ML Lab3, block 1

Rojan Karakaya and Suhani Ariga

31/01/2021

Contents

Contributions

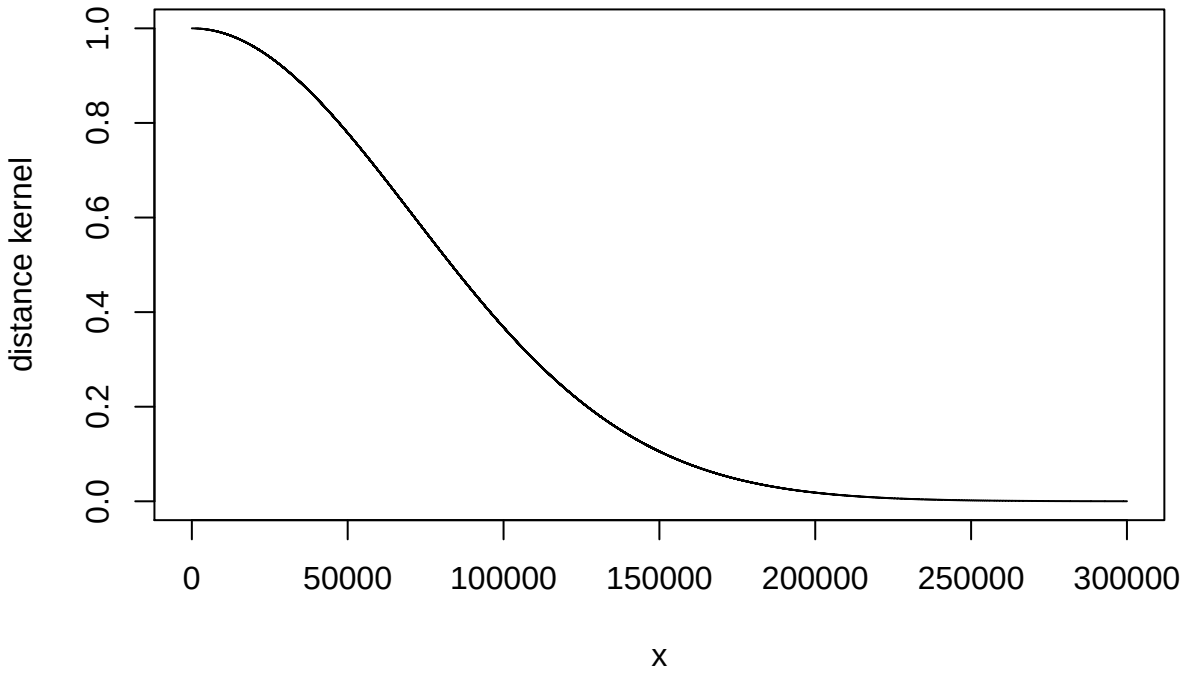
For Lab 3, Suhani focused on assignment 1 and Rojan focused on assignment 3.

Assignment 1. Kernel Methods

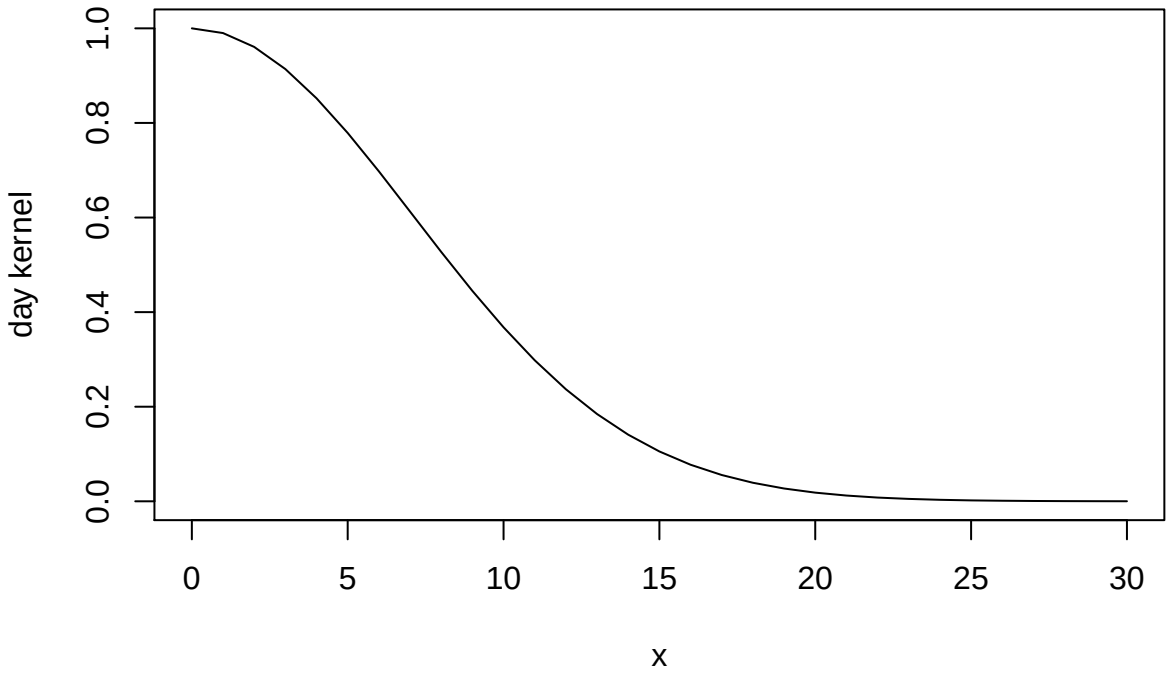
```
set.seed(1234567890)
library(geosphere)
```

```
## Warning: package 'geosphere' was built under R version 4.0.3
```

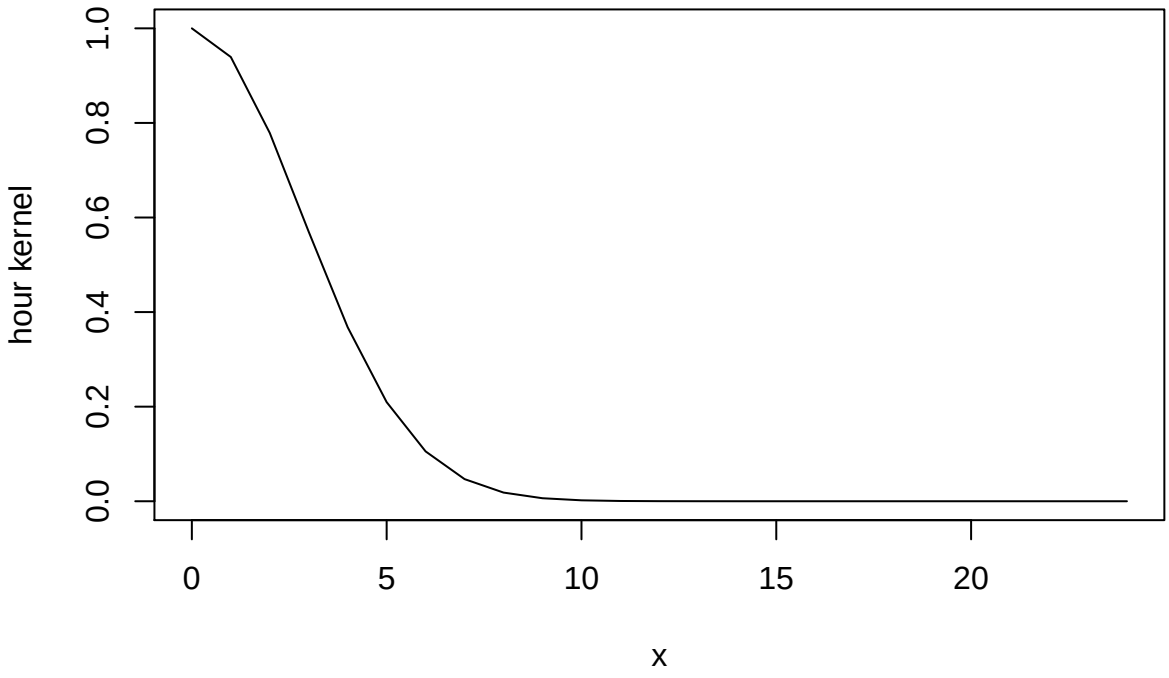
```
stations <- read.csv("stations.csv")
temps <- read.csv("temps50k.csv")
st <- merge(stations,temps,by="station_number")
h_distance <- 100000
h_date <- 10
h_time <- 4
a <- 58.4274
b <- 14.826
date <- "2013-8-15" # The date to predict (up to the students)
times <- c("04:00:00", "06:00:00", "08:00:00", "10:00:00", "12:00:00",
"14:00:00", "16:00:00", "18:00:00", "20:00:00", "22:00:00",
"24:00:00")

distance_plot <- function(x, h){
y <- exp(-(x/h)^2)
plot(x = x, y = y, type="l", ylab = "distance kernel")
}

day_plot <- function(x, h){
y <- exp(-(x/h)^2)
plot(x = x, y = y, type="l", ylab = "day kernel")
}

hour_plot <- function(x, h) {
y <- exp(-(x/h)^2)
plot(x = x, y = y, type="l", ylab = "hour kernel")
}

distance_plot(seq(0,300000,1),100000)
```




```
day_plot(seq(0,30,1),10)
```

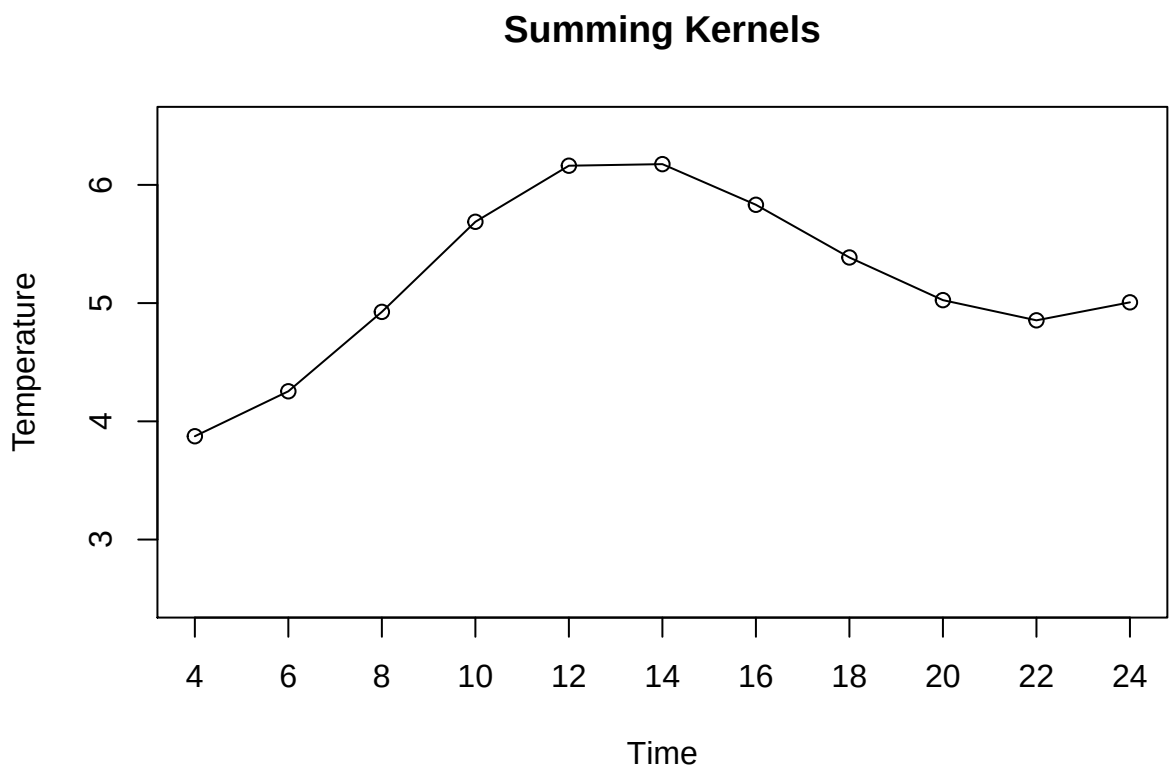



```
hour_plot(seq(0,24,1),4)
```

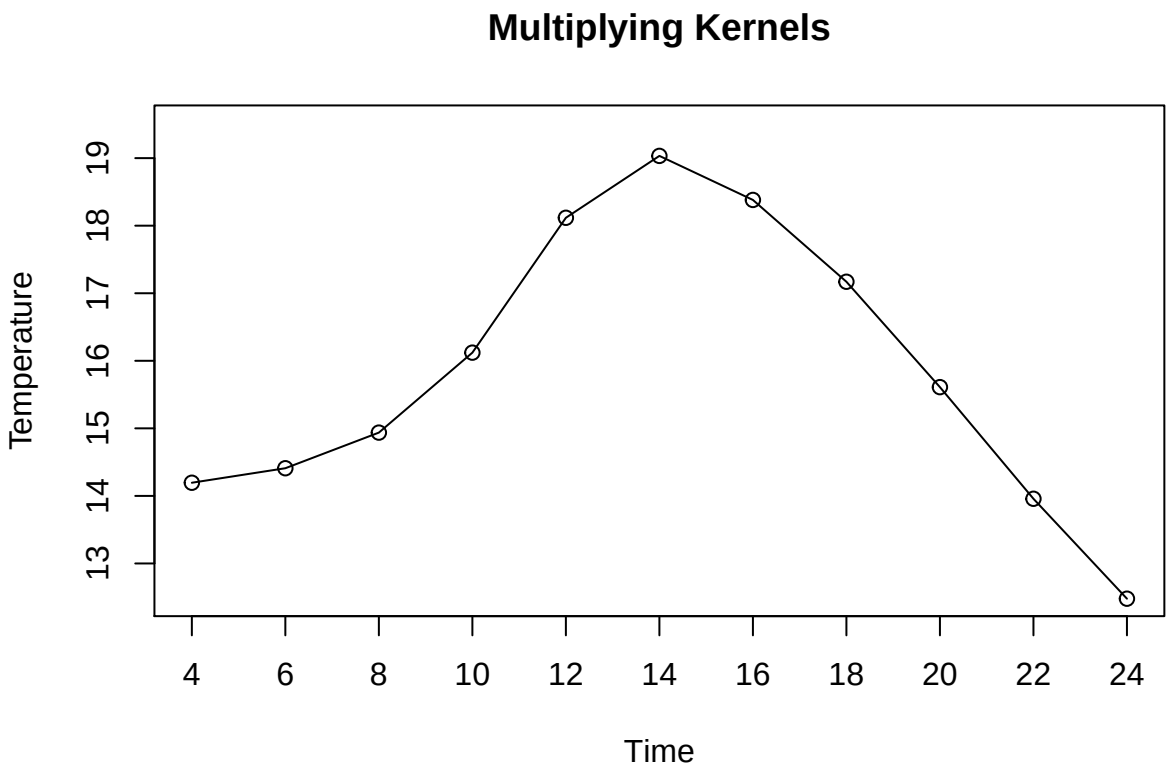



```
filter_posterior <- function(st, date){
return(st[as.Date(st$date) < as.Date(date),])
}

filtered_data <- filter_posterior(st = st, date = date)

get_position = function(data, pre){
  res = abs(distHaversine(data, pre)/h_distance)
  return(exp(-(res^2)))
}

point_of_interest = c(b,a)
g1 <- get_position(filtered_data[,c("longitude","latitude")], point_of_interest)

get_day = function(data, pre){
  res = as.numeric((difftime(data, pre, units ="days"))/h_date)
  return(exp(-(res^2)))
}

g2 <- get_day(filtered_data$date, date)

get_hour <- function(data, pre) {
  res = as.numeric(difftime(strptime(data, format="%H:%M:%S"),strptime(pre, format="%H:%M:%S"), units =
  return(exp(-(res)^2))
}
```

```
temp_sum <- vector(length=length(times))
temp_mul <- vector(length=length(times))

for (i in 1:length(times)){
g3 <- get_hour(filtered_data$time, times[i])
temp_sum[i] <- sum((g1 + g2 + g3) %*% filtered_data$air_temperature)/sum(g1 + g2 + g3)
temp_mul[i] <- sum((g1 * g2 * g3) %*% filtered_data$air_temperature)/sum(g1 * g2 * g3)
}

plot (temp_sum, type ="o", xaxt = "n", xlab ="Time", ylab = "Temperature", main = "Summing Kernels", yli
axis (1, at =1:11, labels = seq (04 ,24 ,2))
```




```
plot (temp_mul, type ="o", xaxt = "n", xlab ="Time", ylab = "Temperature", main = "Multiplying Kernels"
axis (1, at =1:11, labels = seq (04 ,24 ,2))
```



The different results for the product and the summation of kernels are due to the independence of the kernels: the temperature of a given place is independent of the temperature of a given hour of the day, which is independent of the temperature of that day of the year. For the total kernel, each kernel with a higher weight would play an important role when using the sum of the three kernels. However, the large single kernel value will be eliminated by other very small kernel values when we follow the multiplication of three kernels. At the same time, if one kernel value is extremely small, i.e. close to zero, the final kernel value, such as near zero, would be very small. That means the three Gaussian kernels of one point must all have high values to make this point important in the decision when the multiplication of three kernel functions.

Assignment 3: Neural Networks

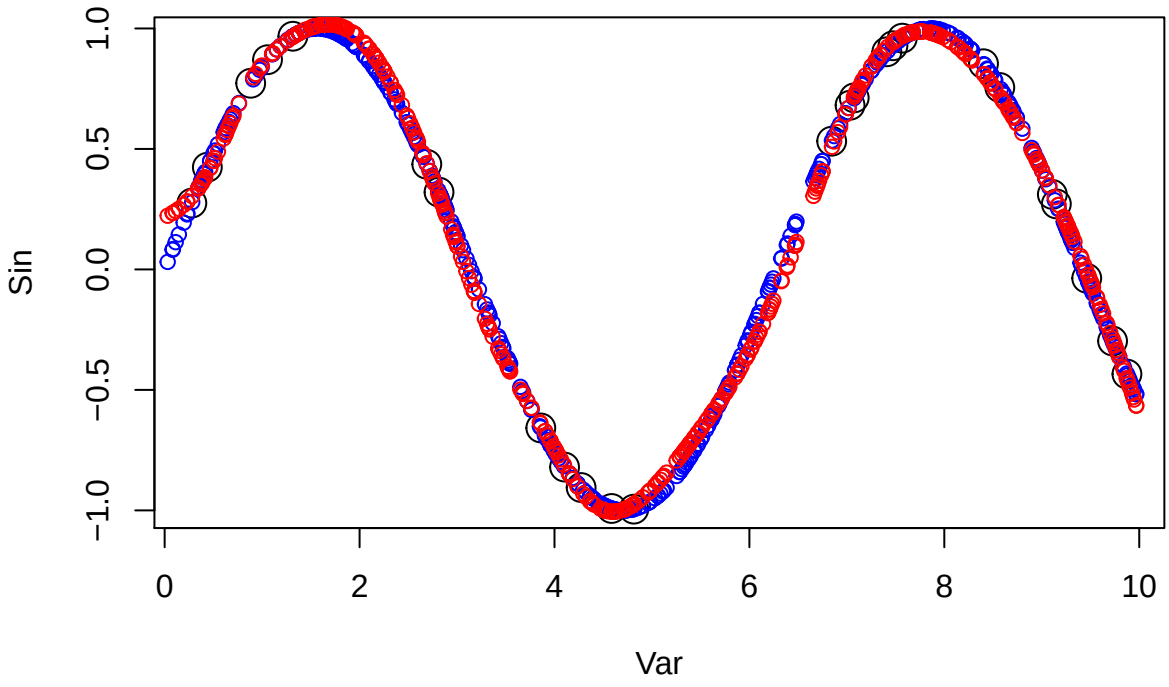
Part 1

```
library(neuralnet)
```

```
## Warning: package 'neuralnet' was built under R version 4.0.3
```

```
set.seed(1234567890)
Var <- runif(500, 0, 10)
mydata <- data.frame(Var, Sin=sin(Var))
tr <- mydata[1:25,] # Training
te <- mydata[26:500,] # Test
```

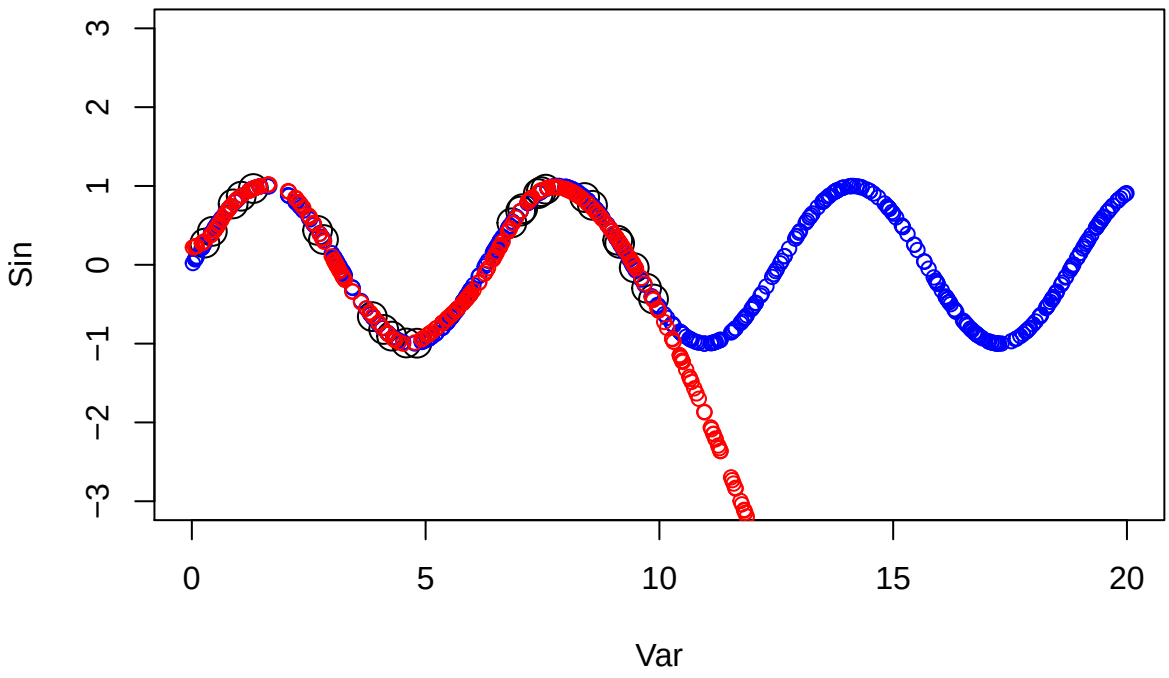
```r
nn <- neuralnet(Sin ~Var, tr, hidden = c(4, 4))

#3.1
# Plot of the training data (black), test data (blue), and predictions (red)
plot(tr, cex=2)
points(te, col = "blue", cex=1)
points(te[,1],predict(nn,te), col="red", cex=1)
```



We have chosen a modest network with 2 hidden layers with four neurons in each, since the function that we are to approximate is fairly simple. As can be seen in the chart above, the network seems to generalize well to the test set.

Part 2

```r
Var2 <- runif(500, 0, 20)
mydata2 <- data.frame(Var=Var2, Sin=sin(Var2))
tr2 <- mydata2[1:25,] # Training
te2 <- mydata2[26:500,] # Test

plot(tr, cex=2,ylim=c(-3,3),xlim=c(0,20))
points(te2, col = "blue", cex=1)
points(te2[,1],predict(nn,te2), col="red", cex=1)
```

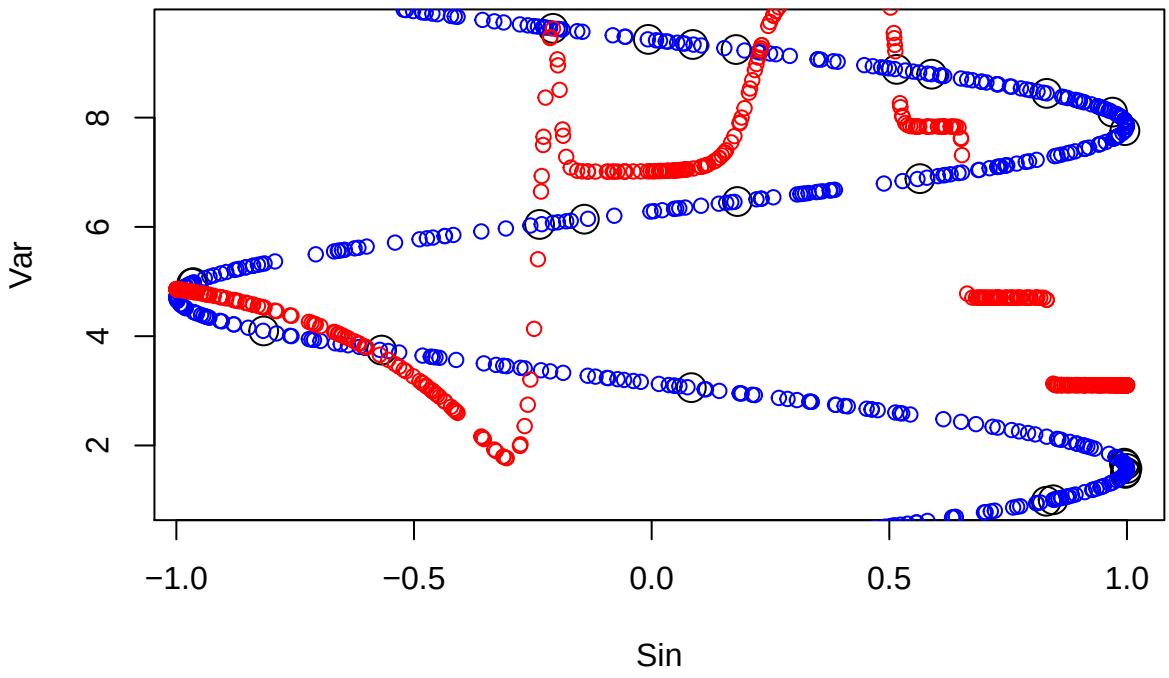


While the network still generalizes well in the input range 0 to 10, it misses the mark completely when it comes to the second cycle of the sine wave. Clearly, it has not learned the periodical nature of the target function.

Each hidden layer of the neural network is like a look-up table, i.e. a very simple approximation of a function. The more hidden layers that are added, the more complex the approximated function can be (as the network becomes a series of nested approximate functions). These, however, only approximate the target function within the domain for which there are data points. Outside of the domain of the training data, a neural network (or any supervised learning method) will not extrapolate particularly well.

Part 3

```
Var3 <- runif(500, 0, 10)
mydata3 <- data.frame(Var=Var3, Sin=sin(Var3))
tr3 <- mydata3[1:25,] # Training
te3 <- mydata3[26:500,] # Test


nn3 <- neuralnet(Var ~ Sin, tr3, hidden = c(3,5))

plot(tr3[,2:1], cex=2,)
points(te3[,2:1], col = "blue", cex=1)
points(te3[,2],predict(nn3,te3), col="red", cex=1)
```



The network works terribly for this problem. The reason for this is that neural networks approximate *functions*, and while functions can map multiple different arguments to the same output, it can only map each set of input variables to a single output. Thus, no neural network is going to solve this problem unless the representation of the data is changed.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
set.seed(1234567890)
library(geosphere)

stations <- read.csv("stations.csv")
temps <- read.csv("temps50k.csv")
st <- merge(stations,temps,by="station_number")
h_distance <- 100000
h_date <- 10
h_time <- 4
a <- 58.4274
b <- 14.826
date <- "2013-8-15" # The date to predict (up to the students)
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")
```

```r
distance_plot <- function(x, h){
y <- exp(-(x/h)^2)
plot(x = x, y = y, type="l", ylab = "distance kernel")
}

day_plot <- function(x, h){
y <- exp(-(x/h)^2)
plot(x = x, y = y, type="l", ylab = "day kernel")
}

hour_plot <- function(x, h) {
y <- exp(-(x/h)^2)
plot(x = x, y = y, type="l", ylab = "hour kernel")
}

distance_plot(seq(0,300000,1),100000)
day_plot(seq(0,30,1),10)
hour_plot(seq(0,24,1),4)

filter_posterior <- function(st, date){
return(st[as.Date(st$date) < as.Date(date),])
}

filtered_data <- filter_posterior(st = st, date = date)

get_position = function(data, pre){
  res = abs(distHaversine(data, pre)/h_distance)
  return(exp(-(res^2)))
}

point_of_interest = c(b,a)
g1 <- get_position(filtered_data[,c("longitude","latitude")], point_of_interest)

get_day = function(data, pre){
  res = as.numeric((difftime(data, pre, units ="days"))/h_date)
  return(exp(-(res^2)))
}

g2 <- get_day(filtered_data$date, date)

get_hour <- function(data, pre) {
  res = as.numeric(difftime(strptime(data, format="%H:%M:%S"),strptime(pre, format="%H:%M:%S"), units =
  return(exp(-(res)^2))
}

temp_sum <- vector(length=length(times))
temp_mul <- vector(length=length(times))

for (i in 1:length(times)){
g3 <- get_hour(filtered_data$time, times[i])
temp_sum[i] <- sum((g1 + g2 + g3) %*% filtered_data$air_temperature)/sum(g1 + g2 + g3)
temp_mul[i] <- sum((g1 * g2 * g3) %*% filtered_data$air_temperature)/sum(g1 * g2 * g3)
}
```

```
plot (temp_sum, type ="o", xaxt = "n", xlab ="Time", ylab = "Temperature", main = "Summing Kernels", yli
axis (1, at =1:11, labels = seq (04 ,24 ,2))

plot (temp_mul, type ="o", xaxt = "n", xlab ="Time", ylab = "Temperature", main = "Multiplying Kernels"
axis (1, at =1:11, labels = seq (04 ,24 ,2))

library(neuralnet)
set.seed(1234567890)
Var <- runif(500, 0, 10)
mydata <- data.frame(Var, Sin=sin(Var))
tr <- mydata[1:25,] # Training
te <- mydata[26:500,] # Test
nn <- neuralnet(Sin ~Var, tr, hidden = c(4, 4))

#3.1
# Plot of the training data (black), test data (blue), and predictions (red)
plot(tr, cex=2)
points(te, col = "blue", cex=1)
points(te[,1],predict(nn,te), col="red", cex=1)
Var2 <- runif(500, 0, 20)
mydata2 <- data.frame(Var=Var2, Sin=sin(Var2))
tr2 <- mydata2[1:25,] # Training
te2 <- mydata2[26:500,] # Test

plot(tr, cex=2,ylim=c(-3,3),xlim=c(0,20))
points(te2, col = "blue", cex=1)
points(te2[,1],predict(nn,te2), col="red", cex=1)
Var3 <- runif(500, 0, 10)
mydata3 <- data.frame(Var=Var3, Sin=sin(Var3))
tr3 <- mydata3[1:25,] # Training
te3 <- mydata3[26:500,] # Test


nn3 <- neuralnet(Var ~ Sin, tr3, hidden = c(3,5))

plot(tr3[,2:1], cex=2,)
points(te3[,2:1], col = "blue", cex=1)
points(te3[,2],predict(nn3,te3), col="red", cex=1)

set.seed(1234567890)
library(geosphere)

stations <- read.csv("stations.csv")
temps <- read.csv("temps50k.csv")
st <- merge(stations,temps,by="station_number")
h_distance <- 100000
h_date <- 10
h_time <- 4
a <- 58.4274
b <- 14.826
date <- "2013-8-15" # The date to predict (up to the students)
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")

distance_plot <- function(x, h){
y <- exp(-(x/h)^2)
plot(x = x, y = y, type="l", ylab = "distance kernel")
}

day_plot <- function(x, h){
y <- exp(-(x/h)^2)
plot(x = x, y = y, type="l", ylab = "day kernel")
}

hour_plot <- function(x, h) {
y <- exp(-(x/h)^2)
plot(x = x, y = y, type="l", ylab = "hour kernel")
}

distance_plot(seq(0,300000,1),100000)
day_plot(seq(0,30,1),10)
hour_plot(seq(0,24,1),4)

filter_posterior <- function(st, date){
return(st[as.Date(st$date) < as.Date(date),])
}

filtered_data <- filter_posterior(st = st, date = date)

get_position = function(data, pre){
  res = abs(distHaversine(data, pre)/h_distance)
  return(exp(-(res^2)))
}

point_of_interest = c(b,a)
g1 <- get_position(filtered_data[,c("longitude","latitude")], point_of_interest)

get_day = function(data, pre){
  res = as.numeric((difftime(data, pre, units ="days"))/h_date)
  return(exp(-(res^2)))
}

g2 <- get_day(filtered_data$date, date)

get_hour <- function(data, pre) {
  res = as.numeric(difftime(strptime(data, format="%H:%M:%S"),strptime(pre, format="%H:%M:%S"), units =
  return(exp(-(res)^2))
}

temp_sum <- vector(length=length(times))
temp_mul <- vector(length=length(times))

for (i in 1:length(times)){
g3 <- get_hour(filtered_data$time, times[i])
temp_sum[i] <- sum((g1 + g2 + g3) %*% filtered_data$air_temperature)/sum(g1 + g2 + g3)
```

```
temp_mul[i] <- sum((g1 * g2 * g3) %*% filtered_data$air_temperature)/sum(g1 * g2 * g3)
}

plot (temp_sum, type ="o", xaxt = "n", xlab ="Time", ylab = "Temperature", main = "Summing Kernels", yli
axis (1, at =1:11, labels = seq (04 ,24 ,2))

plot (temp_mul, type ="o", xaxt = "n", xlab ="Time", ylab = "Temperature", main = "Multiplying Kernels"
axis (1, at =1:11, labels = seq (04 ,24 ,2))




###Assignment 3
#part 1
library(neuralnet)
set.seed(1234567890)
Var <- runif(500, 0, 10)
mydata <- data.frame(Var, Sin=sin(Var))
tr <- mydata[1:25,] # Training
te <- mydata[26:500,] # Test



nn <- neuralnet(Sin ~Var, tr, hidden = c(4, 4))

#3.1
# Plot of the training data (black), test data (blue), and predictions (red)
plot(tr, cex=2)
points(te, col = "blue", cex=1)
points(te[,1],predict(nn,te), col="red", cex=1)


#part 2
Var2 <- runif(500, 0, 20)
mydata2 <- data.frame(Var=Var2, Sin=sin(Var2))
tr2 <- mydata2[1:25,] # Training
te2 <- mydata2[26:500,] # Test

plot(tr, cex=2,ylim=c(-3,3),xlim=c(0,20))
points(te2, col = "blue", cex=1)
points(te2[,1],predict(nn,te2), col="red", cex=1)


#part 3
Var3 <- runif(500, 0, 10)
mydata3 <- data.frame(Var=Var3, Sin=sin(Var3))
tr3 <- mydata3[1:25,] # Training
te3 <- mydata3[26:500,] # Test


nn3 <- neuralnet(Var ~ Sin, tr3, hidden = c(3,5))
```

```
plot(tr3[,2:1], cex=2,)
points(te3[,2:1], col = "blue", cex=1)
points(te3[,2],predict(nn3,te3), col="red", cex=1)
```