An Example of Cross-validation (CV)

October 23, 2017

We have already learned about the Validation set approach (creating separate training and test datasets; fitting a model using the training data, then using the fitted model to predict for the test data) to assessing the accuracy of a predictive model. One drawback to this approach is that the estimate of predictive accuracy can vary a lot depending on how the data are randomized into the test and training sets. It can also overestimate the prediction error. See page 176 in Introduction to Statistical Learning with R by James, witten, Hastie and Tibshirani for more discussion of the drawbacks of the validation set approach. Cross-validation is usually a better way to assess how well your model predicts. I will explain for 3-fold cross-validation, then generalize.

First you randomly split the data into 3 roughly equal size groups.

```
> library(MASS)
> head(fgl)

     RI    Na   Mg   Al    Si    K   Ca Ba   Fe type
1  3.01 13.64 4.49 1.10 71.78 0.06 8.75  0 0.00 WinF
2 -0.39 13.89 3.60 1.36 72.73 0.48 7.83  0 0.00 WinF
3 -1.82 13.53 3.55 1.54 72.99 0.39 7.78  0 0.00 WinF
4 -0.34 13.21 3.69 1.29 72.61 0.57 8.22  0 0.00 WinF
5 -0.58 13.27 3.62 1.24 73.08 0.55 8.07  0 0.00 WinF
6 -2.04 12.79 3.61 1.62 72.97 0.64 8.07  0 0.26 WinF

> set.seed(366)
> n <- nrow(fgl)  # 214 rows in fgl
> gp.vec <- c(rep(1,71),rep(2,71),rep(3,72))
> gp.vec.random.order <- sample(gp.vec,n,replace=FALSE)
> table(gp.vec.random.order)

gp.vec.random.order
 1  2  3
71 71 72

> fgl$cvgroup <- gp.vec.random.order  #randomly assign each subject to each cross-validation group
> head(fgl,n=6)

     RI    Na   Mg   Al    Si    K   Ca Ba   Fe type cvgroup
1  3.01 13.64 4.49 1.10 71.78 0.06 8.75  0 0.00 WinF       1
2 -0.39 13.89 3.60 1.36 72.73 0.48 7.83  0 0.00 WinF       2
3 -1.82 13.53 3.55 1.54 72.99 0.39 7.78  0 0.00 WinF       2
4 -0.34 13.21 3.69 1.29 72.61 0.57 8.22  0 0.00 WinF       1
5 -0.58 13.27 3.62 1.24 73.08 0.55 8.07  0 0.00 WinF       2
6 -2.04 12.79 3.61 1.62 72.97 0.64 8.07  0 0.26 WinF       3
```

The idea is to hold one group out, say group 1, then build a tree using the remaining data for groups 2 and 3. Then use the tree to predict for group 1. Next, we hold out group 2, build a tree using groups 1 and 3 data, the use the tree to predict for the group 2 data. Finally, hold out group 3, build a tree using group 1 and 2 data and use the tree to predict group 3 glass type.

```
> #tree constructed leaving out group 1, then used to predict for group 1
> library(tree)
```

```
> fgl.Notgroup1 <- subset(fgl,cvgroup!=1)
> treeNotgroup1 <- tree(type~.-cvgroup,    data=fgl.Notgroup1)
> set.seed(45)
> fgl.group1 <- subset(fgl,cvgroup==1)
> pred.gp1 <- predict(treeNotgroup1, newdata=fgl.group1, type="class")
> sum(pred.gp1==fgl.group1$type)  #71-35=36 misclassifications

[1] 35

> table(pred.gp1,truth=fgl$type[fgl$cvgroup==1])  #summary of prediction against actual

        truth
pred.gp1 WinF WinNF Veh Con Tabl Head
   WinF     9     3   2   0    0    0
   WinNF   15    11   1   4    0    3
   Veh      0     0   1   0    0    0
   Con      0     0   0   0    0    0
   Tabl     0     1   0   2    3    2
   Head     0     0   0   2    1   11

>
```

Do the same thing, holding out group 2 when making the tree.

```
> #tree constructed leaving out group 2, then use tree to predict for group 2
>
> fgl.Notgroup2 <- subset(fgl,cvgroup!=2)
> treeNotgroup2 <- tree(type~.-cvgroup,    data=fgl.Notgroup2)
> set.seed(98)
> fgl.group2 <- subset(fgl,cvgroup==2)
> pred.gp2 <- predict(treeNotgroup2, newdata=fgl.group2, type="class")
> sum(pred.gp2==fgl.group2$type)

[1] 46

> 71-46  #number of misclassification in group 2

[1] 25

>
```

Seems like a great job for a loop.

```
> #loop to repeat this process for i=1,2,3
> set.seed(98)
> num.correct.class <- NULL
> for (i in 1:3)
+   {fgl.Notgroup.i <- subset(fgl,cvgroup!=i)
+   treeNotgroup.i <- tree(type~.-cvgroup,    data=fgl.Notgroup.i)
+
+
+   fgl.group.i <- subset(fgl,cvgroup==i)
+
+   pred.gp.i <- predict(treeNotgroup.i, newdata=fgl.group.i, type="class")
+   num.correct.class[i] <- sum(pred.gp.i==fgl.group.i$type)
+   }
> total.correct.class <- sum(num.correct.class)
> total.misclass <- dim(fgl)[1]-total.correct.class
>
```

You can also do n-fold cross-validation (CV). If n=10 (the default in cv.tree), you start by splitting the data randomly into 10 groups, and implement the process just described 10 times, each time leaving out one group from tree-building for prediction. Typically 5 or 10 fold CV will do better than just splitting the data into a test and training set.