7.2

Joseph Elikishvili

December 11, 2017

### 7.2

#### Chapter 7 talks about non linear regressions. It goes over several models. Specifically MARS, SVM and K-NN. Exercise 7.2 goes over evaluating different models and comparing the results to better understand how to chose the appropriate model for a specific situation.

Friedman (1991) introduced several benchmark data sets create by simulation. One of these simulations used the following nonlinear equation to create data: y = 10 sin(??x1x2) + 20(x3 ??? 0.5)2 + 10x4 + 5x5 + N(0, ??2) where the x values are random variables uniformly distributed between [0, 1] (there are also 5 other non-informative variables also created in the simulation). The package mlbench contains a function called mlbench.friedman1 that

We will preload the necessary libraries and generate the training and test data, then try the data on KNN, MARS and SVM and finally we will compare the results.

library(mlbench)  
set.seed(200)  
trainingData <- mlbench.friedman1(200, sd = 1)  
## We convert the 'x' data from a matrix to a data frame  
## One reason is that this will give the columns names.  
trainingData$x <- data.frame(trainingData$x)  
## Look at the data using  
#featurePlot(trainingData$x, trainingData$y)  
## or other methods.

## This creates a list with a vector 'y' and a matrix  
 ## of predictors 'x'. Also simulate a large test set to  
 ## estimate the true error rate with good precision:  
 testData <- mlbench.friedman1(5000, sd = 1)  
 testData$x <- data.frame(testData$x)

### KNN

First we will use KNN model and view the results.

library(caret)  
knnModel <- train(x = trainingData$x, y = trainingData$y, method = "knn", preProc = c("center", "scale"), tuneLength = 10)  
knnModel

## k-Nearest Neighbors   
##   
## 200 samples  
## 10 predictor  
##   
## Pre-processing: centered (10), scaled (10)   
## Resampling: Bootstrapped (25 reps)   
## Summary of sample sizes: 200, 200, 200, 200, 200, 200, ...   
## Resampling results across tuning parameters:  
##   
## k RMSE Rsquared MAE   
## 5 3.565620 0.4887976 2.886629  
## 7 3.422420 0.5300524 2.752964  
## 9 3.368072 0.5536927 2.715310  
## 11 3.323010 0.5779056 2.669375  
## 13 3.275835 0.6030846 2.628663  
## 15 3.261864 0.6163510 2.621192  
## 17 3.261973 0.6267032 2.616956  
## 19 3.286299 0.6281075 2.640585  
## 21 3.280950 0.6390386 2.643807  
## 23 3.292397 0.6440392 2.656080  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was k = 15.

Next we will use the trained model to make a prediction and review the RMSE

knnPred <- predict(knnModel, newdata = testData$x)  
## The function 'postResample' can be used to get the test set  
## perforamnce values  
postResample(pred = knnPred, obs = testData$y)

## RMSE Rsquared MAE   
## 3.1750657 0.6785946 2.5443169

We get RMSE = 3.17

### MARS

Next we will take a look at MARS model

set.seed(101)   
mars\_grid = expand.grid(degree =1:2, nprune=seq(2,14,by=2))  
marsTune = train(x= trainingData$x, y=trainingData$y, method='earth', tuneGrid=mars\_grid, trControl = trainControl(method = "cv"))  
marsTune

## Multivariate Adaptive Regression Spline   
##   
## 200 samples  
## 10 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 180, 180, 180, 180, 180, 180, ...   
## Resampling results across tuning parameters:  
##   
## degree nprune RMSE Rsquared MAE   
## 1 2 4.417014 0.2350732 3.656244  
## 1 4 2.606487 0.7293074 2.069700  
## 1 6 2.192319 0.8094378 1.742533  
## 1 8 1.700573 0.8889867 1.313446  
## 1 10 1.640852 0.8967593 1.277871  
## 1 12 1.641261 0.8950167 1.297462  
## 1 14 1.642807 0.8964578 1.287479  
## 2 2 4.417014 0.2350732 3.656244  
## 2 4 2.606487 0.7293074 2.069700  
## 2 6 2.218192 0.8021446 1.729223  
## 2 8 1.750853 0.8789636 1.370531  
## 2 10 1.449480 0.9177568 1.111537  
## 2 12 1.316041 0.9299167 1.026115  
## 2 14 1.321235 0.9300051 1.048692  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final values used for the model were nprune = 12 and degree = 2.

We will review the Variable importance

#Variable Importance  
varImp(marsTune)

## earth variable importance  
##   
## Overall  
## X1 100.00  
## X4 85.05  
## X2 69.03  
## X5 48.88  
## X3 39.40  
## X10 0.00  
## X6 0.00  
## X9 0.00  
## X7 0.00  
## X8 0.00

It appears MARS is only using x1-x5 to make a prediction.

Next we will make the prediction using MARS and review the results.

mars\_pred = predict (marsTune, testData$x)  
postResample(pred = mars\_pred, obs = testData$y)

## RMSE Rsquared MAE   
## 1.2803060 0.9335241 1.0168673

We get RMSE = 1.28 which is a definite improvement from KNN

### SVM

Next we will use SVM model

set.seed(102)  
svmTune = train(x= trainingData$x, y=trainingData$y, method='svmRadial', tuneLength = 14, trControl = trainControl(method = "cv"))  
svmTune$finalModel

## Support Vector Machine object of class "ksvm"   
##   
## SV type: eps-svr (regression)   
## parameter : epsilon = 0.1 cost C = 4   
##   
## Gaussian Radial Basis kernel function.   
## Hyperparameter : sigma = 1   
##   
## Number of Support Vectors : 186   
##   
## Objective Function Value : -82.2151   
## Training error : 0.009563

Lets review the variable importance

varImp(svmTune)

## loess r-squared variable importance  
##   
## Overall  
## X4 100.0000  
## X1 95.5047  
## X2 89.6186  
## X5 45.2170  
## X3 29.9330  
## X9 6.3299  
## X10 5.5182  
## X8 3.2527  
## X6 0.8884  
## X7 0.0000

Looks like x1-x5 are still the most important variables although in this case only x7 is not used, Also it appears x7 is the most important variable for SVM, although x7 was not even used by MARS

Lets review the results

svmPred <- predict(svmTune, newdata = testData$x)  
postResample(pred = svmPred, obs = testData$y)

## RMSE Rsquared MAE   
## 4.8618512 0.2038278 3.9582613

We get RMSE = 4.86

### Conclusion

We conclude that MARS is the most accurate of the 3 models in this particular case. It resulted in the lowest RMSE of 1.28

### 7.5. Exercise 6.3 describes data for a chemical manufacturing process. Use the same data imputation, data splitting, and pre-processing steps as before and train several nonlinear regression models.

We will import imputation and data splitting code from 6.3 below.

library(AppliedPredictiveModeling)  
library(caret)  
  
data(ChemicalManufacturingProcess)  
df = ChemicalManufacturingProcess  
#summary(df)  
  
library(missForest)  
df\_imp1 = missForest(df)

## missForest iteration 1 in progress...done!  
## missForest iteration 2 in progress...done!  
## missForest iteration 3 in progress...done!  
## missForest iteration 4 in progress...done!

df\_imp = df\_imp1$ximp  
  
data = df\_imp[,2:58]  
target = df\_imp[,1]  
  
training = createDataPartition( target, p=0.75 )  
predictor\_training = data[training$Resample1,]  
target\_training = target[training$Resample]  
  
predictor\_testing = data[-training$Resample1,]  
target\_testing = target[-training$Resample1]

### KNN

We will start with KNN model and will measure the performance of the model.

knnModel\_75 <- train(x = predictor\_training, y = target\_training, method = "knn", preProc = c("center", "scale"), tuneLength = 10)  
knnModel\_75

## k-Nearest Neighbors   
##   
## 132 samples  
## 57 predictor  
##   
## Pre-processing: centered (57), scaled (57)   
## Resampling: Bootstrapped (25 reps)   
## Summary of sample sizes: 132, 132, 132, 132, 132, 132, ...   
## Resampling results across tuning parameters:  
##   
## k RMSE Rsquared MAE   
## 5 1.384146 0.3768763 1.078726  
## 7 1.387710 0.3669445 1.109488  
## 9 1.413321 0.3455230 1.141322  
## 11 1.429472 0.3305082 1.160532  
## 13 1.440057 0.3194528 1.172922  
## 15 1.444468 0.3156153 1.177875  
## 17 1.451980 0.3073148 1.189475  
## 19 1.454195 0.3029005 1.191154  
## 21 1.449979 0.3093282 1.187930  
## 23 1.448196 0.3127738 1.188744  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was k = 5.

Looks like we get the best RMSE metrics with n=5. We will use the model to make the prediction.

knnPred\_75 <- predict(knnModel\_75, newdata = predictor\_testing)  
postResample(pred = knnPred\_75, obs = target\_testing)

## RMSE Rsquared MAE   
## 1.5495677 0.4973659 1.2286818

KNN based predictions have RMSE of 1.26.

#### Next we will use MARS model and measure the predictive performance.

mars\_grid = expand.grid(degree =1:2, nprune=seq(2,14,by=2))  
marsTune\_75 = train(x = predictor\_training, y = target\_training, method='earth', tuneGrid=mars\_grid, trControl = trainControl(method = "cv"))  
marsTune\_75

## Multivariate Adaptive Regression Spline   
##   
## 132 samples  
## 57 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 120, 119, 119, 119, 118, 118, ...   
## Resampling results across tuning parameters:  
##   
## degree nprune RMSE Rsquared MAE   
## 1 2 1.388833 0.4162738 1.0937821  
## 1 4 1.168918 0.5991866 0.9458691  
## 1 6 1.265335 0.5485218 1.0050530  
## 1 8 1.336531 0.4986164 1.0847822  
## 1 10 1.217123 0.5945703 0.9454200  
## 1 12 1.209208 0.5854100 0.9321785  
## 1 14 1.198510 0.5961953 0.9504527  
## 2 2 1.389164 0.4160174 1.0940570  
## 2 4 1.192578 0.5690754 0.9631306  
## 2 6 1.231714 0.5424268 0.9690872  
## 2 8 1.233921 0.5448525 0.9706489  
## 2 10 1.226324 0.5563709 0.9701000  
## 2 12 1.281809 0.5263668 0.9913089  
## 2 14 1.246238 0.5583326 0.9787434  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final values used for the model were nprune = 4 and degree = 1.

We will review Variable importance

Next we will make the prediction using MARS and review the results.

marspred\_75 = predict (marsTune\_75, predictor\_testing)  
postResample(pred = marspred\_75, obs = target\_testing)

## RMSE Rsquared MAE   
## 1.1724850 0.7060344 0.9081153

We have RMSE of 1.17 which is a slight improvement over KNN model.

#### Next we will use SVM model to make the prediction on the data set.

svmTune\_75 = train(x = predictor\_training, y = target\_training, method='svmRadial', tuneLength = 14, trControl = trainControl(method = "cv"))  
svmTune\_75

## Support Vector Machines with Radial Basis Function Kernel   
##   
## 132 samples  
## 57 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 118, 119, 120, 119, 117, 119, ...   
## Resampling results across tuning parameters:  
##   
## C RMSE Rsquared MAE   
## 0.25 1.749680 0.04943935 1.430692  
## 0.50 1.746559 0.05023220 1.432773  
## 1.00 1.743879 0.05216393 1.436458  
## 2.00 1.744816 0.05261132 1.443579  
## 4.00 1.744883 0.05257954 1.442838  
## 8.00 1.744883 0.05257954 1.442838  
## 16.00 1.744883 0.05257954 1.442838  
## 32.00 1.744883 0.05257954 1.442838  
## 64.00 1.744883 0.05257954 1.442838  
## 128.00 1.744883 0.05257954 1.442838  
## 256.00 1.744883 0.05257954 1.442838  
## 512.00 1.744883 0.05257954 1.442838  
## 1024.00 1.744883 0.05257954 1.442838  
## 2048.00 1.744883 0.05257954 1.442838  
##   
## Tuning parameter 'sigma' was held constant at a value of 0.01305284  
## RMSE was used to select the optimal model using the smallest value.  
## The final values used for the model were sigma = 0.01305284 and C = 1.

Lets review the results

svmPred\_75 <- predict(svmTune\_75, newdata = predictor\_testing)  
postResample(pred = svmPred\_75, obs = target\_testing)

## RMSE Rsquared MAE   
## 2.08911441 0.03096715 1.65214815

We get an RMSE of 2.09 with SVM which is slightly worse then the previous models.

### A. Which nonlinear regression model gives the optimal resampling and test set performance?

Out of 3 non linear models we used, MARS has the best accuracy with RMSE of 1.17

### B. Which predictors are most important in the optimal nonlinear regression model? Do either the biological or process variables dominate the list? How do the top ten important predictors compare to the top ten predictors from the optimal linear model?

Lets review predictor importance of MARS and KNN models as they have the best RSME scores.

#### KNN top predictors

#plot(varImp(knnModel))

#### MARS top predictors

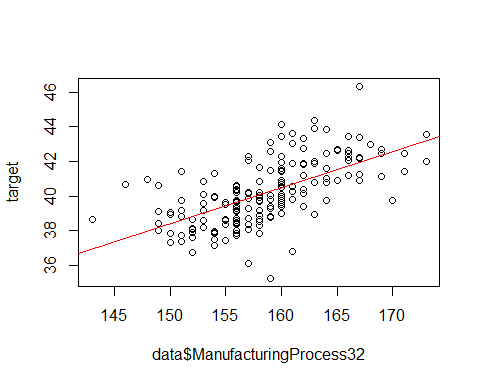
#plot(varImp(marsTune))

Interestingly enough we see pretty much same predictors selected by both models. Knn shows same top picks as PLS and MARS model although only selects top 2 predictos, still has the same top 2 as PLS. ManufacturingProcess32 and manifacturingprocess09 remain top predictors and interestingly enough MARS was able to predict using only those 2 predictors and had comparable results to PLS, which means we could very well need only 2 predictors to capture most of the predictions.

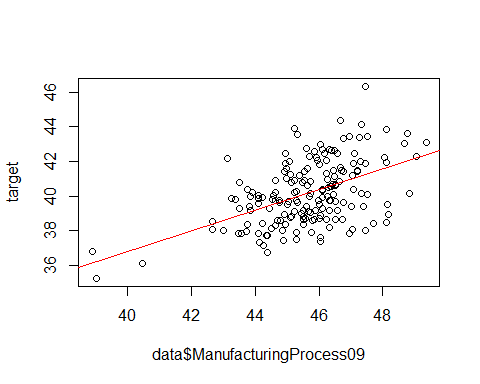
#### C. Explore the relationships between the top predictors and the response for the predictors that are unique to the optimal nonlinear regression model. Do these plots reveal intuition about the biological or process predictors and their relationship with yield?

We have the same predictors as PLS, so we will use those

plot(data$ManufacturingProcess32, target)  
abline(lm(target~data$ManufacturingProcess32),col="red",lwd=1.5)



plot(data$ManufacturingProcess09, target)  
abline(lm(target~data$ManufacturingProcess09),col="red",lwd=1.5)



Both predictors have positive corelation. Lets check the exact corelation

cor(data$ManufacturingProcess32, target)

## [1] 0.6083321

cor(data$ManufacturingProcess09, target)

## [1] 0.5034705

### Conclusion

It appears that we have come to the same conclusion using non-linear models, which is a good indicator that PLS was not the worst choice for this task, We also identified that most of the predictions are coming from 2 top predictors and therefore any potential optimization work should be directed at those 2 predictors.