HMS Harmful Brain Activity Classification

Viba Courtney-Battista 01847210 vc320@ic.ac.uk

April 2024

1 Introduction

The aim of this project is to create a model that can detect and classify seizures and other types of harmful brain activity using signature kernel methods. The data is composed of EEG signals taken from critically ill patients who are experiencing one of six types of seizures, the targets are a distribution of votes over the six seizures. Currently EEG monitoring is conducted with manual analysis by specialised neurologists, which is a labor-intensive bottleneck, prone to errors and has reliability issues between experts, hence this project is aiming to provide a basis for developing models that will allow faster and more accurate treatments to patients.

2 Overview of the dataset

2.1 Metadata

The train data is organised via the train metadata dataframe, train.csv. Expert annotators reviewed 50 seconds of EEG samples with matched spectrograms covering a 10 minute window centred at the same time and labeled the central 10 seconds. These samples may overlap, and is organised by:

- eeg id - A unique identifier for the entire EEG recording.
- eeg sub id - An ID for the specific 50 second long subsample this row's labels apply to.
- eeg label offset seconds - The time between the beginning of the consolidated EEG and this subsample.
- spectrogram id - A unique identifier for the entire EEG recording.

- spectrogram sub id - An ID for the specific 10 minute subsample this row's labels apply to.

- spectogram label offset seconds - The time between the beginning of the consolidated spectrogram and this subsample.

- label id - An ID for this set of labels.

- patient id - An ID for the patient who donated the data.

- expert consensus - The consensus annotator label. Provided for convenience only.

- seizure/lpd/gpd/lrda/grda/other vote - The count of annotator votes for a given brain activity class.

- activity classes - lpd: lateralized periodic discharges, gpd: generalized periodic discharges, lrd: lateralized rhythmic delta activity, and grda: generalized rhythmic delta activity.

The test metadata consists of only the eeg id and the spectrogram id.

2.2 EEG data

Each EEG id corresponds to a parquet file containing the raw EEG data, this is what I will be working with in this project. The EEG sample is a signal taken from 19 sensors on the brain and one on the heart, and is sampled at 200Hz.

3 Data Loading and preprocessing

3.1 Sampling from the dataset

When sampling from the dataset it is important to consider how the model can introduce bias, for example out of the 17,000+ EEG files each with overlapping sub EEGs and multiple taken from one patient may introduce a bias. To avoid this I filter the dataset by unique EEG id/vote distribution pairs, so two sub EEGs in the same EEG are only considered if they are distinct seizure types. This method filters the overlapping samples that may introduce a bias and only keep relevent 50 second snippets from the entire dataset.
In addition to this the dataset has majority 'other' votes, introducing another possibility for the model to become biased. To avoid this we can undersample the dataset aiming for a more even distribution of seizure types. A datapoints

expert consesnus can be categorised into Ideal, Proto or Edge depending on the split of votes for seizures/votes for 'other'. This introduces another way we can subsample the dataset to obtain 'stronger' datapoints by filtering into only idealised votes.
Each datapoint has varying numbers of total votes, so the final way to subsample the dataset that I implemented is to filter by high number of votes. This method has the benefit of not ignoring split cases and uncertain cases ensuring the model can still identify which type of split/uncertain case to classify the sample. Indeed this method yielded the most accurate results.

3.2 Feature engineering

This section is heavily reliant on expert knowledge. Since this model relies on the kernel trick and performs a regression I do not encode and decode to find richer features I have used expert knowledge to conduct the feature engineering:

3.2.1 Missing values

EEG data can contain missing values, which can be dealt with in multiple ways such as interpolating (high frequency recording so linear may suffice, higher order interpolation may improve performance due to the high frequency waves), setting NaN to a constant, back fill/forward filling. A specific method should be chosen based on the architecture of the preprocessing.

3.2.2 Montaging

Many experts agree on the importance of the differences in the signal between neighboring nodes, incorperating this into my model involves taking the difference between the most important neighboring nodes and there are a few ways to do this. The first is bipolar montaging, which defines neighbors to be parallel to the direction the patient is facing, taking this further to reduce the complexity of the model we can take three of the four nodes on the left and right side as the general consensus is they hold the most important information. This is the approach I take when training my model which reduces the time series dimensionality from 20 to 8. Another way to montage is the circumferential montage, considering the outer nodes and defining neighbors in a circular way. I chose the former as this reduces the dimensionality of the path and hence the time complexity significantly, whilst still agreeing with expert consensus.

3.2.3 Denoising

In practice EEG signals can be heavily influenced by artefacts such as blinking, subtle muscle movement, medicines or even oily hair. This introduces an unwanted level of noise that should be reduced. Using a discrete wavelet transform maintains the signal uniqueness whilst still denoising the signal.

3.2.4 Spectral analysis

EEG signals are waves, so it is intuitive that the most important features come from spectral analysis, which may ease the burden of the regression emphasising the periodic patterns. Performing a Fourier transform is a convenient and fast way to transform to a frequency space and can also ease the burden of denoising by only including relevant frequencies such as the frequencies for alpha, beta, theta, delta and gamma waves.

3.2.5 Downsampling

Ten seconds of EEG signals is very memory intensive when creating a kernel matrix, so downsampling must be performed. Slicing the EEG signal directly has the benefit of maintaining the raw data signal, including periodic patterns. Slicing after a Fourier transform loses more periodic patterns so including only the most prominent frequencies is a preferable option, slicing in the frequency space has the benefit of not losing too much spatial information. I use a combination of the two, slicing by 2 in the frequency space followed by 4 on the raw signal. The method could be improved as above, by focussing on the most prominent frequencies enabling a more effective downsample.

3.2.6 Normalisation

In order to get meaningful results the paths must be normalised, I chose to scale by the maximum absolute value in each dimension to obtain all readings between -1 and 1. I chose this because centralising may have destroyed important features that could have identified harmful activity, such as relative position. Considering the scaling with respect to all dimesnions may retain more features.

4 Model structure

The order of the preprocessing can be done in multiple ways, I forward and backward filled NaNs, as interpolation would have had minimal impact on a fourier transform. I then passed the signal through the montage using the most important nodes, yielding an 8 dimensional new path, followed by a fourier tranform where I filter the signal to frequencies of importance as above (consequently denoising the signal) and downsample. I then inverse transform and slice again. I am left with the denoised downsampled montage with the most relevant relationships between nodes and frequencies.
Now the data is prepared and it is an 8 dimensional path in time a regression model can be implemented. The kernel of two datapoints is defined as the signature kernel, which allows us to create a kernel matrix to perform regression on. The signature kernel matrix is computed using the sigkerax package using jax, and it calculates the kernel by solving the Goursat equation, without signature truncation up to a refinement factor. This is very memory intensive so I implemented a batching function to calculate the kernel matrix in small chunks.

The regression method I chose was ridge regression so I was able to control the regularisation factor to reduce overfitting. After performing regression I applied ReLu to the outcomes to ensure the negative predictions predict a 0 probability, and then scale to ensure the predictions sum to one. The reason softmax was not used is the regression happens ignorant of the later scaling, so we would like negative values to be zero and the positive values to be as close as possible to their model output values.

5 Next steps

After basing the preprocessing order and operations on expert consensus the model is ready for fine tuning, some hyperparameters include σ: a scale for the paths in preparation for computation of the signature kernel, the refinement factor for computing the signature kernel: This defines the granularity of the PDE solver used to calculate the signature kernel, alpha: the parameter that regularises the magnitude of the weights, help prevent overfitting, downsampling constant: this defines how much data to keep and whether to downsample the raw signal or frequency space signal.
In my implementation I trained on 6000 datapoints and created a test set of approximately 4000 datapoints, many choices of hyperparameter and preprocessing techniques lead to a leaderboard score of 0.93 and similar KLDiv scores, and the test scores were marginally higher than the train scores; suggesting my model was overfitting. As discussed above the next steps to improve this are a combination of data sampling, and hyperparameter tuning. Finally running with more memory and faster GPU would have enabled a much larger trainset length as constructing the matrix scales with $O(N^2)$ which proved to be another limitation.
I believe implementing this model style with the addition of convolutional layers to encode and decode, and including multihead attention over neighboring channels would have improved this models efficiecy and accuracy greatly.

6 Sources

Montaging information: https://www.learningeeg.com/montages-and-technical-components Dataset information: https://www.kaggle.com/competitions/hms-harmful-brain-activity-classification/data