A Comparison of Machine Learning Algorithms in Hand Movement Classification from EEG Data

Jason Nyam Ngollong & Tim Kalnins

9 May 2021

Literature Review & Objectives

Classification problems involving electroencephalography (EEG) signals have been garning the interest of researchers for some time. To quantify this interest Craik, He, and Contreras-Vidal performed a systematic literature review of papers involving EEG classification and deep learning. Their findings, following an extensive search of the Web of Science and PubMed databases, discovered that between the years 2015 and 2018 more than 90 papers have been published on the application of deep learning to EEG classification tasks [1].

Our paper however will not be focusing on deep learning, instead we seek to emulate two other papers by providing a comparitive analysis of the performance that different machine learning algorithms have when applied to an EEG classification problem. The two papers in question are *Bhattacharyya & Khasnobish (2010)* and *Höller & Bergmann (2013)*. In *Höller & Bergmann (2013)* linear discriminant analysis (LDA), k-nearest neighbor classification (KNC), and support vector machines (SVM) are compared while attempting to classify patients with normal brain function from patients with disorders of consciousness (individuals with brain damage) [2]. In *Bhattacharyya & Khasnobish (2010)* quadratic discriminant analysis (QDA), LDA, and KNC are compared while attempting to classify hand movements (left or right) from EEG signals generated from participants engaged in motor imagery [3]. As a point of clarification, motor imagery refers to the mental process of visualizing or imagining the performance of a certain physical action. Imagining oneself picking up a ball or pushing a button with a specific finger are simple examples.

Our analysis works with EEG data of a similar vein, in that we will be comparing the effectiveness of various algorithms as they attempt to classify hand movements from EEG signals generated by participants engaged in motor imagery. The algorithms of interest are as follows: LDA, QDA, KNC, SVM, and a simple fully connected neural network (FCNN) composed of two hidden layers.

What is Electroencephalography & Why Should We Care?

Electroencephalography (EEG) is a method of observing electical brain activity via the minute electrical current present on the surface of the scalp. These electromagnetic signals are created when populations of neurons are active in conjunction with one another [4]. Electrodes are placed on the surface of the scalp to record these signals as waves of varying shape, frequency, and amplitude. These waves are categorized into five types of differing frequencies: delta (0.5-3 Hz), theta (4-7 Hz), alpha (8-12 Hz), beta (15- 30 Hz), and gamma (>30 Hz) [5]. Two frequent uses of EEG is to measure how brain activity changes in response to some event/stimulus, or to measure spontaneous brain activity in the abscence of an aforementioned

[1] Craik, He, and Contreras-Vidal 2019: Deep learning for eeg classification tasks: a review

[2] Höller & Bergmann 2013: Comparison of EEG-Features and Classification Methodsfor Motor Imagery in Patients with Disorders of Consciousness

[3] Bhattacharyya & Khasnobish 2010: Performance Analysis of LDA, QDA and KNN Algorithms in Left-Right limb movement classification from EEG data

[4] Kirschstein, Köhling 2009: What is the source of the EEG?

[5] Wikipedia: Alpha Wave

event/stimulus. Clinical applications of EEG include: to diagnose sleeping disorders, epilepsi, and other potential brain dysfunction[6].

However it has been speculated that the signals recorded by EEG could be used as some form of communication channel. Which in turn hints at the possiblity of an interface between the brain and some other medium. In the literature this concept is referred to as a Brain-Computer Interface (BCI), which refers to "a system that translates the brain activity patterns of a user into messages or commands for an interactive application" [7]. BCIs have a huge number of use cases: from aiding the disabled or physically impaired, to aiding in rehabiliatory efforts in treating patients suffering from brain damage, or to acting as a gaming interface [8]. An important part of creating an effective BCI is the component of being able to properly classify brain activity corresponding to the correct stimulus that generated the afforementioned signal [9]. Hence the importance of analyzing the performance of various classification algorithms within the domain of EEG data.

Data Origins & Structure

The EEG data used for analysis was sourced from a Brazilian study of EEG signals and hand movements. The study recorded the EEG signals of four participants to stimuli indicating various hand movements: motor action of the right hand, motor action of the left hand, and no motor action of either hand. Of the four paritcipants, three were shown images depicting the afforementioned motor actions: a left facing arrow indicating movement of the left hand, a right facing arrow indicating movement of the right hand, and a circle indicating no movement. The fourth participant differs only in that they were instructed to keep their eyes closed until hearing an audible beep, whereupon the participant was allowed to open their eyes to view the same visual stimulus as the other three paritcipants. Upon viewing this visual stimulus, each participant was asked to imagine or visualize themselves performing the specified action.

The EEG signals were recored using a 14 channel EMOTIV electroencephalogram, with electrode channels placed on the following channel locations: AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8 and AF4 (indicated by purple circles on Figure 1). Each channel used a sampling frequency of 128 Hz at each reading. When accounting for all 14 channels, the provided sample would be of the form of a 16-bit 128x14 matrix. This sample would then be passed through a fast Fourier transform, cutting the frequency range to be between 0 and 30 Hz (128x14 matrix to 30x14 matrix): in turn reducing the EEG signals to be of only delta, theta, alpha, and beta waves. After the fast fourier transform, and the resulting binning of frequencies into delta/theta/alpha/beta categories, the mean and standard deviation of each wave was computed. Creating the following number of features for a single observation:

$$\#\{channels\} \times \#\{waves\} \times \#\{\mu, \sigma\} = 14 \times 4 \times 2 = 112 \ features$$

Each participant was administerd a hand movement stimulus 2880 times over the course of the study, resulting in 11520 total observations. We also generated a one-hot encoded feature, specifying the user corresponding to a given observation. All of which results in a data structure of 11520 observations and 116 features. The target, the given hand movement, is a 1-dimensional array of 11520 entries encoded using the following schema:

$$\text{left hand movement} \longrightarrow 0 \quad \text{right hand movement} \longrightarrow 1 \quad \text{no hand movement} \longrightarrow 2$$

The data was provided by Fabricio Torquato, who made it publicly available on Kaggle [10].

[6] Mayo Clinic: EEG (electroencephalogram)
[7] Fabien Lotte, Laurent Bougrain, Maureen Clerc 2015: EEG-based Brain-ComputerInterfaces
[8] Fabien Lotte, Laurent Bougrain, Maureen Clerc 2015: EEG-based Brain-ComputerInterfaces
[9] Fabien Lotte, Laurent Bougrain, Maureen Clerc 2015: EEG-based Brain-ComputerInterfaces
[10] Torquator: EEG data from hands movement (Kaggle)

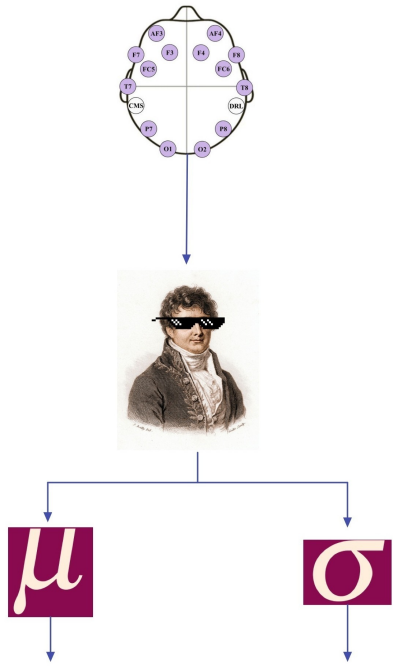

Figure 1: Graphical Representation of Data Processing

Models to Be Evaluated

As mentioned earlier, in our study we will analyze the performance of linear discriminant analysis (LDA), quadratic discriminant analysis (QDA) and k-nearest neighbor classifier (KNC), support vector machine (SVM) and fully connected neural network (FCNN) with regards to the task of hand movement classification. For LDA, QDA, KNC, and SVM the implementations provided by `scikit-learn` were used. While our FCNN was built using `TensorFlow` and `Keras`.

Some preliminaries regarding training. Data was randomized and split into test and training sets. In the case of LDA, QDA, KNC, and SVM a broad parameter grid search was performed to tune various hyperparameters. For each parameter grid search, 5-fold cross-validation was used via `sklearn.model_selection.GridSearchCV`. For our FCNN, the original training set was randomly split into a secondary training and validation sets. Following which, we used the two newly created sets for training and validation purposes.

A small note on the provided plots. Our hyperparameter tuning was performed on multiple model parameters, not simply the single parameter varying in the plots. Each plot uses the optimal set of model parameters, with a single numerical parameter allowed vary.

LDA and QDA

We trained and validated an LDA and QDA algorithm to our data and plot the results. These algorithms are not performing well. As we can see, less than 50% of validation and training accuracy for the LDA and less than 60% for the QDA. After cross validation of the parameters the best validation accuracy was: 45.23% for LDA and 49.76% for QDA. Potential reasons for this poor performance could be that our three classes might not have Gaussian densities. As that can be a potential cause of poor performance for both algorithms. With that being said, this is a form of educated speculation on our part.

The following plots are the training and validation accuracies versus the shrinkage (note that the shrinkage is a log scale) and regularization of the covariance matrix parameters.

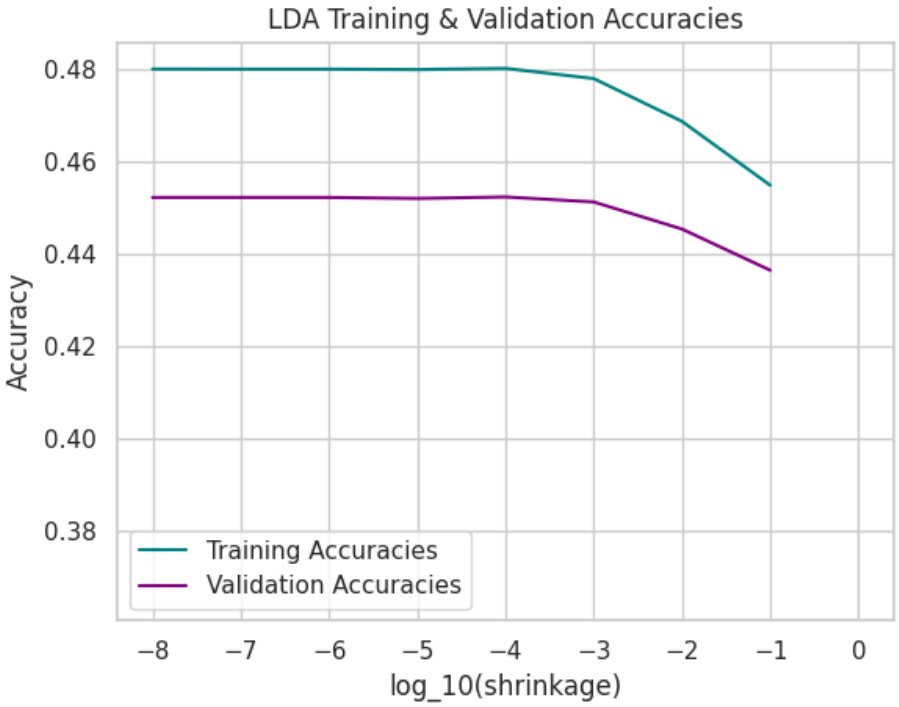

Figure 2: LDA Train vs Val

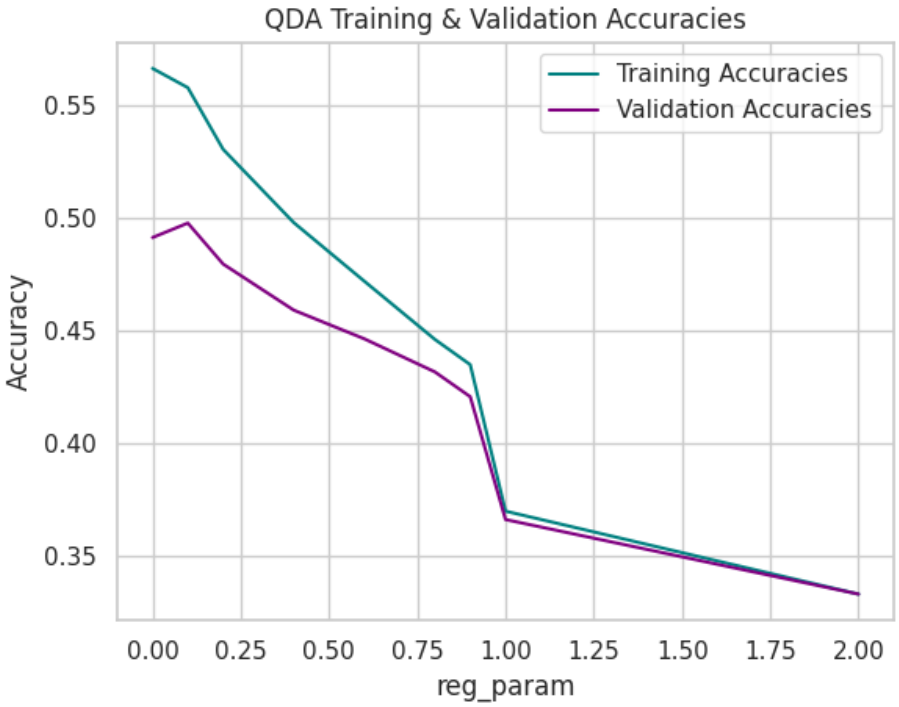

Figure 3: QDA Train vs Val

KNN

We trained and validated the KNN on our design matrix. We tried using different parameters such as the number of neighbors and type of norm used for the distance. We also looked at the performance when the points are weighted by the inverse of their distance in the decision function. Our best result is a weighted function with 3 neighbors using the L1 norm. We have a perfect training accuracy and our best validation accuracy is 80.69%

Here is a plot of the accuracy as the number of neihbors vary.

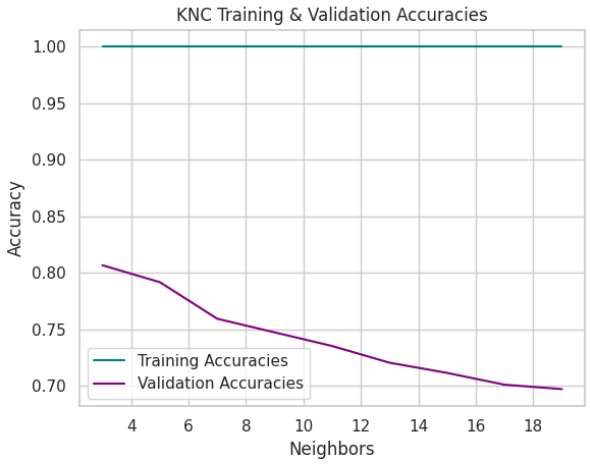

Figure 4: KNC Train vs Val

SVM

The Support Vector Machine on the design matrix with a Radial Basis Function kernel (RBF). We cross validate the shrinkage parameter C and the gamma of the RBF, and found that our best model utilized a C value of 10 and a gamma value of 0.1. Our best results were about 71.17% on validation accuracy.

Here is a plot of the accuracy versus the shrinkage C (again shrinkage is on a log scale).

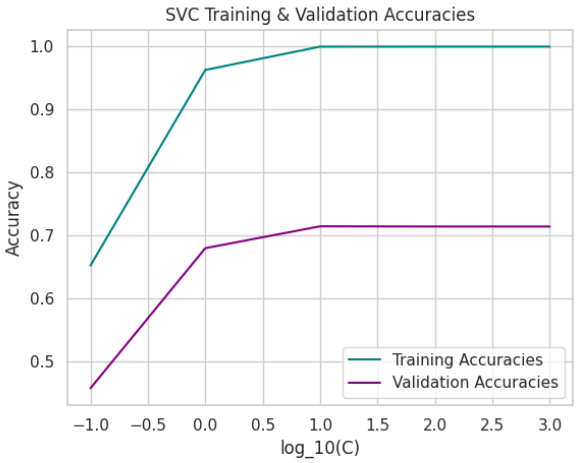

Figure 5: SVC Train vs Val

FCNN

Using `Keras` and `TensorFlow` we created a fully connected neural network with 2 hidden layers (with 150 nodes at each layer). Each of the hidden layers has a Relu activation function. Through trial and error we found improved performance through the use of a uniform kernel initializer when defining our hidden layers. A uniform kernel initializer is simply the use of a uniform distribution to set the values of the initial weights of a given layer. The last layer has a softmax function. We train and validate on 40 epochs and got a final validation accuracy of 71.53%

Here is the training and validation accuracy versus the number of epochs.

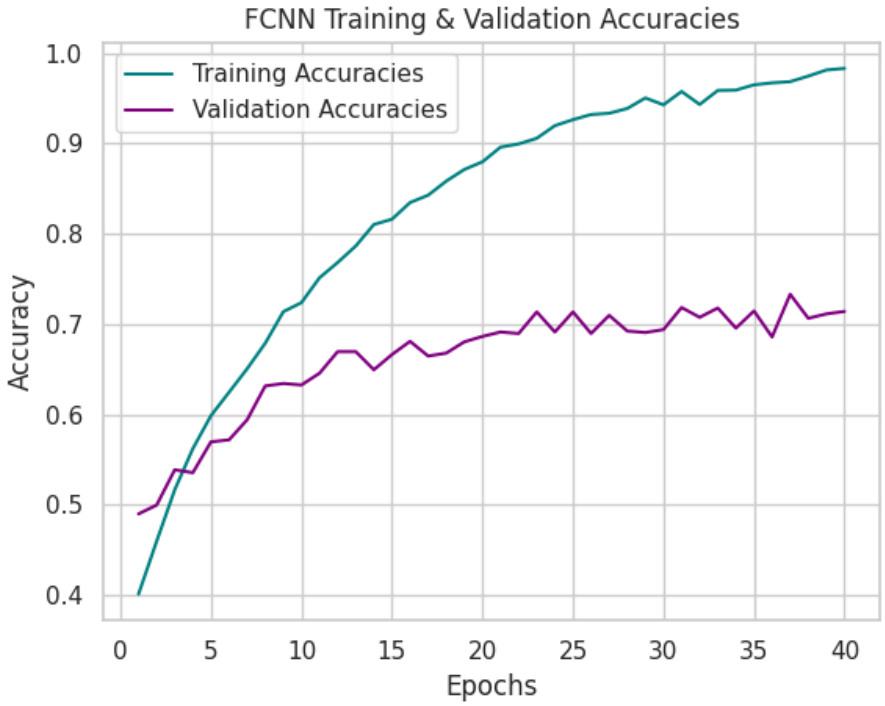

Figure 6: FCNN Train vs Val

Results

After completing the hyperparameter tuning to determine our best models, we were finally ready to evaluate our models using the test data. We would like to stress that we followed proper protocols, in the sense that these evaluations on the test data were run only once at the end, and that the test and training data were separated during all model training and tuning. The results are as follows: LDA and QDA performed the worst with test accuracies of 47.14% and 48.61% respectively. SVM and our FCNN performed better with test accuracies of 73.47% and 70.14%. Which leaves the best performing model, KNC, with a test accuracy of 84.07%. These results are presented on the next page as a table and a bar plot.

Algorithm	Test accuracy
LDA	47.14%
QDA	48.61%
KNC	84.07%
SVM	73.47%
FCNN	70.14%

Figure 7: Test Accuracies

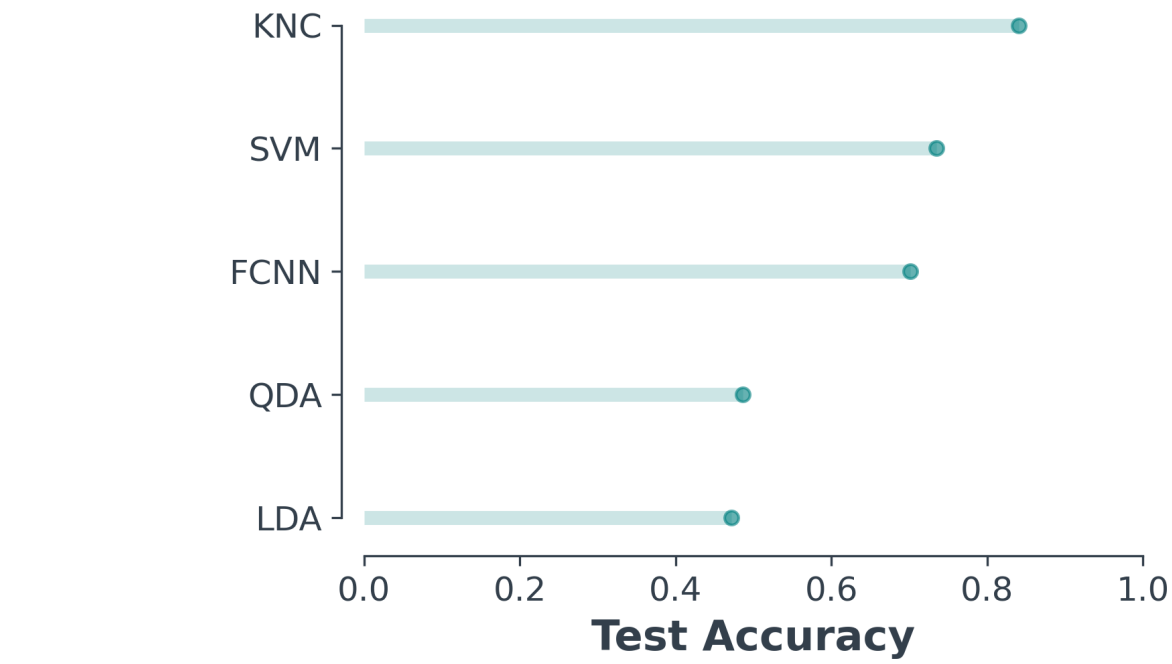

Figure 8: Test Accuracies

Conclusions & Criticisms

What have we learned through the completion of this project? First and foremost, classification problems involving EEG signals are becomming a topic of interst in the field of applied machine learning [11]. And that even for green practitioners like ourselves, interesting and fairly effective results can be obtained. Reviewing our results and comparing them to the findings of *Bhattacharyya & Khasnobish (2010)*, we find an intersting similarity. In one of their attempts to classify left and right hand movements, Bhattacharyya & Khasnobish first extracted features from raw EEG data using a method called average bandpower estimation. While not identical to the form of averaging used in our data, there is similarity in the sense that each utilize some form of sample averaging to reduce an EEG signal to a single data point. With those specifics in mind, Bhattacharyya & Khasnobish found that a k-nearest neighbor algorithm performed the best (compared against LDA and QDA) in classifying left and right hand movements with an accuracy of 84.29% [12]. A result very similar to our own, which was a k-nearest neighbor algorithm with a test accuracy of 84.07%.

With regards to self critique, the preprocessed dimensional reduction of the EEG data used in our project was somewhat self-limiting. Having more dimensions could have proved interesting. As it would have permitted us to be more creative with our selection of algorithms, and open the door for us to try our hand at feature extraction and signal processing. Another critique would be that the number of observations, 11520, was itself a serious limiting factor in our choices of algorithms to compare. Which is a valid concern, as deep learning methods would certainly benefit from a more expansive dataset to train upon. Or looking at the issue from another angle, the use of a transfer learner as another model to compare against would have made for an interesting option as well. With that said, we are still satisfied with our efforts and results.

Project's GitHub Page

https://www.github.com/ka1a1t/eeg_cs289a_proj

[11] Craik, He, and Contreras-Vidal 2019: Deep learning for eeg classification tasks: a review

[12] Bhattacharyya & Khasnobish 2010: Performance Analysis of LDA, QDA and KNN Algorithms in Left-Right limb movement classification from EEG data