Classification of Population Activity in Parkinson's Disease

Gustav Röhss, Míriam Vall

VT 2020

Abstract

Sammanfattning

Contents

1 Introduction

1.1 Purpose

TODO: Using ”brain activity” as supplement for LFP, spiking rates, etc

The purpose of this project is to attempt to find one or several models for classification of the brain activity in patients with Parkinson’s disease.

Furthermore this project aims to, to some extent, use any produced model(s) to evaluate differences is brain activity of different categories. It is of interest to consider what differences any such model(s) show when comparing brain activity from different brain regions. It is also of interest to make a similar comparison for the brain activity in different patients.

1.2 Delimitations

The authors of this report are not well educated or experienced in studying brain activity. The model(s) produced are mainly means to serve as a strong foundation for further research into deeper understanding of Parkinson’s disease. The authors attempt to describe and interpret output produced by the model(s), but do so outside of any broader implications the model(s) show in the further study of Parkinson’s disease.

No software used, or produced, both by others and by the authors, is subject to extensive formal verification within the scope of this project. The same is true for the datasets used within the scope of this project. The datasets used are instead assumed to have been produced/recorded to a satisfactory quality for their uses within the scope of this project.

1.3 Research questions

- How can effective models for classification of brain activity in patients with Parkinson’s disease be produced?
- How can such models be used to distinguish between brain activity from different regions of brains in patients with Parkinson’s disease?
- How can such models be used to distinguish between brain activity from different patients with Parkinson’s disease?

2 Background

2.1 Discrete Fourier transform

The Fourier transform, or more specifically, the family of Fourier transforms, are mathematical tools with a long and rich history and many use cases. The Discrete Fourier Transform (DFT) is the version of the Fourier Transform used on discrete points of data (rather than e.g. a continuous function). One use case of the Fourier transform is to transform a function of time into a function of frequency. The DFT can be said to convert data from the *temporal* (time) domain to the *spectral* (frequency) domain.

Specifically, the Fourier transform can be used to approximately decompose a function, or a series, into a large number of waves of different frequencies and amplitudes. This method can be used to approximate the amplitude or power of activity in specific frequencies in a signal made up of waves of many frequencies (MathWorks n.d.).

NumPy, a software library, provides useful tools for usage of the DFT in its' `fft` package, specifically the `numpy.fft.fft` (`FFT`) function (Oliphant 2006–).

`FFT` can also be given input to pad the input array with additional zeroes. The user then receives a *higher fidelity* output; output information for a larger amount of frequencies. This can be shown by the `FFT`-helper function, `numpy.fft.fftfreq` (`FFTFREQ`). The `FFTFREQ` function takes arguments *window length* and *sample spacing*, and returns an array of *unit frequency bin centers*.

The amplitude spectrum for `FFT` output is obtained by taking the absolute values of the output from the `FFT` complex-valued output array, specifically using the `numpy.abs` (`ABS`) function (Oliphant 2006–).

2.2 k-Means

The k-Means algorithm is a clustering algorithm. One noticeable peculiarity of the k-Means algorithm is that the user makes a choice of k, the amount of clusters.

The algorithm works by first randomly generating k initial *cluster mean vectors*. These are vectors with the same dimensionality as the data samples to be clustered. The algorithm then attempts to minimize the *within-cluster sum of squares* of samples assigned to each cluster; the sum of square (Euclidean) distances from the cluster mean vectors, taken over the individual data samples. The algorithm works iteratively.

- Each sample is assigned to the cluster for which the square distance is minimized.
- New cluster mean vectors are created from the mean of the new assignments of samples to clusters.

The iteration ends when the cluster mean vectors no longer change (possibly within some tolerance), or a set amount of iterations is reached (Bruce & Bruce 2017, p258-260).

The *scikit learn* software library has an implementation of the k-Means algorithm in its' `sklearn.cluster.KMeans` module `(KMeans)`, and is the implementation used in this project (Pedregosa et al. 2011).

2.3 Principal component analysis

One method for extracting lower-dimensional features from higher-dimensional data is by using principal component analysis (PCA). Specifically, PCA refers to the computation and use of principal components (PCs). For a set of data, a PC is a direction in the space of the data's features along which the data samples are highly variable. It's possible for a linear combination of PCs to describe all samples in a dataset with a great degree of accuracy. If the amount of PCs required for this is lower than the amount of features in the data samples, PCA becomes an effective means of feature reduction for that dataset. The PCs produced can also be used to visualize the data, and the individual components can be interesting for analyzing the data in their own right (James et al. 2017, p374-380).

The *scikit learn* software library enables easy computation of PCs using `sklearn.decomposition.PCA` (`PCA`) It also allows the user to transform members of a dataset into their respective representation under a certain set of PCs. It should be noted that such a representation is often approximate. Furthermore, the user is able to see the *explained variance* and *ratio of explained variance* for each PC produced for a specific dataset (Pedregosa et al. 2011).

3 Methods

Some methods in this project focused on amplitudes of activity in the spectral domain for the data used in the scope of this project. The reasoning behind this is that previous research on Parkinson's disease has shown that LFP activity in the beta-range is abnormally synchronized compared to that of the same activity in subjects unaffected by Parkinson's disease. **TODO: Source** This suggests that such activity might embed additional useful information for research, and possibly a means for classification.

3.1 Spectrum feature extraction

One important method for feature extraction used in this project was the DFT, using `FFT`. The data was first split into uniform-sized (in array length) *epochs*. Software implemented for this end was designed such that the *epoch size* could be varied. Each such epoch was then transformed using `FFT`.

Interpreting `FFTFREQ` for the data used in this project, it takes input epoch size (in number of points) and time between samples (multiplicative inverse of sampling frequency). It returns an *array index-to-frequency in Hertz* mapping for the output of `FFT` (Oliphant 2006–).

The `ABS` function was used to produce the amplitude spectrum of `FFT` output. `FFTFREQ` was used to find indices in the `FFT` output representing frequencies in a certain range. This range was implemented to be variable.

Using this process, for each epoch a *feature vector* was produced. Each value in the vector represents an amplitude of LFP activity for a specific frequency, epoch, channel, and session. For ease of reference, these vectors will be referred to as *spectrum feature vectors* (SFVs) in this report.

3.2 k-Means as a visualization heuristic

Initially, k-Means was considered as a means for clustering SFVs in an attempt at classification. Attempting to support this choice with an argument as to why it is an appropriate choice of algorithm for this particular use case proved difficult. However, a heuristic argument can be presented.

The SFVs represent a portion of an amplitude spectrum. The square of an amplitude spectrum is a power spectrum (Oliphant 2006–). The distance between a cluster mean vector (as generated by k-Means) and a SFV can therefor be interpreted as a "difference in amplitude"-spectrum. As such, the square of this distance can be interpreted as a "difference in power"-spectrum. Then, when using k-Means with the SFVs, the "distance" the algorithm will be attempting to minimize is the sum of this "difference in power"-spectrum; or just difference in power.

To reiterate, the presented argument should not be considered extensive for this use case, but rather should be considered a heuristic motivation for its' use as, primarily, a method of visualization.

With this in mind, `KMeans` was used to produce cluster means and assignments for some subsets of SFVs produced. Specifically, a set of SFVs for all channels of a certain session were first produced. A subset of these SFVs were then used as training data for `KMeans`. The resulting `KMeans` model was then used to classify (predict) the entire session-SFV-set. This procedure was repeated for several different sessions. The parameters used for k for `KMeans`, the epochs size, and the range of frequencies included in the SFVs are able to be varied in order to produce extensive results.

3.3 Usage of principal component analysis

In order to better describe the SFVs, PCA was used.

The specific PCs of the SFVs are of interest, as they describe the spectrum-components along which LFP activity vary the most. Analyzing these specific PCs and how they differ for different subsets of SFVs might highlight key similarities and differences in the LFP activity of different brain regions or subjects.

It is also interesting to consider the PCs of the total set of all SFVs produced. Should a small amount of PCs prove able to explain a high ratio of variance in the dataset, this would serve as a strong means for feature reduction, which could then be used in further research. The PCs produced would then also describe in a more easily digestible manner the key spectral components of any LFP activity in this dataset, and have implications for using similar methods for classification of other sets of LFP activity, and possibly brain activity and time-signals in general. Such a set of PCs could also be used to highlight differences between different LFP categories. Should the distribution of SFVs transformed into PC representations under this model be considerably different for different brain regions or animals, this would have implications for classification attempts of LFP activity.

The process for producing PCs was straightforward. Using `PCA`, either with some subset of available SFVs, or the entire set of SFVs, the PCs for these sets were produced.

4 Results

TODO: Introduction to results.

4.1 Spectrum feature vectors

Produced SFVs can be visualized using the *matplotlib* software library's useful `matplotlib.pyplot.imshow` (`IMSHOW`) function (Hunter 2007).

Figure 1 and figure 2 show SFVs for channels `gp_lfp1` and `str_lfp10` for sessions `NPR052e.10` and `NPR064.b08`, respectively. For both figures, each column represents a single SFV. For both figures, higher brightness represent higher amplitude of LFP activity in that epoch and frequency. For both figures, each column is the SFV generated for a specific epoch of activity, displayed chronologically. Figure 1 shows 780 epochs, and figure 2 shows 778 epochs. For both of these sessions the sampling frequency is 16 kHz. For both of these sets of produced SFVs, the epoch size in amount of data points (sampled at sampling frequency) is 2048, resulting in an epoch size of 128 ms. For each of these sets of produced SFVs, there are 46 frequencies sampled. Specifically, the input was padded with to a length of 16384 in order to produce a higher fidelity output, and all frequency samples in the ranges of 5 Hz - 50 Hz were selected. This results in 46 equally-spaced frequency samples, the lowest being approximately 5.86 Hz and the highest being approximately 49.8 Hz. The parameters can be considered arbitrary due to easily being changed. In this specific configuration, the frequency range was selected to be somewhat wider than the beta-range of frequencies. The epoch size was chosen to be somewhat longer than the shortest length of a beta-burst (Cagnan 2019).

These figures are, clearly, a very small subset of the total set of produced SFVs. They also represent only a very small subset of possible configurations of SFVs in regards to epoch size, sampled frequencies, and amount of frequency samples.

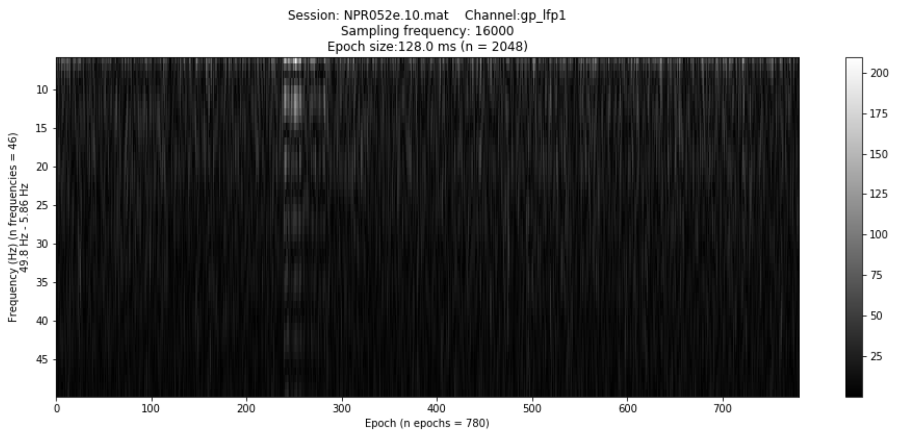

Figure 1: SFVs for Globus Pallidus LFP channel of a specific session.

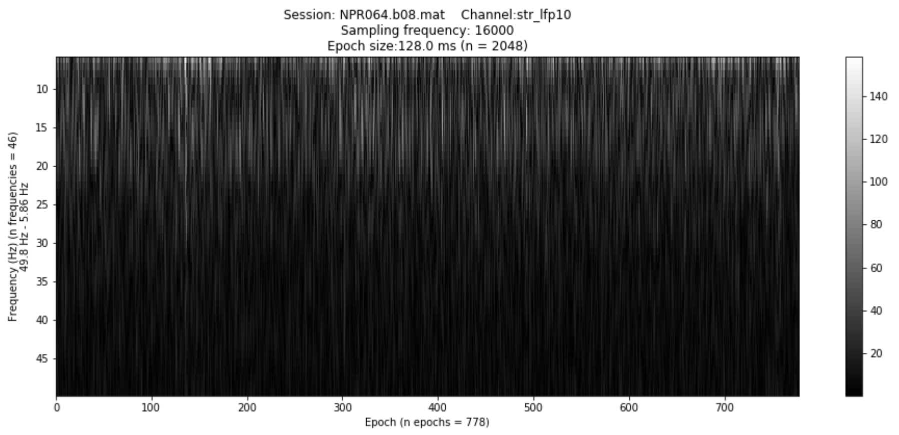

Figure 2: SFVs for Striatum LFP channel of a specific session.

4.2 k-Means

To visualize k-Means predictions for a k-Means model trained on SFVs, `IMSHOW` was used once more. Some of the produced figures are figure 3 and figure 4. In order to better relate to the heuristic argument presented in subsection 3.2, the colors for class assignments were chosen to be sorted by the "sum of power spectrum" (sum of squares) of the cluster centers. Lower class index (refer to color bar in either figure) represents lower sum of power spectrum. The indices of class assignments will be presented as "low" and "high" for brevity.

In the figures, each row represents the class assignments for a specific channel in the set. The columns represent epochs. Each colored "slice" is the prediction of a single SFV to a class, the result of a trained `KMeans` model. The particular channel names are not shown in these figures, but include both STR and GP channels. The SFVs are identical in production parameters to those presented in the previous section.

The most important thing to note about these figures is how different channels in the same (or close) epochs tend to be given equal class assignments excessively. The authors interpret the class assignments for adjacent epochs to often be ”close”, meaning that class assignments seem to be followed by slightly higher or slightly lower class assignments, this was however not researched more thoroughly and should be considered an informal observation. Also of note are ”streaks” of higher class assignments for specific channels. Examples include for channel 4 at about epoch 700, and channel 12 at about epochs 330-420 (rough estimates). There are also ”pillars” of excessively similar assignments for all channels at about epochs 140, 290, and 550 (rough estimates).

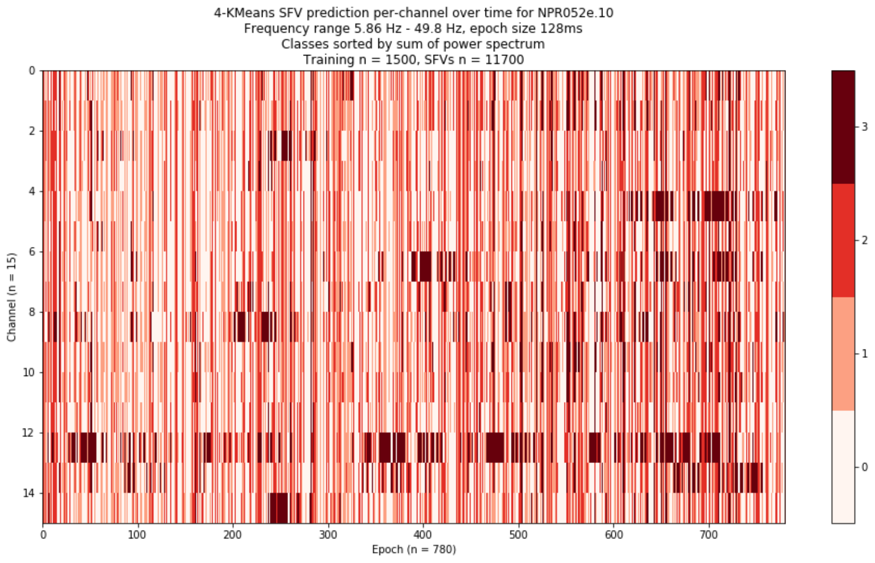

Figure 3: 4-k-Means of channels for specific session

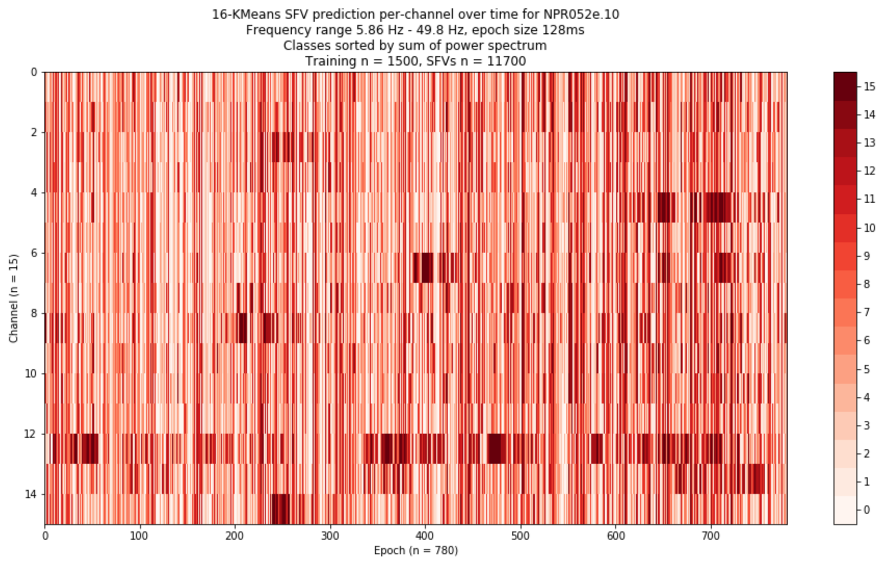

Figure 4: 16-k-Means of channels for specific session

4.3 Principal component analysis of spectrum feature vectors

TODO: Slightly increase height of PCS plots. TODO: Crosscorrelations within GP, STR.

The method outlined in subsection 3.3 was performed using a set of 362,184 SFVs produced with parameters identical to those described in subsection 4.1. Indeed, the SFVs shown in figures 1 and 2 are part of the set used to compute the `PCA` model.

The first thing to consider about the PCs computed for the set of SFVs are the PCs themselves. Figure 5 shows these, as well as their respective explained variance ratios. This figure was produced using the `matplotlib.pyplot.plot` (`PLOT`) function. In this specific case, eight PCs were produced, as this was the points where their cumulative sum of explained variance ratio was above 90%. These PCs can be interpreted as being parts of which a linear combination approximately describe all of the SFVs from which they were computed.

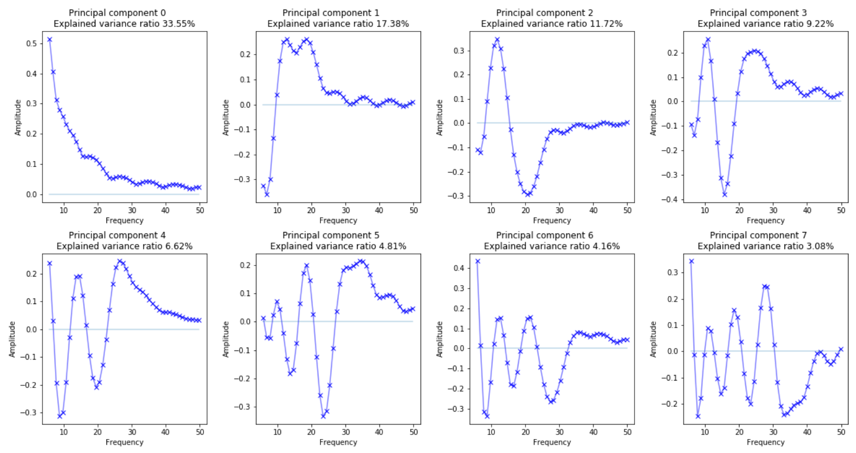

Figure 5: Principal components of SFV-set

Having computed these PCs, the approximate probability distributions of the individual components as they appear in the set of `PCA`-transformed SFVs are shown in figure 6. In order to better visualize the long tails of these distributions, the probability distributions of the logarithms (base 2) of the individual components are shown in figure 7.

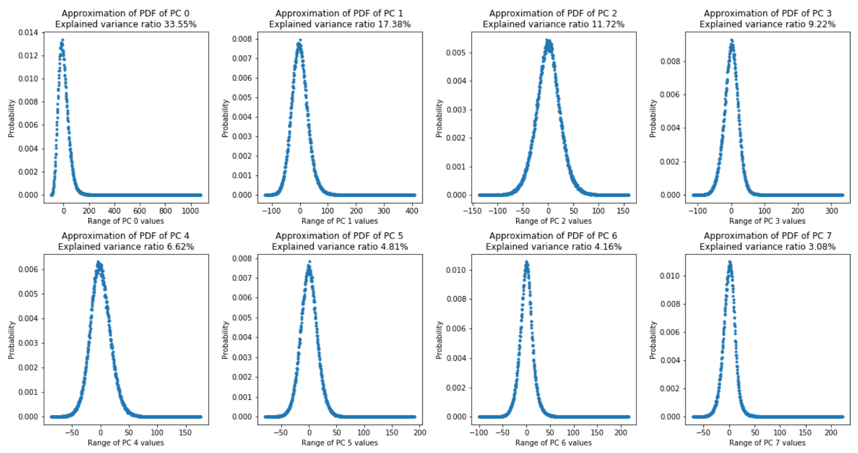

Figure 6: Approximate distributions of principal components of SFV-set

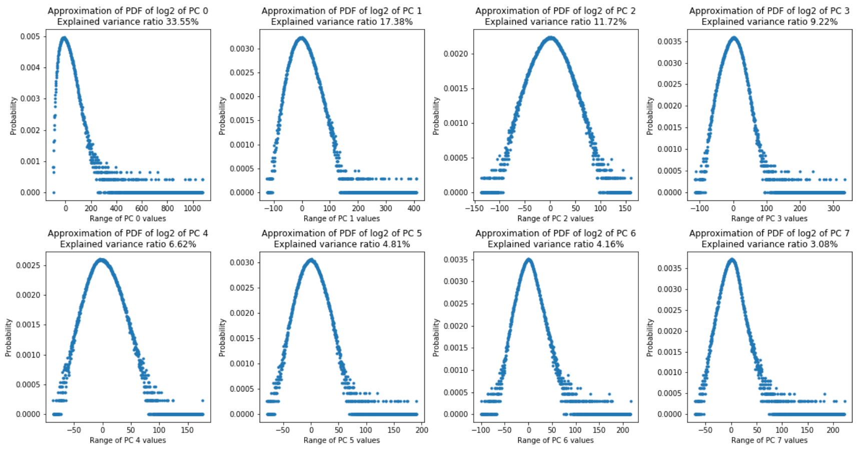

Figure 7: Approximate distributions of log_2 of principal components of SFV-set

The information shown in figures 6 and 7 is also shown in figures 8 and 9, except with channels from STR kept separate from those of GP. Notably, GP exhibits distributions with more width and higher peaks.

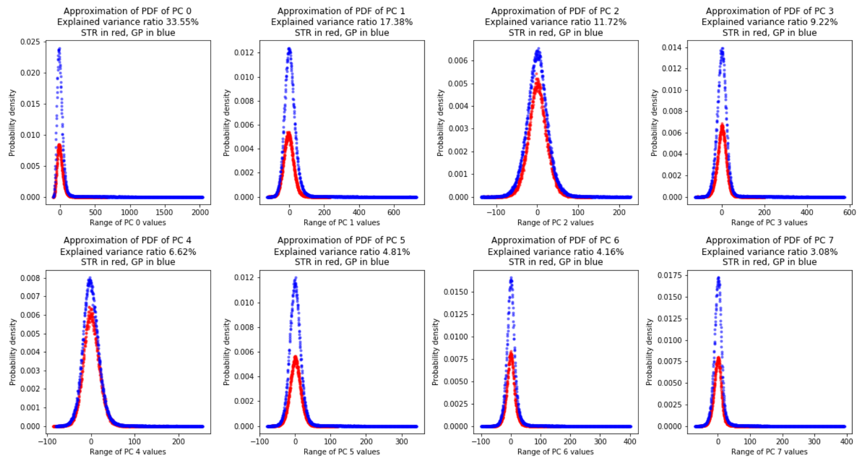

Figure 8: Approximate distributions of principal components of SFV-set, separated by channel type

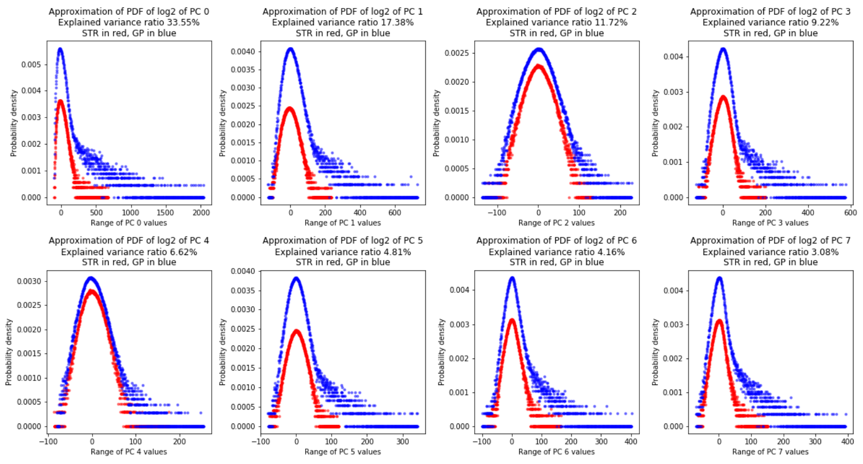

Figure 9: Approximate distributions of log_2 of principal components of SFV-set, separated by channel type

These figures, of course, don't show how these distributions correlate. Tables 1 and 2 show the cross-correlation of `PCA`-transformed SFVs, for GP and STR channels respectively. Notably, cross-correlation within STR channels is consistently somewhat low, while it is consistently quite higher within GP channels.

	PC 0	PC 1	PC 2	PC 3	PC 4	PC 5	PC 6	PC 7
PC 0	1.0	0.43	0.17	0.44	0.23	0.39	0.44	0.5
PC 1	0.43	1.0	0.1	0.26	0.13	0.26	0.26	0.31
PC 2	0.17	0.1	1.0	0.11	0.07	0.08	0.12	0.13
PC 3	0.44	0.26	0.11	1.0	0.16	0.26	0.31	0.35
PC 4	0.23	0.13	0.07	0.16	1.0	0.13	0.16	0.19
PC 5	0.39	0.26	0.08	0.26	0.13	1.0	0.27	0.32
PC 6	0.44	0.26	0.12	0.31	0.16	0.27	1.0	0.35
PC 7	0.5	0.31	0.13	0.35	0.19	0.32	0.35	1.0

Table 1: Cross-correlation of PCs, GP channels, rounded to two decimals.

	PC 0	PC 1	PC 2	PC 3	PC 4	PC 5	PC 6	PC 7
PC 0	1.0	-0.05	0.01	0.0	0.0	-0.01	-0.0	0.01
PC 1	-0.05	1.0	0.05	0.08	0.03	0.01	0.04	0.03
PC 2	0.01	0.05	1.0	0.03	-0.02	0.02	0.02	-0.01
PC 3	0.0	0.08	0.03	1.0	-0.0	0.03	-0.02	0.01
PC 4	0.0	0.03	-0.02	-0.0	1.0	0.01	0.01	-0.0
PC 5	-0.01	0.01	0.02	0.03	0.01	1.0	-0.0	-0.01
PC 6	-0.0	0.04	0.02	-0.02	0.01	-0.0	1.0	0.0
PC 7	0.01	0.03	-0.01	0.01	-0.0	-0.01	0.0	1.0

Table 2: Cross-correlation of PCs, STR channels, rounded to two decimals.

5 Discussion

6 Conclusion

7 References

References

Bruce, P. & Bruce, A. (2017), *Practical Statistics for Data Scientists*, O'Reilly Media, Inc.

Cagnan, H. e. a. (2019), 'Temporal evolution of beta bursts in the parkinsonian cortical and basal ganglia network', *PNAS* **116**(32), 16095–16104.

Hunter, J. D. (2007), 'Matplotlib: A 2d graphics environment', *Computing in Science & Engineering* **9**(3), 90–95. `https://matplotlib.org/index.html` [051220], `https://matplotlib.org/3.2.1/api/_as_gen/matplotlib.pyplot.plot.html` [051220], `https://matplotlib.org/3.2.1/api/_as_gen/matplotlib.pyplot.imshow.html` [051220].

James, G., Witten, D., Hastie, T. & Tibshirani, R. (2017), *An Introduction to Statistical Learning*, Springer.

MathWorks (n.d.), 'Fourier transforms'. `https://se.mathworks.com/help/matlab/math/fourier-transforms.html` [032520].

Oliphant, T. (2006–), 'NumPy: A guide to NumPy', USA: Trelgol Publishing. `https://numpy.org/` [051320], `https://numpy.org/doc/1.18/reference/routines.fft.html` [051320] `https://numpy.org/doc/1.18/reference/generated/numpy.fft.fft.html` [051320], `https://numpy.org/doc/1.18/reference/generated/numpy.fft.fftfreq.html` [051320], `https://numpy.org/doc/1.18/reference/generated/numpy.absolute.html` [051320].

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011), 'Scikit-learn: Machine learning in Python', *Journal of Machine Learning Research* **12**, 2825–2830. `https://scikit-learn.org/` [050520], `https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html` [051120], `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html` [051120].