Activity Recognition Project
CSE 572: Data Mining
Spring 2020

Kenji Mah[1]

[1]**kmmah@asu.edu**

ABSTRACT

The purpose of this project is to gain practice in performing data pre-processing tasks such as data cleaning, data organization, feature extraction, and feature selection with Principle Component Analysis. The overall goal for this project is to determine if a user is eating or not eating given a labeled data collection of EMG and IMU sensors.

Keywords: data pre-processing, feature extraction, feature selection

PHASE 1

For this phase there is one precondition that needs to be addressed and that is in order to run the code you must add a comma in the file 1503609551913.txt, from user19/spoon. I also noticed that in the ground truth file "groundTruth\user32\spoon\1503701139779.txt" one of the entries contained an invalid eating time frame where the end frame < start frame which I just ignored when labeling eating vs non-eating. I decided to separate the eating and non-eating instances fore every user and concatenated them into 2 pandas data frames (eatingMatrix and nonEatingMatrix). In order to syncronize the EMG and IMU data with the video data I needed to convert the ground truth file's eating frames by multiplying it by 100/30 for EMG and 50/30 for IMU. I decided to implement the labeling process stated in the DataMining_project_help.doc document where it suggests starting from the last frame and to use the more precise video frame rate from the video.csv file to get a more accurate eating labels.

eatingMatrix - DataFrame

Index	TimeStamp	Orientation_X	Orientation_Y	Orientation_Z	Orientation_W	Accelerometer_X	Accelerometer_Y	Accelerometer_Z	Gyroscope_X	Gyroscope_Y	Gyroscope_Z	label
0	1503510457691	-0.76	-0.645	-0.01	0.081	0.931	0.009	-0.141	-23.125	-5.125	-9.625	eating
1	1503510457713	-0.763	-0.642	-0.006	0.08	1.034	-0.09	-0.111	-14.438	-23.875	-14.5	eating
2	1503510457736	-0.767	-0.637	-0.002	0.079	1.106	-0.117	-0.109	-16.938	-36.5	-27.562	eating
3	1503510457751	-0.774	-0.629	0.005	0.074	1.059	0.021	-0.215	-12.688	-57.688	-39.375	eating
4	1503510457775	-0.782	-0.62	0.009	0.069	0.951	-0.023	-0.198	-4.312	-70.375	-38.75	eating

nonEatingMatrix - DataFrame

Index	TimeStamp	Orientation_X	Orientation_Y	Orientation_Z	Orientation_W	Accelerometer_X	Accelerometer_Y	Accelerometer_Z	Gyroscope_X	Gyroscope_Y	Gyroscope_Z	label
0	1503510449725	-0.666	-0.665	0.292	0.169	1.013	0.174	0.083	-1.438	0.188	0.062	not-eating
1	1503510449747	-0.667	-0.665	0.292	0.169	1	0.182	0.086	-1.062	-1.75	-0.5	not-eating
2	1503510449763	-0.667	-0.664	0.292	0.169	0.977	0.183	0.095	-0.938	-2	-0.812	not-eating
3	1503510449785	-0.667	-0.664	0.293	0.169	0.967	0.183	0.098	-1.312	-1.312	-0.25	not-eating
4	1503510449808	-0.667	-0.664	0.293	0.169	0.972	0.179	0.097	-1.625	-0.75	0.125	not-eating

I also extracted distribution of the span of frames for the eating class to further understand the data in Figure 1. This visualization was extracted from the IMU conversion of the ground truth, but it is important to note the EMG data is the same distribution but at a different scale (2 times to be exact).

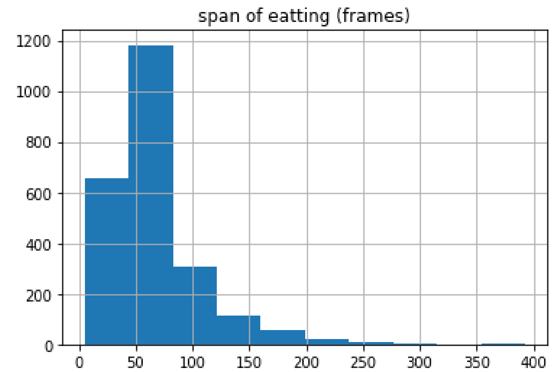


Figure 1. IMU distribution of eating frame lengths

PHASE 2

In this phase we will be extracting features using column wise aggregation. The selected methods of extraction are as follows: Fast Fourier Transformation, Mean, Standard Deviation, Max and Min. They will all be normalized using sklearn's standard scalar function to better visualize the data on a similar scale. These features are extracted from the entire eating matrix and a subset of the non-eating matrix with the same number of samples in each to balance the number of eating and non-eating samples.

1. Fast Fourier Transformation

The Fast Fourier Transformation converts data representing a signal, into data representing the frequencies that make up that signal's mean amplitude. It is important to keep the same time sequence of eating and non-eating actions because we are trying to see if there are any frequency patterns in the data, so I made sure to keep the samples consistent with their temporal positions when separating eating and non-eating instances. I primarily followed a tutorial by balzer82 (2014) for converting the signal with the Fast Fourier Transformation. I then used the argmax function to extract the top 5 frequencies from the transformation. The reasoning behind taking the top 5 is that the 0 *hz* frequency typically shows up as the highest because I decided not to center the signal. Another reason is to look beyond the potential noise frequencies that the sensors pick up naturally.

I thought this method would be able to capture frequencies patterns primarily acceleration and deceleration patterns and maybe muscle patterns from the EMG data. Below are the results of FFT and it shows that there may be potential differences in the gyroscope data, but it's hard to tell when looking at the EMG data. There seems to be possible differences EMG1, EMG3, EMG5, and EMG8.

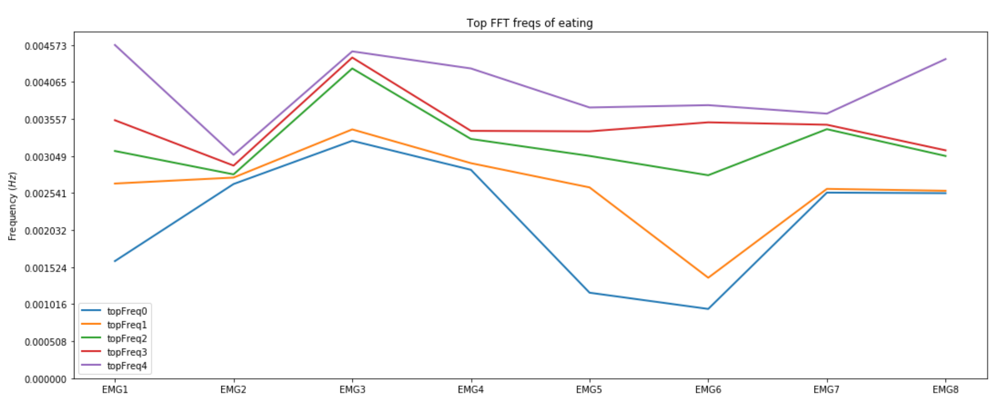


Figure 2. FFT's

2. Average

Average is taken by summing up the values and dividing by the number of values summed. My intuition behind this is so that we could hopefully find distinguishing orientation because your hand oriented a certain way when you eat and taking the average is able to summarize an eating movement and distinguish it from a non-eating one. There seems to be a difference in all all EMG's as well as x and w orientations. There also seems to be differences in the accelerometer and gyroscope data.

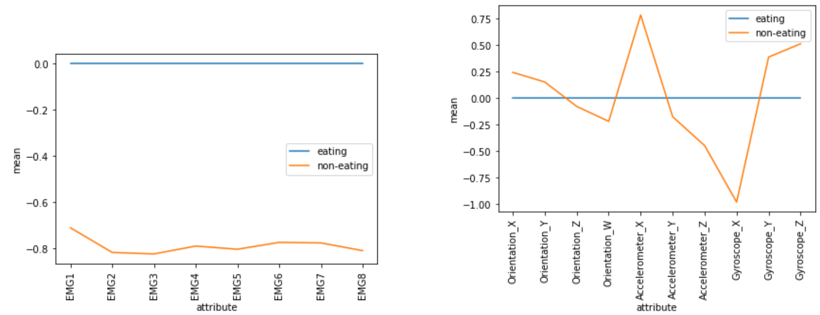


Figure 3. Mean

3. Standard Deviation

The standard deviation is taken with the help of python functions to find the spread of the data. By analyzing the spread of the data we might be able to detect if there is frequent change is values for the sensors, and it seems that there are great differences in EMG1, EMG2, EMG5, and EMG8 as well as the gyroscopes.

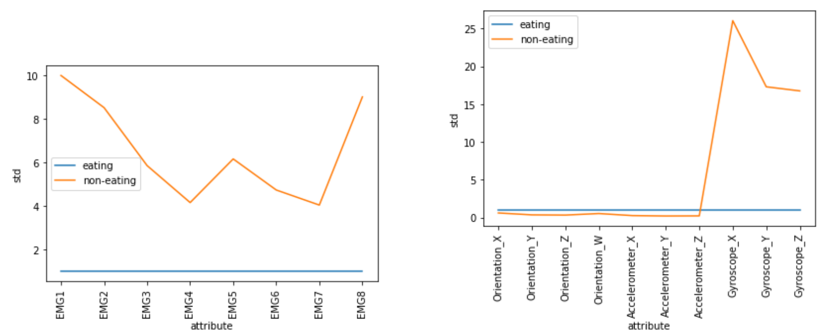


Figure 4. Standard Deviation

4. Max

Max is found by taking the highest value. The intuition behind max is that any profound action will be captured by max, and hopefully it can distinguish at least activity from no activity. It seems that there is a similar scenario as the standard deviation.

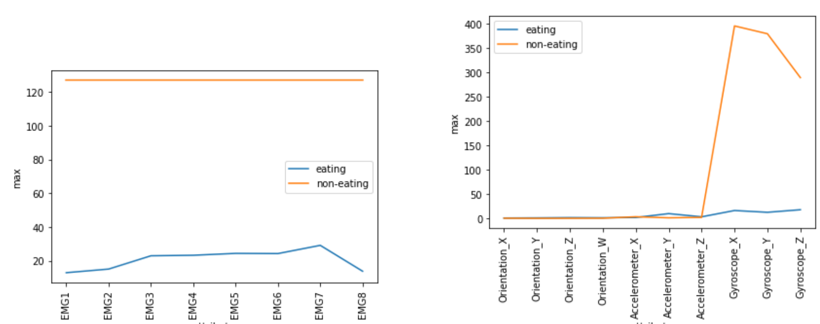


Figure 5. Max

5. Min

Min is found by taking the lowest value. Similar intuition as for max, but this is useful if an eating activity produces lower values for a particular sensor. This also has a similar scenario as the max and standard deviation.

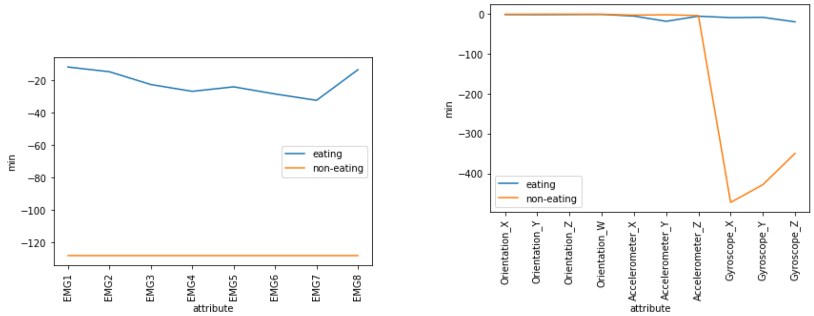


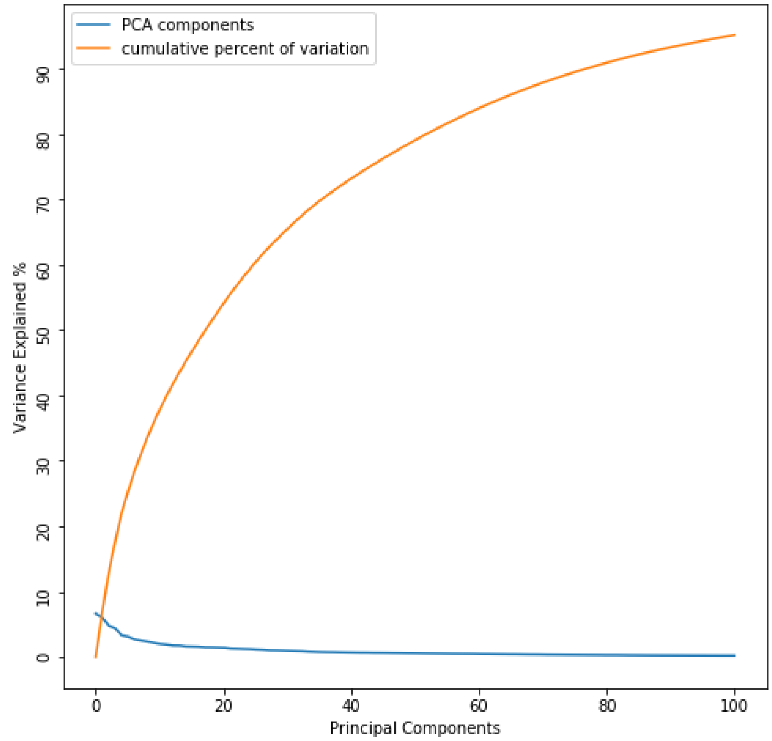
Figure 6. Min

PHASE 3

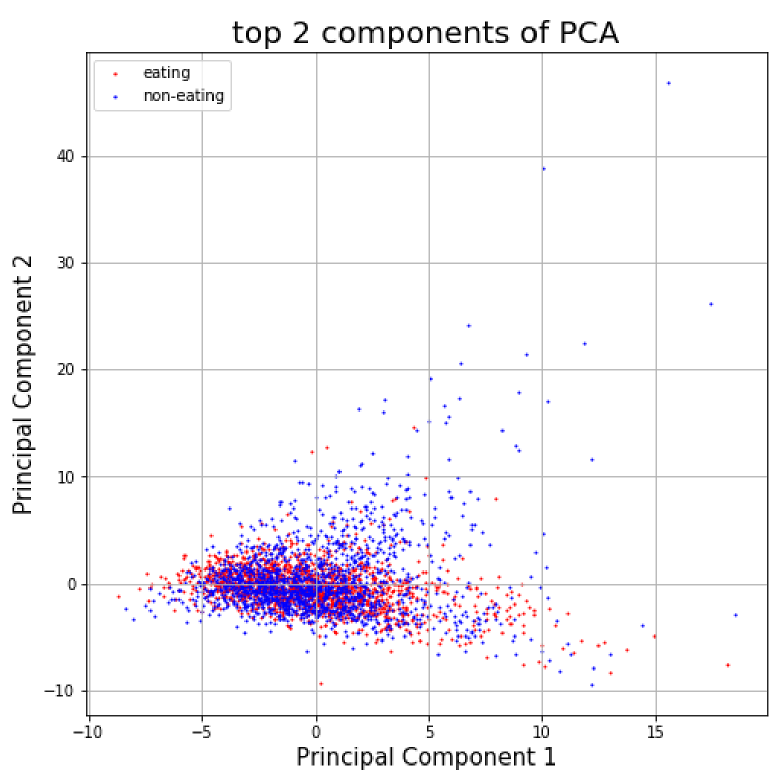
In this phase of the project we needed to set up the input matrix for our principal component analysis. I decided to use a window of 100 frames per window for IMU data and 200 frames per window for EMG data to sync them up. I would then apply the 5 new features to each of these windows which would then total to 162 features (that's because every FFT calculation actually adds 5 new features). Of course it is important to keep the classes separate when doing this so the aggregated matrices in phase 1 will come in handy. After extracting the new features we end up with a matrix with 3172 samples and 162 features. Now we can normalize the data using sklearn libraries and input it into the sklearn PCA function as followed by Michael Galarnyk's PCA tutorialGalarnyk (2017). Here are some of the eigenvalues for the PC's. The rows represent the principle component number. The higher the eigenvalue the higher the explained variance

	0
0	10.7092
1	9.85135
2	7.76292
3	7.1592
4	5.35805
5	5.02146
6	4.3645
7	4.06612
8	3.79573
9	3.48968
10	3.19343
11	3.04388
12	2.81081

The PCA object created in python stores the explained variance ratio as a vector in its explained_variance_ratio_ variable, so we don't have to calculate it using the eigenvalues and the total common variance. After extracting the top 100 components and graphing them with their respective variance explained percentages, it produces the following plot:



We can see that the first principle component accounts for about 6.6% of the variation and the first 100 Principle components accounts for 95.4% total variance. doing further analysis, if we wanted to achieve 99% of the total variation we would need to keep the top 131 components which is a bit less than the total 162. In the next plot we show the PCA feature matrix using the top two principle components of our PCA.



There seems to be some separation of the eating and non eating labels, some eating labels stray slightly to the right while some non eating labels stray slightly upper right. It hard to tell but that is why we need to keep more principle components in order to achieve a decent amount of explained variation. With these 2 components, they only account for 12.7% of the total variance.

Now lets compare our PCA with our intuitions. The eigenvectors are stored in the components_ variable By taking the top 5 PC's and by looking at the top 14 features (because 162 would be too much) we can compare our results with our previous phase. PC's 1 to 5 have the following percent explained variance:

PC1: 6.6% PC2:6.1% PC3:4.8% PC4:4.4% PC5:3.3%

PC1Vecs - Series

Index	PC1
EMG6 std	0.0445089
EMG4 std	0.0410174
EMG6 max	0.0398368
EMG7 std	0.0384081
EMG5 std	0.0363719
EMG4 max	0.0331789
EMG6 min	0.0313781
EMG8 std	0.0309995
EMG5 max	0.0303327
EMG4 min	0.0280108
EMG7 max	0.0279128
EMG3 std	0.0274184
EMG7 min	0.0266334
EMG8 max	0.0253865

PC2Vecs - Series

Index	PC2
Accelerometer_Z topFFT3	0.038758
Accelerometer_Y topFFT3	0.0380639
Accelerometer_Y topFFT4	0.0371129
Accelerometer_Z topFFT4	0.0358158
Gyroscope_X topFFT4	0.0333229
Gyroscope_X topFFT5	0.0314504
Gyroscope_X topFFT3	0.029564
Gyroscope_Y topFFT3	0.0283112
Gyroscope_Z topFFT3	0.0271431
Accelerometer_Z topFFT2	0.0270253
Gyroscope_Z topFFT4	0.0264225
Accelerometer_Y topFFT5	0.0258666
Gyroscope_Y topFFT2	0.0256822
Accelerometer_Z topFFT5	0.0252033

PC3Vecs - Series

Index	PC3
Gyroscope_X std	0.0576582
Gyroscope_Z std	0.0558404
Gyroscope_Y std	0.054341
Gyroscope_X max	0.0509169
Gyroscope_Z min	0.0507939
Gyroscope_Y max	0.0487892
Accelerometer_Y std	0.0449438
Accelerometer_X std	0.0428685
Accelerometer_Z std	0.0355421
Accelerometer_Y min	0.0344867
Gyroscope_Z max	0.0322166
Orientation_Y std	0.0237792
Orientation_Z std	0.0225131
Orientation_Z min	0.0197859

PC4Vecs - Series

Index	PC4
Orientation_X min	0.0856767
Orientation_X mean	0.0853672
Orientation_X max	0.0833226
Orientation_Y min	0.0728267
Orientation_Y mean	0.0688931
Orientation_Y max	0.0650418
Orientation_W max	0.0445174
Orientation_W mean	0.0438352
Orientation_W min	0.0412796
Accelerometer_X max	0.0218555
Accelerometer_X mean	0.0196896
EMG6 std	0.0188251
EMG6 max	0.0177236
EMG2 std	0.0160751

PC5Vecs - Series

Index	PC5
Orientation_Z max	0.0618517
Orientation_Z mean	0.0577005
Accelerometer_X min	0.0558036
Orientation_Z min	0.0530801
Accelerometer_X mean	0.0489743
Accelerometer_X max	0.0385597
EMG1 min	0.0337015
Gyroscope_Y mean	0.0323102
EMG1 std	0.0322395
EMG2 std	0.0305827
EMG1 max	0.0296792
EMG2 max	0.0241957
Gyroscope_X max	0.022521
EMG2 min	0.0220914

Most of the eigenvectors agrees with our graphical analysis's of the previous phase except the FFT accelerometer data. This could be due to the different scope of how we analyzed the data. From our graphical analysis we transformed the entire eating and non-eating samples, whereas in PCA we aggregated each in rows of 100, then looked at the variation. At least PCA confirmed our initial gut intuition for using FFT as a feature that we decided to overlook using the graphical analysis.

In summary PCA is very helpful for doing a mathematical analysis for the most variant features because it can determine variance that can be overlooked when doing an empirical analysis of the data. However, graphical analysis can be use as a quick, general understanding of the variance. Now we can reduce the number of features from 162, however that depends on a lot of factors such as how much percent of variance do we want to keep, how many features we want to keep, etc. Typically 99% of the variance is the common cutoff point for most projects so I will probably be using only the first 131 PC's.

REFERENCES

balzer82 (2014). Fft with python. *https://github.com/balzer82/FFT-Python/blob/master/FFT-Tutorial.ipynb*.

Galarnyk, M. (2017). Pca using python (scikit-learn). *https://towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60*.