Reading EEG Signals to Determine Hand Motion

Intro to Machine Learning

CS-UY 4563

Zonayed Rahman

Elijah Whittle

Introduction

In the final project, we used a dataset from Kaggle that contained the brainwaves of participants who are performing activities in order to determine whether they are moving their left arm, right arm, or neither. The deeper question this poses is whether we can successfully use machine learning techniques from this course to understand simple brain signal data. The dataset contains 113 columns (including the target column, which indicates the user's intended movement). The original columns consists of the mean and standard deviation of reading alpha, beta, delta, and theta waves from the following 14 electrode channels: AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8 and AF4. This is a multi-class classification problem, with the goal of predicting what movement one is making based on the hand-movement data.The movement is encoded as 0 for left arm movement, 1 for right arm movement, and 2 for no movement. In this project, we experiment with three different classification models: logistic, support vector machines (SVM), and neural networks.

Data Preparation

The 4 raw datasets included no incomplete features with null entries. We decided to train the 4 datasets separately, as it gives a brighter insight into how different individuals' neural signals differ. The only categorical column was "Class," which is the target movement, with three categories ranging from 0 to 2. To ensure none of the categories heavily weighted in, their occurrence was analyzed. In all the tested models, the features were scaled using the StandardScaler package from sklearn.preprocessing, which zero-centered the data and scaled

data points by calculating the z-score of that point and transforming the data in each feature to normally distributed data. This allowed for a reduction of bias within the data. The data was split into a training set, a validation set, and a testing set. For logistic regression and SVMs, the training set was 80% of the original set, while the test set was both 20% of the original set. For neural networks, the training set was 60% of the original set, while the test set was 40% of the original set. There were also versions of the training set prepared only providing the mean values or the standard deviation values for training with the neural network.

Analysis

Logistic Regression

The first model used for movement prediction with this data set is logistic regression. The outline for acquiring the best hyperparameters was as follows:

1. Create polynomial transforms of the feature matrix of different degrees with no regularization.
2. Using the best feature transformation from the previous step, experiment with both L1 regularization and L2 regularization. The optimal regularization strength hyperparameter for L1 and L2 regularization shall be found with K-fold cross validation.

In the first two models, Sklearn's PolynomialFeatures was utilized, as its "degree" parameter allowed for easy transformation of the features. Thus, it allowed for comparison between degrees 1 and 2. Since the original feature matrix already begins with 112 features, any number of degrees higher resulted in too many features, resulting in too much time being consumed during training. Therefore, only the first two transformations had to suffice. This

resulted in a logistic regression model with polynomial transformation degree 1 and no regularization in an average of the 4 training accuracies of 64.68% and average testing accuracy of 57.86%. Logistic regression model with polynomial transformation degree 2 and no regularization resulted in perfect accuracy for all 4 datasets and average testing accuracy of 82.73%.

Models of Polynomial Degree 1 and 2

Datasets	**Degree = 1**	**Degree = 2**
User A	0.63021	0.85938
User B	0.63715	0.89063
User C	0.50868	0.71181
User D	0.53819	0.84722

Datasets	**Degree = 1**		**Degree = 2**	
	Precision (0,1,2)	Recall (0,1 2)	Precision (0,1,2)	Recall (0,1 2)
User A	0.71 0.61 0.57	0.68 0.54 0.67	.87 .84 .86	.89 .81 .88
User B	0.64 0.63 0.63	0.64 0.68 0.60	.95 .86 .87	.87 .91 .89
User C	0.54 0.52 0.46	.53 .55 .45	.73 .72 .68	.7 .74 .7
User D	.61 .5 .5	.57 .49 .55	.89 .84 .8	.84 .86 .84

There was considerable increase in accuracy with the second model, almost over 20% across the board. Therefore, regularization was applied to this model to further improve the accuracy.

Regularization with K-fold validation

As stated, both L1 and L2 regularization were performed on the model with 2nd degree polynomial transformation. K-fold validation where K ranges from 2 to 10 was utilized. C values ranging from 0.001 to 10000 were tested on each fold and the best validation and C values were recorded of each dataset.

L1 Regularization

	User_a		User_b		User_c		User_d	
K	**C**	**Score**	**C**	**Score**	**C**	**Score**	**C**	**Score**
2	0.1	0.80972	100	0.84236	10	0.63333	1000	0.77222
3	10	0.85	100	0.87708	1000	0.68333	1	0.80208
4	10000	0.8625	1	0.88888	0.1	0.69444	100	0.82638
5	1	0.87673	1	0.91145	1000	0.72569	10	0.85416
6	1	0.8875	10	0.93333	1	0.74375	1	0.86875
7	100	0.90776	10	0.92214	1000	0.73058	10	0.85922
8	1	0.91111	10	0.93333	0.01	0.74722	1000	0.86667
9	100	0.90937	1	0.9375	0.1	0.76875	1	0.89063
10	1	0.89930	10	0.9375	10000	0.75347	100	0.89236

L2 Regularization

	User_a		User_b		User_c		User_d	
K	**C**	**Score**	**C**	**Score**	**C**	**Score**	**C**	**Score**
2	0.01	0.81389	0.01	0.86597	1	0.65	0.01	0.79930
3	0.01	0.85416	0.1	0.90313	0.1	0.70520	1	0.82917
4	0.1	0.86805	0.01	0.90556	1	0.71528	0.01	0.83611
5	1	0.88368	0.1	0.91146	10	0.74306	1	0.87152
6	0.1	0.8875	1	0.93542	0.01	0.74583	0.1	0.87292
7	0.1	0.90754	0.1	0.92700	0.01	0.74029	1	0.88107
8	100	0.90833	0.01	0.94166	0.1	0.76667	0.1	0.88611
9	0.01	0.91563	100	0.93438	0.1	0.75313	0.1	0.8875
10	1	0.90278	0.01	0.93056	10000	0.76736	10	0.89583

The best scores were:

	L1	**L2**
User A	0.91111	0.91563
User B	0.93333	0.94166
User C	0.75347	0.76667
User D	0.89236	0.89583

Conclusion

Based on the results presented in the L1 and L2 regularization tables, the best logistic regression model for the datasets is the model with 2nd degree polynomial feature transformation with L2 regularization. A large increase was observed once the degree was increased from 1 to 2, which suggests the data is not very linearly separable. Both L1 and L2 regularization also showed an improvement on the 2nd degree polynomial transformation with no regularization of almost 4-5% across all 4 dataset. However, the L2 model did prove to be superior in the K-fold results. This suggests no regularization model was overfitting the data, which was unexpected from preprocessing analysis.

SVM

With the Scikit-Learn python library, linear kernel learning models with L1 and L2 normalization and polynomial kernels with degrees ranging from 2 to 10 were developed to attempt to accomplish a higher accuracy score. Linear kernels of L1 and L2 normalization models were implemented with various C values ranging from 0.001 and 1000000.

Linear Kernels

	L1 Norm		L2 Norm	
	C	**Score**	**C**	**Score**
User A: Train	1	0.68099	0.1	0.67014
User A: Test	10	0.64583	0.1	0.64063
User B: Train	100	0.71224	1	0.71181
User B: Test	0.01	0.64410	0.01	0.64757
User C: Train	100	0.59071	0.1	0.58681
User C: Test	1	0.50868	0.1	0.51042
User D: Train	100	0.60807	0.1	0.60547
User D: Test	0.1	0.55729	0.001	0.55208

For some of the L2 norm results, C values around 1 and above resulted in the model failing to converge, even when max_iter(max iterations for gradient) was set 20 times the default value and all the features had been scaled. Thus, it was concluded the models with such high C value were just non-optimal.

SVM with a polynomial kernel of degrees 1 to 10 were also tested on the dataset. The models with degree 2 and 3 performed best and larger degrees continually performed worse, which usually results due to overfitting. Lower degrees performing better can also show a strong correlation between the features and target.

User A:

Degree	Training	Test
1	0.63454	0.58854
2	0.84418	0.73438
3	0.84983	0.72049
4	0.75477	0.61979
5	0.65972	0.56597
6	0.66970	0.57465
7	0.58116	0.49132
8	0.53472	0.47049
9	0.51606	0.44097
10	0.50477	0.42708

User B:

Degree	Training	Test
1	0.69618	0.64930
2	0.93837	0.82986
3	0.96658	0.86458
4	0.81944	0.66319
5	0.6875	0.56424
6	0.65104	0.53472
7	0.61979	0.51389
8	0.60199	0.48264
9	0.58159	0.46701
10	0.57204	0.44444

User C:

Degree	Training	Test
1	0.57421	0.51562
2	0.76909	0.61111
3	0.82204	0.61631
4	0.64539	0.48090
5	0.60763	0.42361
6	0.55989	0.38888
7	0.54383	0.36979
8	0.53211	0.36631
9	0.52907	0.35069
10	0.52517	0.35763

User D:

Degree	Training	Test
1	0.59635	0.54513
2	0.89583	0.76736
3	0.89149	0.76736
4	0.69487	0.54513
5	0.62586	0.47222
6	0.60329	0.45138
7	0.61328	0.44270
8	0.65321	0.47048
9	0.58420	0.41666
10	0.54644	0.39930

Best performing polynomial kernel models' degrees and score:

Datasets	Training		Test	
Users	Degree	Score	Degree	Score
User A	3	0.84983	2	0.73438
User B	3	0.96658	3	0.86458
User C	3	0.82204	3	0.61631
User D	2	0.89583	2	0.76736

As can be seen from each user's respective best performing polynomial kernel models, the degrees vary between 2 and 3. The difference between the two degrees amongst each user is also significant by almost 5% difference in accuracy. K-fold validation with K=10 was implemented with degrees varying from 2 to 10. Polynomial kernel of a degree of 3 almost dominated all the folds for all 4 datasets. It also reaffirms our idea of the data not being very linearly separable from logistic regression analysis.

Best Results of K-fold validation with Degrees 2 to 10

K	User A		User B		User C		User D	
	Degree	Score	Degree	Score	Degree	Score	Degree	Score
2	2	0.72152	3	0.79722	3	0.56388	3	0.72291
3	2	0.75625	3	0.87187	3	0.62083	3	0.76979
4	2	0.775	3	0.88055	3	0.62222	3	0.79444
5	2	0.77951	3	0.90104	3	0.64583	3	0.80902
6	3	0.79583	3	0.90416	3	0.65833	3	0.8125
7	2	0.79318	3	0.90510	3	0.64563	3	0.82281
8	3	0.8	3	0.90833	3	0.65277	3	0.82777
9	3	0.80625	3	0.90625	3	0.675	3	0.83125
10	3	0.80208	3	0.90972	3	0.67361	3	0.83333

The accuracy score also increased as the fold numbers increased, which is expected. As fold numbers increase, the training set also increases, resulting in better fitting and risking overfitting. However for us, this seemed to prove useful.

Conclusion

The effectiveness of L1 and L2 regularization is almost negligible on the linear kernel SVM. Furthermore, it proved to be very inefficient in capturing the arm movements. The highest accuracies of the polynomial kernel SVM models were observed with degrees 2 and 3, however, even their best results were incomparable to the polynomial logistic regression models. The best results for each user were observed at the highest fold where K=10. A substantial improvement

amongst scores was also observed through the K-fold validation. And it proved the 3rd degree model to be superior to any other polynomial kernel model, but it is still yet to beat logistic regression highest scores.

Neural Network

The neural network structures trained used 1 hidden layer of 112 nodes, 3 hidden layers of 112 nodes, or 4 hidden layers of nodes having the following structure: (112, 56, 30, 10). We trained models using the logistic function with L2 regularization and the tanh function with L2 regularization. We used alpha values ranging from 0.001 to 1.0, and lambda values ranging from 0.00001 to 1000.

Here are the top testing accuracies for each user and neural network structure for a training size of 60%, and the most accurate models' structures, alpha values, and lambda values:

Top testing accuracy information for Users A-D

	User A (logistic)	User A (tanh)	User B (logistic)	User B (tanh)	User C (logistic)	User C (tanh)	User D (logistic)	User D (tanh)
Test accuracy	85.1563%	86.8056%	88.6285%	91.4063%	70.4861%	67.1007%	81.3368%	81.5972%
Precision % (0, 1, 2)	83.8, 84.3, 84.0	87.1, 86.3, 84.2	90.5, 85.1, 88.9	84.9, 79.9, 80.1	70.0, 70.0, 61.7	68.2, 66.1, 65.1	83.7, 81.7, 82.5	81.6, 80.4, 79.2
Recall % (0, 1, 2)	86.7, 80.9, 84.4	86.2, 83.3, 88.3	88.6, 88.5, 87.3	85.8, 77.4, 81.7	68.4, 72.6, 60.7	71.0, 68.7, 59.8	82.4, 83.4, 82.2	82.7, 82.4, 76.2
Confusion Matrix	332 28 23 32 305 40 32 29 331	332 32 21 27 333 40 22 21 324	334 27 16 17 337 27 18 32 344	337 27 29 36 291 49 24 46 313	268 46 78 46 284 61 69 76 224	274 53 59 56 263 64 72 82 229	318 36 38 27 318 29 35 35 316	305 22 42 33 324 36 36 57 297
Alpha	1.0	0.1	0.1	1.0	1.0	0.001	1.0	0.01
Lambda	0.1	1.0	0.1	0.1	0.1	1.0	0.1	1.0

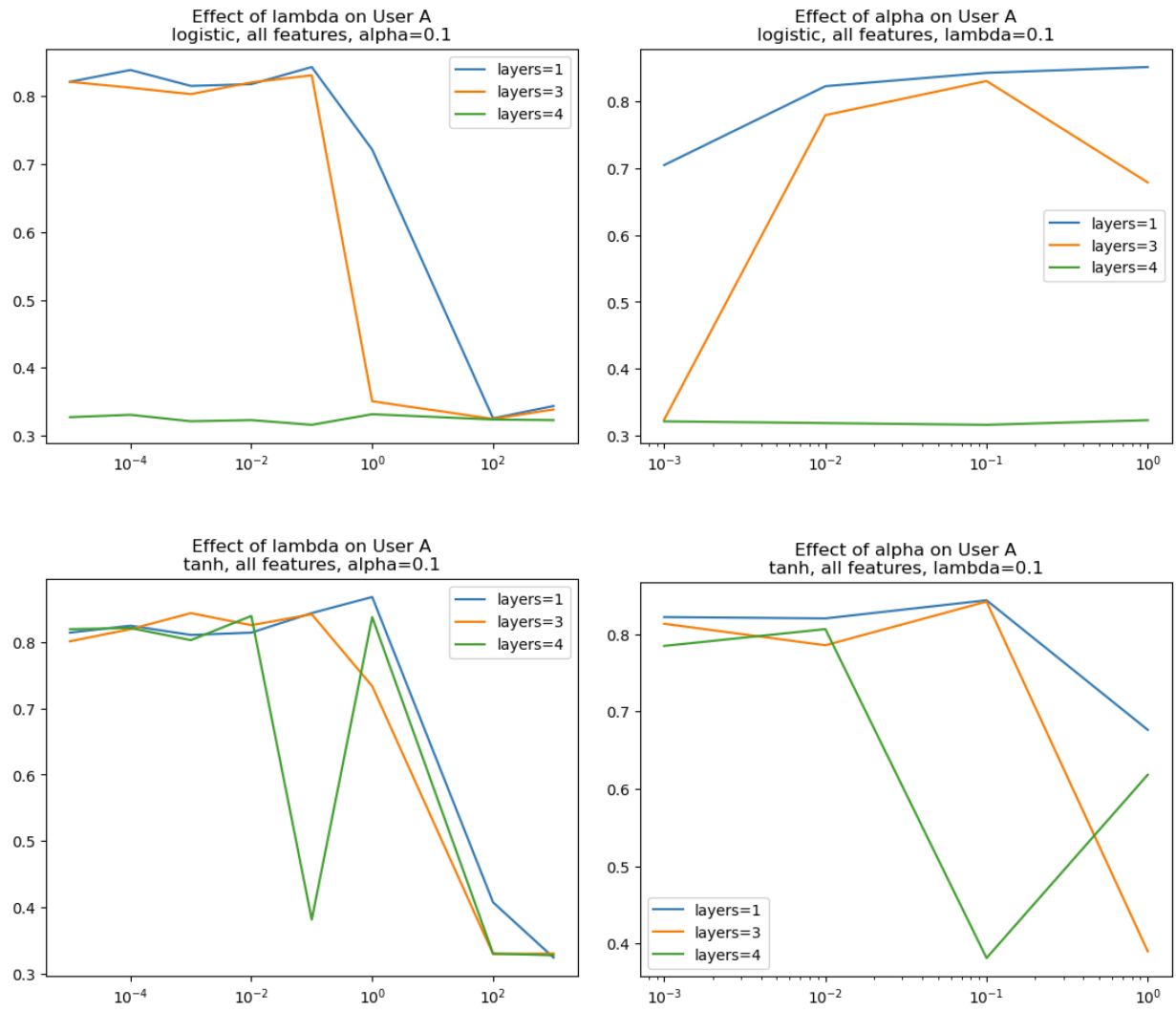
Hidden Layer Structure	(112)	(112)	(112, 112, 112)	(112)	(112)	(112)	(112)	(112, 112, 112)
Features used	all	all	all	all	all	all	all	all
# models above 85% test accuracy	1 (0.1736%)	2 (0.3472%)	22 (3.8194%)	41 (7.1181%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)

During testing, neural networks trained using the tanh activation function produced roughly twice as many models with test accuracy above 85% than neural networks trained using the logistic activation function for users A and B. There is little difference in accuracy between the logistic activation function and the tanh activation function for users C and D. Overall, it appears that the activation functions performed roughly equally, even when comparing precision and recall.

Among the models above 85% testing accuracy, many only have one hidden layer of 112 nodes, though it is important to note that there was much more variability in structure for the tanh activation neural networks. The top performing models also often used all features, although there were some which used only the mean features (which account for half of the features provided).

Using the data from User A, here are plots showing the effect of lambda on the mode top alpha score and the effect of alpha on the mode top lambda score:

Figures: effect of alpha and lambda on User A



As the graph shows, an alpha score around 0.1 improves the accuracy of all models, and low lambda scores contribute to the accuracy as well.

Results Table

Logistic Regression

Best Logistic Regression Model Precision and Recall(2nd Degree Polynomial Feature Transformation with L2 Regularization):

	Precision (0,1,2)	Recall (0,1 2)	Confusion Matrix
User A	.87 .84 .85	.89 .79 .89	186 16 7 17 145 21 10 11 163
User B	.94 .88 .87	.87 .92 .90	182 12 15 5 169 10 6 12 165
User C	.71 .70 .67	.67 .74 .68	139 30 40 25 136 22 32 27 125
User D	.89 .86 .81	.86 .85 .84	179 11 19 9 156 18 14 15 155

SVM

Best Model(3rd degree Polynomial Kernel):

	Precision (0,1,2)	Recall (0,1 2)	Confusion Matrix
User A	.84 .68 .67	.6 .77 .82	125 44 40 10 140 33 13 21 150
User B	.97 .76 .90	.82 .96 .82	171 26 12 2 177 5 4 29 150
User C	.71 .58 .59	.48 .75 .63	101 50 58 23 138 22 18 50 116
User D	.94 .91 .61	.70 .66 .95	147 5 57 6 121 56 3 7 174

Neural Networks

The following tables show the training and testing accuracy of neural networks trained with differences in alpha values, lambda values, and layer structure. The neural networks were trained using either the logistic function with L2 regularization or the tanh function with L2 regularization. Each neural network corresponding to the accuracy score shown was trained on 60% of the data for a particular user, and tested using data from the same user. The neural networks were trained using all 112 features of the dataset. For additional training data, see our GitHub link here: neural network data can be found in the nn_data folder.

The alpha values tested were from the following list: [0.001, 0.01, 0.1, 1.0]. The lambda values tested were from the following list: [0.00001, 0.0001, 0.001, 0.01, 0.1, 1.0, 100, 1000]. The following tables show the accuracy for users A and B. Users C and D are not included here, but their data can be found in the GitHub repository as mentioned above.

User A accuracy with logistic activation function

Lambda \ best model info	data	0.00001	0.0001	0.001	0.01	0.1	1	100	1000
a=0.001	train (1)*	0.807292	0.826389	0.824074	0.826968	0.815394	0.62037	0.34375	0.338542
a=0.001	test (1)*	0.744792	0.692708	0.727431	0.703125	0.704861	0.59809	0.317708	0.325521
a=0.001	train (3)***	0.33912	0.340856	0.346065	0.334491	0.339699	0.338542	0.336227	0.344329
a=0.001	test (3)***	0.320313	0.322049	0.333333	0.328125	0.323785	0.325521	0.328993	0.31684
a=0.001	train (4)****	0.33912	0.333912	0.336227	0.337963	0.341435	0.340278	0.336806	0.329861
a=0.001	test (4)****	0.324653	0.332465	0.328993	0.326389	0.321181	0.322917	0.328125	0.338542
a=0.01	train (1)*	1	1	1	1	0.998843	0.806134	0.341435	0.340278
a=0.01	test (1)*	0.800347	0.815972	0.828125	0.820313	0.822917	0.751736	0.321181	0.322917
a=0.01	train (3)***	0.359375	0.333333	0.332755	0.342014	1	0.340856	0.339699	0.334491
a=0.01	test (3)***	0.358507	0.333333	0.334201	0.320313	0.779514	0.322049	0.323785	0.331597
a=0.01	train (4)****	0.340278	0.336806	0.340856	0.353009	0.343171	0.339699	0.339699	0.337384
a=0.01	test (4)****	0.322917	0.328125	0.322049	0.303819	0.318576	0.323785	0.323785	0.327257
a=0.1	train (1)*	1	1	1	1	1	0.844907	0.338542	0.326389
a=0.1	test (1)*	0.821181	0.838542	0.815104	0.817708	0.842882	0.721354	0.325521	0.34375
a=0.1	train (3)***	1	1	1	1	1	0.321759	0.33912	0.329861
a=0.1	test (3)***	0.821181	0.8125	0.802951	0.820313	0.830729	0.350694	0.324653	0.338542
a=0.1	train (4)****	0.337384	0.335069	0.341435	0.340278	0.344907	0.334491	0.339699	0.340278
a=0.1	test (4)****	0.327257	0.330729	0.321181	0.322917	0.315972	0.331597	0.323785	0.322917
a=1.0	train (1)*	1	1	1	1	1	0.805556	0.344329	0.333912
a=1.0	test (1)*	0.828125	0.805556	0.8125	0.830729	0.851563	0.719618	0.31684	0.332465
a=1.0	train (3)***	0.333912	0.326968	0.342014	0.340278	0.966435	0.323495	0.320602	0.347222
a=1.0	test (3)***	0.332465	0.342882	0.320313	0.322917	0.678819	0.34809	0.352431	0.3125
a=1.0	train (4)****	0.331019	0.340856	0.33044	0.337384	0.340278	0.336227	0.346644	0.30787
a=1.0	test (4)****	0.336806	0.322049	0.337674	0.327257	0.322917	0.328993	0.313368	0.315104

* 1-layer structure: (112)

*** 3-layer structure: (112, 112, 112)

**** 4-layer structure: (112, 56, 30, 10)

† Models shown have been trained on all features and 60% of the data. Additional models have been trained but are not shown.

User A accuracy with tanh activation function

Lambda \ best model info	data	0.00001	0.0001	0.001	0.01	0.1	1	100	1000
a=0.001	train (1)*	0.998843	0.998843	0.999421	0.998843	0.998843	0.991898	0.414931	0.329282
a=0.001	test (1)*	0.824653	0.809896	0.803819	0.789931	0.822049	0.826389	0.377604	0.33941
a=0.001	train (3)***	1	1	1	1	1	1	0.336806	0.326968
a=0.001	test (3)***	0.790799	0.801215	0.799479	0.782986	0.813368	0.845486	0.328125	0.342882
a=0.001	train (4)****	1	1	1	1	1	1	0.342014	0.337384
a=0.001	test (4)****	0.789931	0.760417	0.789063	0.789931	0.784722	0.820313	0.320313	0.327257
a=0.01	train (1)*	1	1	1	1	1	0.998264	0.40625	0.342014
a=0.01	test (1)*	0.806424	0.799479	0.831597	0.835938	0.820313	0.84375	0.391493	0.320313
a=0.01	train (3)***	1	1	1	1	1	1	0.340856	0.342593
a=0.01	test (3)***	0.815104	0.80816	0.788194	0.811632	0.78559	0.835938	0.322049	0.319444
a=0.01	train (4)****	1	1	1	1	1	1	0.346644	0.339699
a=0.01	test (4)****	0.796875	0.796875	0.789931	0.811632	0.806424	0.853299	0.313368	0.323785
a=0.1	train (1)*	1	1	1	1	1	0.999421	0.427083	0.339699
a=0.1	test (1)*	0.814236	0.824653	0.810764	0.814236	0.84375	0.868056	0.407118	0.323785
a=0.1	train (3)***	1	1	1	1	1	0.876157	0.336227	0.335648
a=0.1	test (3)***	0.801215	0.819444	0.84375	0.825521	0.842014	0.733507	0.328993	0.329861
a=0.1	train (4)****	1	1	1	1	0.393519	0.996528	0.335648	0.337384
a=0.1	test (4)****	0.819444	0.821181	0.802951	0.83941	0.381076	0.837674	0.329861	0.327257
a=1.0	train (1)*	1	1	1	1	0.855324	0.622685	0.339699	0.298032
a=1.0	test (1)*	0.710069	0.730035	0.697049	0.799479	0.676215	0.546007	0.323785	0.286458
a=1.0	train (3)***	0.295139	0.449074	0.404514	0.370949	0.402199	0.435764	0.338542	0.320602
a=1.0	test (3)***	0.274306	0.417535	0.396701	0.331597	0.389757	0.421007	0.325521	0.323785
a=1.0	train (4)****	0.690394	0.397569	0.509838	0.474537	0.712963	0.346065	0.326968	0.273727
a=1.0	test (4)****	0.555556	0.386285	0.449653	0.434028	0.618056	0.314236	0.342882	0.298611

* 1-layer structure: (112)

*** 3-layer structure: (112, 112, 112)

**** 4-layer structure: (112, 56, 30, 10)

† Models shown have been trained on all features and 60% of the data. Additional models have been trained but are not shown.

User B accuracy with logistic activation function

Lambda \ best model info	data	0.00001	0.0001	0.001	0.01	0.1	1	100	1000
a=0.001	train (1)*	0.882523	0.87963	0.869213	0.877894	0.851273	0.702546	0.344907	0.339699
a=0.001	test (1)*	0.770833	0.758681	0.778646	0.769965	0.772569	0.632813	0.315972	0.323785
a=0.001	train (3)***	0.340278	0.340278	0.343171	0.354167	0.340278	0.34375	0.342593	0.33912
a=0.001	test (3)***	0.322917	0.322917	0.318576	0.302083	0.322917	0.317708	0.319444	0.324653
a=0.001	train (4)****	0.336806	0.338542	0.337384	0.323495	0.342593	0.33912	0.336806	0.336227
a=0.001	test (4)****	0.328125	0.325521	0.327257	0.326389	0.345486	0.324653	0.328125	0.328993
a=0.01	train (1)*	1	1	1	1	1	0.911458	0.331597	0.327546
a=0.01	test (1)*	0.875868	0.840278	0.858507	0.868924	0.878472	0.769965	0.335938	0.342014
a=0.01	train (3)***	0.33912	0.34375	0.337384	0.337963	0.340278	0.345486	0.343171	0.328125
a=0.01	test (3)***	0.324653	0.317708	0.327257	0.326389	0.322917	0.315104	0.318576	0.341146
a=0.01	train (4)****	0.340856	0.339699	0.337963	0.339699	0.338542	0.335648	0.342593	0.338542
a=0.01	test (4)****	0.322049	0.323785	0.326389	0.323785	0.325521	0.329861	0.319444	0.325521
a=0.1	train (1)*	1	1	1	1	1	0.915509	0.340278	0.326968
a=0.1	test (1)*	0.874132	0.861111	0.880208	0.864583	0.883681	0.790799	0.322917	0.342882
a=0.1	train (3)***	1	1	1	1	1	0.335069	0.346644	0.322917
a=0.1	test (3)***	0.847222	0.837674	0.859375	0.872396	0.886285	0.330729	0.313368	0.348958
a=0.1	train (4)****	0.34375	0.341435	0.332755	0.343171	0.34838	0.345486	0.340278	0.336227
a=0.1	test (4)****	0.317708	0.321181	0.334201	0.318576	0.310764	0.315104	0.322917	0.328993
a=1.0	train (1)*	1	1	1	1	1	0.907986	0.34375	0.32581
a=1.0	test (1)*	0.857639	0.862847	0.847222	0.864583	0.875868	0.81684	0.317708	0.34809
a=1.0	train (3)***	0.333912	0.331597	0.335069	0.328704	0.963542	0.326968	0.324653	0.335648
a=1.0	test (3)***	0.332465	0.335938	0.330729	0.340278	0.773438	0.342882	0.346354	0.329861
a=1.0	train (4)****	0.333912	0.341435	0.340278	0.342593	0.331597	0.328125	0.343171	0.33044
a=1.0	test (4)****	0.332465	0.321181	0.322917	0.319444	0.335938	0.341146	0.318576	0.326389

* 1-layer structure: (112)

*** 3-layer structure: (112, 112, 112)

**** 4-layer structure: (112, 56, 30, 10)

† Models shown have been trained on all features and 60% of the data. Additional models have been trained but are not shown.

User B accuracy with tanh activation function

Lambda \ best model info	data	0.00001	0.0001	0.001	0.01	0.1	1	100	1000
a=0.001	train (1)*	1	1	1	1	1	1	0.435185	0.340278
a=0.001	test (1)*	0.841146	0.826389	0.84375	0.831597	0.823785	0.836806	0.405382	0.322917
a=0.001	train (3)***	1	1	1	1	1	1	0.340278	0.335069
a=0.001	test (3)***	0.829861	0.782986	0.8125	0.790799	0.80816	0.868924	0.322917	0.330729
a=0.001	train (4)****	1	1	1	1	1	1	0.333333	0.33912
a=0.001	test (4)****	0.824653	0.829861	0.807292	0.799479	0.806424	0.855903	0.333333	0.324653
a=0.01	train (1)*	1	1	1	1	1	1	0.436921	0.337384
a=0.01	test (1)*	0.832465	0.81684	0.850694	0.84809	0.865451	0.898438	0.407118	0.327257
a=0.01	train (3)***	1	1	1	1	1	1	0.335648	0.342014
a=0.01	test (3)***	0.805556	0.822917	0.835069	0.81684	0.826389	0.898438	0.329861	0.320313
a=0.01	train (4)****	1	1	1	1	1	1	0.345486	0.340278
a=0.01	test (4)****	0.81684	0.793403	0.828125	0.844618	0.83941	0.894097	0.315104	0.322917
a=0.1	train (1)*	1	1	1	1	1	1	0.439236	0.340278
a=0.1	test (1)*	0.822917	0.863715	0.852431	0.862847	0.872396	0.901042	0.397569	0.322917
a=0.1	train (3)***	1	1	1	1	1	1	0.339699	0.335648
a=0.1	test (3)***	0.844618	0.836806	0.84375	0.831597	0.867188	0.908854	0.323785	0.329861
a=0.1	train (4)****	1	1	1	1	1	1	0.337963	0.337384
a=0.1	test (4)****	0.866319	0.855903	0.84375	0.84375	0.877604	0.902778	0.326389	0.327257
a=1.0	train (1)*	1	1	1	1	1	0.745949	0.33912	0.302662
a=1.0	test (1)*	0.822049	0.828993	0.828993	0.87934	0.914063	0.676215	0.324653	0.28559
a=1.0	train (3)***	0.550347	0.509838	0.550926	0.418403	0.551505	0.395833	0.333333	0.321759
a=1.0	test (3)***	0.524306	0.458333	0.507813	0.401042	0.522569	0.416667	0.333333	0.321181
a=1.0	train (4)****	0.609375	0.82581	0.566551	0.646412	0.57581	0.638889	0.337963	0.300926
a=1.0	test (4)****	0.560764	0.743056	0.53125	0.605035	0.552083	0.59375	0.326389	0.306424

* 1-layer structure: (112)

*** 3-layer structure: (112, 112, 112)

**** 4-layer structure: (112, 56, 30, 10)

† Models shown have been trained on all features and 60% of the data. Additional models have been trained but are not shown.

Conclusion

Using EEG mean and std readings, we were able to classify users A and B with high levels of accuracy, precision, and recall. However, training for users C and D proved difficult for every model, with most achieving below 85% accuracy. Here are the test accuracy results for each user's best model:

User A

Logistic	SVMs	Neural Networking
0.91944	0.80208	0.86806

User B

Logistic	SVMs	Neural Networking
0.94166	0.90972	0.91406

User C

Logistic	SVMs	Neural Networking
0.76667	0.67361	0.70486

User D

Logistic	SVMs	Neural Networking
0.89583	0.83333	0.81597

Overall, the best-performing models used logistic regression; particularly, using 2nd degree polynomial transformation, L2 regularization, and K-fold cross validation. This could have worked best due to the relatively low sample size, with each classification only having 960 x-values for one user.

It is important to note, however, that the neural network competed well with the other models despite being trained on 25% less data.

There is a likelihood that noise in the EEG readings accounted for some level of variance in the models. This could have been accounted for by removing outliers during preprocessing.

Although we did not have time to test it, we also believe that it would be difficult to generalize the model to work accurately on all users. As the data shows, there are already lower levels of success when predicting the movement of users C and D, even though the models were specially trained for them. Training one model on every user's data may lead to underfitting. One reason for this may be the variability from one brain to the next; like how the weights and biases between neural nets can vastly differ, each brain is unique, and that can make it hard to parse brain waves without specializing to a particular user.