_0$ and compute $\hat{g}(x_\star, T)$. Repeat $N$ times and store the results in a vector. Choose the regularization parameter yourself.

(b) Estimate $g(x_\star) = \mathbb{E}_T[y(x_\star, T)]$ from your vector of $\hat{g}(x_\star, T)$. Compare your result to your theoretical findings in 5.3b.

(c) Estimate the square bias $\mathbb{E}_*[\{(g(x_\star) - f(x_\star))^2\}]$ using your result from b) and your knowledge about the true $f(x)$. Compare your result to your theoretical findings in 5.3c.

(d) Estimate the variance $\mathbb{E}_*[\mathbb{E}_T[(y(x_\star, T) - y_\star)^2]]$ using your vector of $\hat{g}(x_\star, T)$ from a) and your result from b). Compare your result to your theoretical findings in 5.3d.

(e) Estimate the expected new data error $\bar{E}_{\text{new}} = \mathbb{E}_T[E_{\text{new}}] = \mathbb{E}_T[\mathbb{E}_*[\{(y(x_\star, T) - g(x_\star))^2\}]$ by, for each $\hat{g}(x_\star, T)$ in your vector from a), simulate $N$ copies of $y_*$. Compare your result to your theoretical findings in 5.3f.

(f) Make a loop over different values for the regularization parameter $\gamma$ and plot bias, variance and $\bar{E}_{\text{new}}$ as a function of $\gamma$. Also plot your theoretical findings from 5.3 in the same plot.

6.4 **Implementing problem 5.5**

Design an experiment (similarly to 6.3) where you numerically confirm the results from problem 5.5.
Solutions

6.1 (a) $k = 1$ has the best performance. That is, however, expected and is not a good measure of which $k$ should be used.

(b) $k$ between 5 and 20 seems optimal, but the result seems noisy and hard to interpret.

(c) It is now rather clear that $k$ between 5 and 40 are all in the same ballpark.

(d) A similar result as in (c).
CrossValidation1.R

library(MASS)
library(class)
set.seed(1)

X <- as.matrix(biopsy[,c("V3","V4","V5")])
y <- as.matrix(biopsy[,"class"])

# a
N.K <- 200 # biggest number i kNN to try
error.training <- c()
for (kt in 1:N.K)
{
  pred <- knn(train=X,test=X,cl=y,k=kt)
  error.training[kt] <- 1-mean(pred==y)
}
plot(x=1:N.K,y=error.training,type="l",xlab="k",ylab="training error",main="training error for kNN")

# b
training.indices <- sample(nrow(biopsy), size = 600, replace = FALSE)
X.training <- X[training.indices,]
y.training <- y[training.indices,]
X.validation <- X[-training.indices,]
y.validation <- y[-training.indices,]

error.validation <- c()
for (kt in 1:N.K)
{
  pred <- knn(train=X.training,test=X.validation,cl=y.training,k=kt)
  error.validation[kt] <- 1-mean(pred==y.validation)
}
plot(x=1:N.K,y=error.validation,type="l",xlab="k",ylab="validation set error",main="validation set error for kNN")

# c
N.CV <- 10 # repeat 10 times
error.crossvalidation1 <- matrix(0,N.K,N.CV)
for (i in 1:N.CV)
{
  training.indices <- sample(nrow(biopsy), size = 600, replace = FALSE)
  X.training <- X[training.indices,]
  y.training <- y[training.indices,]
  X.validation <- X[-training.indices,]
  y.validation <- y[-training.indices,]

  for (kt in 1:N.K)
  {
    pred <- knn(train=X.training,test=X.validation,cl=y.training,k=kt)
    error.crossvalidation1[kt,i] <- 1-mean(pred==y.validation)
  }
}
plot(x=1:N.K,y=rowMeans(error.crossvalidation1),type="l",xlab="k",ylab="validation error",main="cross validation error for kNN")
6.2 (a) - 
(b) The error rates are 0.464 for logistic regression, 0.23 for LDA and QDA, and 0.13 for k-NN. This is, however, not a good way to assess the performance, any enough flexible method will fit the training data well, but may face serious problem in generalizing the behavior and perform bad when facing previously unseen data (i.e., test data). As an extreme example, with \( k = 1 \) the error rate for k-NN would have been 0 (why?).

(c) The plot suggests that the performance of linear regression and LDA is comparable for this problem. Also the performance of QDA is very similar. k-NN performs, however, worse (at least for \( k = 2 \)). Based on this, any choice of logistic regression or LDA seems reasonable for this problem.

(d) – (e) The error rate on the test data is 0.199 for logistic regression, 0.202 for LDA, 0.229 for QDA and 0.289 for k-NN. This was indeed anticipated based on the cross validation in (c).
```R
ER <- function(y, yhat)
  r <- 1 - mean(y == yhat)
  return(r)
}

glm.model <- glm(formula = type ~ ., data = Pima.tr, family = binomial)
glm.probs <- predict(object = glm.model, newdata = Pima.tr, type="response")
glm.predictions <- rep("No", nrow(Pima.te))
glm.predictions[gelm.probs>.5] <- "Yes"
glm.ER <- ER(y = Pima.te$type, yhat = glm.predictions)

lda.model <- lda(formula = type ~ ., data = Pima.tr)
lda.predictions <- predict(object = lda.model, newdata = Pima.tr)
lda.ER <- ER(y = Pima.tr$type, yhat = lda.predictions$y)

qda.model <- qda(formula = type ~ ., data = Pima.tr)
qda.predictions <- predict(object = qda.model, newdata = Pima.tr)
qda.ER <- ER(y = Pima.tr$type, yhat = qda.predictions$y)

kNN.predictions = knn(train = as.matrix(Pima.tr[-8]),
test = as.matrix(Pima.tr[-8]),
cl=Pima.tr$type, k=1)
kNN.ER <- ER(y = Pima.tr$type, yhat = kNN.predictions)

#c
N.cv = 10 # number of cross validation
ER.CV = data.frame(lin.reg=double(), # intialize a data frame for storing results
  lda=double(),
  qda=double(),
  kNN=double())
randomize.indices <- sample(nrow(Pima.tr), size = nrow(Pima.tr), replace = FALSE)
Pima.tr.randomized <- Pima.tr[randomize.indices,]
for (i in 1:N.cv) {
  start.index = (i-1)*ceiling(nrow(Pima.tr)/N.CV)+1
  end.index = min(i*ceiling(nrow(Pima.tr)/N.CV), nrow(Pima.tr))
  validation.indices <- seq(from = start.index, to = end.index, by = 1)
  validation.data <- Pima.tr.randomized[validation.indices,]
  training.data <- Pima.tr.randomized[-validation.indices,]
  glm.model <- glm(formula = type ~ ., data = training.data, family = binomial)
  glm.probs <- predict(object = glm.model, newdata = validation.data, type="response")
  glm.predictions <- rep("No", nrow(validation.data))
  glm.predictions[glm.probs>.5] <- "Yes"
  glm.ER <- ER(y = validation.data$type, yhat = glm.predictions)
  lda.model <- lda(formula = type ~ ., data = training.data)
  lda.predictions <- predict(object = lda.model, newdata = validation.data)
  lda.ER <- ER(y = validation.data$type, yhat = lda.predictions$y)
  qda.model <- qda(formula = type ~ ., data = training.data)
  qda.predictions <- predict(object = qda.model, newdata = validation.data)
  qda.ER <- ER(y = validation.data$type, yhat = qda.predictions$y)
  kNN.predictions <- knn(train = as.matrix(training.data[-8]),
test = as.matrix(validation.data[-8]),
cl=training.data$type, k=2)
  kNN.ER <- ER(y = validation.data$type, yhat = kNN.predictions)
  ER.CV[nrow(ER.CV)+1,] <- c(glm.ER, lda.ER, qda.ER, kNN.ER)
}
boxplot(ER.CV)

#d-e
glm.model <- glm(formula = type ~ ., data = Pima.tr, family = binomial)
glm.probs <- predict(object = glm.model, newdata = Pima.te, type="response")
glm.predictions <- rep("No", nrow(Pima.te))
glm.predictions[glm.probs>.5] <- "Yes"
glm.ER <- ER(y = Pima.te$type, yhat = glm.predictions)
```

February 4, 2019
```r
lda.model <- lda(formula = type ~ ., data = Pima.tr)
lda.predictions <- predict(object = lda.model, newdata = Pima.te)
lda.ER <- ER(y = Pima.te$type, yhat = lda.predictions$class)

qda.model <- qda(formula = type ~ ., data = Pima.tr)
qda.predictions <- predict(object = qda.model, newdata = Pima.te)
qda.ER <- ER(y = Pima.te$type, yhat = qda.predictions$class)

kNN.predictions <- knn(train = as.matrix(Pima.tr[-8]),
                       test = as.matrix(Pima.te[-8]),
                       cl = Pima.tr$type, k = 2)
kNN.ER <- ER(y = Pima.te$type, yhat = kNN.predictions)
```

6.3 Solution not written; See the Python solution for an idea.

6.4 Solution not written; See the Python solution for an idea.