Fitting AR(1) Processes to EEG Signals for Comparison

Ronak Mehta, Vidur Kailash

October 29, 2017

Package Installation

The following packages are required, and the directory is set.

```
require(knitr, quietly = TRUE)
require(MTS, quietly = TRUE)
```

```
## Warning: package 'MTS' was built under R version 3.4.2
```

```
require(data.table, quietly = TRUE)
```

Purpose and Background

In this analysis, we fit the AR(1) process to various multivariate time series produced from EEG signal data. The components represent the first 100 channels from which EEG voltage is recorded. A multivariate AR(1) process for an n-dimenstional time series is as follows:

$$X_t = \phi_1 X_{t-1} + \epsilon_t$$

Where X_t represents an n-dimensional vector at timestep t, ϵ_t represents random noise, and the $n \times n$ matrix ϕ represents the linear transition between an observation at one timestep to the next. By fitting this model to multiple subjects' EEG time series we hope to show that subjects who perform the same task will have similar ϕ matrices, while subject data from different tasks will have dissimilar ϕ matrices.

Comparison Method

To compare matrices, we will take their pairwise difference matrices and calculate the Frobenius norm of each one. we will then average these values to get a difference measure within groups and between groups.

Loading Data and Fitting AR(1) Model

We use the `MTS` package to fit the model. Detailed information about this package can be found here: `https://cran.r-project.org/web/packages/MTS/MTS.pdf` We will use the following five subjects from the Healthy Brain Network Biobank (HBNB) dataset. Their subject IDs are listed below. The three tasks that we use data from will be Resting State, Video 1 (a passive video), and VizLearn (watching an educational video).

```
# Create matrix of IDs
ids <- c("NDARAA075AMK", "NDARAD481FXF", "NDARAE199TDD", "NDARAJ366ZFA", "NDARAM277WZT")
options(knitr.kable.NA = '')
kable(ids, col.names = "Subject ID", align = 'c')
```

Subject ID
NDARAA075AMK
NDARAD481FXF
NDARAE199TDD

Subject ID
NDARAJ366ZFA
NDARAM277WZT

Next, we load the data into memory.

```
# Set file name vectors and data lists.
lemur_dir <- "C:/Users/Ronak Mehta/Desktop/NeuroData/lemur-f17s18/data/"
resting_files <- c()
video_files <- c()
learning_files <- c()
resting_data <- list()
video_data <- list()
learning_data <- list()
num_channels <- 111
good_channels <- c(2:7, 9:13, 15:19, 21:23, 25:30, 32:111)

# Downsample by 100 within the first 100,000 observations.
columns <- seq(1, 50001, 100)

# Name files
for (i in 1:length(ids)) {
  resting_files[i] <- paste(lemur_dir,
                            ids[i],
                            "/EEG/preprocessed/csv_format/",
                            ids[i],
                            "_RestingState_data.csv",
                            sep = "")
  video_files[i] <- paste(lemur_dir,
                          ids[i],
                          "/EEG/preprocessed/csv_format/",
                          ids[i],
                          "_Video1_data.csv",
                          sep = "")
  learning_files[i] <- paste(lemur_dir,
                             ids[i],
                             "/EEG/preprocessed/csv_format/",
                             ids[i],
                             "_vis_learn_data.csv",
                             sep = "")
}

# Injest CSV of subject's data.
for (i in 1:length(ids)) {
  resting_data[[i]] <-  as.matrix(transpose(fread(input = resting_files[i],
                                                  sep = ",",
                                                  nrows = num_channels,
                                                  header = FALSE,
                                                  showProgress = TRUE,
                                                  select = columns,
                                                  data.table = FALSE)))[, good_channels]
  video_data[[i]] <-    as.matrix(transpose(fread(input = video_files[i],
                                                  sep = ",",
```

```
                                                    nrows = num_channels,
                                                    header = FALSE,
                                                    showProgress = TRUE,
                                                    select = columns,
                                                    data.table = FALSE)))[, good_channels]
  learning_data[[i]] <- as.matrix(transpose(fread(input = learning_files[i],
                                                    sep = ",",
                                                    nrows = num_channels,
                                                    header = FALSE,
                                                    showProgress = TRUE,
                                                    select = columns,
                                                    data.table = FALSE)))[, good_channels]
}
```

After the data has been input and downsampled by the selected parameters, we can fit the model.

```
# Initialize list to hold the phi-matrix for each subject's data.
resting_matrix <- list()
video_matrix <- list()
learning_matrix <- list()
for (i in 1:length(ids)) {
  # Calculate matrix for subject i for each task.
  resting_matrix[[i]] <-  VAR(resting_data[[i]],p=1)$coef
  video_matrix[[i]] <-    VAR(video_data[[i]],p=1)$coef
  learning_matrix[[i]] <- VAR(learning_data[[i]],p=1)$coef
}
```

Results

Similarity Within Tasks

Here, we take the distance measure for each ϕ matrix difference within the groups. With 5 subjects, we have 10 differences to take.

```
# Calculate difference matrices.
combs <- list(c(1, 2),
              c(1, 3),
              c(1, 4),
              c(1, 5),
              c(2, 3),
              c(2, 4),
              c(2, 5),
              c(3, 4),
              c(3, 5),
              c(4, 5))
resting_dist <- 0
video_dist <- 0
learning_dist <- 0
num_diffs <- length(combs)
for (i in 1:length(combs)) {
  # Resting Distances
  resting_dist <- resting_dist +
    norm(resting_matrix[[combs[[i]][1]]] - resting_matrix[[combs[[i]][2]]], type = "F")/num_diffs
```

```
  # Video Distances
  video_dist <- video_dist +
    norm(video_matrix[[combs[[i]][1]]] - video_matrix[[combs[[i]][2]]], type = "F")/num_diffs

  # Learning Distances
  learning_dist <- learning_dist +
    norm(learning_matrix[[combs[[i]][1]]] - learning_matrix[[combs[[i]][2]]], type = "F")/num_diffs
}
```

Similarity Between Tasks

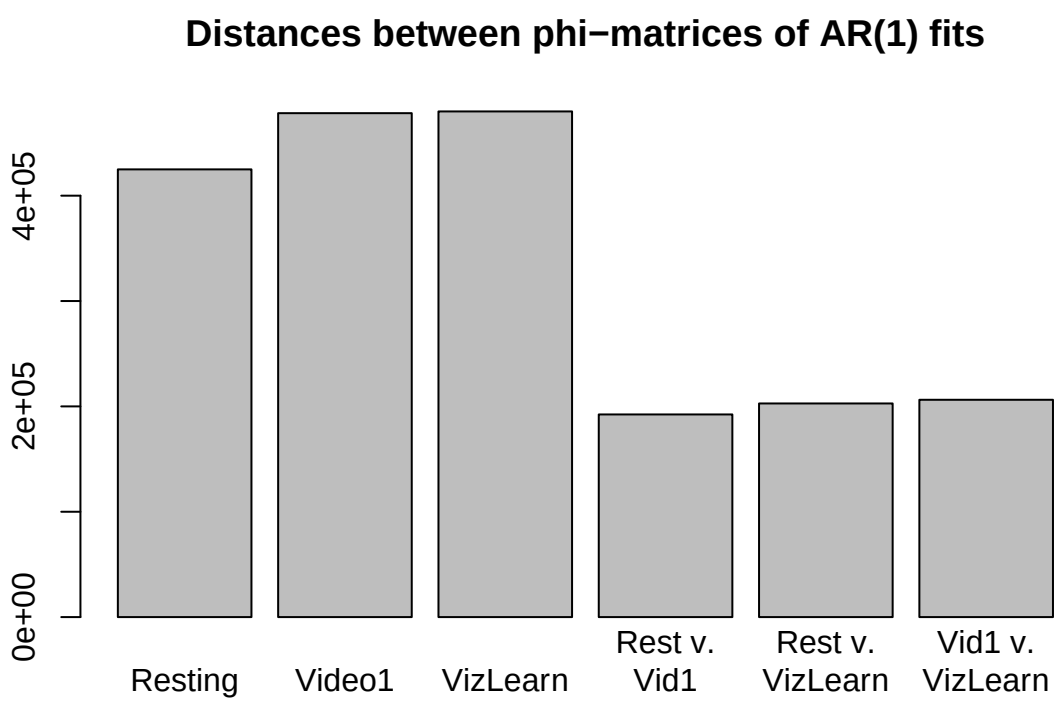
Here, we take the average ϕ matrix within the groups, and compare pairwiase distances.

```
# Calculate difference matrices.
avg_resting_matrix <- matrix(rep(0, 106*105), nrow = 106)
avg_video_matrix <- matrix(rep(0, 106*105), nrow = 106)
avg_learning_matrix <- matrix(rep(0, 106*105), nrow = 106)
for (i in 1:length(ids)) {
  avg_resting_matrix <- avg_resting_matrix + resting_matrix[[i]]/5
  avg_video_matrix <- avg_video_matrix + video_matrix[[i]]/5
  avg_learning_matrix <- avg_learning_matrix + learning_matrix[[i]]/5
}

# Calculate distances
rest_vid_dist <- norm(avg_resting_matrix - avg_video_matrix, type = "F")
rest_learn_dist <- norm(avg_resting_matrix - avg_learning_matrix, type = "F")
vid_learn_dist <- norm(avg_video_matrix - avg_learning_matrix, type = "F")
```

Finally, we plot a bar graph showing the differences in comparison to each other.

```
# Concatnate distance values.
dat <- c(resting_dist,
         video_dist,
         learning_dist,
         rest_vid_dist,
         rest_learn_dist,
         vid_learn_dist)
labs <- c("Resting",
          "Video1",
          "VizLearn",
          "Rest v.\nVid1",
          "Rest v.\nVizLearn",
          "Vid1 v.\nVizLearn")
barplot(dat, main = "Distances between phi-matrices of AR(1) fits", names.arg=labs)
```



Conclusion and Future Analyses

Suprisingly, the between group variation is significantly less than the within group variation. It would seem that the "effect" of the different tasks would be negligible on the EEG signal. However, there are many considerations that we must make. The natural variability must be very high, and averaging the ϕ matrices within each group must have decreased that variability. The distanes between each pairwise group (Resting, Video1, and VizLearn) seem to be on the same order of magnitude.

While this was exploratory, more thorough analysis can follow a few paths. First, we must validate the time series model we use in the first place - we assumed that the data followed the AR(1) process, and compared the only coefficient matrix we had. More complex models will add more $n \times n$ matrices to consider.

Then we looked at the time series itself. How does stationarity generalize to multiple dimensions, and what edits do we need to make in order to use these models more properly? We also plan to study cointegration, and other phenomena that are particular to multidimensional time series data that do not exist in the univariate case.

Similarly, our distance function was simple and easy, but could be looked at more critically. Of the many matrix distance functions, which are applicable to this project? Of these measures, how can we normalize them to understand our results better?

Now that we are more familiar with the data, we plan to explore these questions, ask more opinions, and use more data next week when we extend this analysis.