Week Five Homework

Jacob Tarnowski

12/4/2019

set.seed(101)  
require(data.table)

## Loading required package: data.table

require(keras)

## Loading required package: keras

require(tidyverse)

## Loading required package: tidyverse

## -- Attaching packages ----------- tidyverse 1.3.0 --

## v ggplot2 3.2.1 v purrr 0.3.3  
## v tibble 2.1.3 v dplyr 0.8.3  
## v tidyr 1.0.0 v stringr 1.4.0  
## v readr 1.3.1 v forcats 0.4.0

## -- Conflicts -------------- tidyverse\_conflicts() --  
## x dplyr::between() masks data.table::between()  
## x dplyr::filter() masks stats::filter()  
## x dplyr::first() masks data.table::first()  
## x dplyr::lag() masks stats::lag()  
## x dplyr::last() masks data.table::last()  
## x purrr::transpose() masks data.table::transpose()

require(rsample)

## Loading required package: rsample

require(tseries)

## Loading required package: tseries

## Registered S3 method overwritten by 'xts':  
## method from  
## as.zoo.xts zoo

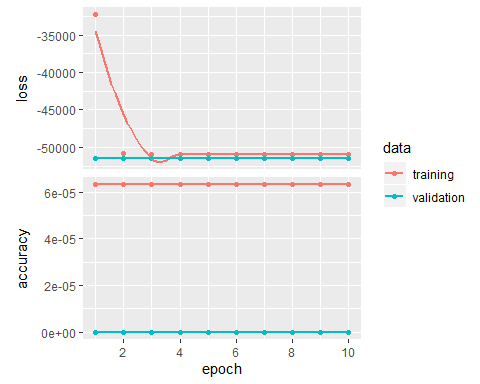
## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

#Question 1

#Load in the data  
df.elec <- fread("C:/Users/jaket/OneDrive/Desktop/GDAT 626/electricity.txt")  
  
#Split the data into test, training and target data  
elec\_split <- initial\_split(df.elec)  
train.elec<- training(elec\_split)  
test.elec <- testing(elec\_split)  
  
#Last column becomes the target data  
target.data <- train.elec[,321]  
train.elec <- train.elec[,-321]  
  
model <- keras\_model\_sequential()  
  
# define and compile the model (random parameters)  
model %>%   
 layer\_dense(units = 20, activation = 'relu', input\_shape = c(320)) %>% #320 is the number of column in the data frame  
 layer\_dense(units = 10, activation = 'softmax') %>%   
 layer\_dense(units = 1)%>%  
 compile(  
 optimizer = 'rmsprop',  
 loss = 'binary\_crossentropy',  
 metrics = c('accuracy')  
 )  
  
  
#Train the model with random parameters  
trained <- model %>% fit( x = as.matrix(train.elec), y = as.matrix(target.data) ,epochs=10, batch\_size=32, validation\_split = 0.2)  
#Look at stats and plots  
trained

## Trained on 15,782 samples (batch\_size=32, epochs=10)  
## Final epoch (plot to see history):  
## loss: -50,982  
## accuracy: 0.00006336  
## val\_loss: -51,449  
## val\_accuracy: 0

plot(trained)

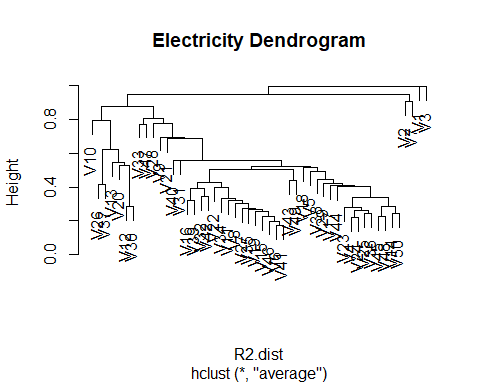


#keras::predict\_classes(trained, test.elec)

For this question, I began by loading in the electricity data, which was already in a form where I could build a deep learning model using it. So I split the data into training and testing data then took a row off my training data and made it my target data. I then moved onto actually building and compiling my model where all of the parameters were picked out of thin air. After constructing the model I had to train the data, again all of the parameters in the fit command were chosen out of thin air. I was unable to get the predict function to work with this model. I kept getting the error: “‘what’ must be a function or character string” and was unable to find a solution in several hours of searching and playing with the function. As a result, I just looked at the model summary and the plot, and due to having and accuracy close to zero and a high/low (negative) of losses. I can say that this model performed rather terribly.

#Question 2

#Load in the data  
elec.df <- fread("C:/Users/jaket/OneDrive/Desktop/GDAT 626/electricity.txt")  
#Only want the first 50 sensors  
elec.df <- elec.df[,1:50]  
#correlation matrix  
cor(elec.df, elec.df,   
 method = "pearson",  
 use = "pairwise.complete.obs") -> cor.mat  
#Distance matrix  
1 - cor.mat \* cor.mat -> R2dist.mat  
as.dist(R2dist.mat) -> R2.dist  
#Cluster and plot  
hclust(R2.dist, method = "average") -> elec.cluster  
plot(elec.cluster, main = "Electricity Dendrogram")



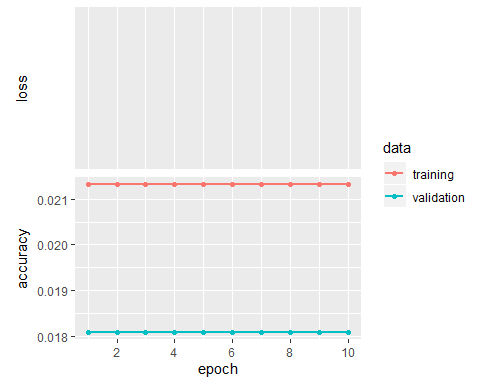
For this problem, I began by loading in the electricity data again and selecting the first 50 columns/sensors (The data was already in a wide format so there was no need to worry about converting it to a wide format). I then made a correlation matrix by finding a correlation between the data frame and itself. I then subtracted the square of the matrix from 1 to convert it to a distance matrix. I then used the hclust function to cluster the data using the average as the method. I then plotted the dendrogram, which appears to show 4 or five distinct groups in our data. These clusters represent sensors that had similar data to other sensors in the cluster.

#Question 3

#Load in all of the data  
Beijing <- fread("C:/Users/jaket/OneDrive/Desktop/GDAT 626/Beijing.csv")  
Beijing<- Beijing[,c(1,10)]  
names(Beijing)[2] <- "Bei"  
  
Chengdu <- fread("C:/Users/jaket/OneDrive/Desktop/GDAT 626/Beijing.csv")  
Chengdu <- Chengdu[,c(1,10)]  
names(Chengdu)[2] <- "Che"  
  
Guangzhou <- fread("C:/Users/jaket/OneDrive/Desktop/GDAT 626/Guangzhou.csv")  
Guangzhou <- Guangzhou[,c(1,10)]  
names(Guangzhou)[2] <- "Gua"  
  
Shang <- fread("C:/Users/jaket/OneDrive/Desktop/GDAT 626/Shang.csv")  
Shang <- Shang[,c(1,10)]  
names(Shang)[2] <- "Sha"  
  
Sheny <- fread("C:/Users/jaket/OneDrive/Desktop/GDAT 626/Sheny.csv")  
Sheny <- Sheny[,c(1,10)]  
names(Sheny)[2] <- "She"  
  
#Join the data by observation number  
{  
df <- left\_join(Beijing,Chengdu, by = "No")  
df <- left\_join(df,Guangzhou, by = "No")  
df <- left\_join(df,Shang, by = "No")  
df <- left\_join(df,Sheny, by = "No")  
}  
  
#NA stop in all of the cities at row 611  
  
df <- df[-c(1:611),]  
#Drop No column  
df <- df[,-1]  
#Split into training, target, and test data  
df\_split <- initial\_split(df)  
train.china<- training(df\_split)  
test.china <- testing(df\_split)  
  
#5th row is target data  
target.china <- train.china[,5]  
train.china <- train.china[,-5]  
  
model <- keras\_model\_sequential()  
  
# define and compile the model  
model %>%   
 layer\_dense(units = 25, activation = 'relu', input\_shape = c(4)) %>% #4 is number of columns in the training data  
 layer\_dense(units = 12, activation = 'softmax') %>%   
 layer\_dense(units = 1)%>%  
 compile(  
 optimizer = 'rmsprop',  
 loss = 'binary\_crossentropy',  
 metrics = c('accuracy')  
 )  
  
  
#Train the model (random parameters)  
trained <- model %>% fit( x = as.matrix(train.china), y = as.matrix(target.china) ,epochs=10, batch\_size=32, validation\_split = 0.2)  
trained

## Trained on 31,184 samples (batch\_size=32, epochs=10)  
## Final epoch (plot to see history):  
## loss: NaN  
## accuracy: 0.02133  
## val\_loss: NaN  
## val\_accuracy: 0.01809

plot(trained)

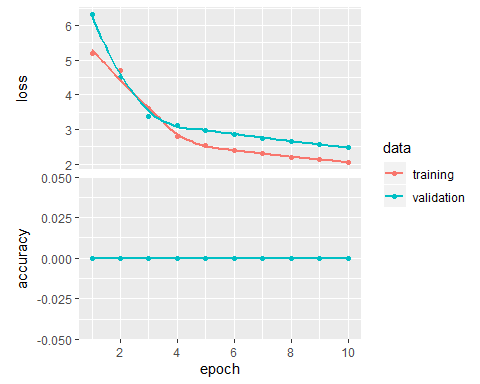


#Would usually use predict but I couldn't get it to work

For this question, I started loading in the data from all five of the cities in the Chinese cities data set. Then I merged them all into one data set by the “No” column. I choose to use the different cities as predictors because we don’t have to worry about the dynamics behind the data with deep learning since they don’t affect these models. My next step was to cut off the top of the data to the point where there were no more NA’s because NA’s were only located at the top of the data frame. At this point, the data had been cleaned and ready to be split into training and testing data. I then removed the column that represented the city of Shenyang and used that as my target data. Again built, compiled and fit my data using random parameters, and like I said in the previous problem I was not able to get the predict/to evaluate function to work. Because the model returned a low accuracy, I can say again that this is not a good model.

#Question 4

data("NelPlo")  
df <- as.data.frame(NelPlo)  
#NA stop at line 50  
df <- na.omit(df)  
  
#Split into testing training and target  
nel\_split <- initial\_split(df)  
train.nel<- training(nel\_split)  
test.nel <- testing(nel\_split)  
  
#What we want to predict is the first column  
target.nel <- train.nel[,1]  
train.nel <- train.nel[,-1]  
  
  
model <- keras\_model\_sequential()  
  
# define and compile the model (random parameters)  
model %>%   
 layer\_dense(units = 20, activation = 'relu', input\_shape = c(13)) %>% #13 is the number of coulmn in training data  
 layer\_dense(units = 10, activation = 'softmax') %>%   
 layer\_dense(units = 1)%>%  
 compile(  
 optimizer = 'rmsprop',  
 loss = 'binary\_crossentropy',  
 metrics = c('accuracy')  
 )  
  
  
#Train the model (random paramaters)  
trained <- model %>% fit( x = as.matrix(train.nel), y = as.matrix(target.nel) ,epochs=10, batch\_size=32, validation\_split = 0.2)  
  
plot(trained)

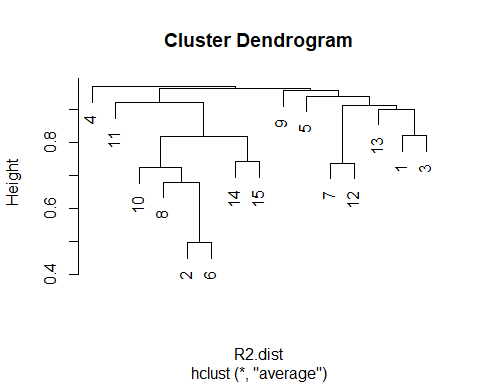


#Would usually predict here but I couldn't get it to work

For this problem, I loaded in the NelPlo data and then cut off the top of the data frame to the point the NA’s were all gone. I then split the data into testing and training data and removed the first column (CPI) from the training data to use it as the target data (its what we are trying to predict). Next, I built, compiled, and trained the model, and due to the relatively high number of loses and accuracy of 0 I built a model that is either historically terrible or doesn’t work. I believe that there is a problem with my model causing it to be this bad. There is an accuracy of zero.

#Question 5

{  
{  
df <- fread("C:/Users/jaket/OneDrive/Desktop/GDAT 626/active/1.csv")  
#Find total acceleration  
df$total <- df$V2 + df$V3 + df$V4  
#Just select the first excersize  
df <- df[df$V5 == 1]  
#Just want 1st column (to join by) and 6th column (variable we want to cluster by)  
df <- df[,c(1,6)]  
#Make sure the first column goes up byone per observation  
df$V1 <- 1:nrow(df)  
#Rename vectors so they don't match other subjects  
names(df)[2]<- "1"  
names(df)[1] <- "row"   
One <- df  
}  
  
{  
df <- fread("C:/Users/jaket/OneDrive/Desktop/GDAT 626/active/2.csv")  
   
df$total <- df$V2 + df$V3 + df$V4  
df <- df[df$V5 == 1]  
df <- df[,c(1,6)]  
df$V1 <- 1:nrow(df)  
names(df)[2]<- "2"  
names(df)[1] <- "row"   
Two <- df  
}   
  
{  
df <- fread("C:/Users/jaket/OneDrive/Desktop/GDAT 626/active/3.csv")  
  
df$total <- df$V2 + df$V3 + df$V4  
df <- df[df$V5 == 1]  
df <- df[,c(1,6)]  
df$V1 <- 1:nrow(df)  
names(df)[2]<- "3"  
names(df)[1] <- "row"   
Three <- df  
}  
  
{  
df <- fread("C:/Users/jaket/OneDrive/Desktop/GDAT 626/active/4.csv")  
   
df$total <- df$V2 + df$V3 + df$V4  
df <- df[df$V5 == 1]  
df <- df[,c(1,6)]  
df$V1 <- 1:nrow(df)  
names(df)[2]<- "4"  
names(df)[1] <- "row"   
Four <- df  
}   
  
{  
df <- fread("C:/Users/jaket/OneDrive/Desktop/GDAT 626/active/5.csv")  
  
df$total <- df$V2 + df$V3 + df$V4  
df <- df[df$V5 == 1]  
df <- df[,c(1,6)]  
df$V1 <- 1:nrow(df)  
names(df)[2]<- "5"  
names(df)[1] <- "row"   
Five <- df  
}  
  
{  
df <- fread("C:/Users/jaket/OneDrive/Desktop/GDAT 626/active/6.csv")  
   
df$total <- df$V2 + df$V3 + df$V4  
df <- df[df$V5 == 1]  
df <- df[,c(1,6)]  
df$V1 <- 1:nrow(df)  
names(df)[2]<- "6"  
names(df)[1] <- "row"   
Six <- df  
}   
  
{  
df <- fread("C:/Users/jaket/OneDrive/Desktop/GDAT 626/active/7.csv")  
  
df$total <- df$V2 + df$V3 + df$V4  
df <- df[df$V5 == 1]  
df <- df[,c(1,6)]  
df$V1 <- 1:nrow(df)  
names(df)[2]<- "7"  
names(df)[1] <- "row"   
Seven <- df  
}  
  
{  
df <- fread("C:/Users/jaket/OneDrive/Desktop/GDAT 626/active/8.csv")  
   
df$total <- df$V2 + df$V3 + df$V4  
df <- df[df$V5 == 1]  
df <- df[,c(1,6)]  
df$V1 <- 1:nrow(df)  
names(df)[2]<- "8"  
names(df)[1] <- "row"   
Eight <- df  
}   
  
{  
df <- fread("C:/Users/jaket/OneDrive/Desktop/GDAT 626/active/9.csv")  
  
df$total <- df$V2 + df$V3 + df$V4  
df <- df[df$V5 == 1]  
df <- df[,c(1,6)]  
df$V1 <- 1:nrow(df)  
names(df)[2]<- "9"  
names(df)[1] <- "row"   
Nine <- df  
}  
  
{  
df <- fread("C:/Users/jaket/OneDrive/Desktop/GDAT 626/active/10.csv")  
   
df$total <- df$V2 + df$V3 + df$V4  
df <- df[df$V5 == 1]  
df <- df[,c(1,6)]  
df$V1 <- 1:nrow(df)  
names(df)[2]<- "10"  
names(df)[1] <- "row"   
Ten <- df  
}   
  
{  
df <- fread("C:/Users/jaket/OneDrive/Desktop/GDAT 626/active/11.csv")  
  
df$total <- df$V2 + df$V3 + df$V4  
df <- df[df$V5 == 1]  
df <- df[,c(1,6)]  
df$V1 <- 1:nrow(df)  
names(df)[2]<-"11"  
names(df)[1] <- "row"   
Eleven <- df  
}  
  
{  
df <- fread("C:/Users/jaket/OneDrive/Desktop/GDAT 626/active/12.csv")  
   
df$total <- df$V2 + df$V3 + df$V4  
df <- df[df$V5 == 1]  
df <- df[,c(1,6)]  
df$V1 <- 1:nrow(df)  
names(df)[2]<- "12"  
names(df)[1] <- "row"   
Twelve <- df  
}  
  
{  
df <- fread("C:/Users/jaket/OneDrive/Desktop/GDAT 626/active/13.csv")  
  
df$total <- df$V2 + df$V3 + df$V4  
df <- df[df$V5 == 1]  
df <- df[,c(1,6)]  
df$V1 <- 1:nrow(df)  
names(df)[2]<- "13"  
names(df)[1] <- "row"   
Thirteen <- df  
}  
  
{  
df <- fread("C:/Users/jaket/OneDrive/Desktop/GDAT 626/active/14.csv")  
   
df$total <- df$V2 + df$V3 + df$V4  
df <- df[df$V5 == 1]  
df <- df[,c(1,6)]  
df$V1 <- 1:nrow(df)  
names(df)[2]<- "14"  
names(df)[1] <- "row"   
Fourteen <- df  
}   
  
{  
df <- fread("C:/Users/jaket/OneDrive/Desktop/GDAT 626/active/15.csv")  
  
df$total <- df$V2 + df$V3 + df$V4  
df <- df[df$V5 == 1]  
df <- df[,c(1,6)]  
df$V1 <- 1:nrow(df)  
names(df)[2]<- "15"  
names(df)[1] <- "row"   
Fifteen <- df  
}  
}  
  
{  
One <- One[1:18280,]  
Two <- Two[1:18280,]  
Three <- Three[1:18280,]  
Four <- Four[1:18280,]  
Five <- Five[1:18280,]  
Six <- Six[1:18280,]  
Seven <- Seven[1:18280,]  
Eight <- Eight[1:18280,]  
Nine <- Nine[1:18280,]  
Ten <- Ten[1:18280,]  
Eleven <- Eleven[1:18280,]  
Twelve <- Twelve[1:18280,]  
Thirteen <- Thirteen[1:18280,]  
Fourteen <- Fourteen[1:18280,]  
Fifteen <- Fifteen[1:18280,]  
}  
  
{  
df <- left\_join(One,Two, by = "row")  
df <- left\_join(df,Three, by = "row")  
df <- left\_join(df,Four, by = "row")  
df <- left\_join(df,Five, by = "row")  
df <- left\_join(df,Six, by = "row")  
df <- left\_join(df,Seven, by = "row")  
df <- left\_join(df,Eight, by = "row")  
df <- left\_join(df,Nine, by = "row")  
df <- left\_join(df,Ten, by = "row")  
df <- left\_join(df,Eleven, by = "row")  
df <- left\_join(df,Twelve, by = "row")  
df <- left\_join(df,Thirteen, by = "row")  
df <- left\_join(df,Fourteen, by = "row")  
df <- left\_join(df,Fifteen, by = "row")  
}  
  
df <- df[,-1]  
cor(df, df,   
 method = "pearson",  
 use = "pairwise.complete.obs") -> cor.mat  
  
# Convert to a distance  
1 - cor.mat \* cor.mat -> R2dist.mat  
as.dist(R2dist.mat) -> R2.dist  
  
# Cluster  
hclust(R2.dist, method = "average") -> df.cluster  
plot(df.cluster)



For this problem, The first thing I did was load in each data frame in the activity data folder, trimmed each data frame so that they all the same length as the shortest one, and joined them all into one data frame by row number. In this step, I also created a total acceleration variable which was just composed of the sum of the three other accelerations. After removing all of the rows except for each subject’s total acceleration, this data is fit for clustering. To cluster, I created a correlation matrix by finding the correlation between the data frame and its self. Then I converted it to a distance matrix by subtracting the square of the data frame from one. Next, I put the distance matrix into the hclust command to cluster the data. When I plot the clustered data, the dendrogram shows us there appear to be approximately 3 or 4 different clusters. These clusters represent subjects whose total acceleration for exercise one was similar.