Introduction to Machine Learning 732A95

Lab 3 Block 1

Eric Herwin
erihe068

Contents

1. Kernel methods

In this task I will implement a kernel method to predict the hourly temperatures for a date and place in Sweden. The forecast are consisting of the predicted temperatures from 4 am to 24 pm in an interval of 2 hours. The kernel I am to implement are a sum of three Gaussian kernels. The Gaussian kernel is represented as:

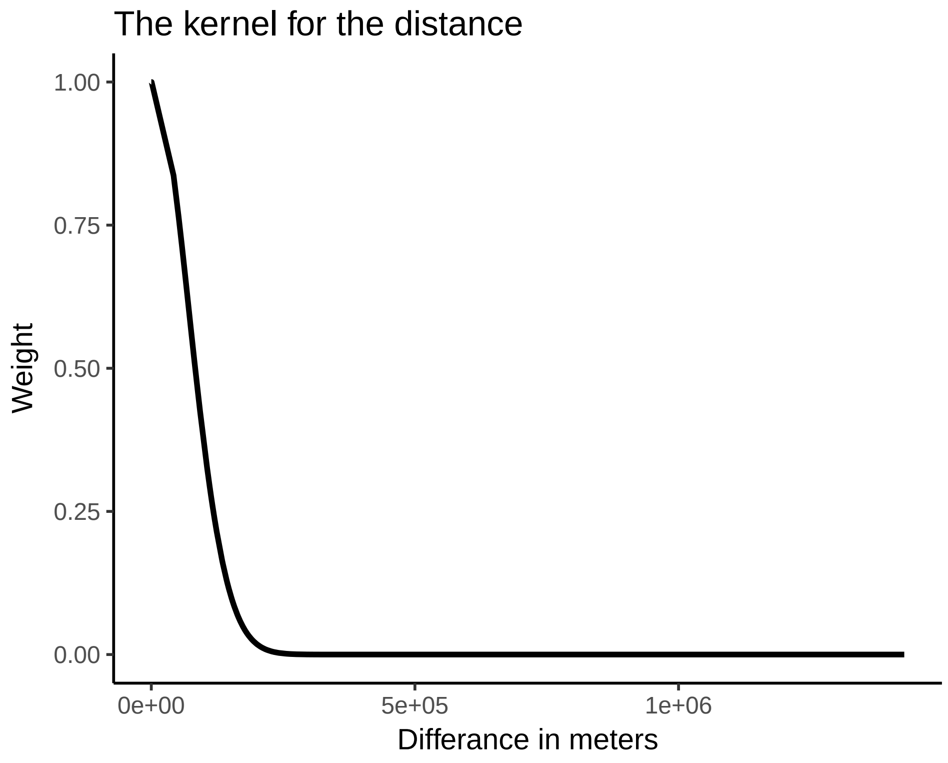
$$k(\mathbf{x}, \mathbf{x}') = exp\left(\frac{\|||\mathbf{x} - \mathbf{x}'||^2}{h^2}\right)$$

where $\mathbf{x}'$ is the point of interest (prediction).

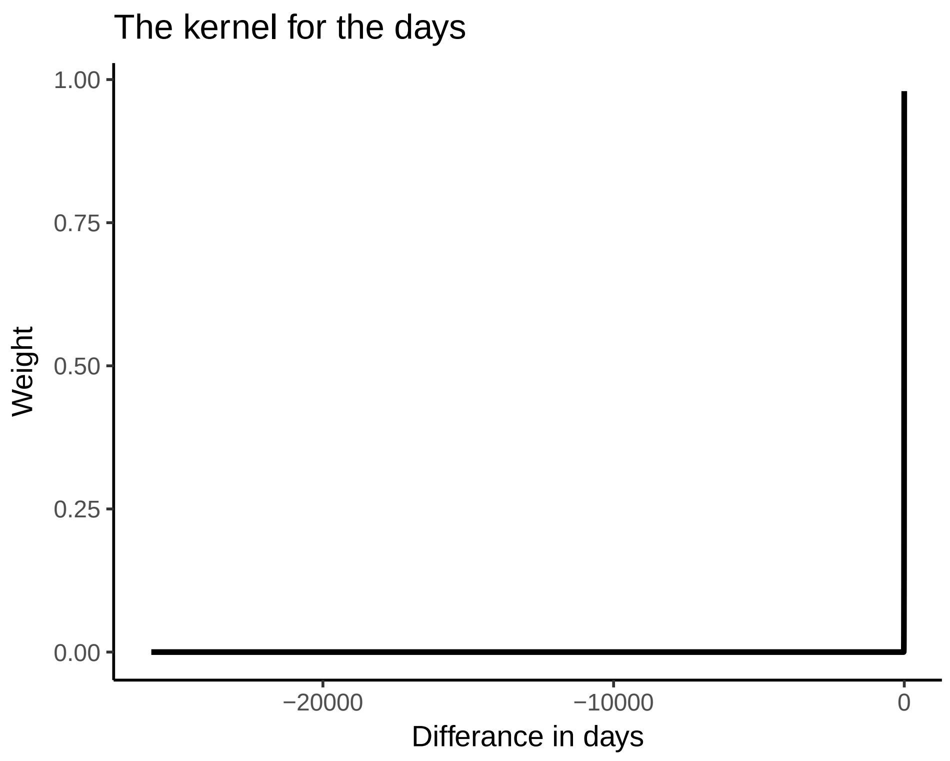
The kernels are taking in to account of the position of the prediction, the day pf the prediction and the hours of the day. The h in the formula are set to: $h_{distance} = 100000$ due to the temperature should be reasonable within a 100km of the prediction location. $h_{date} = 7$ due to that temperatures are not reasonably the same for each week (7 days in a week). $h_{time} = 2$ due to that the temperature can easch second hour of the day.

The data has been sorted so that the date, and the dates after my point of interest, are taken away.

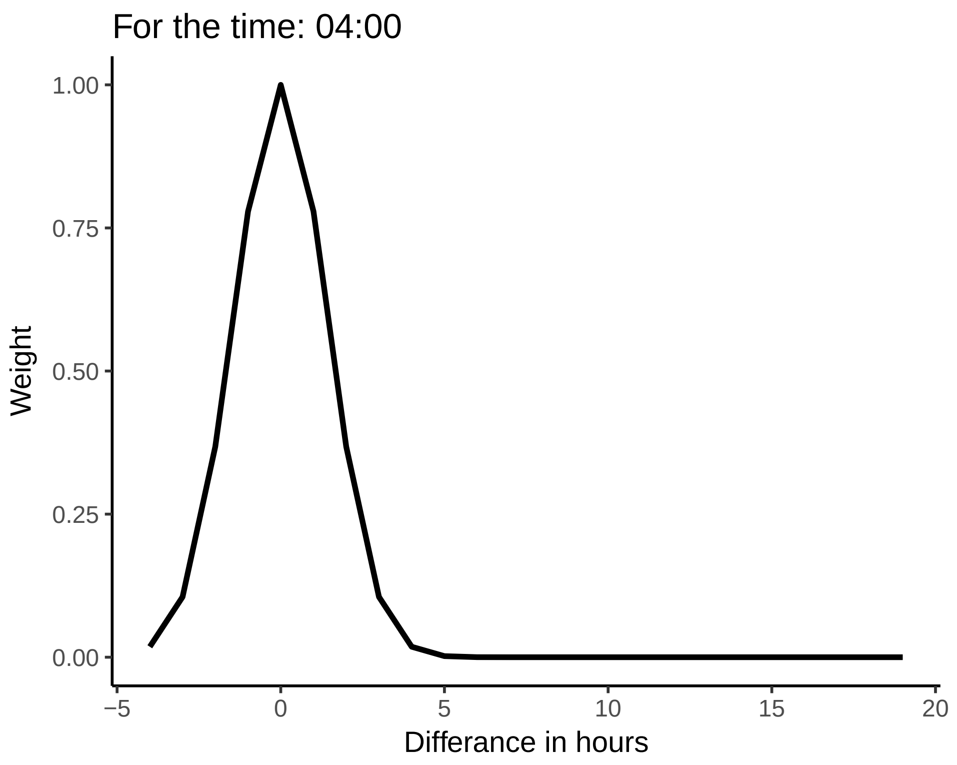
I will first inspect that the h for the different kernels has been chosen appopriately. I will start be looking at the kernel with the distance. The first kernel I will inspect is the distance kernel and the distance has been calculated with the Haversine Distance, which takes in to consideration that the earth is round.



We can see that the closest places of the observation are given higher weight. Next we look at the weights for the kernels with days.



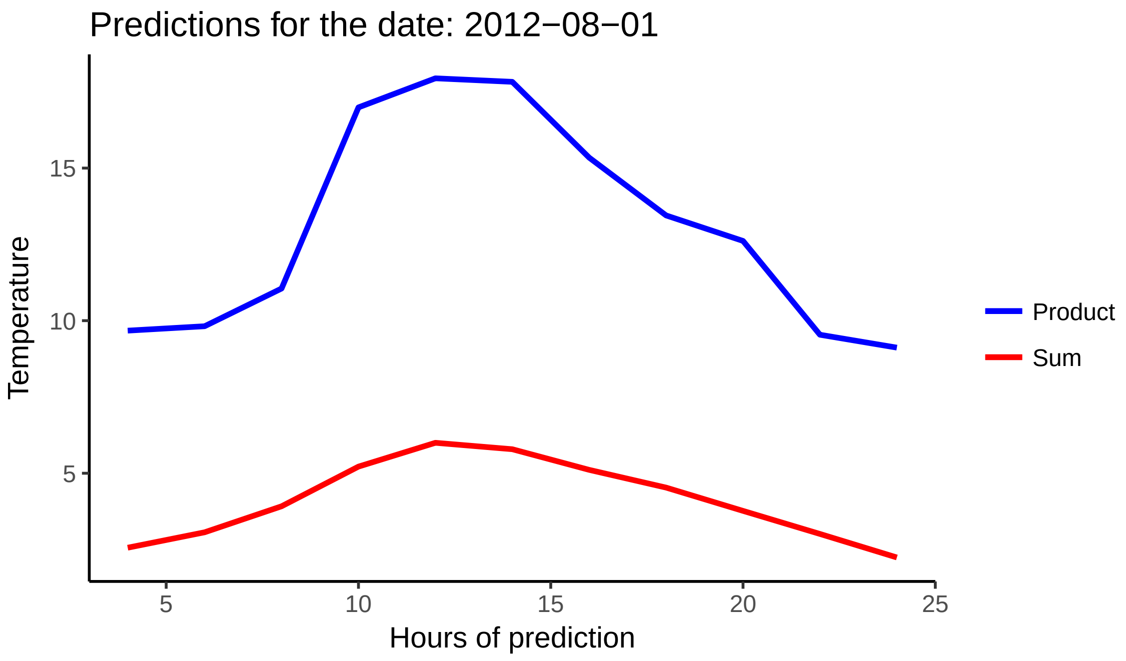
We can see that the closest days of the observation are given higher weight, and the reason why this plot seems so non-smooth is beacuse the low h compared to the values in data. Next we look at the weights for the kernels with time of the day.



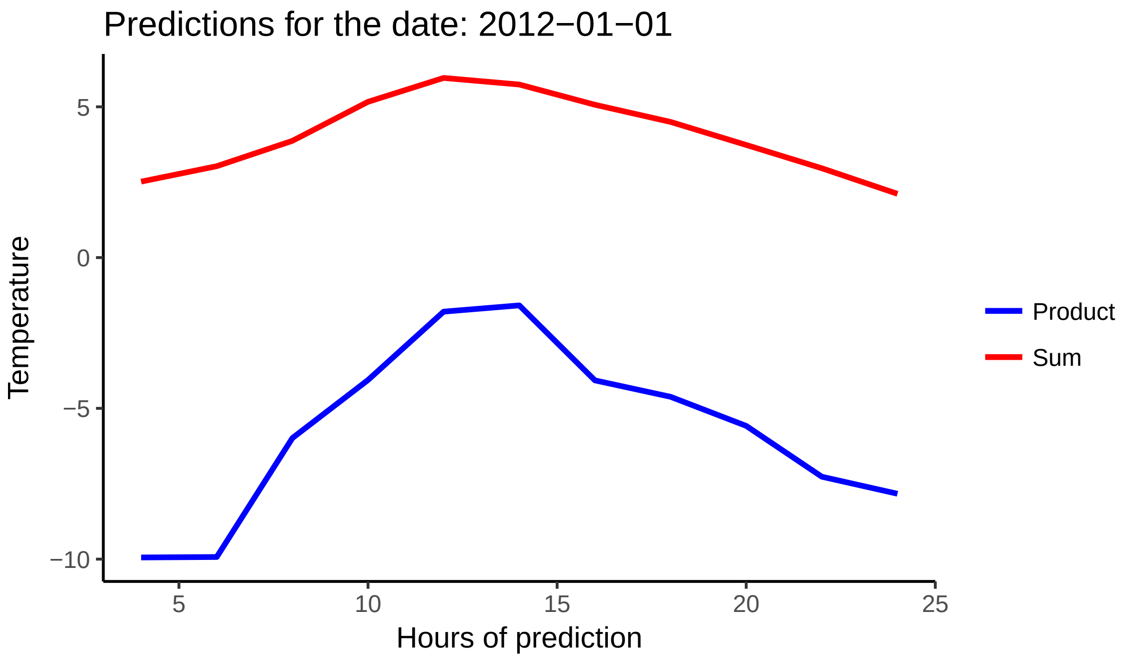
We can see that the closest hours for the observation have the highest weight.

I will do an prediction for latitude = 68.4284 and longitude = 18.1302 (Riksgränsen, Sweden) for the date

2012-08-01. The kernel are supposed to be a sum of kernels, but I will both perform a sum and a product of kernels.



We can see that the prediction is sensible in terms of the shape of the prediction. Due to that the temperature is low on mornings and evenings is logical and that it is a little bit higher during the day. We can note that the product of sums are predicting a higher temperature then the sum. Let's look at the predictions for whe date 2012-01-01.



For this time the product of kernels are much more lower in temperature and the sum are still on the plus temperatue, but lower then before. The product takes a much more "extreme" prediction for the time of the year. This might because in the kernel where I take in to consideration of the date, I only account for the closest days to the prediction due to my choosen h.

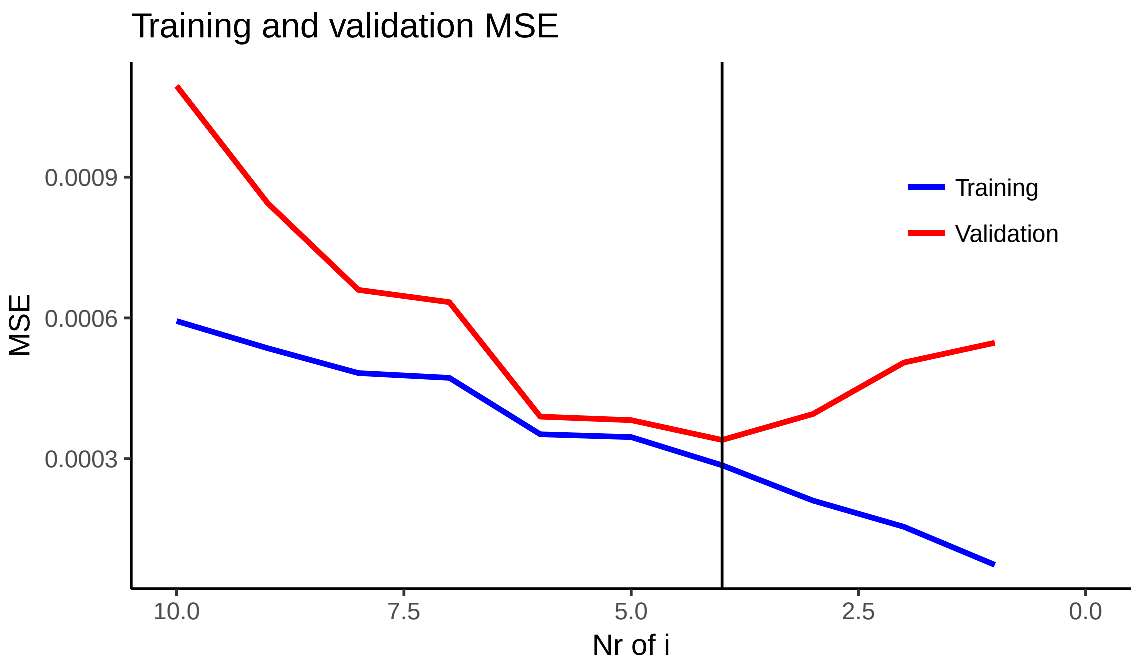
This behaviour for the temperature is because when you take the sum of kernels you are also considering the furthest observations back in time in to the prediction of the temperatue. We would not want to take the temperature of ex. 1970-01-01 when we perform the prediction of 2012-01-01, but we would still want to take the closest years/time in to the caculation. So a note for this kind of predictions is that the prediction would probably be better if we where take a seasonality in to consideration, and not just how far the days are, the prediction would be more accurate.

The product of kernels have a much more realistic prediction in this case. This is due to that the furthest observations back in time, for the date kernel, will have a vey low weight and therefore those observations will be close to 0. Then making the prediction will take the closest observations in to consideration, and in this case for January at Riksgränsen, Sweden will have a much more colder temperature.

2. Neural networks

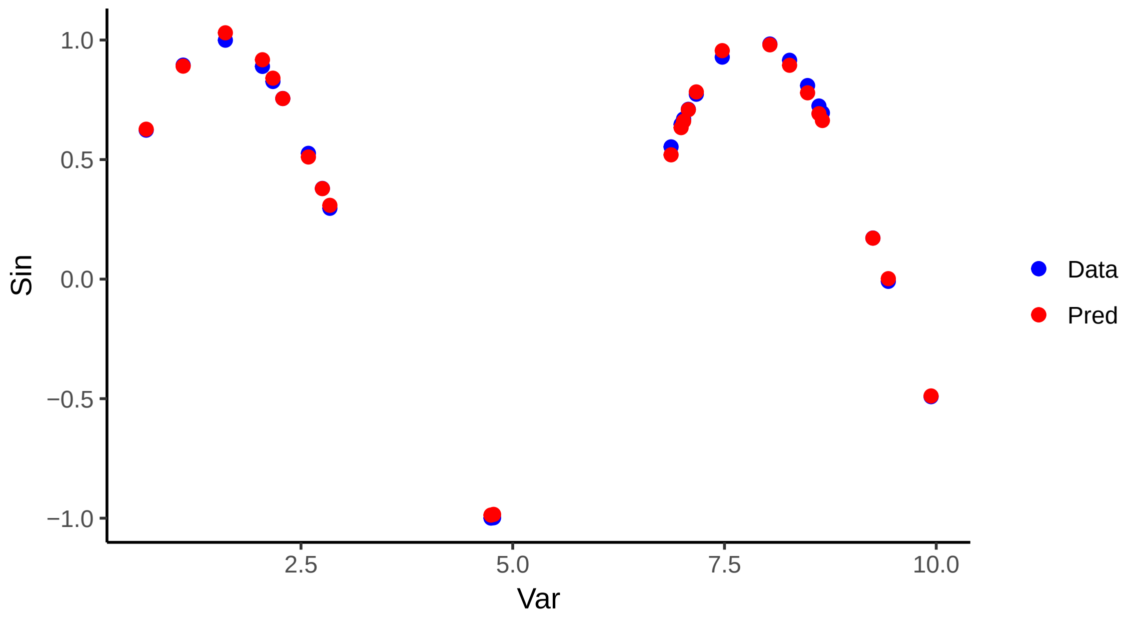
In this task I will train a neural network to learn the trigonometric sine function. I will do this with the function `neuralnet`.

I will have data in 50% in training and 50% validation. I will use the validation set to early stop of the gradient descent. So I will use a threshold to stop the gradient descent where the threshold $= i/1000$ and $i = 1, ..., 10$. The default gradient descent are back probagation. I will initilize weights (w_j and w_k) for the neural network where the weights are random values between [-1, 1]. I will use a neural network with a single hidden layer of 10 units.

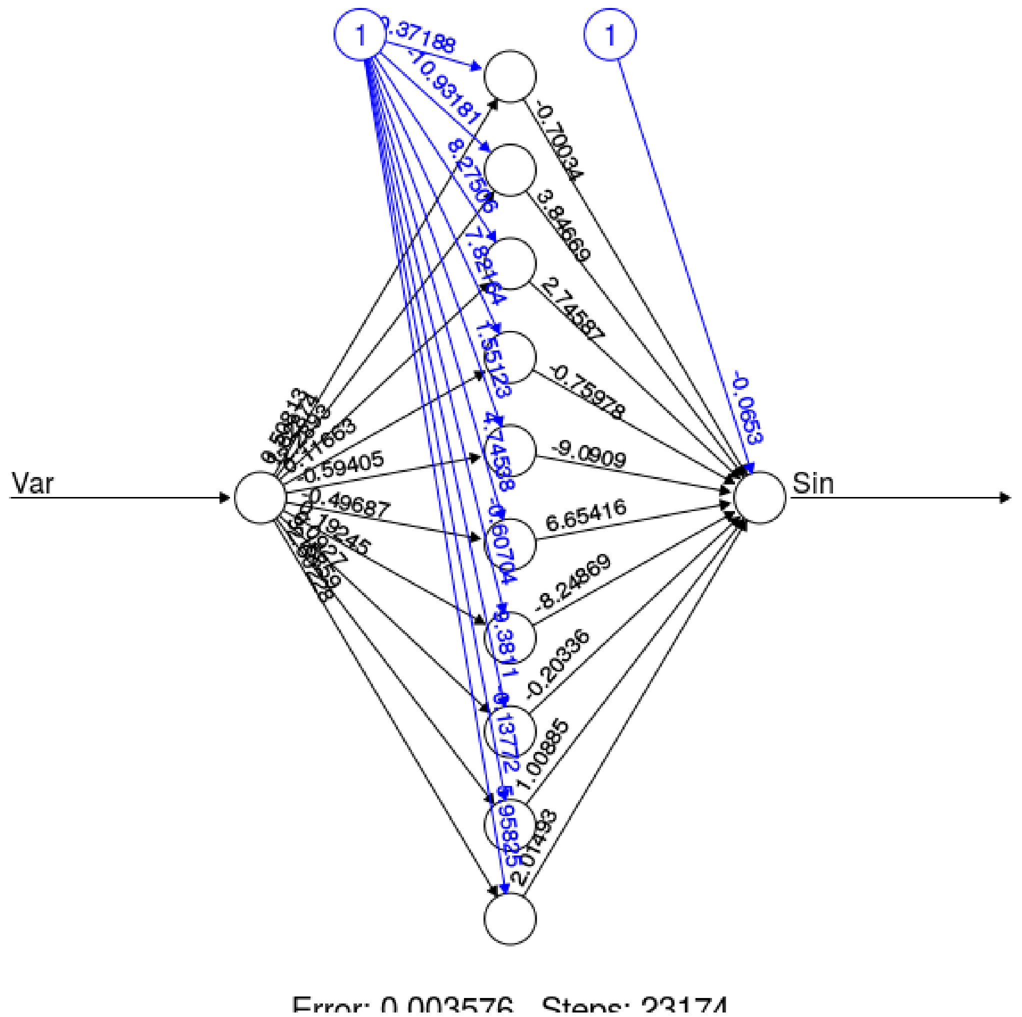


From the plot we can conclude that the number of $i = 4$ is ideal, so the threshold will become $4/1000 = 0.004$. I have choosen this beacuse the validation MSE is lower at this point. We can also note that the training MSE is at it lowest when $i = 1$, and this is because the gradient has found a minimum/ local minimum. So we can say that the lower the i is, the more overfitted the model will be.

For the threshold when $i = 4$ and for the training data, I have the prediction:



We can see that for the theshold the data with prediction looks really good. The final network looks like:



In the plot we can see all the 31 weights as the numbers in the plot and that the variable `Var` is the input variable and the `Sin` is the output.

Appendix

1

```
kern_pred <- function(latitude, longitude, date, data_smhi ){
  ### pre-pross step
  date_pred <- date
  sort_data_date <- data_smhi[order(date(data_smhi$date)),]
  #clean the data
  pos_to_clean <- which(sort_data_date$date == date_pred)
  #take data up to the date of interest
  sort_data_clean <- sort_data_date[1: (min(pos_to_clean)-1),]

  h_date <- 7
  h_dist <- 100000
  h_time <- 2

  #########1:th kern#########
  pr_min <- c(longitude, latitude)

  not_fun_R <- as.character(sort_data_clean$longitude)
  not_fun_R_2 <- as.character(sort_data_clean$latitude)
  to_dat <- data.frame(longitude = (as.numeric(not_fun_R)), latitude = (as.numeric(not_fun_R_2)))
  #assume a spherical earth with distHaversine
  hav_dist <- distHaversine(p2 = to_dat, p1 = pr_min)
  ker1 <- exp(- (((hav_dist)^2)/(h_dist)^2) )

  ##for ploting
  hav_dist_plot <- hav_dist
  ker1_plot <- ker1


  #########2:th kern#########
  #do so that the difference is only dependent on the days of the year, not the days and the year.

  diff_days <- date(sort_data_clean$date) - date(date_pred)
  diff_days <- as.numeric(diff_days)

  ker2 <- exp(-((diff_days^2)/(h_date)^2))

  ##for ploting
  diff_days_plot <- diff_days
  ker2_plot <- ker2

  #########3:th kern#########
  pred_time <- seq(hour(hms("04:00:00")), hour(hms("24:00:00")), 2)

  time_list <- list()
  n <- 0
  for(pred in pred_time){
    n <- n+1
    diff_time <-  hour(hms(sort_data_clean$time)) - pred
    ker3 <- exp(  -((abs(diff_time)^2)/(h_time)^2))
    time_list[[n]] <- ker3
  }

  ##for ploting
  diff_time_plot <- hour(hms(sort_data_clean$time)) - pred_time[1]
  ker3_plot <- time_list[[1]]



  ## pred ##
  t_n <- as.character(sort_data_clean$air_temperature)

  preds_sum <- c()
  preds_prod <- c()
  for(i in 1:length(time_list)){

    sum_of_kern_sum <- (ker1 + ker2 + time_list[[i]])
    sum_of_kern_prod <- ker1* ker2 * time_list[[i]]

    preds_sum[i] <- sum(sum_of_kern_sum * as.numeric(t_n)) / sum(sum_of_kern_sum)
    preds_prod[i] <- sum(sum_of_kern_prod * as.numeric(t_n)) / sum(sum_of_kern_prod)

  }


  ########### Plotting area ###########
  ##pred plot##
  pred_plot <- ggplot() +
    geom_line(aes(x = pred_time, y = preds_sum, colour = "Sum"), size = 1) +
    geom_line(aes(x = pred_time, y = preds_prod, colour = "Product"),  size = 1) +
    scale_color_manual(values = c("blue", "red")) +
    labs(colour = "", title = paste("Predicions for the date:", date), x = "Hours of prediction", y = "
    theme_classic()

  ## time proving plot ##
  diff_time_plot_1 <- ggplot() +
    geom_line(aes(x = diff_time_plot, y = ker3_plot), size = 1)+
    labs(title = "For the time: 04:00", x = "Differance in hours", y = "Weight")+
    theme_classic()

  #ploting just for the closest 183 days
  diff_days_plot_1 <- ggplot() +
    geom_line(aes(x = diff_days_plot, y = ker2_plot), size = 1)+
    labs(title = "The kernel for the days", x = "Differance in days", y = "Weight")+
    theme_classic()

  #ditance weights plot
  hav_dist_plot_1 <- ggplot() +
    geom_line(aes(x = hav_dist_plot, y = ker1_plot), size = 1)+
    labs(title = "The kernel for the distance", x = "Differance in meters", y = "Weight")+
    theme_classic()
```

```r
  ret_list_plots <- list(hav_dist_plot = hav_dist_plot_1,
                         diff_days_plot = diff_days_plot_1,
                         diff_time_plot = diff_time_plot_1,
                         pred_plot = pred_plot)
  return(ret_list_plots)
}
```

2

```r
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(n = 31, min = -1, max = 1)

#Use a neural network with a single hidden layer of 10 units
mse_nn_tr <- c()
mse_nn_va <- c()
for(i in 1:10) {
    nn <- neuralnet(formula = Sin ~ Var, data = tr,
                          threshold = i/1000,
                          startweights = winit, hidden = 10 )
    #training
    pred_nn <- compute(nn, covariate = tr[,1])$net.result
    #validation
    pred_va_nn <- compute(nn, covariate = va[,1])$net.result

    mse_nn_tr[i] <- sum((tr$Sin - pred_nn)^2)/nrow(tr)
    mse_nn_va[i] <- sum((va$Sin - pred_va_nn)^2)/nrow(va)

}


nn <- neuralnet(formula = Sin ~ Var, data = tr,
                threshold = 4/1000,
                startweights = winit, hidden = 10)

pred_nn <- compute(nn, covariate = tr[,1])$net.result

#the plot for training and validation
ggplot() +
  geom_line(aes(x = 1:10, y = (mse_nn_tr), colour = "Training"), size = 1) +
  geom_line(aes(x = 1:10, y = (mse_nn_va), colour = "Validation"), size = 1) +
  scale_color_manual(values = c("blue", "red")) +
  geom_vline(xintercept = 4) +
  scale_x_reverse( lim=c(10,0)) +
  labs(colour = "", title = "Training and validation MSE", x = "Nr of i", y = "MSE") +
  theme_classic()+
  theme(legend.position = c(0.85,0.75)) #legend position
```