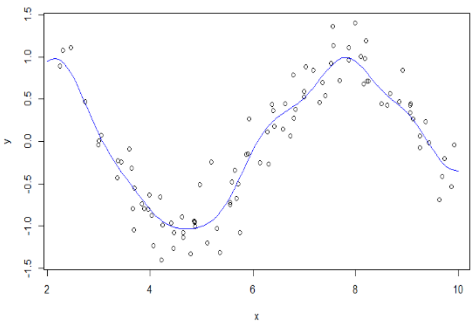
MA4790

Homework 4

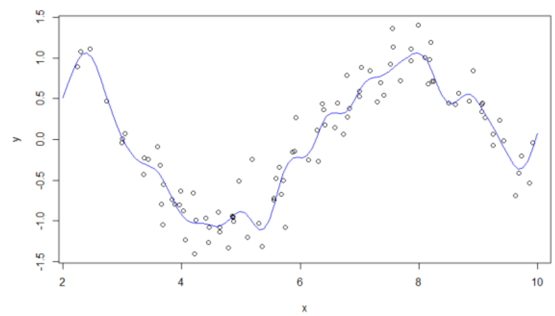
Ian Boulis

7.1 A. Fit different models using a radial basis function and different values of the cost (the C parameter) and ε. Plot the fitted curve.

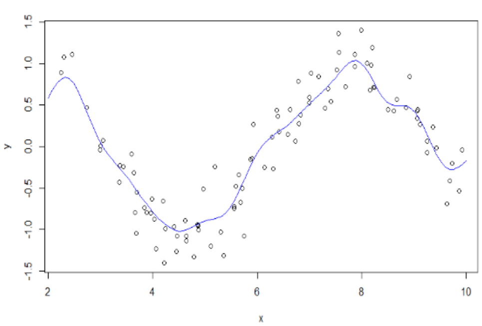
C = 1, epsilon = 0.1



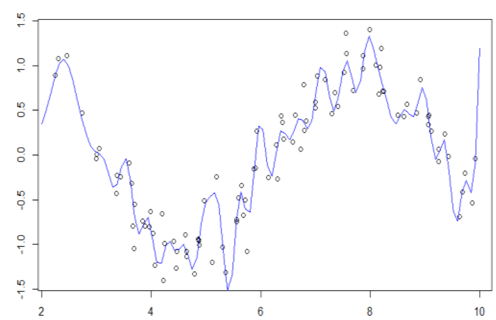
C = 25, epsilon = 0.1



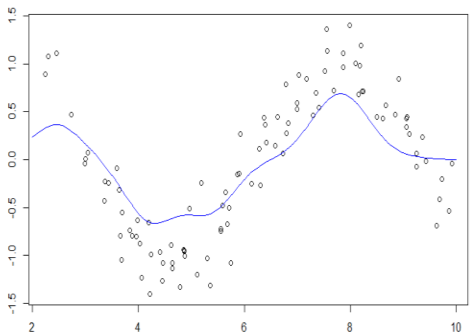
C = 0.5, epsilon = 0.1



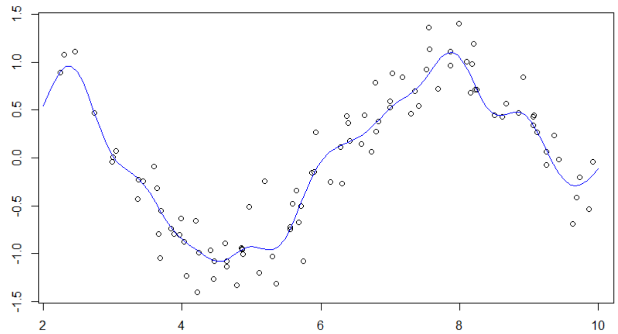
C = 200, epsilon = 0.1



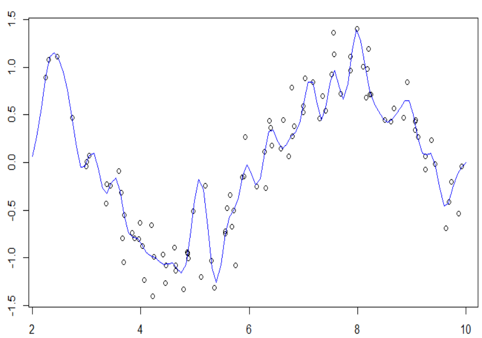
C= 1, epsilon = 1



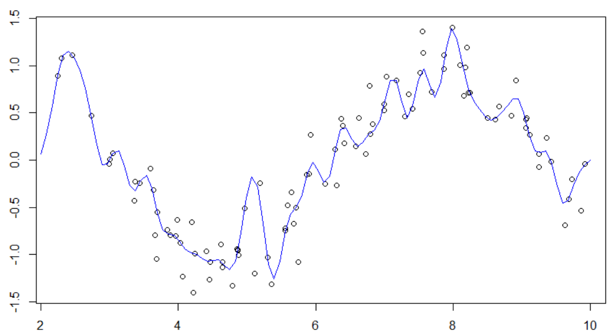
C = 1, epsilon = 0.01



C = 10, epsilon = 0.01



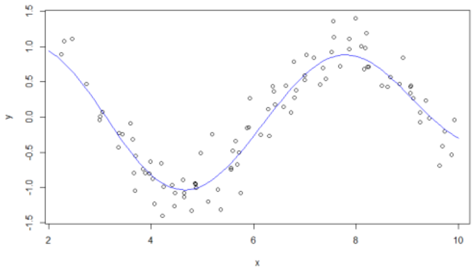
C = 100, epsilon = 0.5



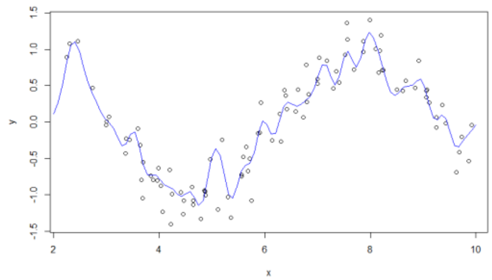
It appears to be that as cost values increase, the model gets more jagged. Along with as epsilon decreases, the model gets more jagged. This could be from R trying to minimalize residuals as much as possible and as a result the model is overfitting at higher cost values and lower epsilon values.

B. The σ parameter can be adjusted using the kpar argument, such as kpar = list(sigma = 1). Try different values of σ to understand how this parameter changes the model fit. How does the cost, ε, and σ values affect the model?

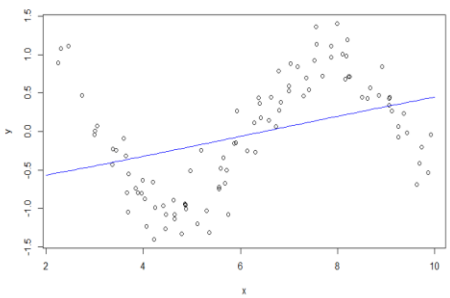
Sigma = 1



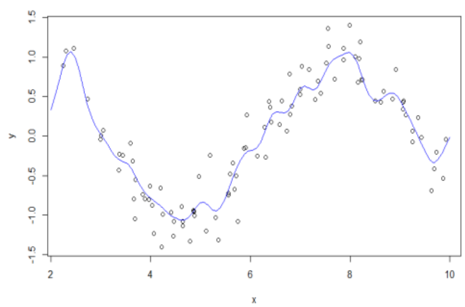
Sigma = 100



Sigma = 0.01



Sigma = 40



It appears that for higher values of sigma the model becomes more jagged, and for lower values it appears to become smoother (even nearly a straight line in the case for sigma = 0.01). Which could mean that for lower values of sigma, the model is underfitting, and conversely for higher values the model is overfitting.

7.2 A. Consider KNN and MARS, which model appears to give the best performance?

Here are the model summaries, as given by the question statement:

```
RMSE Rsquared
3.2286834 0.6871735
```

And here are the model summaries from the MARS model output:

```
> postResample(pred=mPred, obs = testData$y)
     RMSE  Rsquared       MAE
1.8117424 0.8674877 1.3894754
```

Based off information solely from the outputs, the MARS model predicts significantly better. The MARS model has nearly twice the accuracy, with RMSE nearly halving between the two models, and while not as significant the R^2 value increased greatly. Looking outside of the information presented from R, I do not believe the data from the question statement is "set up" very well for a KNN model to outperform various other models. The MARS model gives the better performance.

B. Does MARS select the informative predictors?

This can be easily seen using the varImp function:

```
   Overall
X1  100.00
X4   82.78
X2   64.18
X5   40.21
X3   28.14
X6    0.00
```

The variables X1-X5 are all deemed important in the MARS model, with the remaining variables being zero, or not even showing up in the varImp function. The MARS model does select the informative predictors.

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

Model	Parameters	Root Mean Squared Error
Neural Network	Decay = 0.01	2.010877
Multivariate Adaptive Regression Splines	Nprune = 4 Degree = 2	1.291151
Support Vector Machine	σ = 0.01476372 C = 2	1.188678
K-Nearest Neighbors	K = 5	1.318348

```
Summary of sample sizes: 144, 144, 144, 144, 144, 144, ...
Resampling results across tuning parameters:

  degree  nprune  RMSE      Rsquared   MAE
  1        2      1.553731  0.3788301  1.219976
  1        4      1.454893  0.4920483  1.109310
  1        6      1.512286  0.4627197  1.140674
  1        8      1.550068  0.4485466  1.163790
  1       10      1.645747  0.4321644  1.227985
  1       12      1.884280  0.3805430  1.321740
  1       14      1.945506  0.3777421  1.351101
  1       16      1.948224  0.3768974  1.373473
  1       18      1.964706  0.3748162  1.375178
  1       20      1.957314  0.3771580  1.383809
  2        2      1.559449  0.3675661  1.228820
  2        4      1.291151  0.5656666  1.045678
  2        6      1.395620  0.5107648  1.100235
  2        8      1.392398  0.5227338  1.098944
  2       10      2.093359  0.4403609  1.316996
  2       12      2.346349  0.4185885  1.405435
  2       14      2.406405  0.4178158  1.423316
  2       16      2.647800  0.4010090  1.534583
  2       18      3.293006  0.3603686  1.645092
  2       20      3.144907  0.3540336  1.617087
  3        2      1.563364  0.3601449  1.236325
  3        4      1.365394  0.5142975  1.090141
  3        6      1.414028  0.4995197  1.099302
  3        8      1.355548  0.5405447  1.061740
  3       10      1.460169  0.5147455  1.108351
  3       12      1.553291  0.4820714  1.147303
  3       14      1.662883  0.4600609  1.208669
  3       16      1.774241  0.4388089  1.251695
  3       18      1.926128  0.4271897  1.311022
  3       20      2.151185  0.4006714  1.428667
  4        2      1.559449  0.3675661  1.228820
  4        4      1.391531  0.5049802  1.100165
  4        6      1.424494  0.5089061  1.085703
  4        8      1.493936  0.4889091  1.124027
  4       10      1.576366  0.4748988  1.155194
  4       12      1.638321  0.4732832  1.170935
  4       14      1.753416  0.4439901  1.240316
  4       16      1.855013  0.4244560  1.311751
  4       18      1.988902  0.3917836  1.369496
  4       20      2.351843  0.3458526  1.510616
  5        2      1.559449  0.3675661  1.228820
  5        4      1.391531  0.5049802  1.100165
  5        6      1.424494  0.5089061  1.085703
  5        8      1.493936  0.4889091  1.124027
  5       10      1.606508  0.4663672  1.172291
  5       12      1.705306  0.4504152  1.204845
  5       14      1.749842  0.4448904  1.239983
  5       16      1.863332  0.4258514  1.316626
  5       18      1.985335  0.3903978  1.363101
  5       20      2.338214  0.3516136  1.504393

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were nprune = 4 and degree = 2.
```

```
k-Nearest Neighbors

144 samples
 48 predictor

No pre-processing
Resampling: Bootstrapped (10 reps)
Summary of sample sizes: 144, 144, 144, 144, 144, 144, ...
Resampling results across tuning parameters:

  k   RMSE      Rsquared   MAE
   5  1.318348  0.4723569  1.027885
   7  1.369227  0.4298315  1.096468
   9  1.373588  0.4265968  1.113131
  11  1.384309  0.4172377  1.116934
  13  1.377304  0.4262089  1.119377
  15  1.390814  0.4172017  1.133931
  17  1.408763  0.4055506  1.151707
  19  1.412637  0.4055853  1.149760
  21  1.403181  0.4228085  1.143173
  23  1.414250  0.4179682  1.152148

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was k = 5.
```

```
Support Vector Machines with Radial Basis Function Kernel

144 samples
 47 predictor

No pre-processing
Resampling: Bootstrapped (10 reps)
Summary of sample sizes: 144, 144, 144, 144, 144, 144, ...
Resampling results across tuning parameters:

  C        RMSE      Rsquared   MAE
     0.25  1.329745  0.5068909  1.0623176
     0.50  1.262548  0.5308867  1.0017334
     1.00  1.213730  0.5525226  0.9550785
     2.00  1.188678  0.5702333  0.9280873
     4.00  1.199701  0.5623368  0.9242983
     8.00  1.208211  0.5575732  0.9302723
    16.00  1.208472  0.5573496  0.9305917
    32.00  1.208472  0.5573496  0.9305917
    64.00  1.208472  0.5573496  0.9305917
   128.00  1.208472  0.5573496  0.9305917
   256.00  1.208472  0.5573496  0.9305917
   512.00  1.208472  0.5573496  0.9305917
  1024.00  1.208472  0.5573496  0.9305917
  2048.00  1.208472  0.5573496  0.9305917

Tuning parameter 'sigma' was held constant at a value of 0.01476372
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were sigma = 0.01476372 and C = 2.
```

From the results of R and the table, the optimal resampling and test performance is from the support vector machine. It produced a relatively low RMSE value and the best R^2 value from any of the models. These values did not change much between the training and test set, which implies that the training set is accurate to the test set.

B. Which predictors are most important in the optimal nonlinear regression model?

Output from the varImp function:

```
                       Overall
ManufacturingProcess32  100.00
ManufacturingProcess13   92.18
BiologicalMaterial06     85.55
ManufacturingProcess36   84.94
BiologicalMaterial03     78.34
ManufacturingProcess17   73.15
ManufacturingProcess09   70.54
ManufacturingProcess06   51.29
ManufacturingProcess33   50.71
ManufacturingProcess02   45.85
BiologicalMaterial01     43.97
BiologicalMaterial04     42.24
ManufacturingProcess11   40.27
ManufacturingProcess12   40.16
BiologicalMaterial11     37.05
BiologicalMaterial08     36.21
ManufacturingProcess30   31.14
BiologicalMaterial09     29.12
ManufacturingProcess20   26.04
ManufacturingProcess01   22.17
```

Shows that mostly manufacturing processes are the most important for the nonlinear model, with process 32 being at the top of the list. Looking back at the most important predictors from the linear model there is some crossover in the top ten list, however, they are in a different order between the two models.

R Code

```
##7.1##

set.seed(100)

x <- runif(100, min=2,max=10)

y <- sin(x) + rnorm(length(x))* .25

sinData <- data.frame(x=x,y=y)

plot(x,y)

dataGrid <- data.frame(x=seq(2,10, length=100))


##A##B##

library(kernlab)

rbfSVM <- ksvm(x = x, y = y, data = sinData, kernel ="rbfdot", kpar =  list(sigma = 40), C = 1,
epsilon = 0.1)

modelPrediction <- predict(rbfSVM, newdata = dataGrid)

points(x = dataGrid$x, y = modelPrediction[,1], type = "l", col = "blue")


##7.2##

library(mlbench)

library(caret)
```

```
set.seed(200)

trainingData <- mlbench.friedman1(200, sd = 1)

trainingData$x <- data.frame(trainingData$x)

featurePlot(trainingData$x, trainingData$y)

testData <- mlbench.friedman1(5000, sd = 1)

testData$x <- data.frame(testData$x)

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

knnModel

knnPred <- predict(knnModel, newdata=testData$x)

postResample(pred=knnPred, obs=testData$y)


##A##

mars <- train(x=trainingData$x, y=trainingData$y, method = "earth", preProc =
c("center","scale"), tuneLength=10)

mPred <- predict(mars, newdata = testData$x)

postResample(pred=mPred, obs = testData$y)


##B##
```

```
varImp(mars)

##7.5##

library(AppliedPredictiveModeling)

data("ChemicalManufacturingProcess")

yield <- subset(ChemicalManufacturingProcess, select = "Yield")

predict <- subset(ChemicalManufacturingProcess, select = -Yield)

partition <- createDataPartition(yield$Yield, p=4/5, list = FALSE)

trP <- predict[partition,]

trY <- yield[partition,]

teP <- predict[-partition,]

teY <- yield[-partition, ]

pre <- preProcess(trP, method =c("BoxCox","center","scale","knnImpute"))

trP <- predict(pre, trP)

teP <- predict(pre, teP)

nzv <- nearZeroVar(trP)

trP <- trP[-nzv]

teP <- teP[-nzv]
```

```
cor <- cor(trP)

hc <- findCorrelation(cor)

trP <- trP[, -hc]

teP <- teP[, -hc]

##First 20 or so lines are all taken from 6.3 homework, just to preprocess the data##

##A##

ctrl <- trainControl(method = "boot", number= 10)

##Neural Network##

nnetGrid <- expand.grid(.decay = c(0, 0.01, .1), .size = c(1:10), .bag = FALSE)

nnett <- train(trP, trY, method="avNNet", tunegrid=nnetGrid, trControl=ctrl,preProc =
c("center", "scale"), linout = TRUE, trace = FALSE, MaxNWts = 10 * (ncol(trainXnnet) + 1) +
10 + 1, maxit = 500)

nnet <- nnet(trP, trY,

        size = 5,

        decay = 0.01,

        linout = TRUE,

        trace = FALSE,

        maxit = 500)

nnet
```

```
pred <- predict(nnet, teP)

SSEnnet <- mean((teY-pred)^2)

RSSE <- sqrt(SSEnnet)

Rsquared=(cor(teY, pred))^2

RSSE

Rsquared

##MARS##

marsGrid <- expand.grid(degree = c(1:5), nprune = (1:10) * 2)

mars <- train(trP, trY, method="earth", tuneGrid =marsGrid, trControl=ctrl)

mars

##KNN##

knn <- train(trP, trY, method="knn", tuneLength=10, trControl=ctrl)

knn

##Support Vector Machine##

SVM <- train(trP, trY, method="svmRadial",tuneLength=14, trControl=ctrl)

SVM



##B##
```

varImp(SVM)