Preprocessing

Davide la Manna

2023-03-13

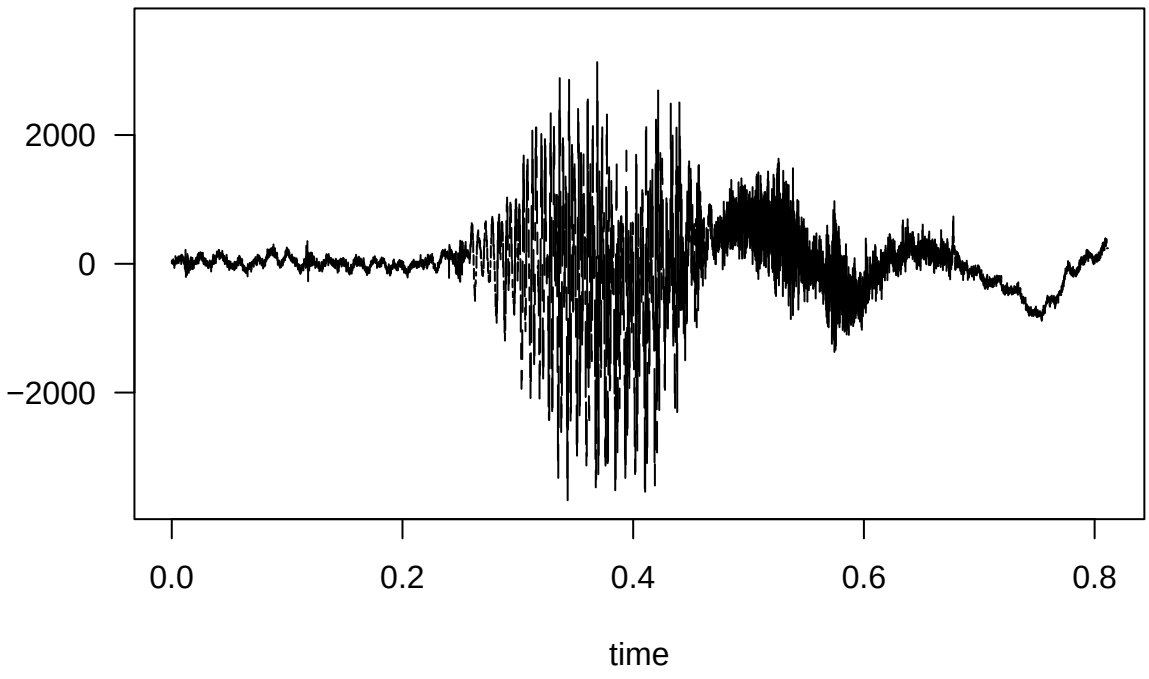
uploading .wav data

To begin with, we will upload a random .wav file representing a “yes”.

```
library(tuneR)
file_list <- list.files("yes")
random_file <- sample(file_list, 1)
audio <- readWave(file.path("yes",random_file))
```

let’s try to display and reproduce the following file

```
plot(audio)
```



```
#play(audio)
```

investigate the class "yes"

Let us now go on to calculate mean and correlation of the MFCC of the word "yes"

```
library(abind)
ncep=13 #number of cepstral coefficients required
time=99 #number of time intervals
nfile=length(file_list)
mfccs <- NULL
path="C:/Users/david/OneDrive/Desktop/semester project/git/SemesterProject/speech_commands_v0.02.tar/sp
# Loop through all the .wav files in the "yes" folder.
for (file in file_list) {
    # load file .wav
  wav <- readWave(file.path(path, file))

  # compute the MFCC
  mfcc <- melfcc(wav, numcep = 13)
  # I solve the problem given by the fact that the audio files do not all have the same time length by
  mfcc99 <- matrix(rep(0, time*ncep), ncol = 13)
  mfcc99[1:dim(mfcc)[1],] <- mfcc


 # Add MFCC to MFCCs

mfccs <- abind(mfccs, mfcc99, along = 3)
}
```

It appears from the analysis that there are 21723 NA values

```
summary(mfccs)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
## -85.836  -5.052  -0.313   2.827   4.240 129.604   21723
```

since it represents a considerably small value compared to `ncep` * `time`*`nfile`, we decide for the moment to replace these NA values with zeros (hoping that our analysis is still good)

```
mfccs[is.nan(mfccs)]<-0
```

We now save the tensor with preprocessed data

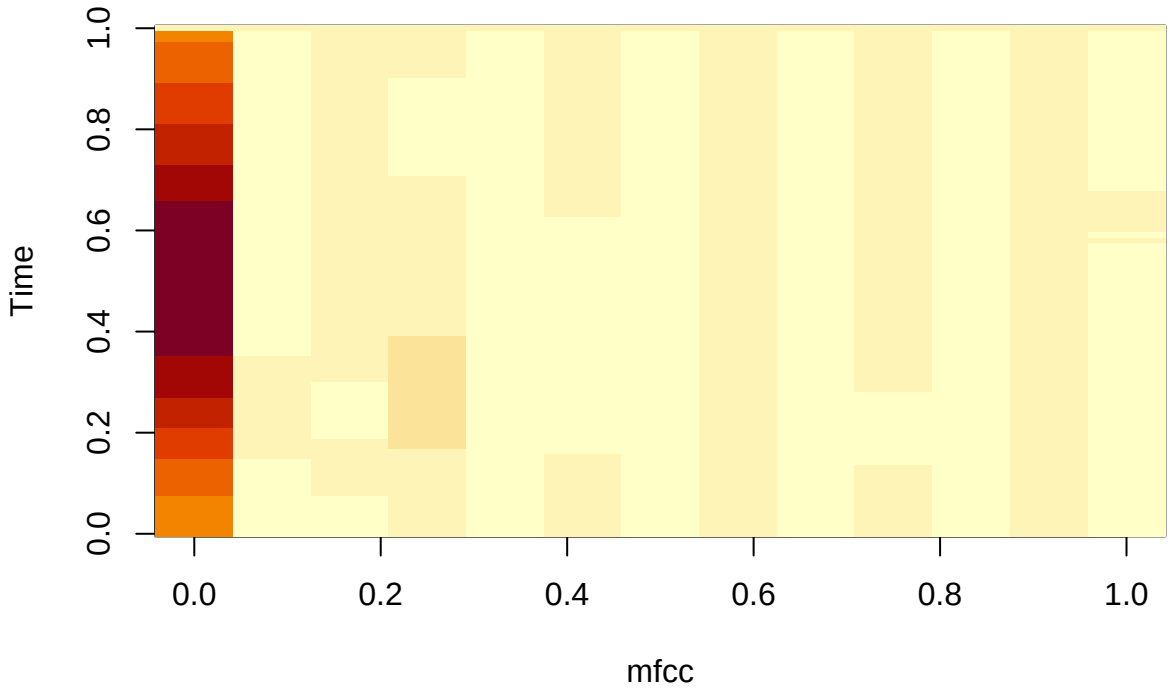
```
save(mfccs, file = "yesP.RData")
```

In order to investigate the data, we decide to calculate the mean and correlation of our data and plot the values

```
mean <- apply(mfccs, c(1,2), mean)
```

this command calculates the average along the third dimension returning us a matrix of size `ncep`*`time` that we plot

```
image(t(mean),xlab = "mfcc", ylab = "Time")
```



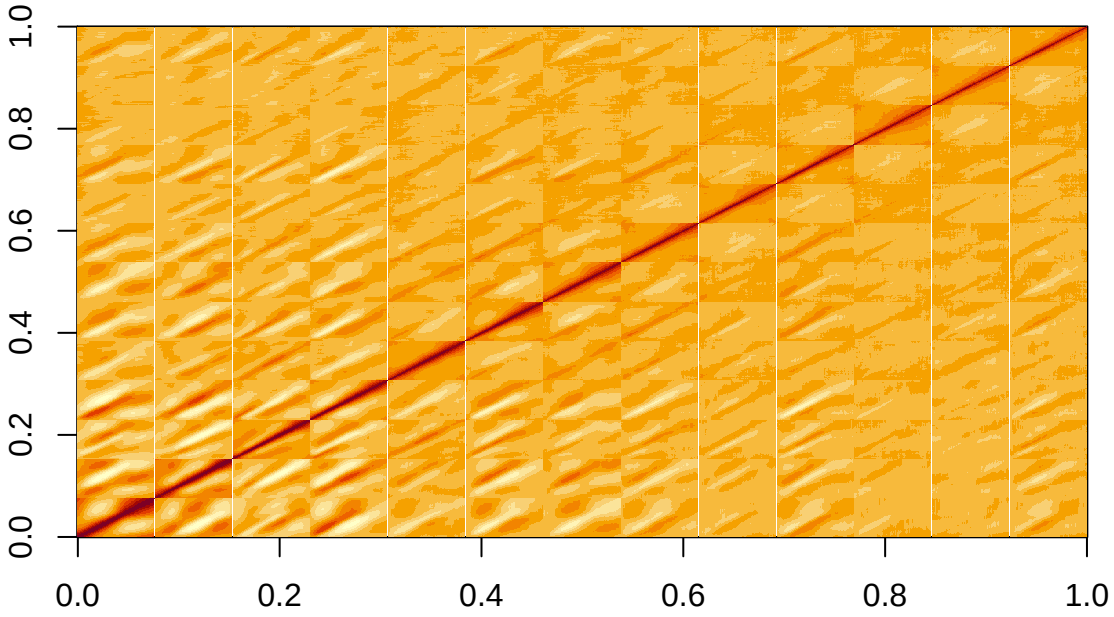
for the correlation we decide to transform the array into a matrix of dimension `ncep` × `time` × `nfile` and calculate the correlation, obtaining a but matrix of dimension `ncep`*`time` ×`ncep`*`time` which we plot

```
vectorized = matrix(mfccs, ncol = ncep*time, byrow = TRUE)
corr=cor(vectorized)
```

```
## Warning in cor(vectorized): the standard deviation is zero
```

As can also be seen from the graph, some correlation values are outliers, creating white lines in the graph. This phenomenon is probably related to the presence of zero values in the data array.

```
image(corr)
```



As we can see, the covariance matrix in this case will be the vectorization of a tensor of size (13,99,13,99), thus a matrix of size 1287 × 1287, we can see how this matrix consists of 13 x 13 blocks of square matrices of size 99. For example, the matrix on the lower left is the sample covariance matrix for the first cepstral coefficient across the 99 time points.

Preprocess all the data

We act the same way with all the other classes. For memory reasons we decide not to save all labeled data in one tensor but save the preprocessed data separately and add labels when the classification process starts.

to this end we need only update the previous code with a for loop to perform the operation on all folders

```
#library(abind)
#ncep=13 #number of cepstral coefficients required
#time=99 #number of time intervals
#
#
#list_dirs=list.dirs()
#list_dirs<-list_dirs[-1] #omit the empty dir
#for (dir in list_dirs) {
#    file_list <- list.files(dir)
#nfile=length(file_list)
#mfccs <- NULL
#
## Loop through all the .wav files in the "yes" folder.
```

```
#for (file in file_list) {
#    # load file .wav
#  wav <- readWave(file.path(dir, file))
#
#  # compute the MFCC
#  mfcc <- melfcc(wav, numcep = 13)
#  # I solve the problem given by the fact that the audio files do not all     have #the same time leng
#  mfcc99 <- matrix(rep(0, time*ncep), ncol = 13)
#  mfcc99[1:dim(mfcc)[1],] <- mfcc
#
#
# # Add MFCC to MFCCs
#
#mfccs <- abind(mfccs, mfcc99, along = 3)
#}
#mfccs[is.nan(mfccs)]<-0
#string <- paste(dir,"P.RData")
#new_str <- sub("./", "", string)
#save(mfccs, file = new_str)
#}
#
```

The output of this file will be provided on github

Conclusion

We were able to observe graphically that the mean and covariance matrices found are very similar to those found in the work of HOFF et al. so we are very satisfied with our analysis, although the presence of audios of different lengths forced us to fill in arbitrary values for the time values of the cepstral coefficients in some cases (which is also evident from the covariance matrix). One way to achieve better data preprocessing might be to force the audios to be the same length or to force the choice of 99 framerates within the command `melfc` Hoping that this will not adversely affect the selection of cepstral coefficients for the purpose of classification.