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 (KMeans)` module, 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 weak visualization heuristic

3.3 PCA of FFVs

4 Results

5 Discussion

6 Conclusion

7 References

References

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

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/` [060520], `https://docs.scipy.org/doc/numpy/reference/generated/numpy.fft.fft.html` [051120], `https://docs.scipy.org/doc/numpy/reference/generated/numpy.fft.fftfreq.html` [051120], `https://docs.scipy.org/doc/numpy/reference/generated/numpy.absolute.html` [051120].

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].