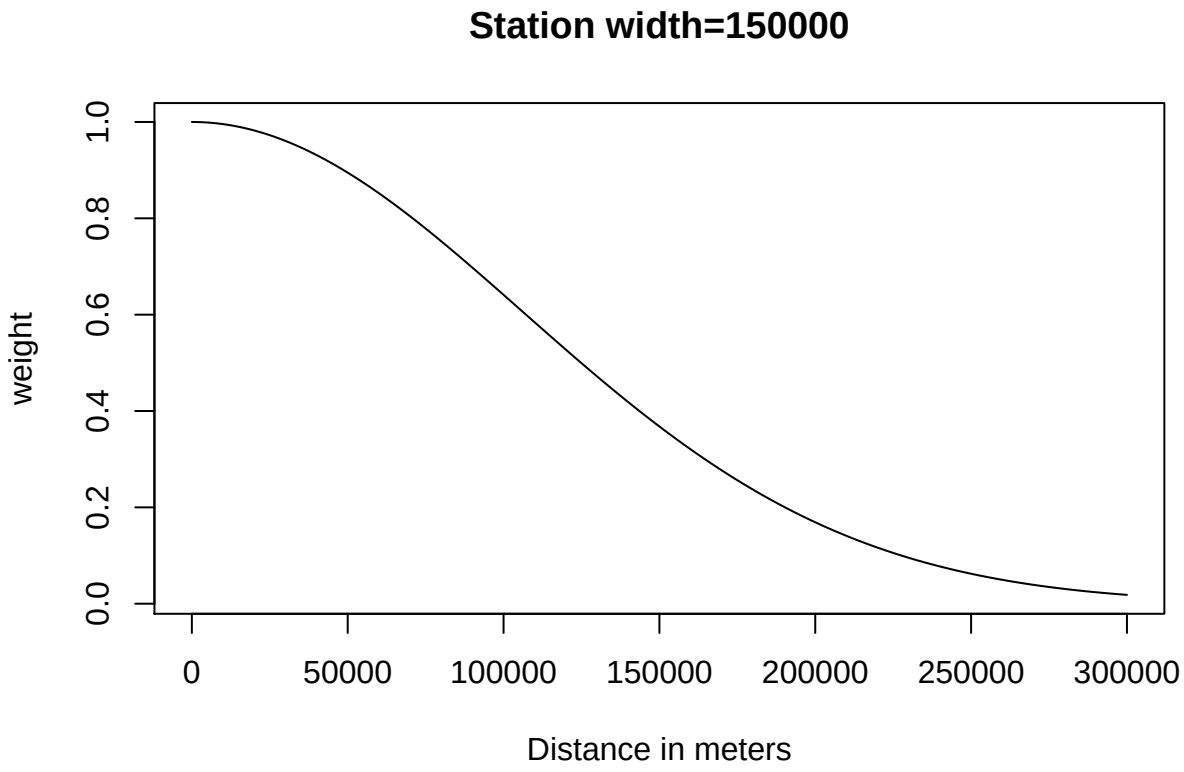
Lab 3

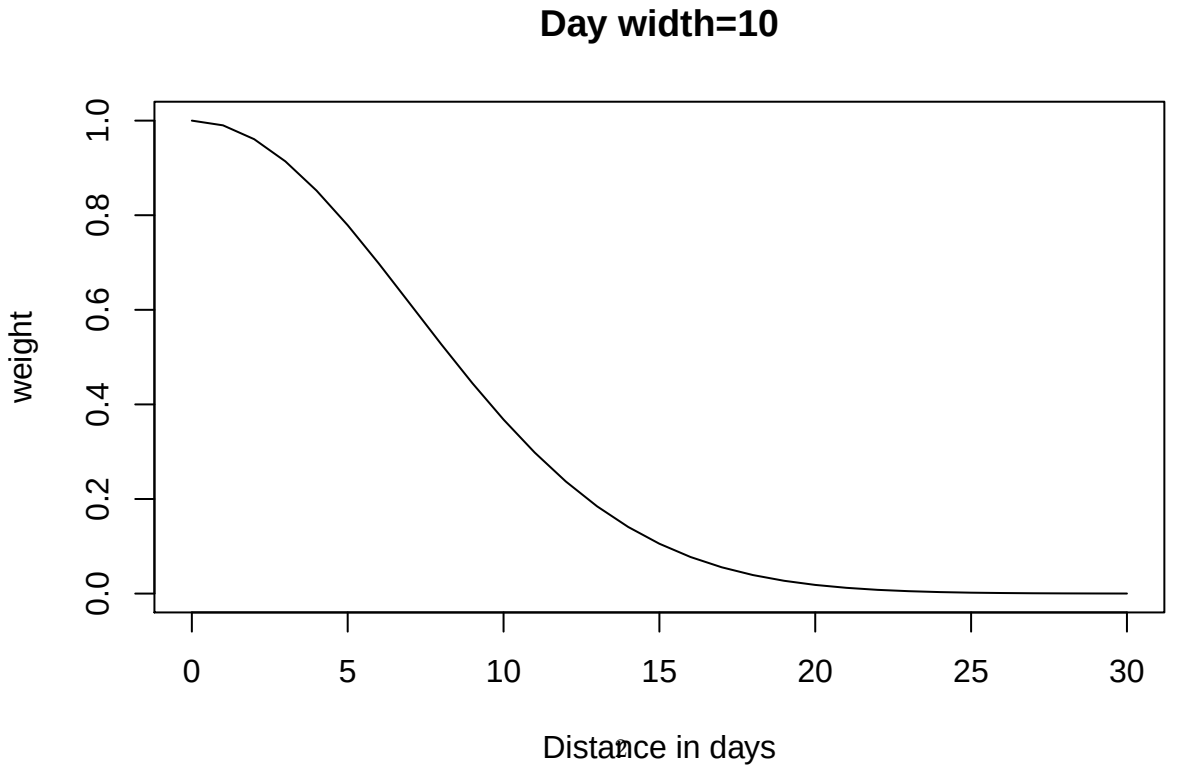
Dylan Maeenpaeae (dylma900), Arturas Aleksaundrauskas (artal938)

2019-12-16

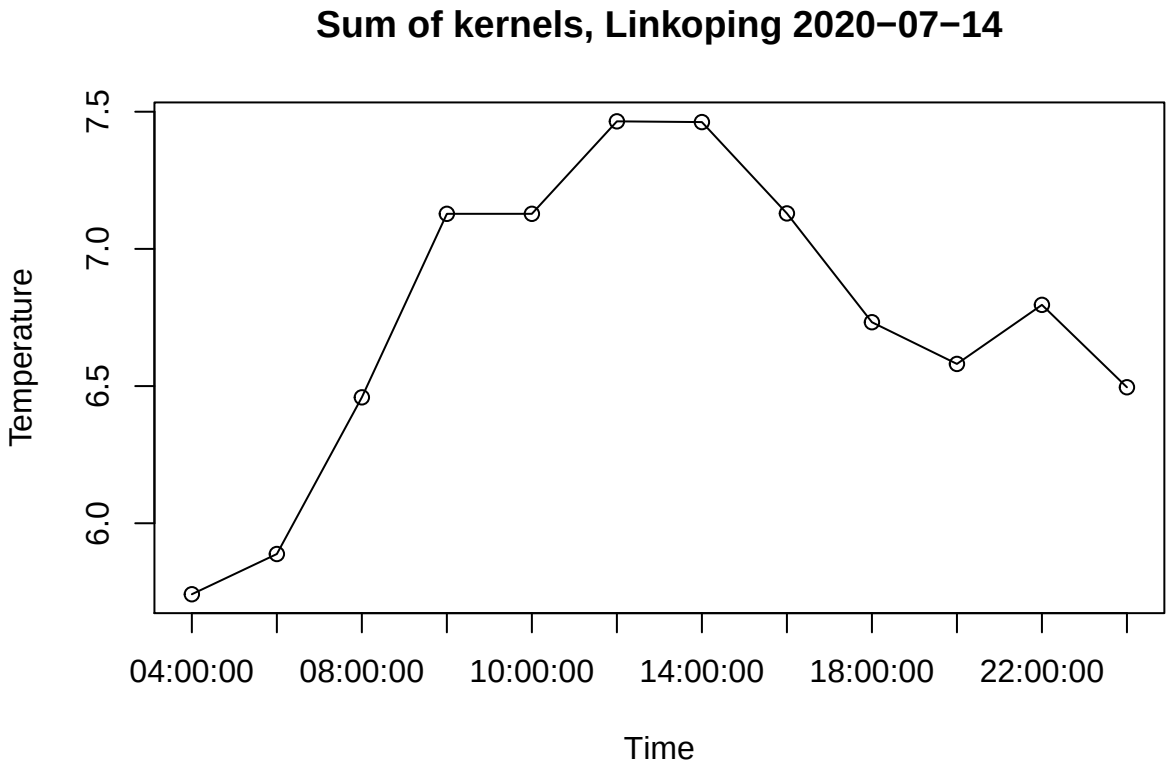
Assignment 1. Kernel Methods

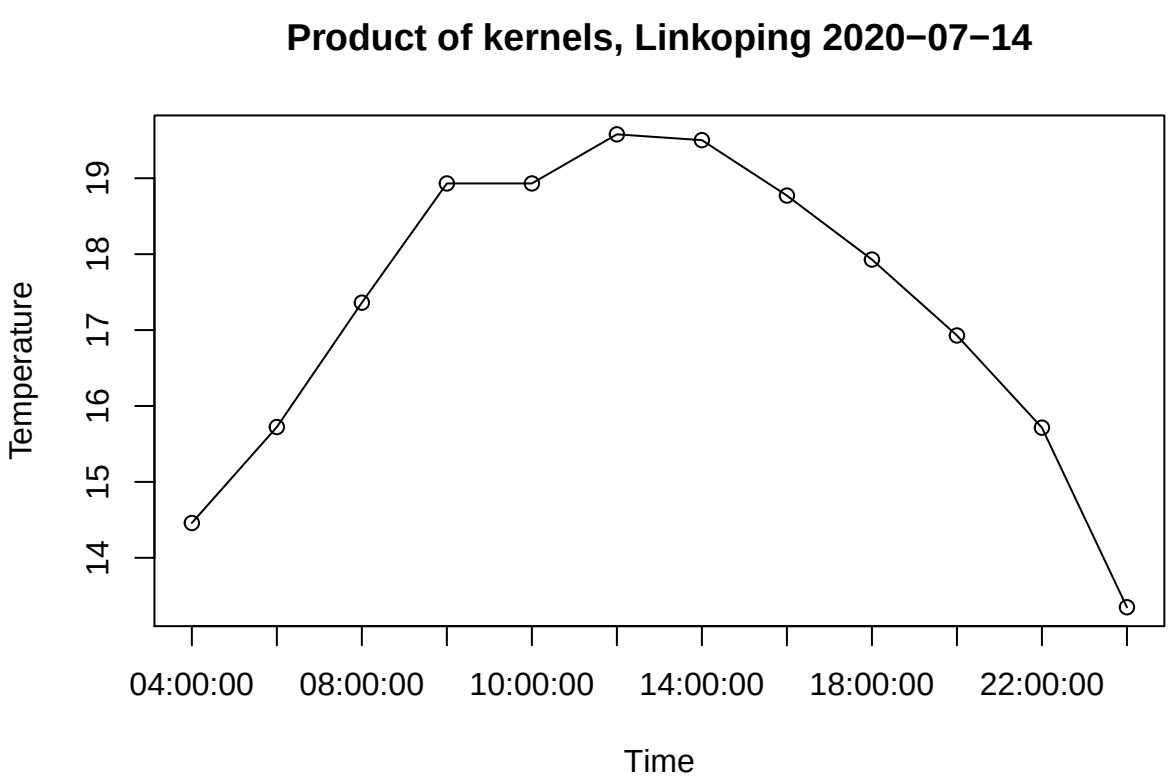
1)





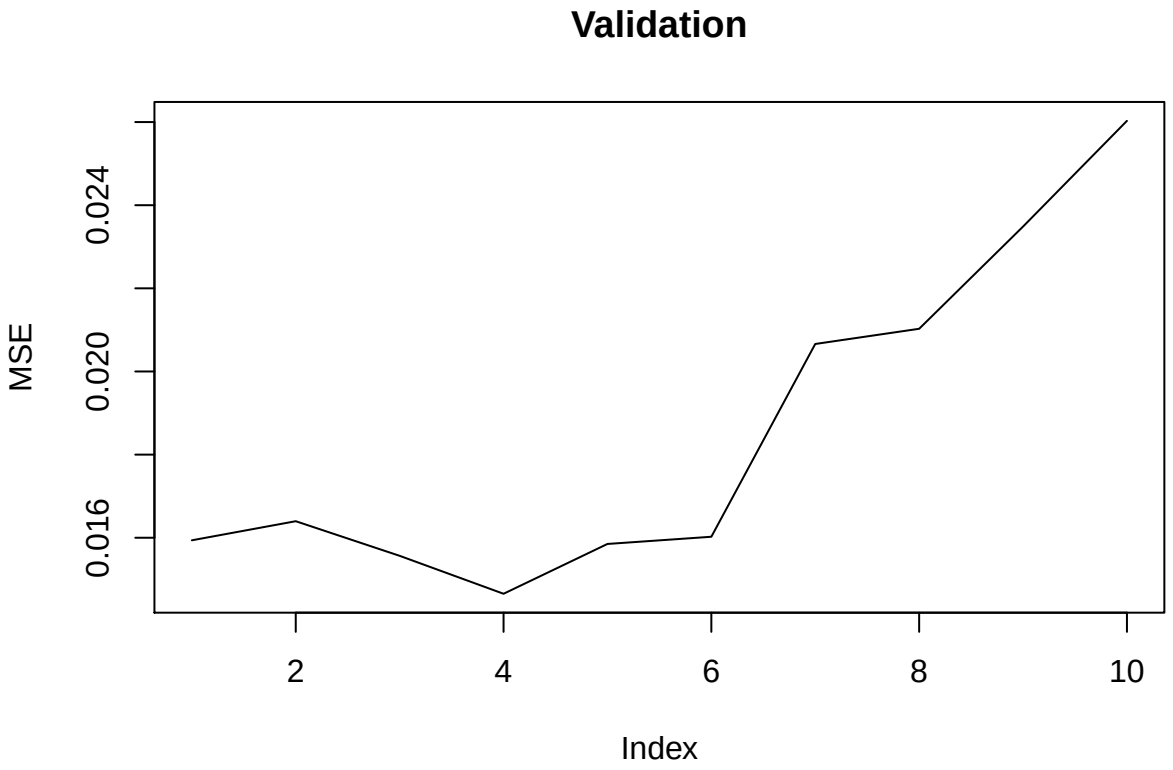
The above plots shows that the kernel widths are sensible, they show that the kernels gives more weight to the closer measurements. The different widths were chosen with some trial and error and some analysis of meaning of them. It is reasonable for stations closer in a radius of 150km to give information about the temperature in the query point. The same reason is applied to hour and date, which is 3 for hour, and 10 width for day. A measurement that was measured at hour difference close to 3 is likely to give information about the query point. For a day, it would not be reasonable to have for example 150 as width, since that will make a query for summer day take winter days in to account, therefore 10 was chosen as width.


Sum of kernels, Linkoping 2020−07−14
Temperature
6.0
6.5
7.0
7.5
04:00:00
08:00:00
10:00:00
14:00:00
18:00:00
22:00:00
Time

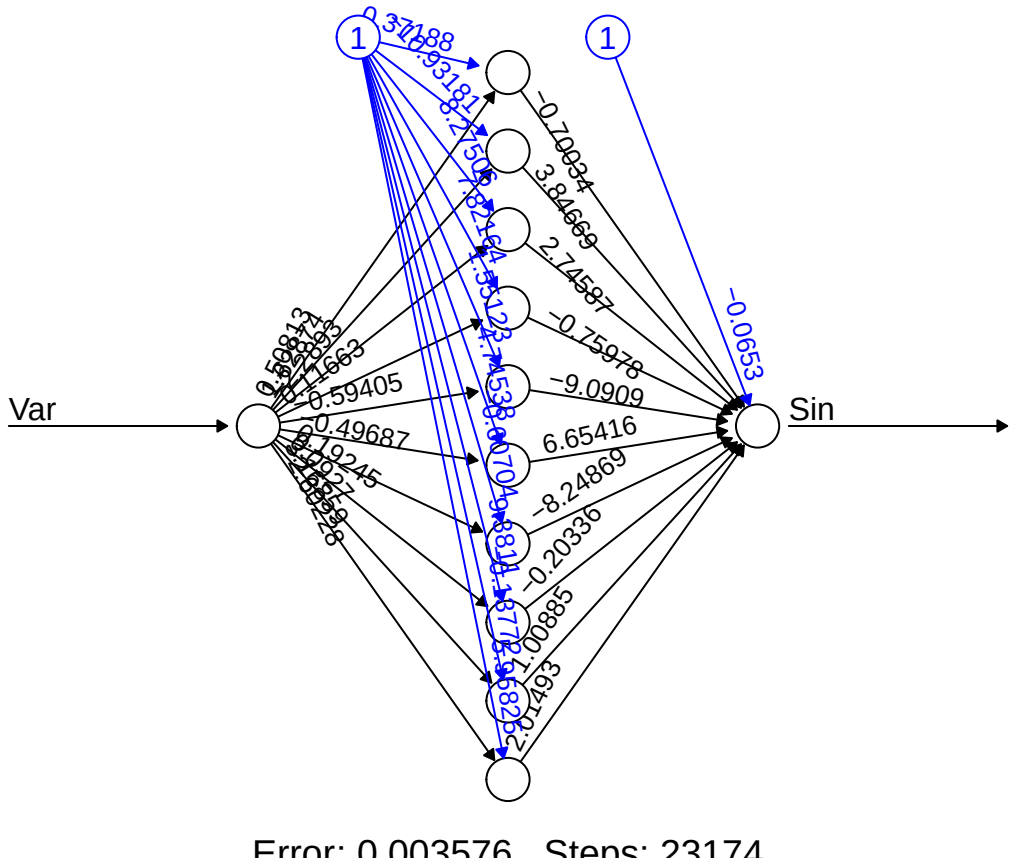


The chosen date was 2020-07-14 in Linkoping. As seen in the sum of kernels, the predicted temperatures is not reasonable since we get a temperature around 7 degrees, which is low for a summer day. The product of kernels shows much more reasonable values, as we get temperatures ranging from 14-19 which can be considered summer temperatures. An explanation for the different values is that the sum of kernels is capable to take measurements in account that is weighted high in two kernels but not in a third. However, in the product of kernels, all three kernels must have high weight values in order for the model to take them in account since a kernel that gives a low value, will impact the result a lot. Therefore all three kernels must have high weights in order for the resulting product to be high. Because of this behavior, I think that the product of kernels is more reasonable, which the plots above also indicates.

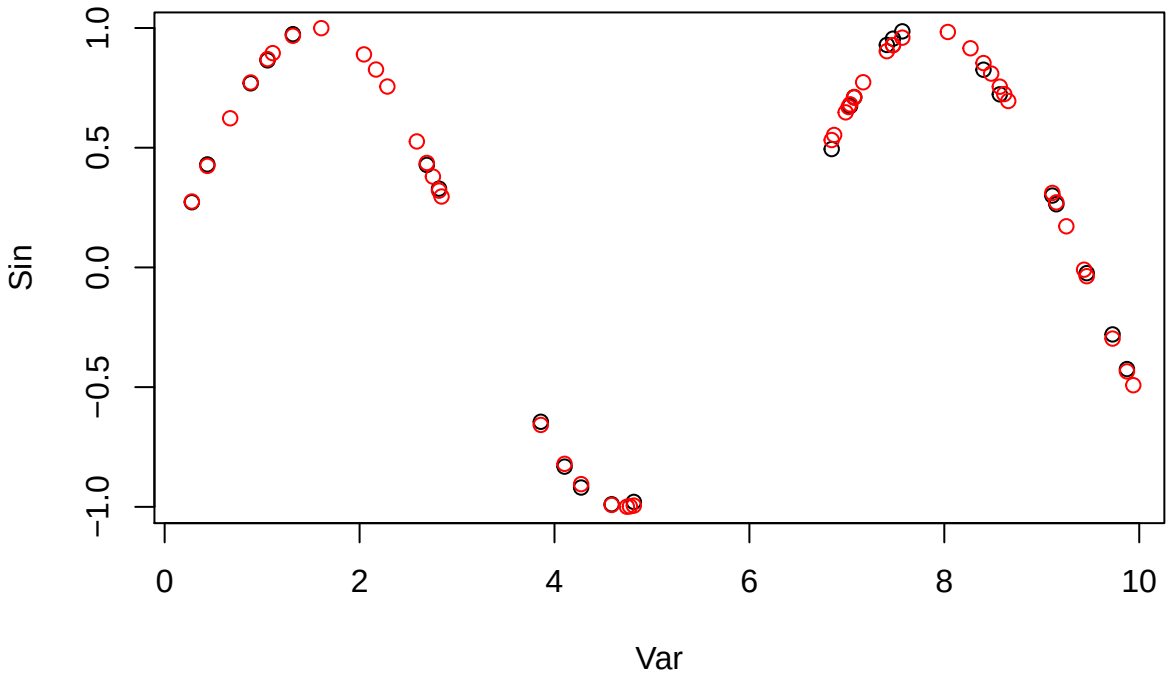
Assignment 3. Neural networks



The selection of the threshold is based on the mean squared error on the validation set. We can see that 4/1000 gives the smallest loss, thus this value is chosen.


1
1
Var
Sin
−0.59405
−0.49687
−0.70034
3.84669
2.74597
−0.75978
−9.0909
6.65416
−8.24869
−0.20336
−1.00885
2.01493
−0.0653
Error: 0.003576 Steps: 23174

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



The above plots shows the network and predictions and real values.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
RNGversion('3.5.1')
Sys.setlocale("LC_ALL", "C")
### 1
set.seed(1234567890)
library(geosphere)
stations <- read.csv("../stations.csv")
temps <- read.csv("../temps50k.csv")
st <- merge(stations,temps,by="station_number")
gaussian_kernel <- function(dist) {
  return(exp(-dist^2))
}

station_kernel <- function(st, lat, lon, station_h) {
  pred_point = c(lon, lat)
  stations_points = data.frame(st$longitude, st$latitude)
  dist = distHaversine(stations_points, pred_point)/station_h
  return(gaussian_kernel(dist))
}
```

```r
day_kernel <- function(st, date, h_day) {
  day_diffs = 365-abs(as.numeric(strftime(st$date, format="%j")) - as.numeric(strftime(date, format="%j
  day_diffs2 = abs(as.numeric(strftime(st$date, format="%j")) - as.numeric(strftime(date, format="%j")))
  for(i in seq(0, length(day_diffs), 1))
  {
    day_diffs[i] = min(day_diffs[i], day_diffs2[i])
  }
  day_diffs = day_diffs / h_day
  return(gaussian_kernel(day_diffs))
}



hour_kernel <- function(st, hour, h_hour) {
  dist = as.numeric(strptime(st$time, format="%H:%M:%S") - strptime(hour, format="%H:%M:%S"))/(3600*h_ho

  return(gaussian_kernel(dist))
}

#Link??ping
lat <- 58.41086
lon <- 15.62157
date <- "2020-07-14"
times <- c("04:00:00", "06:00:00", "08:00:00", "10:00:00", "10:00:00", "12:00:00" , "14:00:00", "16:00:0
h_distance <- 150000
h_day <- 10
h_hour <- 3
# Plot k against distance to find good h-values
station_dist<- seq(0,300000,20)
date_dist <- seq(0,30,1)
hour_dist <- seq(0,10,1)

station_vals_plot = gaussian_kernel(station_dist/h_distance)
date_vals_plot = gaussian_kernel(date_dist/h_day)
hour_vals_plot = gaussian_kernel(hour_dist/h_hour)
plot(station_dist, station_vals_plot, xlab="Distance in meters", ylab="weight", main="Station width=150
plot(date_dist, date_vals_plot, xlab="Distance in days", ylab="weight", type ="l", main="Day width=10")
plot(hour_dist, hour_vals_plot, xlab="Distance in hours", ylab="weight", type="l", main="Hour width=3")
station_vals = station_kernel(st, lat, lon, h_distance)
day_vals = day_kernel(st, date, h_day)
temps <- vector(length = length(times))
temps_prod <- vector(length = length(times))
for(i in seq(0, length(times), 1)) {
  hour = times[i]
  hour_vals = hour_kernel(st, hour, h_hour)
  total_sum = station_vals + day_vals + hour_vals
  total_product = station_vals * day_vals * hour_vals
  temps[i] = sum(total_sum*st$air_temperature)/sum(total_sum)
  temps_prod[i] = sum(total_product*st$air_temperature)/sum(total_product)
}

plot(temps, xaxt = "n", xlab="Time", ylab="Temperature", type="o", main = "Sum of kernels, Linkoping 20
axis(1, at=1:length(times), labels=times)
```

```r
plot(temps_prod, xaxt = "n", xlab="Time", ylab="Temperature", type="o", main = "Product of kernels, Lin
axis(1, at=1:length(times), labels=times)
#install.packages("neuralnet")
library(neuralnet)
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

# Random initialization of the weights in the interval [-1, 1]
winit <- runif(31, -1, 1)
error <- numeric(10)
for(i in 1:10) {
  nn <- neuralnet(Sin~Var, tr, 10, i/1000, startweights=winit)
  # print(nn$weights)
  pred <- predict(nn, va)
  #Compute MSE
  error[i] = mean(sqrt((va$Sin - pred)^2))
}

plot(error, type="l", main = "Validation", ylab="MSE")
nn <- neuralnet(Sin~Var, tr, 10, 4/1000, startweights=winit)
plot(nn, rep="best")
# Plot of the predictions (black dots) and the data (red dots)
plot(prediction(nn)$rep1)
points(trva, col = "red")
```