Lab3

David Nyberg

12/9/2019

Assignment 1

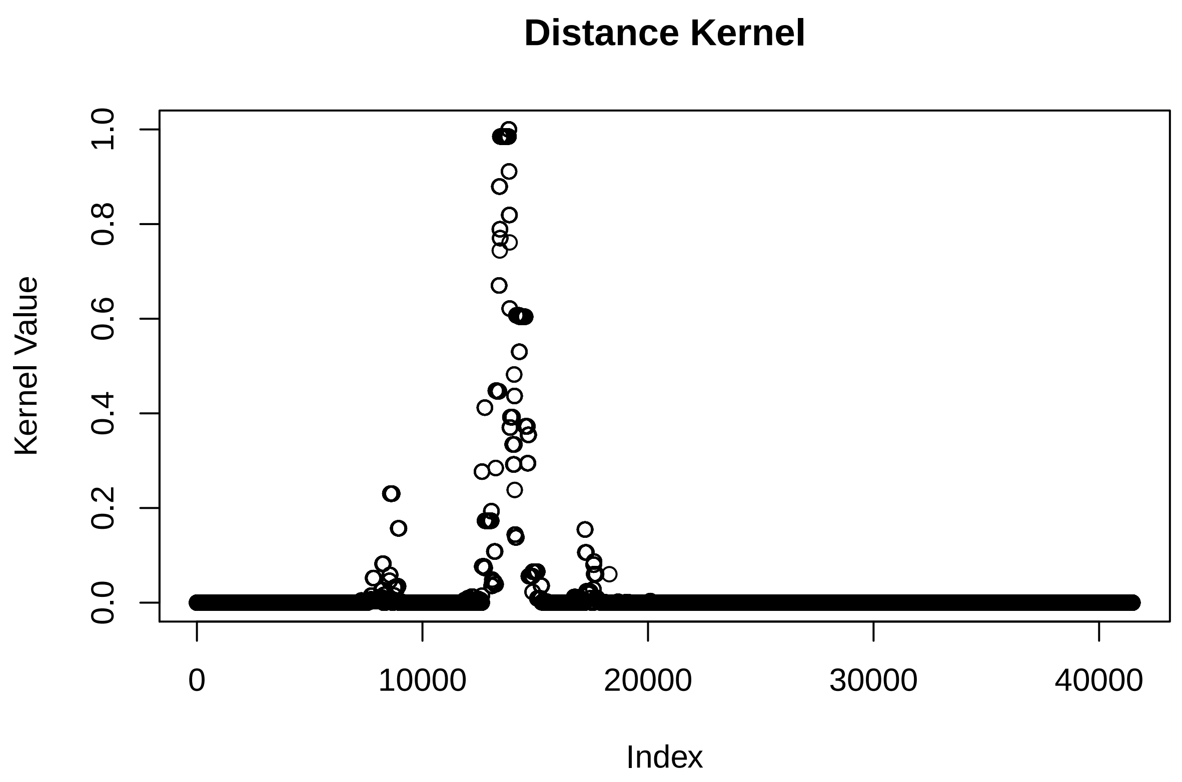
Using three different gaussian kernels to predict weather in Sweden. For my prediction I will predict on Linkoping (15.6333, 58.4166), on the date 2010-05-05. Because we cannot use future data to predict we subset our data to select dates that are before the target date.

The first distance kernel checks the distance between two weather stations. To do this we used the distHaversine function from the geosphere package. This gets the the 'great circle distance' or essentially straight line distance. Then using the guassian kernel we get the values for each distance.

Plotting these kernel values we can see the kernel width showing we give almost zero weight to stations that are far away from our station (linkoping). The hvalue of 50,000 seems reasonable to allow temperature within 50,000m influence the prediction.

```r
#takes in latitude/longitude, predicted date, and h scale factor,
distance_kernel <- function(x1,x2, pred, h) {
  a <- (c(x1, x2))
  tdist <- distHaversine(a, pred)
  return (exp(-((tdist)/h)^2))
}
prediction_distance_kernel <- c()
for (i in 1:len){
  prediction_distance_kernel[i] <- distance_kernel(newst$longitude[i],newst$latitude[i],
                                                   latlong_pred, 50000)
}

plot(prediction_distance_kernel, ylab = "Kernel Value", main = "Distance Kernel")
```



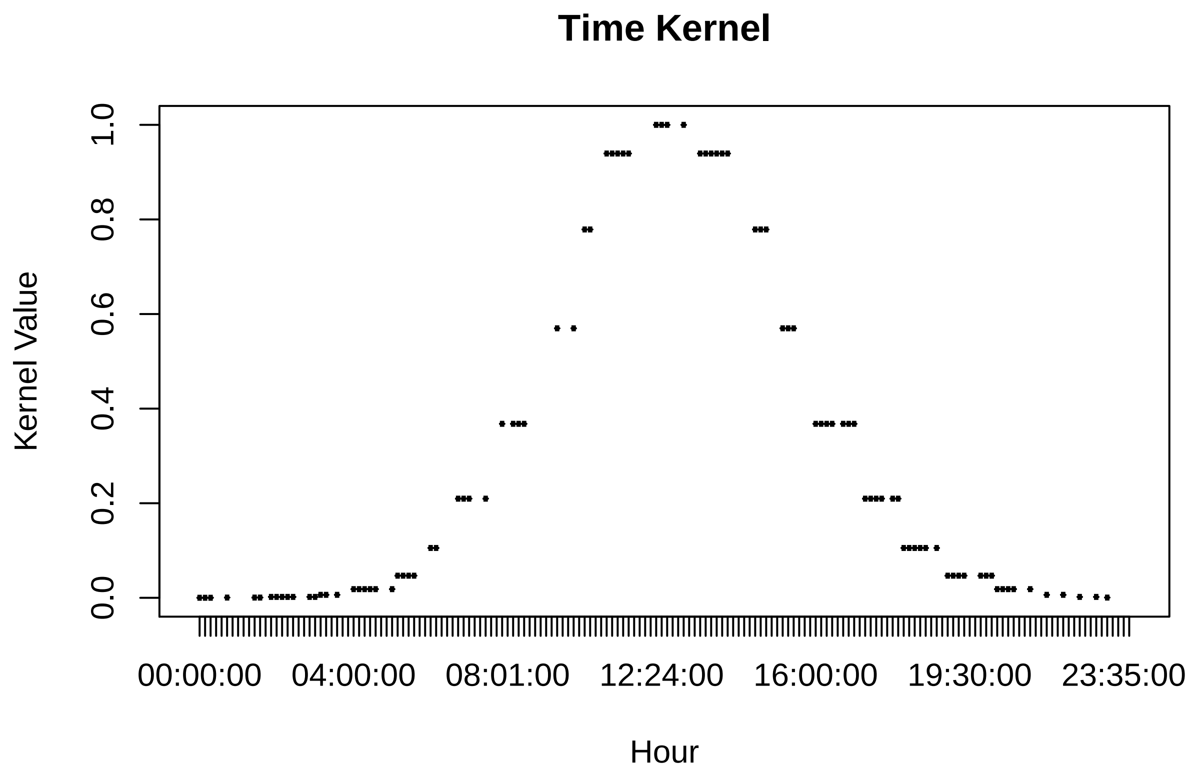
The second kernel time kernel checks the 'distance' between two time observations. Essentially this kernel is returning the distance in hours between two times and then weighted by the gaussian kernel function.

Plotting these kernel values we can see that a h-value of 4 means that we are giving high weight to measurements that are close 4 hours within our predictions, otherwise they have little weight.

```
time_kernel <- function(t1, t2, h) {
  dist <- abs(t1 - t2)
  if (dist > 12){
    return (exp(-((dist-24)/h)^2))
  } else {
    return (exp(-(dist/h)^2))
  }
}

ntime <- newst$time
prediction_time_kernel <- c()
for (i in 1:len) {
  prediction_time_kernel[i] <- (time_kernel(12, as.numeric(substring(ntime[i],1,2)), 4))
}

plot(x = ntime,y = prediction_time_kernel,xlab = "Hour", ylab = "Kernel Value", main = "Time Kernel")
```

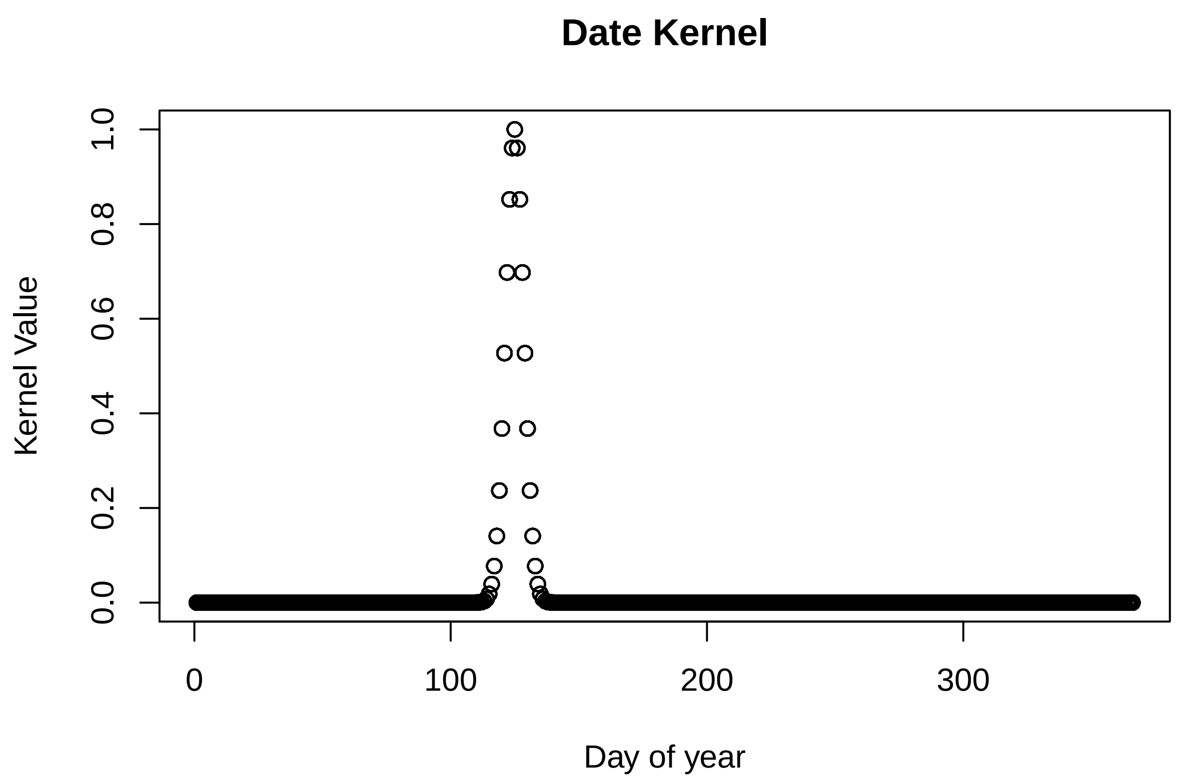


The final kernel takes the distance between two days of the year. For this calculation the year is irrelevant, we care about the day of the year (1-365), then the value passed through the gaussian kernel with an h value of 5. Meaning that within 5 days from our prediction target is relevant and should have higher weight.

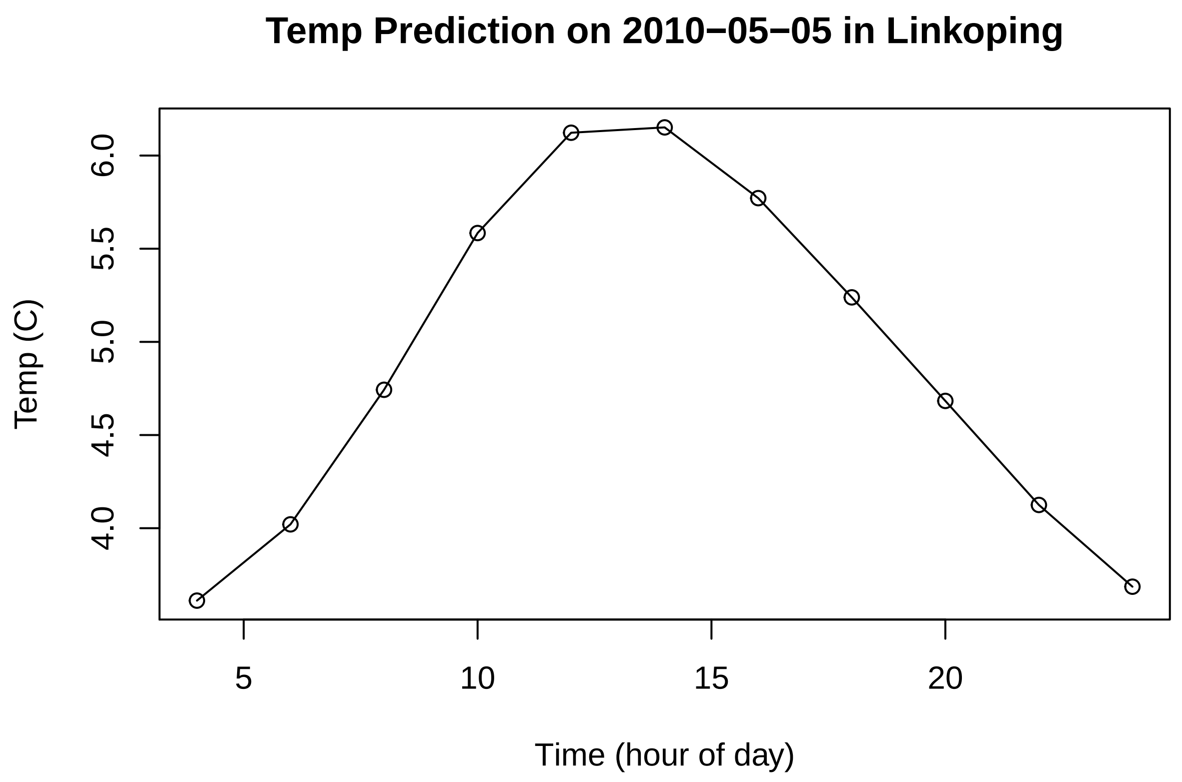
In the plot we can see with an h-value of 5 makes most days of the year have a weight of zero, this is because weather more than 5 days apart should not have a big influence on a prediction.

```r
date_kernel <- function(d1, d2, h) {
  a <- abs(d1 - d2)
  if (a > 182){
    b <- abs(a-365)
    return (exp(-((b/h)^2)))
  } else {
    return (exp(-(a/h)^2))
  }
}
ndate <- c(yday(newst$date))
prediction_date_kernel <- c()
for (i in 1:len) {
  prediction_date_kernel[i] <- (date_kernel(yday(p_date), ndate[i], 5) )
}


#date looks good
plot(x = ndate, y= prediction_date_kernel, main="Date Kernel",
     ylab = "Kernel Value", xlab="Day of year")
```

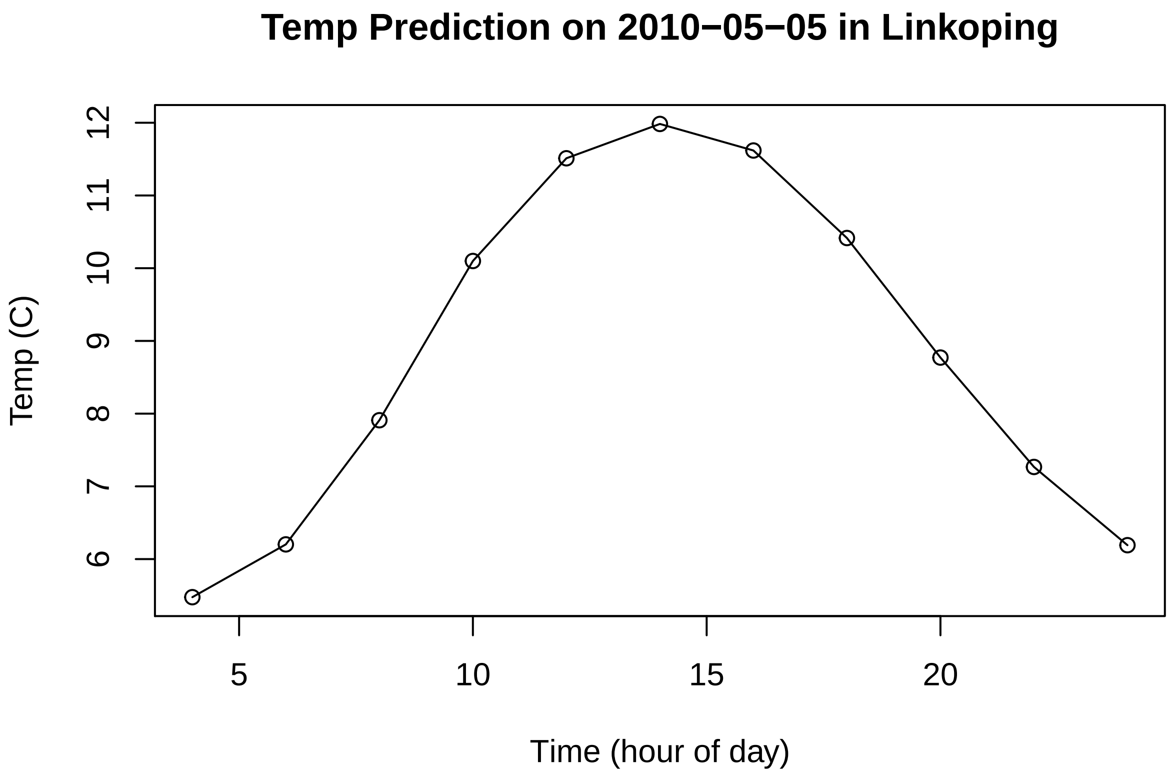


With our three gaussian kernel we can add them together to make a prediction for our date, and location. The prediction seems reasonable, but not very accurate.



We repeat the exact same calculations but now we multiply our kernels together. Now we can see a similar prediction curve, but our temperatures are more realistic. Coldest at the night horus and warmest during midday, and temps from around 5-12 degrees celsius which makes sense for May.

Multiplying gives a more accurate prediction because it weighs all the inputs more accuratly, such as multiplying by zero or almost zero for our weights that shouldn't have much influence on the prediction.

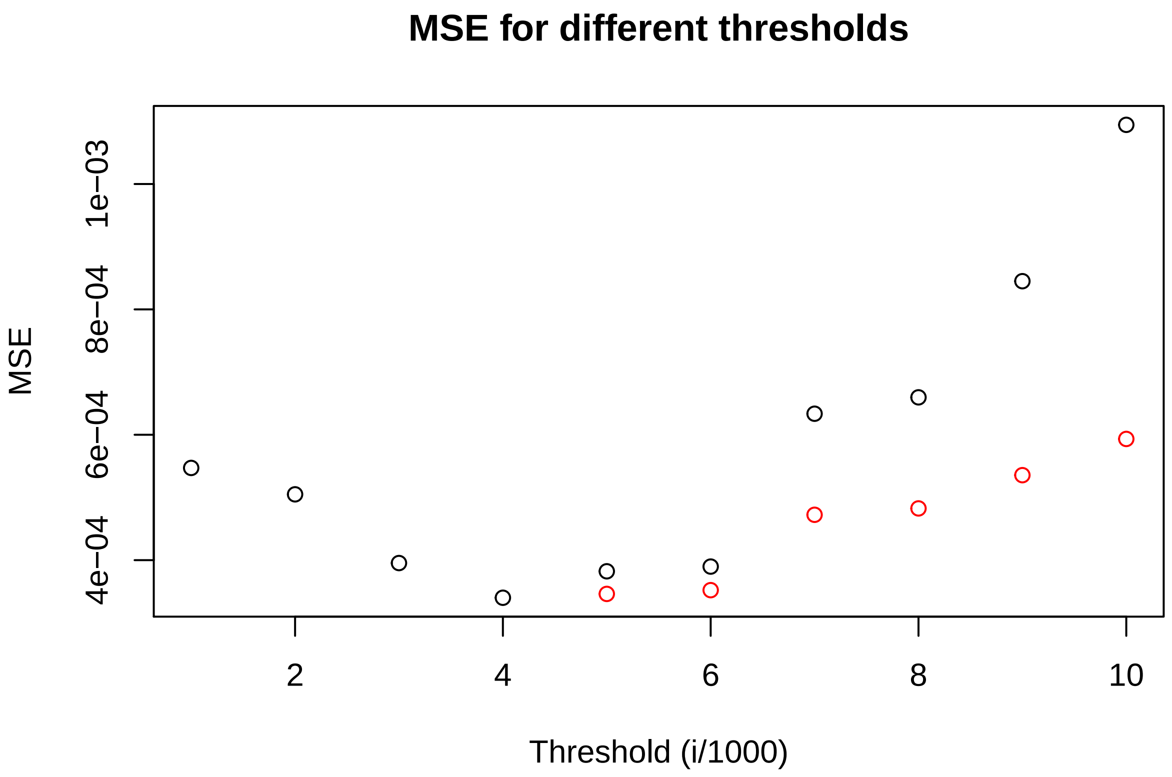


3 Neural network

Training a neural network to learn the sin function.

To start, the weights for the network are randomly initialized from -1 to 1 and we chose 31 weights as there are 10 nodes in the hidden layer giving us 10 input weights, with 10 biases, an output with 10 weights, then 1 bias for the output node. This can be seen in our visualization of the NN below. The thresholds are chosen i/1000 with i=1-1,..10.

In this plot the red points represent the training MSE and the black the validation MSE. You can see based off this plot that our validation set error gets lower as we increase i to 4. After that point we increase both our training and validation error. This means 4/1000 is our optimal threshold for the best neural network.

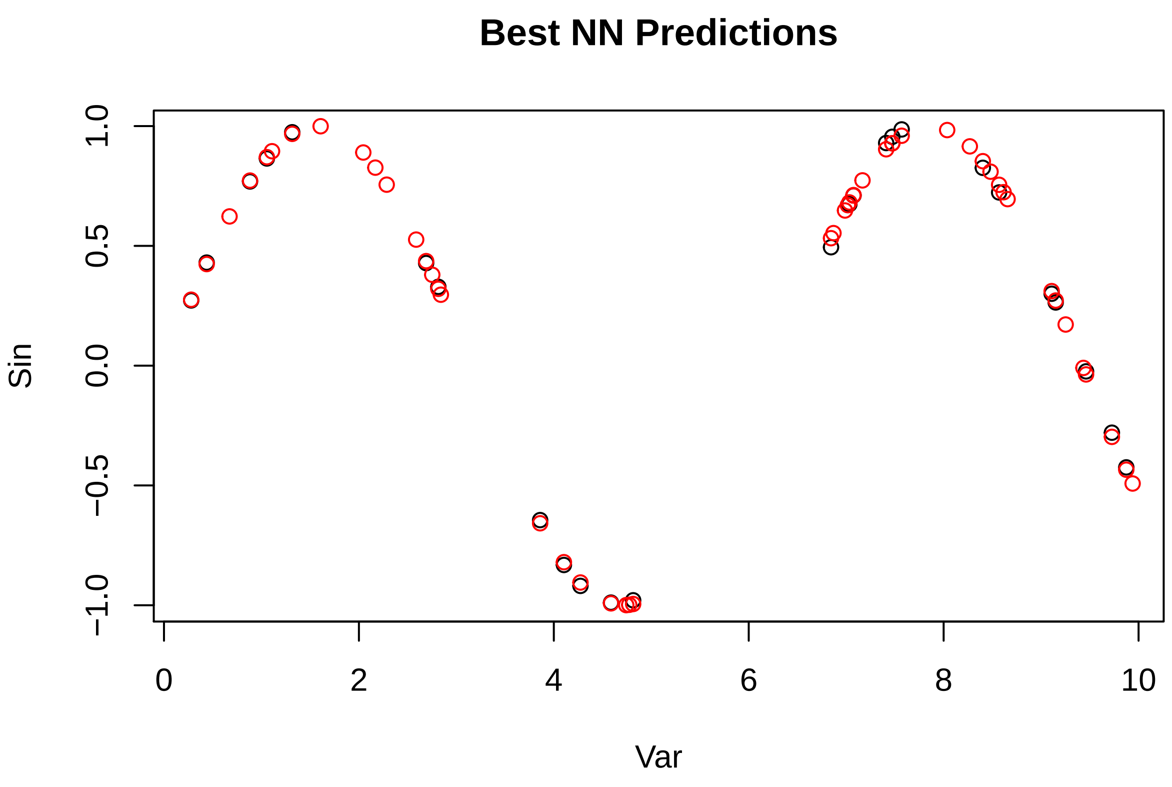


Here our best neural network with threshold = 4/1000 based off our results from checking MSE, to make the best neural network it seems like taking the lowest error seems reasonable.

```
newnn <- neuralnet(formula = Sin ~ Var, data = tr, hidden = 10, threshold = 4/1000, startweights = wini
plot(newnn)
```

Finally, a plot showing predictions from our best NN and we can see the red points is the actual data, and the black is our NN. This shows that the predictions are very accurate.

```
## Data Error:  0;
```



Appendix

```r
knitr::opts_chunk$set(echo = TRUE)
library(geosphere)
library(lubridate)
library(neuralnet)
library(Metrics)
RNGversion("3.5.1")
set.seed(1234567890)
stations <- read.csv("stations.csv", fileEncoding = "latin1")
temps <- read.csv("temps50k.csv")
st <- merge(stations,temps,by="station_number")

latlong_pred <- c(15.6333, 58.4166) #linkoping
p_date <- "2010-05-05" # The date to predict

#take only the dates before our prediction
newst <- subset(st, as.Date(date) < as.Date(p_date))

times <- c("04:00:00", "06:00:00", "08:00:00", "10:00:00", "12:00:00",
           "14:00:00", "16:00:00", "18:00:00", "20:00:00", "22:00:00", "24:00:00")
temp <- vector(length=length(times))

#used to index for loops in kernels
len <- length(newst$date)
#takes in latitude/longitude, predicted date, and h scale factor,
distance_kernel <- function(x1,x2, pred, h) {
  a <- (c(x1, x2))
  tdist <- distHaversine(a, pred)
  return (exp(-((tdist)/h)^2))
}
prediction_distance_kernel <- c()
for (i in 1:len){
  prediction_distance_kernel[i] <- distance_kernel(newst$longitude[i],newst$latitude[i],
                                                   latlong_pred, 50000)
}

plot(prediction_distance_kernel, ylab = "Kernel Value", main = "Distance Kernel")
time_kernel <- function(t1, t2, h) {
  dist <- abs(t1 - t2)
  if (dist > 12){
    return (exp(-((dist-24)/h)^2))
  } else {
    return (exp(-(dist/h)^2))
  }
}

ntime <- newst$time
prediction_time_kernel <- c()
for (i in 1:len) {
  prediction_time_kernel[i] <- (time_kernel(12, as.numeric(substring(ntime[i],1,2)), 4))
}

plot(x = ntime,y = prediction_time_kernel,xlab = "Hour", ylab = "Kernel Value", main = "Time Kernel")
date_kernel <- function(d1, d2, h) {
  a <- abs(d1 - d2)
  if (a > 182){
    b <- abs(a-365)
    return (exp(-((b/h)^2)))
  } else {
    return (exp(-(a/h)^2))
  }
}
ndate <- c(yday(newst$date))
prediction_date_kernel <- c()
for (i in 1:len) {
  prediction_date_kernel[i] <- (date_kernel(yday(p_date), ndate[i], 5) )
}


#date looks good
plot(x = ndate, y= prediction_date_kernel, main="Date Kernel",
     ylab = "Kernel Value", xlab="Day of year")
denom = c()
num = c()
pred = c()
test.prediction_time_kernel = c()
test.prediction_distance_kernel = c()

for (x in 1:length(times))
{
  temp_time <- as.numeric(substring(times[x], 1, 2))

  for (i in 1:length(newst$date)) {
    # temp time will input all the hours of the day into the time kernel and predict for
    # each hour of the day, same location, same date. 4 is the h_value

    test.prediction_time_kernel[i] <- (time_kernel(temp_time, as.numeric(substring(ntime[i],1,2)), 4))
    denom[i] <- (test.prediction_time_kernel[i] + prediction_date_kernel[i] +
                   prediction_distance_kernel[i])
    num[i] <- (denom[i] * newst$air_temperature[i])
  }
  pred[x] <- sum(num) / sum(denom)
}

hrs <- seq(4,24,2)
plot(x = hrs, y = pred, type="o", ylab="Temp (C)", xlab="Time (hour of day)",
     main = "Temp Prediction on 2010-05-05 in Linkoping")
for (x in 1:length(times))
{
  temp_time <- as.numeric(substring(times[x], 1, 2))

  for (i in 1:length(newst$date)) {
    # temp time will input all the hours of the day into the time kernel and predict for
    # each hour of the day, same location, same date. 4 is the h_value

    test.prediction_time_kernel[i] <- (time_kernel(temp_time, as.numeric(substring(ntime[i],1,2)), 4))
    denom[i] <- (test.prediction_time_kernel[i] * prediction_date_kernel[i] *
                   prediction_distance_kernel[i])
    num[i] <- (denom[i] * newst$air_temperature[i])
  }
  pred[x] <- sum(num) / sum(denom)
}

hrs <- seq(4,24,2)
plot(x = hrs, y = pred, type="o", ylab="Temp (C)", xlab="Time (hour of day)",
     main = "Temp Prediction on 2010-05-05 in Linkoping")

set.seed(1234567890)
Var <- runif(50, 0, 10)
trva <- data.frame(Var, Sin=sin(Var))
tr <- trva[1:25,] # Training
va <- trva[26:50,] # Validation

mseva <- c()
msetr <- c()
# Random initialization of the weights in the interval [-1, 1]
winit <- runif(31, -1, 1)
for(i in 1:10) {
  nn <- neuralnet(formula = Sin ~ Var, data = tr, hidden = 10, threshold = i/1000, startweights = winit
  #predict everytime for 1-10
    temp_p <- predict(nn, va)
    temp_p2 <- predict(nn, tr)
  #msetr[i] <- (1/25) * sum((va$Sin - temp_p)^2)
  #mean square error from Sin curve to our predictions
  mseva[i] <- mse(va$Sin, temp_p)
  msetr[i] <- mse(tr$Sin, temp_p2)
}
plot(mseva, xlab = "Threshold (i/1000)", main = "MSE for different thresholds", ylab = "MSE")
points(msetr, col = "red")

newnn <- neuralnet(formula = Sin ~ Var, data = tr, hidden = 10, threshold = 4/1000, startweights = winit
plot(newnn)
#predicting with the best NN, the plot is very accurate
suppressWarnings(plot(prediction(newnn)$rep1,main="Best NN Predictions"))
par(new = TRUE)
points(trva, col = "red")
```