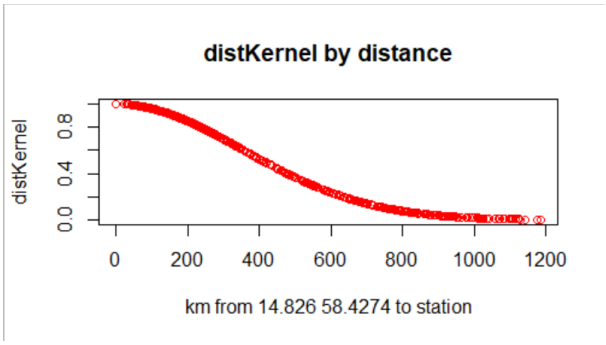
Assignment 1

Kernels

```
h_distance <- 50*10^4
h_date <- 2
h_time <- 4
```

The weights were chosen iteratively and changed intuitively to give temperatures weighted with reasonable correlated measurements.

h_distance

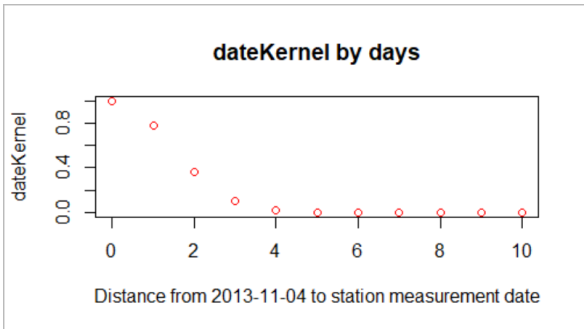


h_distance is set to 50*10*^5 which corresponds to 50 kms.

The range of the graph is about 1000 kms. This almost corresponds to the distance from Stockholm to Piteå. This makes sense, as the temperature at these two places should not correlate much, and especially not points further from each other than this. This is my rationale for my h_distance.

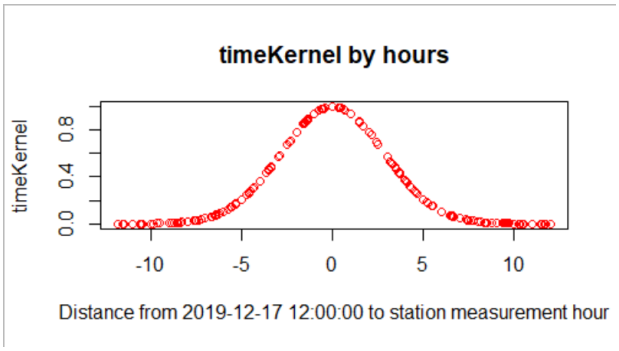
The graph clearly displays, through the negative trend, that station measurements further from the prediction point will be weighted less.

h_date



h_date = 2. This can be seen in the graph as the values decrease to zero shortly after 2 days. This makes sence as only the most nearby days should matter, and these should not be weighted much. The first day is weights at about 80% which makes sense to me, as weather patterns does not often change that fast. But over two days there is a higher risk for short term changes such as rain e.t.c.. This graph clearly weights measurements further away in time less.

h_time



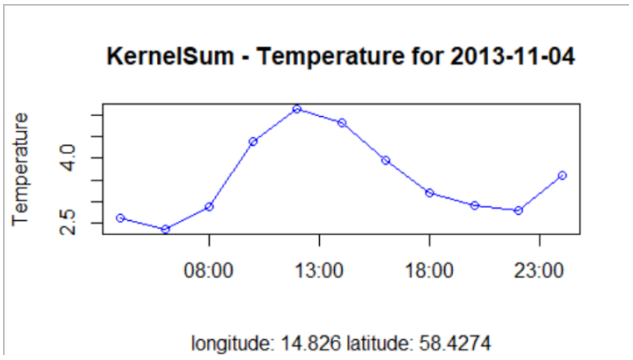
h_time is set to 4 hours. This to get a more smooth distribution for the kernels over the days as a larger smoothing factor will increase the overlap in kernels for different hours. This can be seen in the curve, especially the multiplicative, as it is much smoother than if smaller values are used. This graph clearly weights measurements further away in time less.

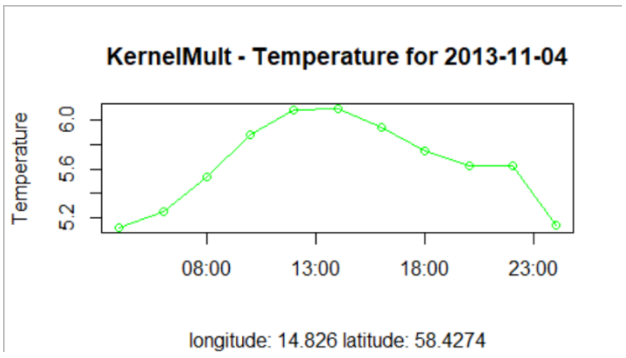
Sum vs multiplicative

Sum has more varied outputs, partly larger range of values and also a more complex pattern. Kernelsums temperature suddenly spikes at 2300 as well, which is unclear intuitively.

I think KernelMult has a much more likely shape, as it is more smooth and also displays a likely pattern for a days temperature. A rise in temperature before lunch, with a slight decrease after lunch folloed by a sharp decrease later at night.

I also looked at the sample MSE for the different methods with a batch size of 100 predictions. The predictions were chosen randomly from observed points of station measurements, and the coordinates for these points and its time and date were used for prediction with my kernels. The MSE were then computed by using the stations real value along with the predicted value.





testTemps calculates this MSE as specified above. The Boolean value at the end represents multiplicative model (true) or additive model (false)

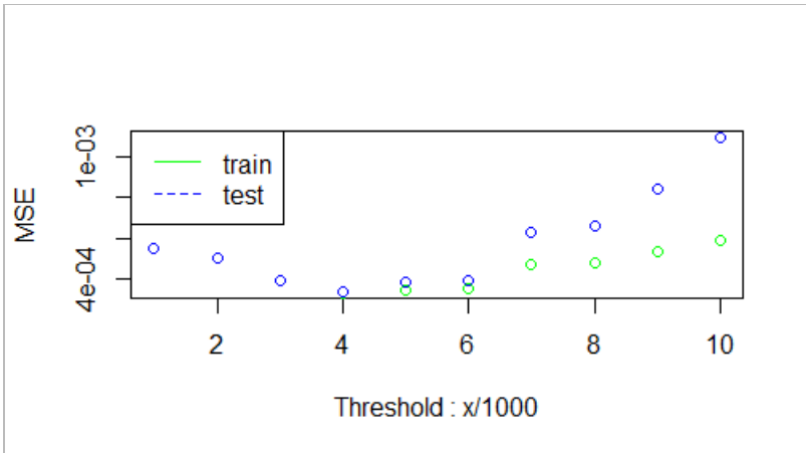
```
> testTemp(100,h_distance,h_date,h_time,TRUE)
[1] 32.12264
> testTemp(100,h_distance,h_date,h_time,FALSE)
[1] 93.4388
```

As observed, a multiplicative model has a third of the MSE of the additive model, which is quite significant, especially as this value is the mean *squared* value.

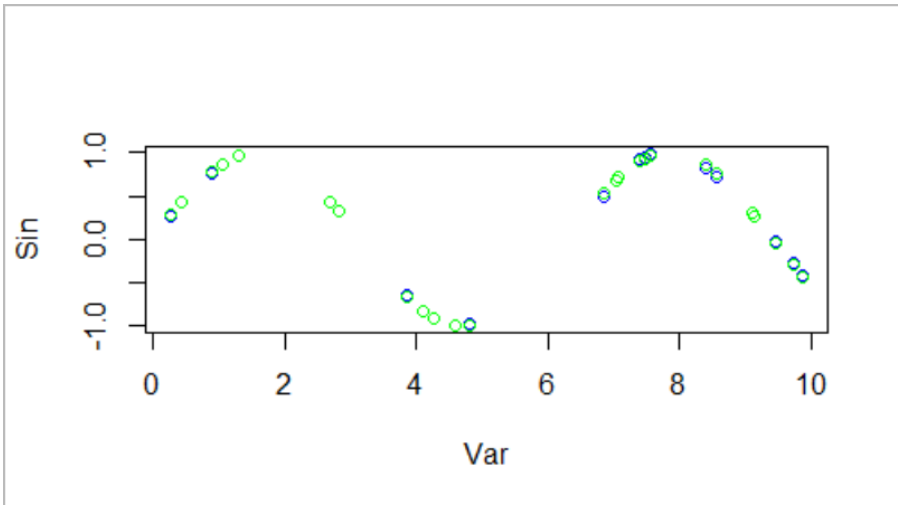
```
testTemp = function(B,h_distance,h_date,h_time,mult=FALSE){
  MSE = vector(mode="double",length(B))
  sample = sample(1:nrow(data.merge),B)
  for(i in 1:B){
    ts = data.merge[sample[i],]
    temp.pred = predictTemp(ts$longitude,ts$latitude,ts$time,
                            ts$date,h_distance,h_date,h_time,mult)
    MSE[i] = (temp.pred - ts$air_temperature)^2
  }

  return (sum(MSE)/B)
}
```

This is the code for the testTempfunction. which calculates the total MSE for all batches. predictTemps is my function for calculating the prediction with the kernels.

Assignment 3



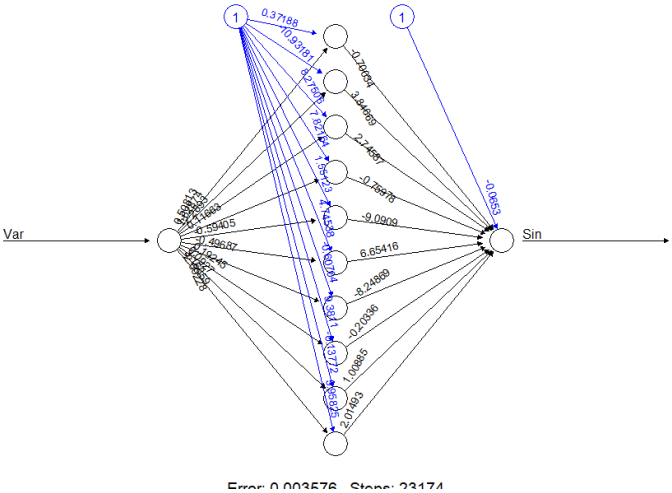
The tresholds are plotted out above. From this we observe that the best threshold is 4/1000. This as MSE for test is min here.



The above plot is the plot for the predicted values. Green points are predicted, and black points are the real values. As can be observed, most, if not all points are following the sin curve pretty closely.

Vikgu708

The resulting neuralnetwork is as follows.



Appendix 1 – Assignment 1 code

```
set.seed(1234567890)

library(geosphere)

library(ggplot2)


stations <- read.csv("stations.csv")

temps <- read.csv("temps50k.csv")

data.merge <- merge(stations,temps,by="station_number")

data.merge$date = as.Date(data.merge$date,format='%Y-%m-%d')

data.merge$time = strptime(data.merge$time,format = "%H:%M:%S")


predictTemp = function(longitude, latitude, time, date, h_distance, h_date,h_time,mult=FALSE){

  st = data.merge[which((data.merge$date < date) | (data.merge$date==date && data.merge$time<= time)),]

  #Distance Kernel

  station.positions = cbind(st$longitude,st$latitude)

  it.dist = distHaversine(station.positions,c(longitude,latitude))

  it.distKernel = distKernel(h_distance,st,longitude,latitude)

  it.dateKernel = dateKernel(h_date,st,date)

  it.timeKernel = timeKernel(h_time,st,time)

  if (mult){

    it.totKernel =  it.distKernel *  it.timeKernel * it.dateKernel


  }else{

    it.totKernel =  it.distKernel +  it.timeKernel + it.dateKernel

  }

  temp.Sum = sum(it.totKernel * st$air_temperature)/sum(it.totKernel)

  return(temp.Sum)

}


testTemp = function(B,h_distance,h_date,h_time,mult=FALSE){
```

```
  MSE = vector(mode="double",length(B))
  sample = sample(1:nrow(data.merge),B)
  for(i in 1:B){
    ts = data.merge[sample[i],]
    temp.pred = predictTemp(ts$longitude,ts$latitude,ts$time,
                            ts$date,h_distance,h_date,h_time,mult)
    MSE[i] = (temp.pred - ts$air_temperature)^2
  }

  return (sum(MSE)/B)
}

gaussianKernel = function(h,diff){
  u = (diff)/h
  return(exp(-u^2))
}

distKernel = function(h,xN,a,b){
  station.positions = cbind(xN$longitude,xN$latitude)
  it.dist = distHaversine(station.positions,c(a,b))
  return(gaussianKernel(h,it.dist))
}

dateKernel = function(h,xN,date){
  return(gaussianKernel(h,as.double(date-xN$date)))
}

timeKernel = function(h,xN,time){
  return(gaussianKernel(h,as.double(time-xN$time)/3600))
}
```

Vikgu708

```
 h_distance <- 50*10^4 # These three values are up to the students
  h_date <- 2
  h_time <- 4

  b <- 58.4274 # The point to predict (up to the students)
  a <- 14.826

date <- "2013-11-04" # 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")

date = as.Date(date,format='%Y-%m-%d')
times = strptime(times,format = "%H:%M:%S")

tempMult = vector(mode="double",length=length(times))
tempSum = vector(mode="double",length=length(times))
for(p in 1:length(times)){
  tempMult[p] = predictTemp(a,b,times[p],date,h_distance,h_date,h_time)
  tempSum[p] = predictTemp(a,b,times[p],date,h_distance,h_date,h_time,TRUE)
}

testTemp(100,h_distance,h_date,h_time,TRUE)
testTemp(100,h_distance,h_date,h_time,FALSE)

#Result
#par(mfrow=c(1,1))
plot(col="green",x=times,y=tempMult, type="o", main=sprintf("KernelMult - Temperature for
%s",date),xlab=" ",ylab="Temperature",sub=sprintf("longitude: %s latitude: %s",a,b))
plot(col="blue",x=times,y=tempSum, type="o", main=sprintf("KernelSum - Temperature for
%s",date),xlab=" ",ylab="Temperature",sub=sprintf("longitude: %s latitude: %s",a,b))
```

Vikgu708

S

#test kernels

station.positions = cbind(data.merge$longitude,data.merge$latitude)

it.dist = distHaversine(station.positions,c(a,b))

plot(x=it.dist/1000,y=distKernel(h_distance,data.merge,a,b),xlab=sprintf("km from %s %s to station",a,b),ylab="distKernel",main="distKernel by distance",col="red")

plot(x=as.double(date-data.merge$date),y=dateKernel(h_date,data.merge,date),xlab=sprintf("Distance from %s to station measurement date",date),ylab="dateKernel",main="dateKernel by days",col="red",xlim=c(0,10))

relTime = times[5]

plot(x=as.double(relTime-data.merge$time)/(3600),y=timeKernel(h_time,data.merge,relTime),xlab=sprintf("Distance from %s to station measurement hour",relTime),ylab="timeKernel",main="timeKernel by hours",col="red")

Appendix 2 – Assignment 3 code

```
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


MSE = function(nn, dat){

 return(sum((predict(nn,dat)-dat$Sin)^2)/nrow(dat))

}


MSE_train =c()

MSE_test =c()


winit = runif(31,-1,1)

 for(i in 1:10) {

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

  MSE_test[i] = MSE(nn,va)

  MSE_train[i] = MSE(nn,tr)


 }

 MSE_test

 MSE_train

 plot(MSE_test,col="blue",xlab="Threshold : x/1000",ylab="MSE")

 points(MSE_train,col="green")

 legend("topleft",legend=c("train","test"),col=c("green","blue"),lty=1:2)

 #plot(nn)

 nn = neuralnet(Sin ~
Var,data=tr,threshold=which.min(MSE_test)/1000,hidden=10,startweights=winit)

 plot(nn)
```

```
 # Plot of the predictions (black dots) and the data (red dots)
prediction(nn)$rep1
plot(x=va$Var,y=(predict(nn,va)), col="blue")
points(va$Var, va$Sin, col = "green")
legend("topleft",legend=c("observed","testPredictions"),col=c("black","blue"),lty=1:2)
```