Demo of Cross Validation

Malathi

5 June 2018

K fold Cross validation

Cross Validation is a technique of validating the machine learning model to avoid under fitting and over fitting. ####Under fitting: The Machine Learning Model is too simple and fails to capture the underlying trend of the data.The model does not perform well for the training set as well as the testing set.

Over Fitting:

The model performs well for the training set and but makes too much of prediction errors for the teting set. Because the model is too complex and in the process captures random patterns and noise.

Cross Validation:

Cross Validation is a technique where you train on the entire data except for a portion of the data set which is used for validation.

Holdout Cross Validation:

In this approach, reserve 30% of dataset for validation and rest 70% for model training. Disadvantage of this approach is that, while training a model on 70% of the data set only, one may loose some interesting information about data i.e. higher bias. This 70-30 ratio can be changed according to the size of the dataset.

Leave one out:

Train the model on the entire data set except one point which is also a test point. So low bias.Every point becomes a test as well as traing point and iteration is for n times as there are n points..But this approach results in high execution time.The prediction is highly determined by this one point and leads to high variation.If this point is noise, it will affect the efficiency of the model.

K-Fold Cross validation

So train a model on large portion of data set to reduce higher bias. Test the model on a reasonable number of points to reduce variance error. Conduct the training and testing process multiple times, and change the train and test data set distribution. This process helps to validate the model effectively.

The k-fold cross validation procedure is as follows. Randomly split the entire dataset into k"folds". For each k folds in the dataset, build the model on k - 1 folds of the data set. Then, test the model to check the effectiveness for kth fold. Record the error on each of the predictions.Repeat this until each of the k folds has served as the test set. The average of the k recorded errors is called the cross-validation error and will serve as the performance metric for the model.

To choose the value of K: Lower value of K leads to higher bias. Higher value leads to less bias, but high variance. K=10 is normal.

Demonstration of K Fold Cross Validation with iris dataset

Packages needed: plyr,dplyr,randomForest

```r
data <- iris
head(data)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

cross validation, using Random Forest model to predict sepal.length

```r
k = 10
data$id <- sample(1:k, nrow(data), replace = TRUE)
head(data)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species id
## 1          5.1         3.5          1.4         0.2  setosa  8
## 2          4.9         3.0          1.4         0.2  setosa  8
## 3          4.7         3.2          1.3         0.2  setosa  4
## 4          4.6         3.1          1.5         0.2  setosa  1
## 5          5.0         3.6          1.4         0.2  setosa  9
## 6          5.4         3.9          1.7         0.4  setosa  1
```

```r
list <- 1:k
```

prediction and test set data frames are added with each iteration over the folds

```r
prediction <- data.frame()
testsetCopy <- data.frame()

dim(prediction)
```

```
## [1] 0 0
```

```r
dim(testsetCopy)
```

```
## [1] 0 0
```

Creating a progress bar to know the status of CV

```r
progress.bar <- create_progress_bar("text")
progress.bar$init(k)
```

```
##
  |
  |                                                                 |   0%
```

Function for k fold.

Remove rows with id i from dataframe to create training set.Select rows with id i to create test set.Run a random forest model.Remove response column 1, Sepal.Length.Append each iteration's predictions to the end of the prediction data frame.Append each iteration's test set to the test set copy data frame.Keep only the Sepal Length Column

```
for(i in 1:k){

trainingset <- subset(data, id %in% list[-i])
testset <- subset(data, id %in% c(i))
print(paste0("iteration:  ", i))
print(dim(trainingset))
print(dim(testset))

mymodel <- randomForest(trainingset$Sepal.Length ~ ., data = trainingset, ntree = 100)


temp <- as.data.frame(predict(mymodel, testset[,-1]))
print(dim(temp))
print(dim(prediction))


prediction <- rbind(prediction, temp)
print(dim(prediction))
testsetCopy <- rbind(testsetCopy, as.data.frame(testset[,1]))
print(dim(testsetCopy))

progress.bar$step()

}
```

```
## [1] "iteration:  1"
## [1] 132   6
## [1] 18  6
## [1] 18  1
## [1] 0 0
## [1] 18  1
## [1] 18  1
##
  |
  |======                                                          |  10%[1] "iteration:  2"
## [1] 140   6
## [1] 10  6
## [1] 10  1
## [1] 18  1
## [1] 28  1
## [1] 28  1
##
  |
  |=============                                                   |  20%[1] "iteration:  3"
## [1] 139   6
## [1] 11  6
## [1] 11  1
## [1] 28  1
## [1] 39  1
## [1] 39  1
```

```
##
  |
  |====================                                             |  30%[1] "iteration:  4"
## [1] 137   6
## [1] 13  6
## [1] 13  1
## [1] 39  1
## [1] 52  1
## [1] 52  1
##
  |
  |==========================                                       |  40%[1] "iteration:  5"
## [1] 132   6
## [1] 18  6
## [1] 18  1
## [1] 52  1
## [1] 70  1
## [1] 70  1
##
  |
  |================================                                 |  50%[1] "iteration:  6"
## [1] 139   6
## [1] 11  6
## [1] 11  1
## [1] 70  1
## [1] 81  1
## [1] 81  1
##
  |
  |=======================================                          |  60%[1] "iteration:  7"
## [1] 133   6
## [1] 17  6
## [1] 17  1
## [1] 81  1
## [1] 98  1
## [1] 98  1
##
  |
  |==============================================                   |  70%[1] "iteration:  8"
## [1] 134   6
## [1] 16  6
## [1] 16  1
## [1] 98  1
## [1] 114   1
## [1] 114   1
##
  |
  |====================================================             |  80%[1] "iteration:  9"
## [1] 134   6
## [1] 16  6
## [1] 16  1
## [1] 114   1
## [1] 130   1
## [1] 130   1
```

```
##
  |
  |============================================================       |  90%[1] "iteration:  10"
## [1] 130   6
## [1] 20  6
## [1] 20  1
## [1] 130   1
## [1] 150   1
## [1] 150   1
##
  |
  |===================================================================| 100%
```

Add predictions and actual Sepal Length values

```
result <- cbind(prediction, testsetCopy[, 1])
names(result) <- c("Predicted", "Actual")
result$Difference <- abs(result$Actual - result$Predicted)
```

Mean Absolute Error as Evalution

```
summary(result$Difference)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.001308 0.091023 0.248860 0.304007 0.469896 1.370149
```