Kmeans clustering of data

Data was from 19 activates using 8 subjects with 5 sensor devices with 3 different reading across 3 dimensions (x, y, z)

Jeremy Swartwood, Assignment 4, Machine Learning, UAA. Spring 2018

# Table of Contents

[Table of Contents 1](#_Toc510881749)

[Overview (Executive Summary): 1](#_Toc510881750)

[Pre-processing: 1](#_Toc510881751)

[Clustering: 1](#_Toc510881752)

[1. Pre-Processing 1](#_Toc510881753)

[1.1. Load data into single numpy array 2](#_Toc510881754)

[1.2. Run FFT 3](#_Toc510881755)

[1.2.1. Run FFT: Exclude first row 4](#_Toc510881756)

[1.3. Run PSD 5](#_Toc510881757)

[1.4. Concatenate rows into 1 row. 6](#_Toc510881758)

[1.5. Run PCA 6](#_Toc510881759)

[2. Clustering 7](#_Toc510881760)

# Overview (Executive Summary):

## Pre-processing:

The first row of the FFT caused skewed and squished data, so it was removed before running PSD. The output of PSD produced 63 columns per column per sample. Those were stacked vertically side-by-side to produce 2835 features per single sample. Finally PCA was run on the 2835 features to reduce down to 689 features.

## Clustering:

K-Means was run on the output to produce the clustering based on Euclidian and Cosine Similarity. Both were run with clusters of 3 and 19. The Mean Entropy of 3 clusters was about 1.4 for Euclidian, and 1.16 for cossim. For 19 clusters about 3.6 for Euclidian, and 3.8 for cossim.

# Pre-Processing

***IMPORTANT NOTE:*** is that I have the 32-bit version of Python installed on my computer. I did not realize this until I was done working on my pre-processing code and already working on my K-means code.

Steps:

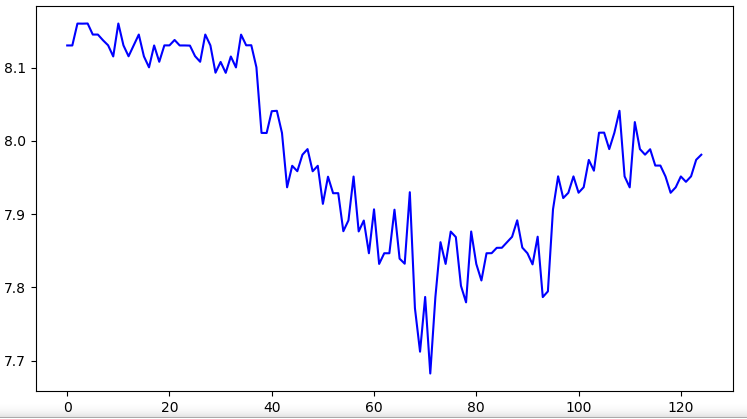


1. Load all data into single numpy array.
2. Run FFT on each of the 45 columns (features), 125 rows (one 5-second sample at a time).
   1. Exclude the first row of the FFT, as it causes a huge spike and throws the range/graph off.
3. Run PSD on each of the 45 columns (features) on the 124 rows from FFT, producing 63 rows per sample.
4. Concatenate all 63 rows for each sample of 45 features into a single row per 5-second sample for 2835 columns.
5. Run PCA on each of the 2835 columns output from the PSD concatenation step, producing 689 features.

## Load data into single numpy array

Load all files into a file array. Loop over the files loading them into memory. Horizontally stack the array to create a single array.



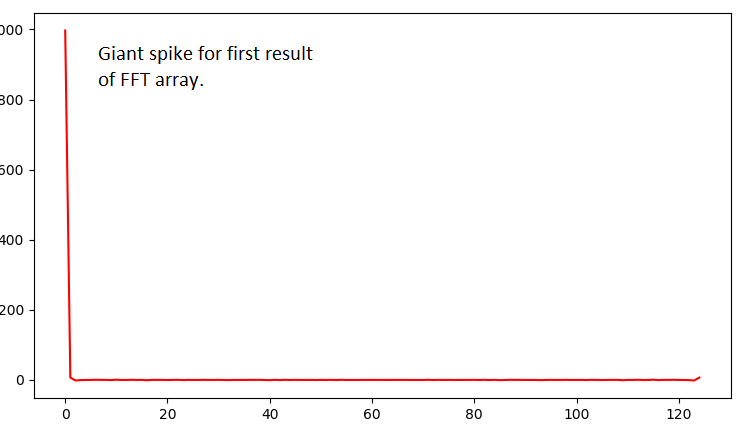
**Example data for one 5-second sample for 1 column:**  


## Run FFT

Take each sample (125 rows of 45 columns) and run each column through FFT (Fast Fourier Transform).

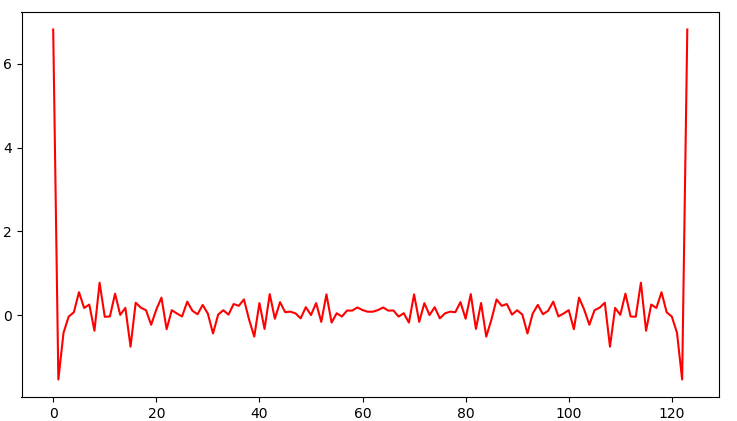


**Output from FFT:**



## Run FFT: Exclude first row

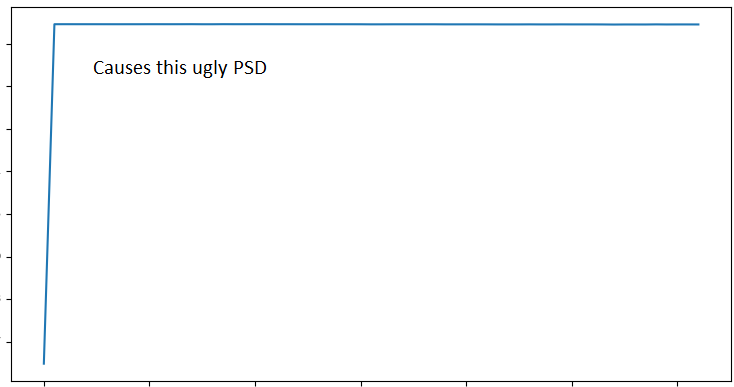
Because of the giant spike (positive or negative) for the first row in the output FFT, the values for the FFT are skewed drastically. Removing the row allows PCA and the K-means to better fit without noise (Assumption).

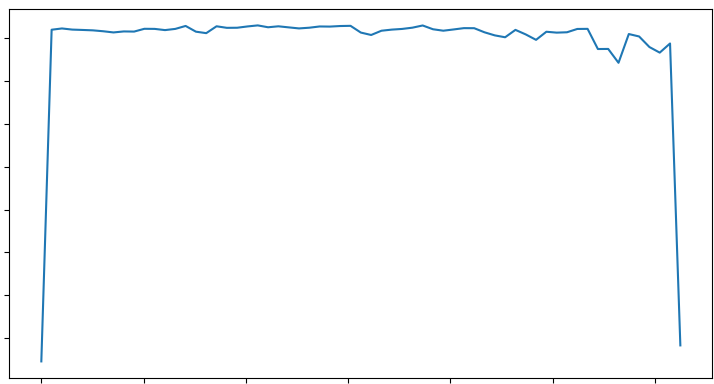
**After removing the first row the resulting FFT looks like this:**

## Run PSD

The resulting 124 rows from the FFT are run through the PSD (Power Spectral Density) which outputs a total of 63 rows.



**Example of PSD without removing 1st column from FFT.**

**Example of PSD after removing 1st column from FFT.**

## Concatenate rows into 1 row.

To make sure PSD runs correctly as 1 row per sample (instead of 63), each of the 63 rows are stacked end-to-end. Producing a (63 \* 45) = 2835 columns



## Run PCA

The output from PCA reduced the 2835 features down to 689 features. PCA was run on the ENTIRE PSD dataset of 9120 by 2835. As I was using 32 bit python, I had to save the PSD data to a file, stop the python code, comment out the FFT/PSD code, and run the PCA code on the loaded file. This is because I was running out of memory in the 32bit address space allotted to python 32bit.



# Clustering

The k-means algorithm was implemented but during testing on the synthetic data provided, centroids from the right side of the data would not move over to the left side, and often the left side would end up with only 2 centroids instead of 3. To combat this, the coherence value for the cluster and separation values between centroids were calculated. Then K / 3 centroids would be randomly replaced in the dimensional space. Then Kmeans would run again. If the result was a lower coherence value, and a higher separation value, the new centroid locations would be kept. This was done multiple times in a count down fashion. The counter would be reset if a better centroid orientation was found. This allows the kmeans to always find the centers of the 8 clusters in the synthetic data. Visualizing 689 dimensions is a struggle, so instead the Mean Entropy was used.

