Classification 02

Resampling and evaluation of prediction methods

Josep Fortiana 2019-10-29

The goal of this laboratory is to attain a practical knowledge of some of the available procedures to assess the quality of prediction methods, classification (categorical response) and regression (numerical response), with a particular emphasis on detecting and avoiding overfitting.

We consider four procedures or families of procedures:

1. Hold-out: Split the dataset in two parts, a training subset and a 'test' subset. The first one is used to train (estimate or learn parameters of) the prediction method. Meanwhile, the 'test' subset is held out, keeping it apart to rule out the optimistic bias inherent in using the same samples for learning to predict and for evaluating goodness of prediction.
2. k-fold cross-validation,
3. Leave-one-out (LOO),
4. Bootstrap.

We already know the first procedure. The second one is a systematic repetition of hold-out in such a way that consecutively each individual is in a training subset and in a test subset. The third procedure is an extreme k-fold cross-validation, where $k = n$, the number of individuals in the sample, in which the test subset is a single individual and the training subset the remaining $(n - 1)$ restantes.

The fourth procedure is an application of a much more general statistical concept, the *bootstrap,* which we review in Section A, before going to its application with the other procedures.

Datasets and prediction methods

A. `wine` dataset

`wine` data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

They can be found in the UCI Machine Learning Repository, which hosts many documented data sets to be used as benchmarks in evaluating Machine Learning methods and algorithms. Alternatively, should the link be broken, you can find the `.csv` file in the Virtual Campus. The following description is taken from the UCI website:

I think that the initial data set had around 30 variables, but for some reason I only have the 13 dimensional version. I had a list of what the 30 or so variables were, but a.) I lost it, and b.), I would not know which 13 variables are included in the set.

The attributes are:

1. Alcohol
2. Malic acid
3. Ash
4. Alcalinity of ash
5. Magnesium
6. Total phenols

7. Flavonoids
8. Nonflavonoid phenols
9. Proanthocyanins
10. Color intensity
11. Hue
12. OD280/OD315 of diluted wines
13. Proline

In a classification context, this is a well posed problem with "well behaved" class structures. A good data set for first testing of a new classifier, but not very challenging.

Since the `.csv` file has no first row with variable names we must set `header=FALSE` in the `read.csv` call (see default values for the optional parameters in the help).

The casting `as.factor()` command has the purpose of conveying the fact that this variable is qualitative, so the R interpreter can use it as such.

```r
wine.url<-"http://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data"
#wine<-read.csv(wine.url,header=FALSE)
wine<-read.csv("wine.csv",header=FALSE)
colnames(wine)<-c("Type","Alcohol","Malic","Ash", "Alcalinity","Magnesium","Phenols","Flavonoids",
                  "Nonflavonoids","Proanthocyanins","Color","Hue", "Dilution","Proline")
wine$Type <- as.factor(wine$Type)
```

Aa. `wine` dataset and k-NN

Aa1. Hold-out

Split the whole dataset into two subsets, `train` with $\approx 60\%$ of data, and `test` the rest.

```r
n<-nrow(wine)
ntrain<-ceiling(0.6*n)
ntest<-n-ntrain
Itrain<-sample(1:n,ntrain,replace=FALSE)
wine.train<-wine[Itrain,]
wine.test<-wine[-Itrain,]
Xtrain<-as.matrix(wine.train[,-1])
ytrain<-wine.train[,1]
Xtest<-as.matrix(wine.test[,-1])
ytest<-wine.test[,1]
```

```r
#install.packages("class",dependencies=TRUE,repos="https://cloud.r-project.org")
require(class)
```

```
## Loading required package: class
```

Confusion matrix and misclassification error estimate

```r
k<-5
y.hat<-knn(Xtrain,Xtest,ytrain,k )
C<-table("True"=ytest,"Predicted"=y.hat)
P.err<-(ntest-sum(diag(C)))/ntest
C
```

```
##     Predicted
## True  1  2  3
##    1 24  0  3
##    2  3 18  9
##    3  1  8  5
```

```r
sprintf("For k =%d,   Prob.err = %5.2f ", k, round(P.err,2))
```

```
## [1] "For k =5,   Prob.err =  0.34 "
```

Which is the optimal k?

Redo the above computation with several values of k and decide which is the optimal k according to the probability of misclassification.

Hint: After doing it *"by hand"* you can try a systematic approach, building a table where the first column is `k` and the second column the estimated probability of misclassification.

Aa.2. k-fold cross-validation

Here we perform k-fold cross-validation with some simple code. An alternative approach is to use functions from a dedicated R package. My personal prejudice is in favor of writing one's own code. Anyway you can see in the Appendix at the end of this notebook some possible such packages and you can judge by yourselves.

```r
kfoldIndexes<-function(n,k){
    l<-floor(n/k)
    Indexes<-c(1,(1:k)*l)
    Indexes[k+1]<-n
    return(Indexes)
    }
```

```r
# Decide a 'K' for K-fold cross-validation (uppercase to avoid notation clash with the k in k-NN)
n<-nrow(wine)
K<-5
J<-kfoldIndexes(n,K)
J
```

```
## [1]   1  35  70 105 140 178
```

```r
Lower<-J[-(K+1)]
Upper<-J[-1]
Lower
```

```
## [1]   1  35  70 105 140
```

```r
Upper
```

```
## [1]  35  70 105 140 178
```

```r
# Una random permutation of indexes
I<-sample(1:n)
```

```r
k<-7 # this is the 'k' in k-NN
# Repeat with several 'k' and choose an optimal value.
P.ERR<-rep(0,K)
for (fold in 1:K){
    Itest<-I[Lower[fold]:Upper[fold]]
    wine.test<-wine[Itest,]
    wine.train<-wine[-Itest,]
    Xtrain<-as.matrix(wine.train[,-1])
    ytrain<-wine.train[,1]
    Xtest<-as.matrix(wine.test[,-1])
    ytest<-wine.test[,1]
    y.hat<-knn(Xtrain,Xtest,ytrain,k)
    C<-table("True"=ytest,"Predicted"=y.hat)
    print(C)
    P.ERR[fold]<-(ntest-sum(diag(C)))/ntest
    }
```

```
##     Predicted
## True  1  2  3
##    1 11  1  1
##    2  0 10  5
##    3  0  3  4
##     Predicted
## True  1  2  3
##    1 11  0  3
##    2  0  9  2
##    3  0  5  6
##     Predicted
## True  1  2  3
##    1  8  0  2
##    2  0 12  6
##    3  3  1  4
##     Predicted
## True  1  2  3
##    1  9  0  1
##    2  1 10  4
##    3  0  4  7
##     Predicted
## True  1  2  3
##    1 12  0  0
##    2  3  7  5
##    3  1  2  9
```

```
round(P.ERR,3)
```

```
## [1] 0.648 0.634 0.662 0.634 0.606
```

```
mean.p.err<-mean(P.ERR)
round(mean.p.err,3)
```

```
## [1] 0.637
```

Aa.3. Leave-one-out *(LOO)*

```
k<-7
# Repeat with several 'k' and choose an optimal value.
g<-nlevels(wine$Type)
C<-matrix(0,nrow=g,ncol=g)
for (i in 1:n){
    wine.test<-wine[i,]
    wine.train<-wine[-i,]
    Xtrain<-as.matrix(wine.train[,-1])
    ytrain<-wine.train[,1]
    Xtest<-as.matrix(wine.test[,-1])
    ytest<-wine.test[,1]
    y.hat<-knn(Xtrain,Xtest,ytrain,k)
    C[ytest,y.hat]=C[ytest,y.hat]+1
    }
print(C)
```

```
##      [,1] [,2] [,3]
## [1,]   53    0    6
## [2,]    5   45   21
## [3,]    3   21   24
```

```
p.err<-(n-sum(diag(C)))/n
round(p.err,3)
```

```
## [1] 0.315
```

Aa.4. *bootstrap*

```
k<-7
# Repeat with several 'k' and choose an optimal value.
n<-nrow(wine)
I<-1:n
# Number of bootstrap resamples
B<-10
P.ERR<-rep(0,B)
for (b in 1:B){
    Ib<-sample(I,n,replace = TRUE)
    oob<-I[is.na(match(I,Ib))]
    Itest<-oob
    ntest<-length(oob)
    print(ntest)
    wine.test<-wine[Itest,]
    wine.train<-wine[-Itest,]
    Xtrain<-as.matrix(wine.train[,-1])
    ytrain<-wine.train[,1]
    Xtest<-as.matrix(wine.test[,-1])
    ytest<-wine.test[,1]
    y.hat<-knn(Xtrain,Xtest,ytrain,k)
    C<-table("True"=ytest,"Predicted"=y.hat)
    print(C)
    P.ERR[b]<-(ntest-sum(diag(C)))/ntest
    }
```

```
## [1] 67
##     Predicted
## True  1  2  3
##    1 18  0  2
##    2  1 23  9
##    3  1  5  8
## [1] 65
##     Predicted
## True  1  2  3
##    1 18  0  5
##    2  1 18  5
##    3  0  8 10
## [1] 59
##     Predicted
## True  1  2  3
##    1 20  0  6
##    2  2  9 11
##    3  0  1 10
## [1] 64
##     Predicted
## True  1  2  3
##    1 15  0  3
##    2  1 18 11
##    3  4  3  9
## [1] 66
##     Predicted
## True  1  2  3
##    1 22  0  3
##    2  1 10 12
##    3  3  5 10
## [1] 61
##     Predicted
## True  1  2  3
##    1 15  0  2
##    2  2 20  5
##    3  1  7  9
## [1] 67
##     Predicted
## True  1  2  3
##    1 22  0  1
##    2  2 17  8
##    3  3  9  5
## [1] 62
##     Predicted
## True  1  2  3
##    1 23  0  1
##    2  2 18  2
##    3  1 10  5
## [1] 72
##     Predicted
## True  1  2  3
##    1 19  1  4
##    2  0 21 10
##    3  1  5 11
## [1] 54
##     Predicted
## True  1  2  3
##    1 16  0  2
##    2  1 16 10
##    3  1  1  7
```

```
round(P.ERR,3)
```

```
##  [1] 0.269 0.292 0.339 0.344 0.364 0.279 0.343 0.258 0.292 0.278
```

```r
mean.p.err<-mean(P.ERR)
round(mean.p.err,3)
```

```
## [1] 0.306
```

B. Auto dataset

Description

Gas mileage, horsepower, and other information for 392 vehicles.

Format

A data frame with 392 observations on the following 9 variables.

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

The orginal data contained 408 observations but 16 observations with missing values were removed.

Source

This dataset was taken from the StatLib library which is maintained at Carnegie Mellon University. The dataset was used in the 1983 American Statistical Association Exposition.

```r
#install.packages("ISLR",dependencies=TRUE,repos="https://cloud.r-project.org")
require(ISLR)
```

```
## Loading required package: ISLR
```

```r
data(Auto)
str(Auto)
```

```
## 'data.frame':    392 obs. of  9 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : num  130 165 150 150 140 198 220 215 225 190 ...
##  $ weight      : num  3504 3693 3436 3433 3449 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : num  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin      : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ name        : Factor w/ 304 levels "amc ambassador brougham",..: 49 36 231 14 161 141 54 223 241
```

```r
# Discard the 'name' variable, irrelevant for prediction
Auto<-Auto[,-9]
str(Auto)
```

```
## 'data.frame':    392 obs. of  8 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : num  130 165 150 150 140 198 220 215 225 190 ...
##  $ weight      : num  3504 3693 3436 3433 3449 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : num  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin      : num  1 1 1 1 1 1 1 1 1 1 ...
```

Ba.1 Hold-out

Ba.2 k-fold crossvalidation

Ba.3 Leave-one-out

Ba.4 Bootstrap

C. `SAheart` dataset

From the `ElemStatLearn` package, `SAheart` is a data frame with 462 observations on the following 10 variables.

1. `sbp`: systolic blood pressure.
2. `tobacco`: cumulative tobacco (kg).
3. `ldl`: low density lipoprotein cholesterol.
4. `adiposity`: a numeric vector.
5. `famhist`: family history of heart disease, a factor with levels `Absent`, `Present`.
6. `typea`: type-A behavior.
7. `obesity`: a numeric vector.
8. `alcohol`: current alcohol consumption.
9. `age`: age at onset
10. `chd`: response, coronary heart disease

Details

A retrospective sample of males in a heart-disease high-risk region of the Western Cape, South Africa. There are roughly two controls per case of CHD. Many of the CHD positive men have undergone blood pressure reduction treatment and other programs to reduce their risk factors after their CHD event. In some cases the measurements were made after these treatments. These data are taken from a larger dataset, described in Rousseauw et al, 1983, South African Medical Journal.

```r
#install.packages("ElemStatLearn",dependencies=TRUE,repos="https://cloud.r-project.org")
require(ElemStatLearn)
```

```
## Loading required package: ElemStatLearn
```

```r
data(SAheart)
```

```r
n<-nrow(SAheart)
ntrain<-ceiling(0.60*n)
Itrain<-sample(1:n,ntrain,replace=FALSE)
n<-nrow(SAheart)
ntrain<-ceiling(0.60*n)
Itrain<-sample(1:n,ntrain,replace=FALSE)
SAheart.train<-SAheart[Itrain,]
SAheart.test<-SAheart[-Itrain,]
```

Ca. SAheart data with logistic regression

```r
SAheart.logit1<-glm(chd~.,data=SAheart.train,family=binomial)
summary(SAheart.logit1)
```

```
##
## Call:
## glm(formula = chd ~ ., family = binomial, data = SAheart.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0853  -0.8362  -0.4674   0.9443   2.4517
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -5.660041   1.685115  -3.359 0.000783 ***
## sbp             0.001443   0.007507   0.192 0.847605
## tobacco         0.075593   0.035667   2.119 0.034056 *
## ldl             0.204639   0.078620   2.603 0.009244 **
## adiposity       0.002886   0.037955   0.076 0.939382
## famhistPresent  0.768758   0.288423   2.665 0.007690 **
## typea           0.040266   0.015661   2.571 0.010140 *
## obesity        -0.039552   0.060281  -0.656 0.511743
## alcohol         0.004266   0.005455   0.782 0.434170
## age             0.045118   0.015064   2.995 0.002745 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 365.45  on 277  degrees of freedom
## Residual deviance: 296.29  on 268  degrees of freedom
## AIC: 316.29
##
## Number of Fisher Scoring iterations: 4
```

Prediction of probabilities that the binary 0/1 response takes the value 1 (here `chd=1`):

```r
SAheart.pred<-predict(SAheart.logit1,newdata=SAheart.test,type="response")
str(SAheart.pred)
```

```
##  Named num [1:184] 0.294 0.244 0.688 0.171 0.539 ...
##  - attr(*, "names")= chr [1:184] "3" "7" "8" "9" ...
```

A *crisp* prediction of 0 or 1 is obtained from the above by taking a cut point, e.g., $L = 0.5$ (this is not a mandatory value, it may depend on the problem or on the *a priori* probabilites) and assign to 0 or 1 according to whether the probability is smaller or larger than this threshold:

```r
SAheart.pred.crisp<-1*(SAheart.pred>=0.5)
C<-table("True"=SAheart.test$chd,"Predicted"=SAheart.pred.crisp)
C
```

```
##     Predicted
## True   0   1
##    0 101  25
##    1  23  35
```

Ca.1 Hold-out

Ca.2 k-fold crossvalidation

Ca.3 Leave-one-out

Ca.4 Bootstrap

Cb. `SAheart` with Fisher's linear discriminant analysis

```r
require(MASS)
```

```
## Loading required package: MASS
```

```r
SAheart.lda1<-lda(chd~.,data=SAheart.train)
SAheart.pred<-predict(SAheart.lda1,newdata=SAheart.test)
C<-table("True"=SAheart.test$chd,"Predicted"=SAheart.pred$class)
C
```

```
##     Predicted
## True   0   1
##    0 102  24
##    1  24  34
```

Cb.1 Hold-out

Cb.2 k-fold crossvalidation

Cb.3 Leave-one-out

Cb.4 Bootstrap

Cb. `SAheart` with Quadratic discriminant

```r
SAheart.qda1<-qda(chd~.,data=SAheart.train)
SAheart.pred<-predict(SAheart.qda1,newdata=SAheart.test)
C<-table("True"=SAheart.test$chd,"Predicted"=SAheart.pred$class)
C
```

```
##     Predicted
## True   0   1
##    0 102  24
##    1  24  34
```

D. `Default` data set

From the `ISLR` package. A simulated data set containing information on ten thousand customers. The aim here is to predict which customers will default on their credit card debt.

A data frame with 10000 observations on the following 4 variables.

1. `default`: A factor with levels `No` and `Yes` indicating whether the customer defaulted on their debt
2. `student`: A factor with levels `No` and `Yes` indicating whether the customer is a student
3. `balance`: The average balance that the customer has remaining on their credit card after making their monthly payment
4. `income`: Income of customer

```r
#install.packages("ISLR",dependencies=TRUE,repos="https://cloud.r-project.org")
require(ISLR)
data(Default)
str(Default)
```

```
## 'data.frame':    10000 obs. of  4 variables:
##  $ default: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ student: Factor w/ 2 levels "No","Yes": 1 2 1 1 1 2 1 2 1 1 ...
##  $ balance: num  730 817 1074 529 786 ...
##  $ income : num  44362 12106 31767 35704 38463 ...
```

Da. `Default` data set with Fisher's linear discriminant

```r
n<-nrow(Default)
ntrain<-ceiling(0.6*n)
set.seed(24025)          # An arbitrary value, fixed for the sake of reproducibility of results
Itrain<-sample(1:n,ntrain,replace=FALSE)
Default.train<-Default[Itrain,]
Default.test<-Default[-Itrain,]
```

```r
require(MASS)
Default.lda<-lda(default~.,data=Default.train)
Default.pred<-predict(Default.lda,newdata=Default.test)
C<-table("True"=Default.test$default,"Predicted"=Default.pred$class)
C
```

```
##      Predicted
## True    No  Yes
##   No  3871    7
##   Yes   87   35
```

`Smarket` data set

This is an S&P Stock Market Data set. Daily percentage returns for the S&P 500 stock index between 2001 and 2005. Contained in the `ISLR` package as a `data.frame` with 1250 observations on the following 9 variables.

1. `Year`: The year that the observation was recorded
2. `Lag1`: Percentage return for previous day
3. `Lag2`: Percentage return for 2 days previous
4. `Lag3`: Percentage return for 3 days previous
5. `Lag4`: Percentage return for 4 days previous
6. `Lag5`: Percentage return for 5 days previous
7. `Volume`: Volume of shares traded (number of daily shares traded in billions)

8. `Today`: Percentage return for today
9. `Direction`: A factor with levels `Down` and `Up` indicating whether the market had a positive or negative return on a given day.

```r
require(ISLR)
data(Smarket)
str(Smarket)
```

```
## 'data.frame':    1250 obs. of  9 variables:
##  $ Year     : num  2001 2001 2001 2001 2001 ...
##  $ Lag1     : num  0.381 0.959 1.032 -0.623 0.614 ...
##  $ Lag2     : num  -0.192 0.381 0.959 1.032 -0.623 ...
##  $ Lag3     : num  -2.624 -0.192 0.381 0.959 1.032 ...
##  $ Lag4     : num  -1.055 -2.624 -0.192 0.381 0.959 ...
##  $ Lag5     : num  5.01 -1.055 -2.624 -0.192 0.381 ...
##  $ Volume   : num  1.19 1.3 1.41 1.28 1.21 ...
##  $ Today    : num  0.959 1.032 -0.623 0.614 0.213 ...
##  $ Direction: Factor w/ 2 levels "Down","Up": 2 2 1 2 2 2 1 2 2 2 ...
```

Appendix: dedicated R packages

Many R packages contain functions implementing methods for validating prediction procedures, for instance: - `caret`, - `CVST`, - `cvTools`, - `dprep`, - `sortinghat`.

One of them (`CVST`) is devoted to a validation technique more sophisticated than the four in this notebook, and the rest are generic, with application to some prediction methods than those in our course.