Identifying Brain Activity from EEG Recordings

Trenton Chang
Department of Computer Science
Stanford University
tchang97@stanford.edu

Caroline Ho
Department of Computer Science
Stanford University
cho19@stanford.edu

Ray Iyer
Department of Computer Science
Stanford University
rri@stanford.edu

1 Introduction

Electroencephalograms (EEGs) are diagnostic tools used to detect anomalies in brain electrical activity, such as epilepsy and other neurological disorders [1]. Additionally, neuroscientists have used them to find patterns that indicate alertness, movement, blinking, and more.

Our goal is to classify EEG recordings of various types of brain activity, using a variety of methods: softmax regression, k-nearest neighbors, hidden Markov models, and convolutional neural networks. Our motivation is that healthcare providers currently must be trained to process vast amounts of data to correctly diagnose EEGs. Due to the pattern-based, high-volume nature of EEG data, this is a compelling application of machine learning.

2 Related Works

Previous works for EEG classification in the context of epilepsy detection have primarily taken a binary class approach, discriminating epileptic EEGs from the non-epileptic case. Below, we discuss previous research that has informed our feature extraction/model selection.

Regarding models, Wang et al. (2018) used a three-layer convolutional neural network (CNN) with a single convolutional layer, a pooling layer, and a fully connected layer to make binary classifications in the time and frequency domain [2]. There is precedent as well for using hidden Markov models on EEG classification (Williams, Daly, and Nasuto , 2018) [3]. In a different problem space (seizure type classification), Roy et al. (2019) found that a k-nearest neighbors (kNN) classifier on Fourier-transformed data achieved the best results (F1 score of 0.907, better than CNN) [4].

For feature engineering, mapping EEGs to the frequency domain via Fourier transform is common; in addition to Roy et al. [4], Polat and Güneş use the transformed data as input to a decision tree, achieving 0.9872 binary classification accuracy [5].

Another feature extraction method utilized by Zhang, et. al. [12] in classifying imagined left/right hand movement for use in brain-computer interface systems was to calculate the power spectral entropy of the raw EEG signal. This study found promising results, achieving efficient calculation speed and 90% accuracy with a time-variable classifier.

3 Data

The Epileptic Seizure Recognition dataset [6] contains EEG recordings from 500 subjects, each split into 23 one-second-long chunks with 178 observations per second. There are five classes, where classes (1)-(3) are activity from epileptic subjects: (1) epileptic seizure, (2) from a tumor, (3) from a healthy area, (4) eyes closed, and (5) eyes open. The data is also preprocessed to ensure stationarity

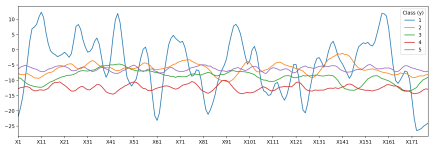

Figure 1: Time vs Mean Value (Aggregated by Class)

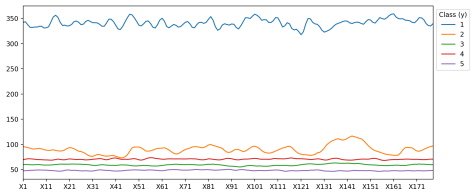

Figure 2: Time vs Standard Deviation (Aggregated by Class)

and removal of artifacts caused by blinking. We further normalize the data by subtracting the mean and dividing by the standard deviation. While past work has achieved high binary classification performance (seizure or not), little has been done in the multiclass case, which requires distinguishing similar non-seizure classes.

In Fig. 1, we see that the epileptic class ($y = 1$) has much higher fluctuation in mean over time than the other classes. In Fig. 2, we observe that the standard deviation is fairly constant over time, yet consistently higher for the epileptic class and slightly stratified among the other classes.

4 Feature extraction methods

Drawing from our data exploration and review of previous research, we use a number of methods to extract features: 1) summary statistics (minimum and standard deviation) over each EEG as a baseline, 2) using raw data, 3) applying the Fourier transform, and 4) calculating spectral entropy. We discuss the Fourier transform and spectral entropy in more depth below.

4.1 Fourier Transform

The Fourier transform transforms data from the time to the frequency domain. Its use was motivated by the hypothesis that neurotypical vs. epileptic EEGs have significant differences in frequency spectra, which we validated by plotting the Fourier transform of averaged data per class. We see that the spectrum for the epileptic class contains more high frequencies (Fig. 3).

4.2 Spectral Entropy

To calculate spectral entropy, we treat the normalized power distribution of the signal in the frequency domain as a probability distribution, then calculate the Shannon entropy of this distribution. Plotting

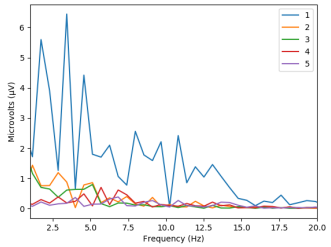

Figure 3: Power spectra obtained via discrete Fourier transform on EEG signals by class average.

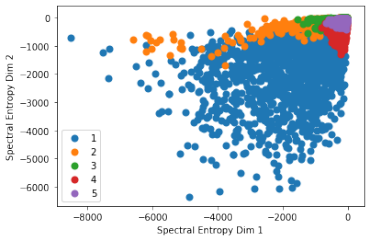

Figure 4: Data separation in 2 dimensions (of 4 total) for spectral entropy of EEG signal.

two of the four dimensions of the extracted spectral entropy, we see a high degree of separability between the five classes (Fig. 4). As expected, the erratic nature of raw EEG signals for the epileptic class ($y = 1$) is reflected in the high spectral entropy.

5 Classification methods

5.1 Softmax Regression

Softmax regression generalizes logistic regression to multiclass problems. Prediction are made by finding the class k that maximizes the softmax function $\phi_k = \frac{\exp(\theta_k^T x)}{\sum_{j=1}^{K} \exp(\theta_j^T x)}$ (which maps to class probability). We use multiclass cross-entropy loss and L_2 regularization in training to optimize θ: $J(\theta) = -(\frac{1}{C}||\theta||_2^2 + \sum_{i=1}^{n} \sum_{j=1}^{K} \mathbf{1}\{y_i = j\} \log \hat{y}_{ij})$. We implement softmax regression with `scikit-learn`'s LogisticRegression classifier [7].

5.2 K-Nearest Neighbors

k-nearest neighbors (kNN) is a supervised learning method which performs classification by calculating the k closest neighbors to a data point and assigning the point to the most common class among those k neighbors. For our distance metric, we use Euclidean distance $d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$. We implement kNN with `scikit-learn`'s kNeighborsClassifier [7].

5.3 Hidden Markov Models

Given that EEGs are time-series data, we hypothesized that we could model EEGs as a Markov model of emissions conditioned on n unobserved component distributions. Hidden Markov Models (HMMs) are probabilistic graphical models used to model noisy, time-distributed data. HMMs in particular assume the existence of n unobservable "states," which correspond to probability distributions that "emit" observations over time. The n states comprise a Markov chain, which can be encoded as a transition probability matrix. This model makes two assumptions:

First, the Markov assumption: $P(X_t \mid X_{0:t-1}) = P(X_t \mid X_{t-1})$, where X_t is the hidden state X at time t, and $0 : t - 1$ denotes a sequence of timesteps 0 through $t - 1$ inclusive. Thus, the hidden state at timestep t depends solely on the previous hidden state. Second, the independence assumption: $P(Y_t \mid X_{0:t}, Y_{1:t-1}) = P(Y_t \mid X_t)$ where Y_t is the emitted variable at timestep t. That is, the observation at time t is solely dependent on the hidden state configuration at time t [8].

To train, we learn a transition matrix of the states, and learn the distributions $P(Y \mid X_k)$ for each hidden state $k \in \{1, ..., n\}$. We assume all distributions are univariate Gaussian, given the emissions are one-dimensional. Formally, we write an HMM $\theta = (\pi, A, B)$, where π is the initial state vector, a categorial distribution over the states, A is the transition matrix, and B is the emission probabilities.

We used a variant of expectation-maximization, the Baum-Welch algorithm, as implemented in the `hmmlearn` Python library for parameter learning and inference [9]. We train a HMM for each class, and then classify by determining which HMM was mostly likely to generate a sequence of observations: $\hat{y} = \arg\max_j \sum_{i=0}^{t} P(Y_t \mid X_t^j) P(X^j)$ where X^j denotes the estimated parameters for the HMM for generating class $j \in \{1, 2, ...5\}$.

Table 1: Results

Features	Model	Train Acc	Acc	Macro F1	1 F1	2 F1	3 F1	4 F1	5 F1
Summary statistics	Softmax reg.	0.465	0.434	0.399	0.873	0.196	0.092	0.398	0.437
	kNN	0.591	0.472	0.466	0.903	0.304	0.291	0.396	0.435
Spectral entropy	Softmax reg.	0.677	0.652	0.629	0.918	0.378	0.507	0.710	0.632
	kNN	**0.768**	**0.681**	**0.653**	**0.919**	**0.478**	**0.517**	**0.697**	**0.653**
Raw data	Softmax reg.	0.303	0.195	0.186	0.323	0.126	0.102	0.219	0.163
	kNN	0.655	0.423	0.421	0.799	0.375	0.391	0.411	0.130
	HMM ($n = 1$)	0.432	0.434	0.420	0.873	0.193	0.232	0.295	0.507
	CNN	**0.927**	**0.780**	**0.773**	**0.974**	**0.705**	**0.622**	**0.775**	**0.789**
Fourier transform	Softmax reg.	0.284	0.187	0.177	0.295	0.135	0.081	0.224	0.152
	kNN	0.656	0.443	0.448	0.805	0.386	0.407	0.430	0.210
	CNN	**0.959**	**0.740**	**0.737**	**0.941**	**0.547**	**0.625**	**0.799**	**0.770**

5.4 Convolutional Neural Networks

A convolutional neural network (CNN) is a neural network with layers that apply sliding fixed-length filters along an input [10], which we hypothesize is useful for sequential data due to its shift invariance. We use multiclass cross-entropy loss to learn the filter parameters and use the following architecture for our CNN (implemented with Keras [11]):

- Two 1D convolutional layers with 64 filters (ReLU activation)
- 1D max pooling layer
- Two 1D convolutional layers with 128 filters (ReLU activation)
- 1D global average pooling layer
- Dropout layer (for regularization)
- Dense layer (softmax activation)

6 Evaluation methods

We evaluate our models using accuracy, macro F1, and per-class F1 scores. Due to the models' differing complexity and hyperparameter searches, we perform evaluation a bit differently on each. While this means results are not directly comparable, we still get a general idea of model performance.

For softmax regression and kNN, we use nested cross validation to optimize hyperparameters (with 10 inner cross-validation folds) and get an unbiased performance estimate (with 10 outer cross-validation folds), dividing data so all of a subject's EEG chunks are in the same fold. For softmax regression, we perform a hyperparameter search over the inverse of regularization strength $C \in \{0.1, 0.5, 1, 5\}$. For kNN, we perform a search over $k \in \{3, 5, 7, 9\}$.

For HMM, which only involves searching over the number of hidden distributions, we use a simple 70/30 train/test split.

For CNN, nested cross-validation is too costly, so we use an 80/20 train/test split and use 10-fold cross-validation on the train split to select hyperparameters. We perform a search over dropout value ($\{0.1, 0.2, 0.5\}$) and sliding window length ($\{3, 7, 11\}$ for Fourier and $\{3, 11, 15, 19\}$ for raw due to different sequence lengths). We train for 25 epochs on a batch size of 32.

7 Results and Discussion

Results are in Table 1. The top 3 models (raw data CNN, Fourier transform CNN, and spectral entropy kNN) are in bold. Examining the results and confusion matrices (Figure 5), we see that these models perform very well on class 1 (epileptic seizure), achieving class F1 scores of above 0.9. Most errors come from within the "super-classes" 2 and 3 (non-seizure activity from epileptic subject) and 4 and 5 (activity from non-epileptic subject).

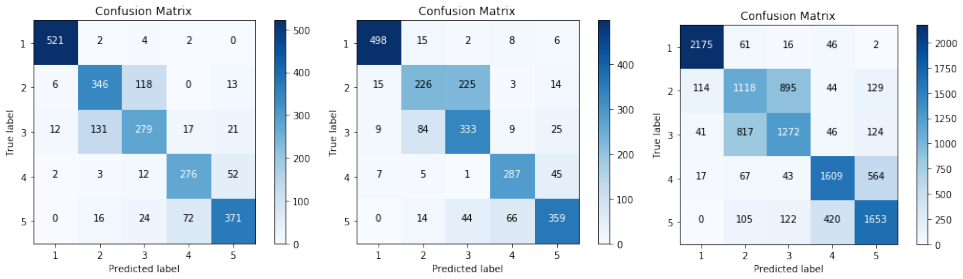

Figure 5: Confusion matrices for raw data CNN, Fourier transform CNN, and spectral entropy kNN.

7.1 Softmax Regression

Softmax regression performs very poorly (roughly equivalent to random guessing) on sequential data (raw data and Fourier transform features) and even performs poorly for class 1, which is likely because it sets a fixed weight for a given time t for an EEG segment and does not take into account the pattern-based, repetitive nature of the data. However, it performs better on spectral entropy and summary statistics, likely because of the separability of the classes in those feature spaces.

7.2 K-Nearest Neighbors

In general, kNN performs somewhat well (or at least better than softmax regression), achieving accuracies and macro F1 scores of at least 0.4. Spectral entropy kNN performs especially well compared to the other baseline models, achieving 0.681 accuracy and a 0.919 class 1 F1 score.

7.3 Hidden Markov Models

HMMs perform poorly. The best performing HMM had a single hidden state; adding more states to led to decreases in accuracy on the order of single-digit percentages. It is likely that the low performance of HMMs is due to erroneous modeling assumptions. More domain knowledge would be required to validate/invalidate our assumptions for this model.

7.4 Convolutional Neural Networks

From cross-validation, we found that the best hyperparameters for raw data are a window of 15 and dropout of 0.1, and those for Fourier tranform are a window of 11 and dropout of 0.2. CNNs perform very well, as the only models to achieve over 0.7 accuracy and above 0.5 F1 on all classes; we attribute their success to their shift invariance, which captures recurring patterns in sequences. The raw data CNN performs best, achieving an accuracy of 0.780 and a class 1 F1 score of 0.974. However, both CNNs overfit, suggesting our models are too complex and/or have trained too long.

8 Conclusions and Future Work

Of all our models, the CNN model and the spectral entropy kNN perform best, achieving high accuracies overall (considering most of the errors are from confusing very similar classes) and high scores on seizure classification. Given these results, there are a few steps we can take in the future. First, given that CNN overfits, we can refine the architecture by applying regularization techniques like weight decay (L1/L2/ElasticNet regularization) or by expanding our hyperparameter search. Secondly, given the success of spectral entropy, we can solicit domain knowledge to improve our feature engineering to see if there exist more suitable signal processing techniques that could improve the separability of our data in the feature space. Finally, we can experiment with other time-distributed architectures such as RNNs.

9 Contributions

Caroline performed softmax regression, kNN, and the CNN experiments. Trenton worked on data visualization and preprocessing for Fourier transform, and ran the HMM experiment. Ray worked on additional visualizations (aggregating and visualizing data per-class) and feature engineering

to maximize data separability. Our code is on GitHub at https://github.com/carolineh101/cs229-epilepsy.

References

[1] Electroencephalogram. (n.d.). Johns Hopkins Medicine. https://www.hopkinsmedicine.org/health/treatment-tests-and-therapies/electroencephalogram-eeg

[2] Zhou M, Tian C, Cao R, Wang B, Niu Y, Hu T, Guo H and Xiang J (2018). Epileptic Seizure Detection Based on EEG Signals and CNN. Front. Neuroinform. 12:95. doi: 10.3389/fninf.2018.00095

[3] Williams NJ, Daly I and Nasuto SJ (2018). Markov Model-Based Method to Analyse Time-Varying Networks in EEG Task-Related Data. Front. Comput. Neurosci. 12:76. doi: 10.3389/fncom.2018.00076

[4] Roy S, Asif U, Tang J, and Harrer S (2019). Machine Learning for Seizure Type Classification: Setting the benchmark. arXiv:1902.01012

[5] Polat K, Güneş S (2007). Classification of epileptiform EEG using a hybrid system based on decision tree classifier and fast Fourier transform. Applied Mathematics and Computation. 187:2. doi: 10.1016/j.amc.2006.09.022

[6] Andrzejak RG, Lehnertz K, Rieke C, Mormann F, David P, Elger CE (2001). Indications of nonlinear deterministic and finite dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state, Phys. Rev. E, 64, 061907

[7] Pedregosa et al. (2011). Scikit-learn: Machine Learning in Python, JMLR 12, pp. 2825-2830.

[8] Koller D, Friedman N (2009). Probabilistic graphic models. MIT Press (Cambridge, MA).

[9] Tu S (n.d.). Derivation of Baum-Welch Algorithm for Hidden Markov Models. https://people.eecs.berkeley.edu/~stephentu/writeups/hmm-baum-welch-derivation.pdf

[10] Kiranyaz S, Ince T, Abdeljaber O, Avci O, Gabbouj M (2019). 1-D Convolutional Neural Networks for Signal Processing Applications. 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

[11] Chollet, François and others (2015). Keras. https://keras.io

[12] Zhang A, Yang B, Huang L et. al. (2008). Feature Extraction of EEG Signals Using Power Spectral Entropy. 2008 International Conference on BioMedical Engineering and Informatics.