CS 5310 Data Mining

Lab – Chapter 3 Lazy Learning – Classification Using Nearest Neighbors

Virtual reality mindfulness intervention for enhancing learning effectiveness research – service-learning

In this lab, we will process the data collected in a mindfulness intervention with virtual reality (VR) experiment, prepare the data for analyzing the effect of mindfulness intervention, and use kNN method to create a model to classify data collected in different brain activities. This is a part of the continuous service-learning project supported by UHD Center for Community Engagement and Service Learning (CCESL). The procedure of the experiment is as follows.

**The procedure**:

*Learning procedure used in the experiment*:

The learning procedure is carried-out on computer. The subject logon to BrainGymmer.com account and choose a brain training game to play. The score of the game is recorded as the subject’s performance index.

*Pre-intervention activity*:

Each volunteer plays the BrainGymmer game as depicted above. The time taken in this step depends on how much time the volunteer uses to finish the game.

*Meditation*:

Each volunteer plays the VR TRIPP meditation game. The TRIPP demo is used to guide the meditation. The entire process take about 10 minutes.

*Post-intervention activities*:

Each volunteer plays the same BrainGymmer game as in the pre-intervention phase. The time taken in this step depends on how much time the volunteer uses to finish the game.

EEG data were recorded in all the above three phases using Compumedics Grael LT 34 channel recorder. Repeat the above procedure for each subject.

The EEG dataset we will process in this lab is saved in 3 Matlab file (with file extension .mat), corresponding to the data collected in the aforementioned 3 phases of the experiment. The following is a code segment that suggests how to open the data file:

import os

import numpy as np

import pandas as pd

from scipy.io import loadmat

os.chdir(‘[The path to the following file]')

filename = 'Acquisition-15-data.mat'

data = loadmat(filename)

eeg = data['data']

“eeg” in the above code segment is a Numpy array with shape 34 X NS, where NS is the number of samples recorded in the corresponding phase. You need to transpose and save it in a data frame that has shape NS X 34 (NS rows, 34 columns), where each column saves the EEG recordings of one channel.

We will use package “pyEEG” to compute power spectrum intensity (PSI) to be used in modeling. Download pyEEG package from <https://github.com/forrestbao/pyeeg> and follow the instructions on the website to install the package. The function we use in this lab is bin\_power(X, Band, Fs), where X is a list of 1-D real time series, Band is a list of boundary frequencies (in Hz) of bins. They can be unequal bins, e.g. [0.5,4,7,12,30] which are delta, theta, alpha and beta respectively. Set Band = [0.5,4,7,12,30]. Fs is an integer indicating the sampling rate in physical frequency, which is 1024 in our case. The function returns a list of PSIs of delta, theta, alpha and beta bands, and a list of corresponding RIRs (*Relative Intensity Ratio*). We only use the alpha PSI (the 3rd value on the returned list).

For more information about pyEEG and its usage, visit:

<https://www.hindawi.com/journals/cin/2011/406391/>

<http://pyeeg.sourceforge.net/>

We will compute an alpha PSI for every second segment, which means *X* should be a list of 1024 EEG readings in each call to bin\_power(X, Band, Fs). In other words, one alpha PSI value is computed out of every second recording for one particular channel. If the entire EEG recording has 10 minutes, then there will be 10 X 60 = 600 alpha PSI values computed for one channel.

Repeat the above process to compute alpha PSI values for each of the 34 channels. Save the results in a data frame of 34 columns, corresponding to 34 channels.

Repeat the above process to compute alpha PSIs for each of the 3 Matlab files, corresponding to the pre-meditation (with file name “Pre”), meditation (with file name “Med”), and post-meditation (with file name “Post”) phases.

For each data set, create a list of brain state labels, which are string “Pre” for the data set corresponding to the pre-meditation phase, “Med” for the data set corresponding to the meditation phase, and “Post” for the data set corresponding to the post-meditation phase. The length of the list equals the number of recorded samples in the corresponding phase (viz., the NS value).

Combine 3 data sets into one by vertically stacking them up (or row binding). Correspondingly, combine the 3 lists of labels into one in the same order.

Remove co-linearity from the data frame: For channels that have a correlation coefficient that is higher than 0.9 or less than -0.9, only one of them is kept in the data frame, and all others are removed. Raise the correlation threshold to a higher value if the number of left-over variables is less than 4.

Min-max Normalize every column of the combined dataset.

Split the combined dataset into training dataset and testing dataset, and train and test the kNN model.

Perform the following activities in Python and submit your codes and results in Blackboard.

1. Load the data in each of the Matlab files into a data frame, resulting in 3 data frames. Each data frame has 34 channels of EEG samples.
2. Compute the alpha PSIs of 34 channels, using 1024 as the segment size, as depicted above, and save the alpha PSIs in a data frame of 34 columns. Repeat this for each of the 3 data frames.
3. Create a list of brain activity labels, corresponding to each data frame, as depicted above.
4. Combine 3 data frames into one by vertical stacking.
5. Create a correlation coefficient matrix of the alpha PSIs of the 34 channels.
6. Remove co-linearity from the combined data frame, as depicted above.
7. Print the correlation coefficient matrix of the alpha PSIs of the remaining channels.
8. Normalize data in every remaining channel using the min-max normalization.
9. Randomly sample 80% of each dataset and save them in a training dataset, and save the remaining 20% of each dataset in a testing dataset. Split the list of labels accordingly.
10. Train the kNN model using the training dataset.
11. Use the kNN model to predict the target feature of the testing dataset.
12. Create a confusion matrix to compare the predicted state activities to the actual activities and compute the accuracy.

Submit the Python source file (.py file), and a Word document containing the correlation matrix of the datasets before and after removing co-linearity (Steps 5 and 7, respectively), and the confusion matrix and accuracy, properly labeled, by the due date.