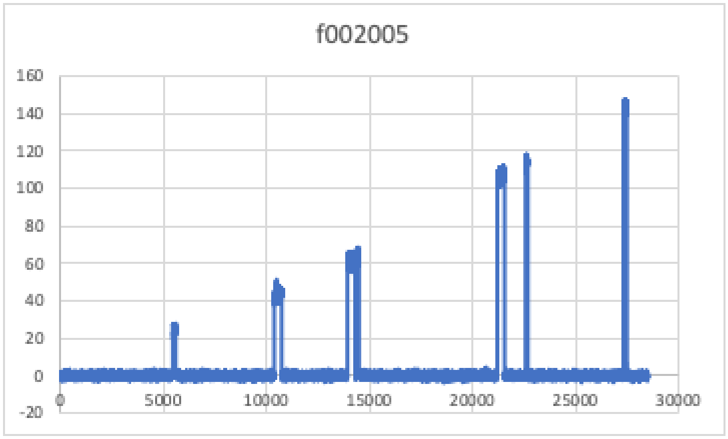
Anomaly Detection & Event Segmentation

Gavin Koma
Department of Bioengineering, Temple University
gavintkoma@temple.edu

Introduction: The overall goal of this assignment is to implement a variety of techniques that we have learned throughout the semester to parse raw data from a time-series event. We then perform segmentation to find unique events in the data and perform sequence decoding to classify these anomalies into a class (0, 1, 2 ,3 or 4). The training data provided has start-times, end-times, and a class value that corresponds to each segment all in addition to the provided signal data (t).

To begin, we first take a moment to look at the provided data. We can see that prior to each pulse, is a uniform distribution of data that centers at 0. Each pulse following is then classified into some class (1-4) while all data in the uniform distribution is Class 0. One important thing to note within these training files is that the durations vary and there is no way to determine if some spacing prior will indicate the class of the next pulse. Another important thing to note is that in each file, the amplitudes of the subsequent classes continue to increase after the previous pulse. As a result of this, we cannot utilize a model like the Hidden Markov Model because they utilize information from the previous state to deduce the next state.

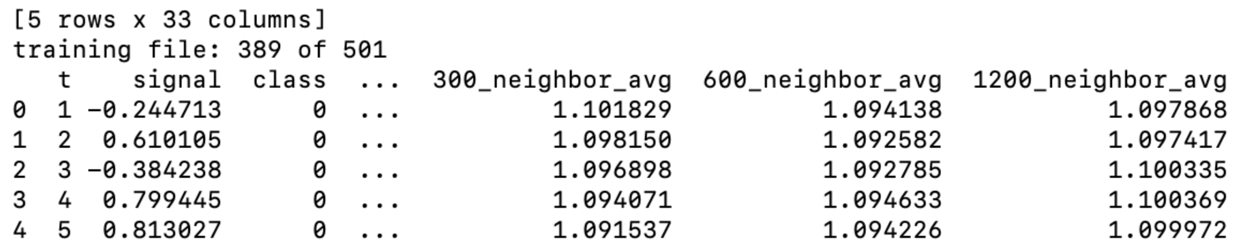


I wanted to implement two models: a multilayer perceptron model (MLP) and a random forest classifier (RFC). Unfortunately, I was unable to implement an RFC model because my computer would fail due to limited memory/cache. Given more time, I would implement this model using PySpark to distribute the workload, but with more finals quickly approaching, I instead focused entirely on MLP.

Data Preprocessing: Once the model to be used was determined, the next step was to perform some basic EDA and load all of the data. I prioritized a dynamic method of coding that iterates through a .txt file and passes the names to a function that will process and load each provided data set in a given folder. I parsed through the commented out sections of the provided data and utilized them to create columns and populate them with values. The length of these datasets are the length of the entire provided signal, thus, training was performed on the entire dataset instead of the segmented chunks. This was necessary for my initial processing. Each file was processed and then passed to a function that performs feature engineering and then partially fits my model in an iterative process.

At initialization and evaluation of this model as is, the error rate was upwards of 70% and this was prior to any segmentation. I figured that it would have been superfluous to move on towards segmentation if my initial model was performing so poorly. Alterations had to be made. First, I changed the layers of my MLP model from 30, 15, 5 to 30, 15, 10, 5. I also switched my activation function from ReLU to TanH. TanH was preferred over sigmoidal as many papers confirm that it tends to perform better in a multilayer neural network. At this point, my model was only performing at about 45% error rate. Not terrible, but could undoubtedly be better. I hypothesized that my error rates should be closer to ~5% considering how much of a notable bias there was to Class 0. Thus, it was time to do a bit of feature engineering.

Feature Engineering: I began by incorporating new columns into my dad to provide my model with more information. This was performed by creating intervals of [10, 20, 50, 100, 300, 600, 1200]. At any time these intervals were implemented, the function would take the bottom half of the value and the top half of the value and perform some mathematical function. I first implemented averages and standard deviations of all these values to add an additional ~35 columns of data to my model. Doing so brought my error rate to ~5% which is what I expected. At this point, my model was successfully predicting classes based on the signal value provided and I figured that it was time to move on to creating the segmented datasets and passing them to the bash script provided by Dr. Picone. The newly computed datasets (with feature engineering) look as such:



```
[5 rows x 33 columns]
training file: 389 of 501
   t    signal  class  ...  300_neighbor_avg  600_neighbor_avg  1200_neighbor_avg
0  1 -0.244713      0  ...          1.101829          1.094138           1.097868
1  2  0.610105      0  ...          1.098150          1.092582           1.097417
2  3 -0.384238      0  ...          1.096898          1.092785           1.100335
3  4  0.799445      0  ...          1.094071          1.094633           1.100369
4  5  0.813027      0  ...          1.091537          1.094226           1.099972
```

Segmentation & GroupBy: The next step was to create a method that segments the data into the proper chunks that are provided in the commented section of the provided data. To do so, I implemented a lambda method that resets the index, groups by the chunks, and provides the mode (majority vote) for each section of the file. This was an interactive process that passed each individual scored data file through the function to output a new given set of hypothesis files that can be passed to the scoring script.

Conclusion/Results: Unfortunately, after a long chat with Dr. Picone tried to figure out why my scoring script was not working, it continued to fail even with the proper hypothesis files provided and path set. As a byproduct, I am unable to provide my final scores and I will have to submit this without my own reported scores and wait to see what the official scores are. Regardless of my own accumulated frustration with python logic, I am proud of the engineering and data process that I did to write this code. I also think that this is some of the most efficient and dynamic code that I have written thus far in my academic career and I am proud to have submitted it.

Unfortunately, during the course of this project, I did not take time to look at my eval data. I try to treat my eval data as a big red box that I need to avoid at all costs. Thus, I was under the impression that my eval data would be able to be passed to my model with little to no issues. It wasn't until I was finally provided the eval data that I realized it was incredibly different and lacked specific values that I needed to provide to my model (chunk duration, start, stop, etc…). As a byproduct, I was unable to calculate a PDF to calculate my confidence and I also was not able to determine a method to guess segmentation lengths. Thus, my eval data is most likely going to have the lowest of scores of the three datasets and with other pressing finals, I am unable to create a method to supplement these values.

To summarize, this assignment's goal was to segment a time-series dataset and perform sequence decoding to predict classes of signals. I utilized an 4 layer MLP model that relied heavily on feature engineering to augment my data and to lower my error rate. Many of my metrics provided little information as to how my model will actually perform on the eval dataset as my metrics are calculated on a per frame basis instead of a per segment basis. While I do hope that my error rate stays low, I am sure that this may be a byproduct of overzealousness on my behalf.