Single Feature Kernel Regression

Jacob Kramp

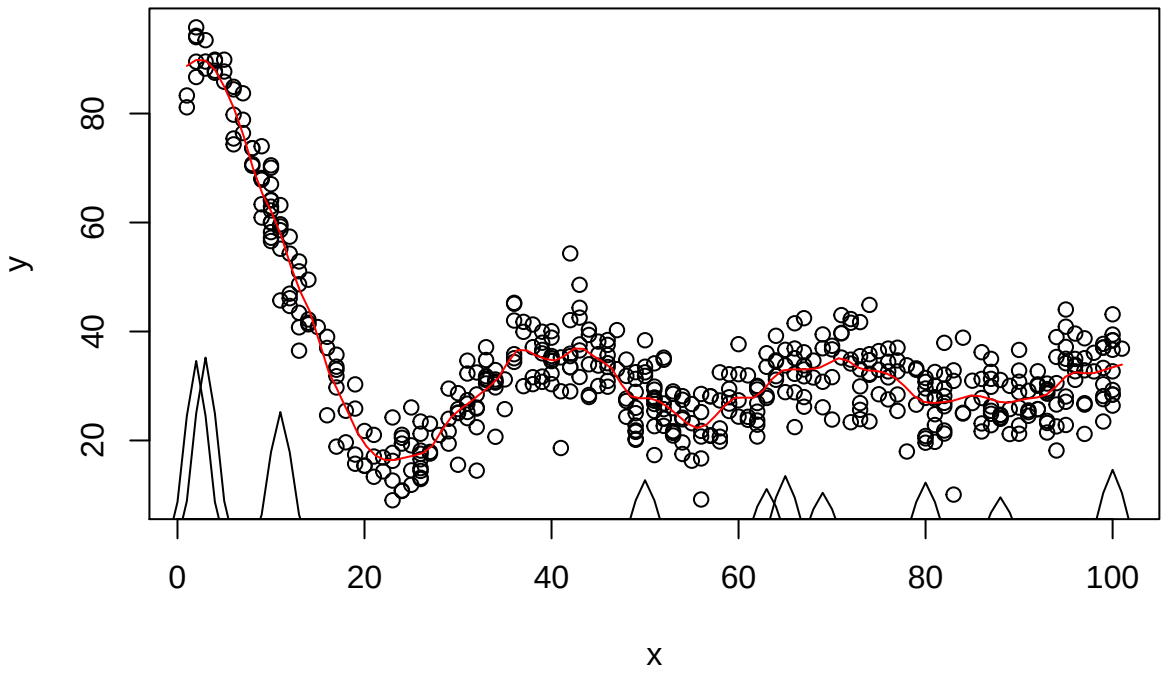
3/14/2021

Introduction

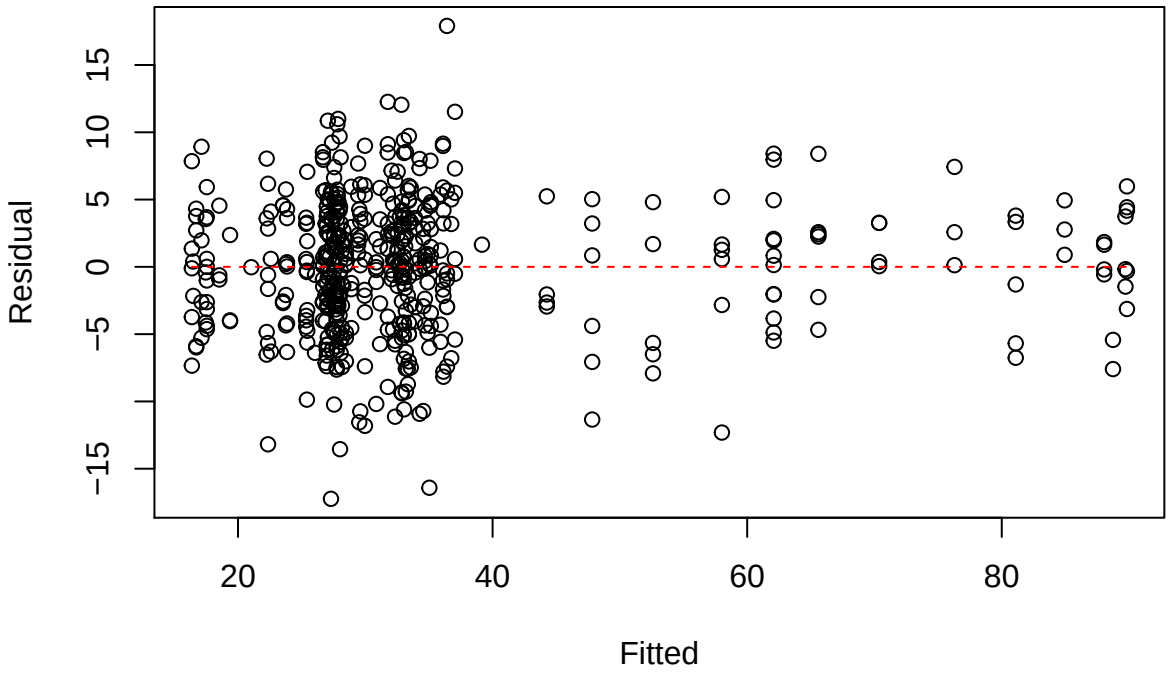
We have been asked to estimate the average cost in having a certain number of employees within a given company.They have specifically (hypothetically) asked for predictions of 1,20, and 120 employees. We will be using a bootstrap kernel regression method to estimate predictions and their associated confidence intervals.

Model Fit

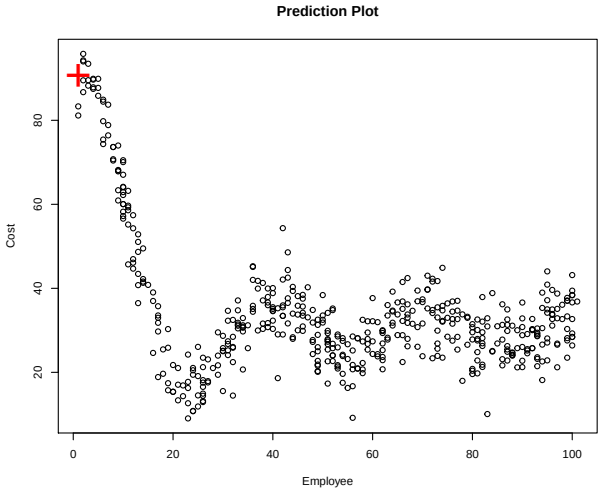
The result of the kernel regression fit is shown below:

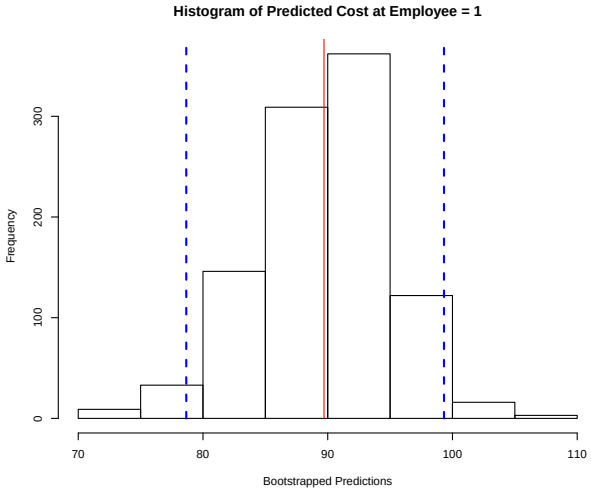


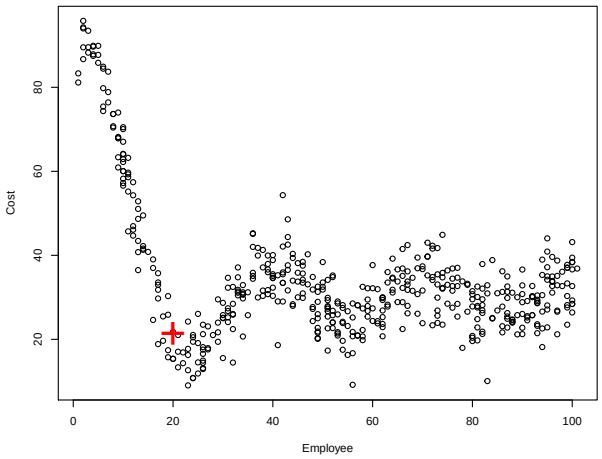
This model looks as though it is fitting our data well and we've chosen an optimal bandwidth for our kernel regression model to minimize our sum of squared residuals.(The bandwidth in kernel regression is a value which dictates the width of the Gaussian kernels. Choosing an optimal bandwidth is key to our estimation process.) Now we need to see if we have constant variance in our residuals in order to be sure that we're not breaking any assumptions needed for kernel regression to be a decent model to make predictions with our data.

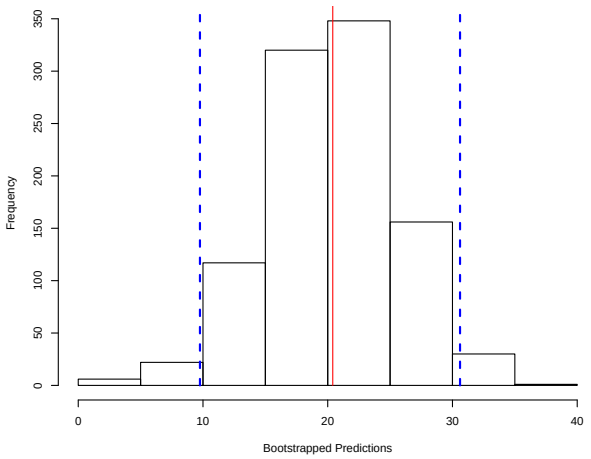


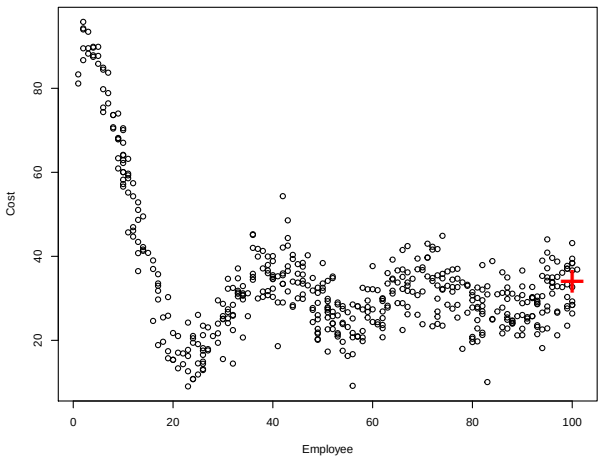
It seems as though we have constant variance as the residuals are centered around 0 for each of our fitted points. Now that we have a best fit model with no broken assumptions, we can make prediction estimates for the requested distance measurements. The below plots will show a red '+' for where each of our predictions lie and their associated confidence intervals shown in a histogram.

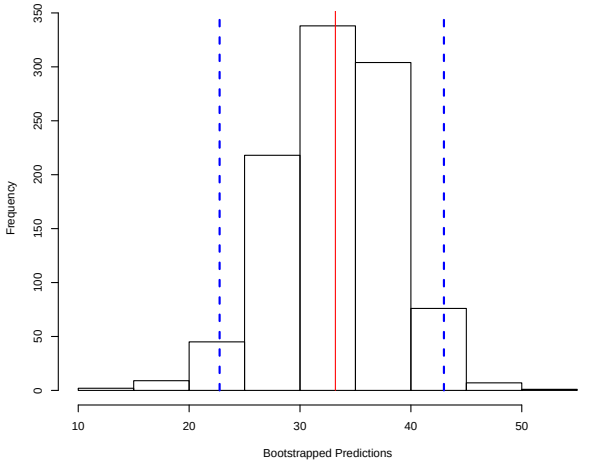

Prediction Plot
Cost
Employee
0
20
40
60
80
100
20
40
60
80


Histogram of Predicted Cost at Employee = 1
Frequency
Bootstrapped Predictions
0
100
200
300
70
80
90
100
110


Prediction Plot
Cost
Employee
0
20
40
60
80
100
20
40
60
80


Histogram of Predicted Cost at Employee = 20
Frequency
Bootstrapped Predictions
0
50
100
150
200
250
300
350
0
10
20
30
40


Prediction Plot
Cost
Employee
0
20
40
60
80
100
20
40
60
80


Histogram of Predicted Cost at Employee = 100
Frequency
Bootstrapped Predictions
0
50
100
150
200
250
300
350
10
20
30
40
50

The exact values of our predictions are shown in the following table as well.

Employee	Cost	lower	upper
1	89.70183	78.660871	99.32351
20	20.40366	9.756242	30.60685
100	33.18960	22.749279	42.97950

Results and Conclusion

We were able to make predictions for an employee count of 1, 20, and 80 (reference table above). We were unable to make a prediction for 200 employees because we only have data for up to 101 employees and we can't make a prediction outside of our data's domain. We would need to collect more data for that number of employees and the behavior of the cost in order to begin forming models beyond our current domain. We included a prediction for 100 employees because that is the furthest value for which can make a reliable estimation for error and the closest estimation we can make to 100 employees.

Index (code)

```
library(data.table)
library(tidyverse)

mydata <- data.frame(fread('Cost per employee.csv'))
attach(mydata)

x <- employee
y <- cost
#Use leave one out cross validation and sum of squared residuals to find the optimal
#bandwidth of our Gaussian kernals
SSR.KR = function(h){
  x1 = x
  y1 = rep(0,length(x1))
  for(i in 1:length(x1)){
    y1[i] = sum((1/(sqrt(2*pi)))*exp(-.5*((x1[i]-x[-i])/h)^2)*y[-i])/
            sum((1/(sqrt(2*pi)))*exp(-.5*((x1[i]-x[-i])/h)^2))
  }
  res = y - y1
  SSR = sum(res^2)
  SSR
}

h = optim(2,SSR.KR)$par


#Now we can run our code again with optimal h

plot(x,y,xlim=c(min(x),max(x)),ylim=c(min(y),max(y)))
lines(c(min(x)-15,max(x)+15),c(0,0),lty=2)
```

```r
###THIS CODE IS JUST FOR PLOTTING####

for(i in 1:1){
  x1 = seq(x[i]-40,x[i]+40,by=1)
  y1 = (1/(sqrt(2*pi)))*exp(-.5*((x[i]-x1)/h)^2)
  lines(x1,y1*y[i])
}

for(i in 2:10){
  x1 = seq(x[i]-40,x[i]+40,by=1)
  y1 = (1/(sqrt(2*pi)))*exp(-.5*((x[i]-x1)/h)^2)
  lines(x1,y1*y[i])
}

###This code is for the actual kernel regression/smoothing.
#Again ,x1 represents the values at which we want to make a prediction or compute E(y|x).
#For now I'm just making 100 predictions evenly spaced out over
#out domain so I can plot the results.
#y1 represents E(y|x1)

x1 = seq(min(x),max(x),by=(max(x)-min(x))/99)
y1 = rep(0,100)
for(i in 1:100){
  y1[i] = sum((1/(sqrt(2*pi)))*exp(-.5*((x1[i]-x)/h)^2)*y)/
          sum((1/(sqrt(2*pi)))*exp(-.5*((x1[i]-x)/h)^2))
}

lines(x1,y1,col=2)

#Gather original residuals and check for constant variance
res.orig <- lapply(1:length(x),function(i){
  #Make prediction for each x in dataset
  yhat <- sum((1/(sqrt(2*pi)))*exp(-.5*((x[i]-x)/h)^2)*y)/sum((1/(sqrt(2*pi)))*exp(-.5*((x[i]-x)/h)^2))
  #Calculate each residual
  e <- y[i] - yhat
  data.frame(yhat = yhat,e = e)
}) %>% bind_rows()

plot(res.orig$yhat,res.orig$e,xlab = 'Fitted',ylab = 'Residual')
lines(c(min(res.orig$yhat),max(res.orig$yhat)),c(0,0),col = 2,lty = 'dashed')

#Now that we know how to make our most accurate predictions with Kernal regression. We can
#Make an attempt to calculate a p[rediction and an associated prediction interval.

#Step 1. Find original residuals from original model (done)
#Step 2. Create new X's.  BS.x = sample n values of x with replacement form x.
#Step 3.  Plug BS.x into original model to get BS.yhat
#Step 4.  BS.yhat + randomly sampled original error makes BS.y
#Step 5.  Create new model using BS.x, BS.y, plug in x.star
#Step 6.  Repeat 10,000 times.

BS.kernel <- function(x,y,h,xstar){
```

```
  #Gather original residuals
  res.orig <- sapply(1:length(x),function(i){
    #Make prediction for each x in dataset
    yhat <- sum((1/(sqrt(2*pi)))*exp(-.5*((x[i]-x)/h)^2)*y)/sum((1/(sqrt(2*pi)))*exp(-.5*((x[i]-x)/h)^2
    #Calculate each residual
    e <- y[i] - yhat
  })

  #Create new x's,bs.yhat, and new model 10k times.
   BS.pred <- sapply(1:1000,function(n){
    #Create BS x
    x.BS <- sample(x,size = length(x),replace = T)
    #For each BS X, calculate BS yhat with orginal model
    yhat.BS <-  sapply(x.BS,function(x.BS){
      sum((1/(sqrt(2*pi)))*exp(-.5*((x.BS-x)/h)^2)*y)/sum((1/(sqrt(2*pi)))*exp(-.5*((x.BS-x)/h)^2))
    })
    #Generate BS Y by adding random residual
    y.BS <- yhat.BS + sample(res.orig,size = length(yhat.BS), replace = T)
    #Calculate pred and add random residual for prediction interval
    pred <- sum((1/(sqrt(2*pi)))*exp(-.5*((xstar-x.BS)/h)^2)*y.BS)/sum((1/(sqrt(2*pi)))*exp(-.5*((xstar
  #print(paste('done',n))
    pred
    })
  plot(x,y,main = 'Prediction Plot',xlab = 'Employee',ylab = 'Cost')
  points(x = xstar,y = mean(BS.pred),col = 'red',pch = '+',cex = 4)

  #hist(e.bs)
  #Now we can create some prediction interval for our estimated predictive value
  a <-  quantile(BS.pred,.025)
  b <-  quantile(BS.pred,.975)
  return(list(BS.pred = BS.pred,pred = mean(BS.pred),lower = a,upper = b))
}

par(mfrow = c(3,2))
#Now we can build a table of the requested predictions
requested.pred <- lapply(c(1,20,100),function(req){
  BS.data <- BS.kernel(x,y,h,xstar = req)
  hist(BS.data$BS.pred,xlab = 'Bootstrapped Predictions',main = paste('Histogram of Predicted Cost at E
  lines(c(BS.data$pred,BS.data$pred),c(0,10000),col = 2)
  lines(c(BS.data$lower,BS.data$lower),c(0,10000),col = 4,lwd=2,lty=2)
  lines(c(BS.data$upper,BS.data$upper),c(0,10000),col = 4,lwd=2,lty=2)
  data.frame(Employee = req,Cost =BS.data$pred,lower = BS.data$lower,upper = BS.data$upper)
})

table <- bind_rows(requested.pred)
rownames(table) <- seq(1:nrow(table))

knitr::kable(table)
```