

COG 189

Final Project



Introduction





Introduction/Motivation

With development of technology, we have been given the opportunity to track the brain to better understand its responses to stimulus. The workings of the brain are still mostly a mystery so by continuing to research and conduct experiments, we are able to get closer to understanding its structure. Our project explores Motor Imagery, which is how the brain reacts to the intention of performing movement. Both datasets explored data from EEG and BCIs interaction. We wanted to explore the differences between invasive and noninvasive techniques as well as learn more about how noninvasive methods can help patients in the future. We also wanted to explore how reliable predictive movement was with the noninvasive techniques.



Related Works



“Hand Movement Direction Decoded from MEG and EEG”

- The goal was to find a non invasive way to predict motor movements.
- They found that each extremity has its own cortical area with at least 4 distinct neural activation patterns within them.
- They were able to predict small hand movements 67% of the time.



“EEG-based discrimination between imagination of right and left hand movement”

- They wanted to see if there was a similarity between imagining and executing motor function.
- They discovered that executing an action activates the opposite side of your cortex.
- They found that when you imagine the motor activity both sides of the cortex activates

Speaking: Silvina Rodriguez



Related Works



“How capable is non-invasive EEG data of predicting the next movement? A mini review”

- They wanted to develop BCIs based on the before-movement
- Further research would lead to better predictions of upcoming movement as well as shorter response times that would seem more natural

“Performance analysis of left/right hand movement classification from EEG signal by intelligent algorithms”



- RBF kernelized SVM classifier indicated the highest performance accuracy of 82.14% with both original and reduced feature set.
- Further study will center around the techniques (optimizing feature selection, extraction and classification methodologies) that are to be implemented in online classification of EEG data for BCI research

Speaking: Michael Lai



Related Works



Currently, Motor imagery (MI) has become an important area of study in the field of neurological rehabilitation research. A previous research paper titled, Target-oriented motor imagery for grasping action: different characteristics of brain activation between kinesthetic and visual imagery, explained that, “the neural mechanisms underlying the process of MI demonstrate functional equivalence with motor execution (ME), with the two sharing similar physiological and anatomical characteristics. During MI, increased brain activity is observed at the premotor, supplementary motor, cingulate, parietal cortical areas, basal ganglia, and cerebellum, which are also the principal brain areas involved in ME.” In another previous paper called, Recognition of Brain Waves of Left and Right Hand Movement Imagery with Portable Electroencephalographs, the paper elaborated on the fact that the obtained motor imagery EEG data of right hand vs left was discriminated between through the C3 and C4 electrodes.

Speaking: (Aliya)

Methods





Methods - Research Question

We deduced that given the evidence, despite meager, from previous studies suggesting the importance of MI in neurological rehabilitation, we could defend the theory on the basis of experiential studies of healthy subjects. Therefore, we decided to examine the dataset from BCI competition III part IVa.

Using the dataset we wanted to analyze the data from the C3 channel electrode and plot it to see what parts of the brain were active during trials of each subject.

Speaking: (Aliya)



Methods - Participants

The dataset consisted of 5 healthy subjects. During the study, the subjects were instructed to sit in a chair with their arms resting on the armrests and then were exposed to one or two types of visual stimuli. The first visual stimulus was where targets were indicated by letters appearing behind a fixation cross, inducing target-correlated eye movements, and the second type was where a randomly moving object indicated targets, inducing target-uncorrelated eye movements. The subjects were also provided with visual cues that determined which of the motor imageries was to be performed: right hand or right foot. Additionally, target cues were intermitted by periods of random length, 1.75 to 2.25 s, in which the subject could relax.

Speaking: (Aliya)



Methods - Research Question

What classification techniques(SVM, Random Forest, and Logistic Regression and Decision Trees) and hyper parameters are effective for classifying EEG data?

Written by: Ryan Cai

Methods - Preprocessing Approaches

Initially the data was in a MatLab We approached preprocessing through the removal of all NAN values and the reshaping of the Y dimension to transform so X is no longer 3 dimensional in terms of Y. We accomplished this by assigning a Y value to each individual X time series. We were following this methodology. (although out of curiosity I decided to train the models with all the features).

Fig. 1



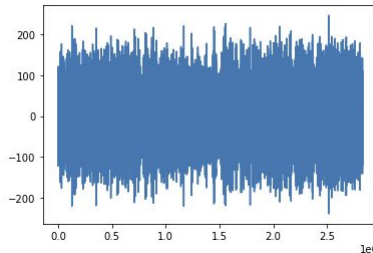
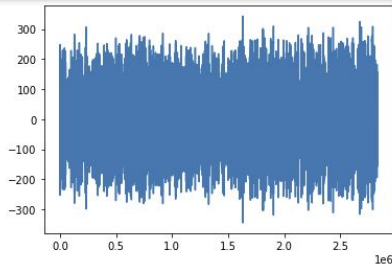
Steps of EEG signal processing and analysis

Methods - Filtering

Along with our Classification algorithms, one other group member did some feature extraction through Filtering.

Band Filter:

```
ay_cnt_filtered_c3 = bandPassFilter(data_set_IVa_ay_cnt[51])  
ay_cnt_filtered_c4 = bandPassFilter(data_set_IVa_ay_cnt[55])  
#ay_cnt_filtered_fp1 = bandPassFilter(data_set_IVa_ay_clab[0])  
  
plt.plot(ay_cnt_filtered_c3)  
plt.show()  
plt.plot(ay_cnt_filtered_c4)  
plt.show()
```



Methods - Filtering

FFT Filter:

```
from scipy.fft import fft, fftfreq

def fft_data(data):
    # Number of samples in normalized_tone
    N = len(data)
    yf = fft(data)
    xf = fftfreq(N, 1 / fs)

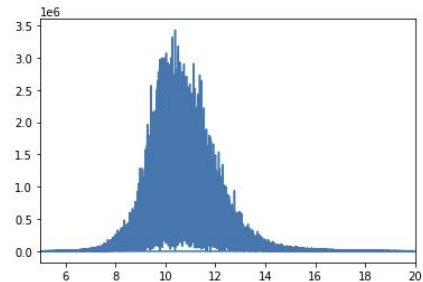
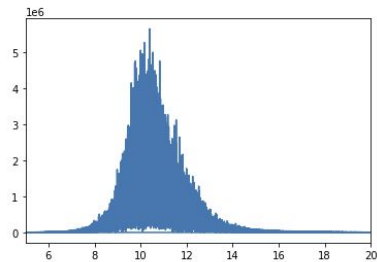
    return xf
```

```
def fft_plot(data):
    # Number of samples in normalized_tone
    N = len(data)
    yf = fft(data)
    xf = fftfreq(N, 1 / fs)
    plt.plot(xf, np.abs(yf))
    plt.xlim(5, 20)
    plt.show()
```

```
fft_plot(ay_cnt_filtered_c3)
```

```
fft_plot(ay_cnt_filtered_c4)
```

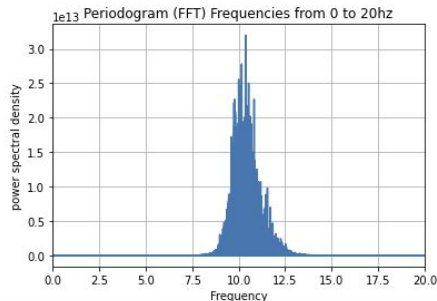
```
#fft_plot(ay_cnt_filtered_fp1)
```



Methods - Filtering

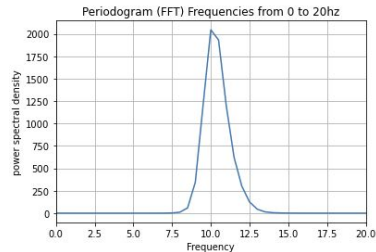
Welches Method:

```
power = np.abs(np.fft.fft(ay_cnt_filtered_c3))**2
frequency = np.fft.fftfreq(len(ay_cnt_filtered_c3),1/fs)
plt.plot(frequency,power)
plt.xlabel('Frequency')
plt.ylabel("power spectral density")
plt.title("Periodogram (FFT) Frequencies from 0 to 20hz")
plt.xlim(0,20)
plt.grid()
```



Welches Method to Estimate the power Spectrum

```
f, pxx = scipy.signal.welch(ay_cnt_filtered_c3, fs = fs, scaling = 'spectrum', nperseg = 2000)
plt.plot(f,pxx)
plt.xlabel('Frequency')
plt.ylabel("power spectral density")
plt.title("Periodogram (FFT) Frequencies from 0 to 20hz")
plt.xlim(0,20)
plt.grid()
```



Welches Method to Estimate the power Spectrum

Methods - Data

Of the five participants, we decided to focus on analyzing the first participant

Subject 1 Data- AY

```
# Untangled data directly from mat files
data_set_IVa_ay_cnt = cnt_df(data_set_IVa_ay['cnt'])
data_set_IVa_ay_name = get_Name(data_set_IVa_ay)
data_set_IVa_ay_fs = get_frequency(data_set_IVa_ay)
data_set_IVa_ay_clab = get_clab(data_set_IVa_ay)
data_set_IVa_ay_xpos = get_xpos(data_set_IVa_ay)
data_set_IVa_ay_ypos = get_ypos(data_set_IVa_ay)
data_set_IVa_ay_y = get_y(data_set_IVa_ay)
data_set_IVa_ay_className = get_className(data_set_IVa_ay)
data_set_IVa_ay_pos = get_pos(data_set_IVa_ay)
```

| | |
|--|-----------------|
| <i>#name of subject</i> | <i># Values</i> |
| <i>1</i> | <i>1</i> |
| <i>#frequency of sample</i> | <i>1</i> |
| <i>#Labels for the Electrodes Positions</i> | <i>118</i> |
| <i>#Labels for the Electrodes Positions</i> | <i>118</i> |
| <i>#Values of the Electrode Positions</i> | <i>118</i> |
| <i>#Value 1 or 2 depending on 'right'/'foot'</i> | <i>280</i> |
| <i>#Label 'Right', "foot"</i> | <i>2</i> |
| <i>#Y position values 61625-2813716</i> | <i>280</i> |

Methods - Classification(Decision Trees)

Using Sklearn, on the first dataset, we ran Decision Tree Classification with different criterion, splittier, max_depth, and min_samples_leaf. Our goal was to avoid overfitting which is a huge problem for noisy EEG data especially when it's analyzed with a Decision Tree. Limiting the max_depth and the min_samples_leaf were effective ways for accomplishing this.

Spoken by: Ryan Cai

```
d1 = {"y": data_set_IVa_ay_y , "pos": data_set_IVa_ay_pos}
data_set_IVa_mrk = pd.DataFrame(d1)
# Transforming Y so Y is expanded making x a 2d array
Y_transformed = []
last_time = 0
for row in range(data_set_IVa_mrk[:28].shape[0]):
    for x in range(data_set_IVa_mrk.iloc[row,1] - last_time):
        Y_transformed.append(data_set_IVa_mrk.iloc[row,0])
    last_time = data_set_IVa_mrk.iloc[row,1]

from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV, StratifiedKFold, train_test_split

dtclassifier = DecisionTreeClassifier()
param_grid = [{'criterion': ['gini','entropy'],
                'splitter': ['best','random'],
                'max_depth': [3, 5, 7, 9, 11, 13, 15, 17, 19, 21],
                'min_samples_leaf': [5, 50, 500, 5000, 50000, 500000]}]

clf = GridSearchCV(estimator = dtclassifier, param_grid = param_grid, cv=StratifiedKFold(n_splits=5),
```


Methods - Classification(Logistic Regression)

Using Sklearn, on the first dataset, we ran Logistic Regression with varying regularization values of C. We also tried different solvers with different norms(l1 and l2).\

Spoken by: Ryan Cai

```
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
pipe = Pipeline([('std', StandardScaler()),
                  ('classifier', LogisticRegression())])


search_space = [{'classifier': [LogisticRegression(max_iter=5000)],
                  'classifier__solver': ['saga'],
                  'classifier__penalty': ['l1', 'l2'],
                  'classifier__C': np.logspace(-2,2, 5)
                },
                {'classifier': [LogisticRegression(max_iter=5000)],
                  'classifier__solver': ['lbfgs'],
                  'classifier__penalty': ['l2'],
                  'classifier__C': np.logspace(-2,2, 5)},
                {'classifier': [LogisticRegression(max_iter=5000)],
                  'classifier__solver': ['lbfgs', 'saga'],
                  'classifier__penalty': ['none']}
               ]

clf = GridSearchCV(pipe, search_space, cv=StratifiedKFold(n_splits=5),
                  scoring=['accuracy', 'roc_auc_ovr_weighted', 'f1_micro'], refit=False,
                  verbose=3, n_jobs = -1)
X_train, X_test, y_train, y_test = train_test_split(X, Y, train_size = 15000, random_state=3087)

best_model_LR = clf.fit(X_train, y_train)
```



Methods - Classification(Random Forest)



For Random Forest, we varied criterion, n_estimators(number of trees in the forest) max_features, min_samples_leaf. The purpose of varying max_features is to limit how many features are considered when splitting and setting a min_samples_leaf which we found to be better for smoothing out the model


```
from sklearn.ensemble import RandomForestClassifier
rfclassifier = RandomForestClassifier()
param_grid = [{'criterion': ['gini', 'entropy'],
               'n_estimators': [10, 100, 1024],
               'max_features': [1, 2, 4, 5, 6, 12, 16, 20],
               'min_samples_leaf': [1, 5, 25, 50, 100, 500, 1000]}]

clf = GridSearchCV(estimator = rfclassifier, param_grid = param_grid, cv=StratifiedKFold(n_splits=5),
                  scoring=['accuracy', 'roc_auc_ovr_weighted', 'f1_micro'], refit=False,
                  verbose=3, n_jobs = -1)
Mush_best_model_RFs = []

X_train, X_test, y_train, y_test = train_test_split(X, Y, train_size = 15000, random_state=254)

best_model_RF = clf.fit(X_train, y_train)
```

Spoken by: Ryan Cai.



Methods - background analysis for dataset 2a



For the dataset 2a, this data set consists of EEG data from 9 subjects. The cue-based BCI paradigm consisted of four different motor imagery tasks, which are the imagination of movement of the left hand (class 1), right hand (class 2), both feet (class 3), and tongue (class 4). Each subject was recorded for two separate sessions on different days. Each session consists of 6 runs which are separated by short breaks. Each run consists of 48 trials (12 trials for each of the four classes), which means there is a total of 288 trials per session.

By studying the dataset, we can see there are 26 features in total, which are time, 22 EEG channels and 3 monopolar EOG channels. The time is in a range from 0 to 2747996 ms, the highpass is set to be 0.5 Hz while the lowpass is set to be 100 Hz. sfreq is also set to be 250 Hz.

Written and spoken by: Bingzhe Wang



Methods - research question for dataset 2a



As pointed out in the dataset description, the EOG data must not be used for classification. Therefore, our main focus should be on the EEG data and hopefully we can try to figure out any possible trends for the EEG data by using machine learning algorithms. In this case, we figure that logistic regression would be an appropriate approach to investigate the data because there are many features. Also, the researchers had already divided the whole data into two separate parts, with training sets marked as “T” and evaluation sets marked as “E”. Thus, we intended to check whether the pattern shown in training set would also work for the evaluation sets or not so that we can find out the relationship between imaging and executing motor functions.

Our research question: Does the pattern of the training sets also apply for the evaluation sets?

Written and spoken by: Bingzhe Wang

Methods - first analysis for dataset 2a

We first downloaded the data files and upload them so that we can analyze them. However, they are all presented in gdf files. Therefore, we downloaded the mne package to load them.

Downloading essential packages for loading in GDF files

```
In [1]: conda install scipy matplotlib scikit-learn mayavi ipython-notebook
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: failed with current_repodata.json, will retry with next repodata source.
Collecting package metadata (repodata.json): done
Solving environment: failed
```

PackagesNotFoundError: The following packages are not available from current channels:

- ipython-notebook

Current channels:

- <https://repo.anaconda.com/pkgs/main/osx-64>
- <https://repo.anaconda.com/pkgs/main/noarch>
- <https://repo.anaconda.com/pkgs/r/osx-64>
- <https://repo.anaconda.com/pkgs/r/noarch>

To search for alternate channels that may provide the conda package you're looking for, navigate to

<https://anaconda.org>

and use the search bar at the top of the page.

Note: you may need to restart the kernel to use updated packages.

```
In [2]: pip install PySurfer
```

```
Requirement already satisfied: PySurfer in ./anaconda3/lib/python3.7/site-packages (0.11.0)
Requirement already satisfied: mayavi in ./anaconda3/lib/python3.7/site-packages (from PySurfer) (4.7.2)
Requirement already satisfied: scipy in ./anaconda3/lib/python3.7/site-packages (from PySurfer) (1.3.0)
Requirement already satisfied: nibabel>=1.2 in ./anaconda3/lib/python3.7/site-packages (from PySurfer) (3.2.1)
Requirement already satisfied: numpy in ./anaconda3/lib/python3.7/site-packages (from PySurfer) (1.16.4)
Requirement already satisfied: matplotlib in ./anaconda3/lib/python3.7/site-packages (from PySurfer) (3.1.0)
Requirement already satisfied: agptools in ./anaconda3/lib/python3.7/site-packages (from mayavi->PySurfer) (5.1.0)
Requirement already satisfied: envisage in ./anaconda3/lib/python3.7/site-packages (from mayavi->PySurfer) (5.0.0)
Requirement already satisfied: pyface>=6.1.1 in ./anaconda3/lib/python3.7/site-packages (from mayavi->PySurfer) (7.3.0)
Requirement already satisfied: pygments in ./anaconda3/lib/python3.7/site-packages (from mayavi->PySurfer) (2.4.2)
Requirement already satisfied: traits>=6.0.0 in ./anaconda3/lib/python3.7/site-packages (from mayavi->PySurfer) (6.2.0)
Requirement already satisfied: traitsui>=7.0.0 in ./anaconda3/lib/python3.7/site-packages (from mayavi->PySurfer) (7.1.1)
Requirement already satisfied: vtk in ./anaconda3/lib/python3.7/site-packages (from mayavi->PySurfer) (9.0.1)
Requirement already satisfied: packaging>=14.3 in ./anaconda3/lib/python3.7/site-packages (from nibabel>=1.2->PySurfer) (19.0)
Requirement already satisfied: cycler>=0.10 in ./anaconda3/lib/python3.7/site-packages (from matplotlib->PySurfer) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in ./anaconda3/lib/python3.7/site-packages (from matplotlib->PySurfer) (1.1.0)
Requirement already satisfied: pyparsing>=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in ./anaconda3/lib/python3.7/site-packages (from fonttools->PySurfer) (2.4.0)
Requirement already satisfied: python-dateutil>=2.1 in ./anaconda3/lib/python3.7/site-packages (from matplotlib->PySurfer) (2.8.0)
Requirement already satisfied: configobj in ./anaconda3/lib/python3.7/site-packages (from apptools->mayavi->PySurfer) (5.0.6)
Requirement already satisfied: setuptools in ./anaconda3/lib/python3.7/site-packages (from envisage->mayavi->PySurfer) (41.0.1)
Requirement already satisfied: importlib-resources>=1.1.0 in ./anaconda3/lib/python3.7/site-packages (from pyface>=6.1.1->PySurfer) (5.10.0)
```

```
In [3]: pip install mne --upgrade
```

```
Requirement already up-to-date: mne in ./anaconda3/lib/python3.7/site-packages (0.22.0)
Requirement already satisfied, skipping upgrade: numpy>=1.11.3 in ./anaconda3/lib/python3.7/site-packages (from mne) (1.16.4)
Requirement already satisfied, skipping upgrade: scipy>=0.17.1 in ./anaconda3/lib/python3.7/site-packages (from mne) (1.3.0)
Note: you may need to restart the kernel to use updated packages.
```

Methods - analysis for dataset 2a

Now that we have downloaded the necessary packages, we try to load the data and convert them to pandas dataframe to see if the information of the data is accordance to the dataset

```
In [4]: import os
import numpy as np
import mne
```

```
In [5]: raw1 = mne.io.read_raw_gdf('A01E.gdf')
print(raw1.info)
```

```
Extracting EDF parameters from /Users/test/A01E.gdf...
GDF file detected
Setting channel info structure...
Creating raw.info structure...
<Info | 7 non-empty values
bads: []
ch_names: EEG-Fz, EEG-0, EEG-1, EEG-2, EEG-3, EEG-4, EEG-5, EEG-C3, EEG-6, ...
chs: 25 EEG
custom_ref_applied: False
highpass: 0.5 Hz
lowpass: 100.0 Hz
meas_date: 2005-01-19 12:00:00 UTC
nchan: 25
projs: []
sfreq: 250.0 Hz
>
```

```
/Users/test/anaconda3/lib/python3.7/site-packages/mne/io/edf/edf.py:1044: DeprecationWarning: The binary mode of from
string is deprecated, as it behaves surprisingly on unicode inputs. Use frombuffer instead
etmode = np.fromstring(etmode, UINT8).tolist()[0]
<ipython-input-5-300e5f9b376e>:1: RuntimeWarning: Channel names are not unique, found duplicates for: {'EEG'}. Applyi
ng running numbers for duplicates.
raw1 = mne.io.read_raw_gdf('A01E.gdf')
```

```
In [6]: raw2 = mne.io.read_raw_gdf('A01T.gdf')
print(raw2.info)
```

```
Extracting EDF parameters from /Users/test/A01T.gdf...
GDF file detected
Setting channel info structure...
Creating raw.info structure...
<Info | 7 non-empty values
bads: []
ch_names: EEG-Fz, EEG-0, EEG-1, EEG-2, EEG-3, EEG-4, EEG-5, EEG-C3, EEG-6, ...
chs: 25 EEG
custom_ref_applied: False
highpass: 0.5 Hz
lowpass: 100.0 Hz
meas_date: 2005-01-17 12:00:00 UTC
nchan: 25
projs: []
sfreq: 250.0 Hz
>
```

```
/Users/test/anaconda3/lib/python3.7/site-packages/mne/io/edf/edf.py:1044: DeprecationWarning: The binary mode of from
string is deprecated, as it behaves surprisingly on unicode inputs. Use frombuffer instead
etmode = np.fromstring(etmode, UINT8).tolist()[0]
<ipython-input-6-8db2a8959b5>:1: RuntimeWarning: Channel names are not unique, found duplicates for: {'EEG'}. Applyi
ng running numbers for duplicates.
raw2 = mne.io.read_raw_gdf('A01T.gdf')
```

Methods - analysis for dataset 2a

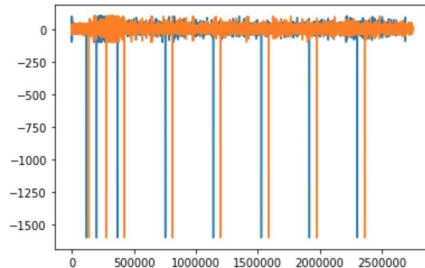
We first plotted the plot of time over EEG-Fz of both training set and evaluation set.

Now we will see what effect has time over the EEG data.

```
In [11]: import matplotlib.pyplot as plt
```

```
In [12]: plt.plot(df_raw2['time'], df_raw2['EEG-Fz'])  
plt.plot(df_raw1['time'], df_raw1['EEG-Fz'])
```

```
Out[12]: [<matplotlib.lines.Line2D at 0x8229c6080>]
```





Methods - analysis for dataset 2a

Now we start to perform logistic regression on the converted dataframes. We first set the training sets and testing sets. We decided to mark EEG values that are above 0 as “1” and those below 0 as “-1”.

```
In [22]: ## Set the training set and the testing set according to the description of datas.
X_train = df_raw2['EEG-Fz'].to_numpy()
X_train = np.transpose(X_train[:5000])
X_test = df_raw1['EEG-Fz'].to_numpy()
X_test = np.transpose(X_test[:5000])
Y_train = (X_train > 0).reshape(-1,1).astype(np.float)
Y_train[Y_train == 0] = -1
Y_test = (X_test > 0).reshape(-1,1).astype(np.float)
Y_test[Y_test == 0] = -1
```


Methods - analysis for dataset 2a

Then, we created helper functions for the classification.

```
In [24]: def sigmoid(z):  
         return 1.0/(1.0 + np.exp(-z))
```

```
In [25]: def judge(a, b):  
         if a != b:  
             return 1  
         else:  
             return 0  
         def f_logistic(x, W, b):  
             n = W.dot(x) + b  
             if sigmoid(n) >= 0.5:  
                 return 1  
             else:  
                 return -1  
         def calc_error(X, Y, W, b):  
             count = 0  
             for (xi, yi) in zip(X, Y):  
                 count += judge(yi, f_logistic(xi, W, b))  
             return (count/len(X))
```

```
def grad_L_W_b(X, Y, W, b):  
    one = np.ones(len(X))  
    P = sigmoid(Y * (X.dot(W) + b * one))  
    grad_W = -X.T.dot(((one - P) * Y))  
    grad_b = -one.T.dot(((one - P) * Y))  
    return grad_W, grad_b
```

```
def L_W_b(X, Y, W, b):  
    one = np.ones(len(X))  
    P = sigmoid(Y * (X.dot(W) + b * one))  
    return -one.T.dot(np.log(P))
```



Methods - analysis for dataset 2a

Finally, we perform logistic regression on the training sets and hopefully to generate the loss function.

```
learning_rate = 0.001
iterations    = 10000
losses = []

W = np.zeros(1)
b = 0

for i in range(iterations):
    grad_W, grad_b = grad_L_W_b(X_train, Y_train, W, b)
    W = W - learning_rate*(grad_W)
    b = b - learning_rate*(grad_b)

    losses.append(L_W_b(X_train, Y_train, W, b))
```

Results





Results - Decision Trees

Accuracy best: {'criterion': 'entropy', 'max_depth': 21, 'min_samples_leaf': 5, 'splitter': 'best'}



Roc auc ovr best : {'criterion': 'entropy', 'max_depth': 21, 'min_samples_leaf': 50, 'splitter': 'best'}


F1 micro best : {'criterion': 'entropy', 'max_depth': 21, 'min_samples_leaf': 5, 'splitter': 'best'}

Spoken by: Ryan Cai



Results - Logistic Regression

F1 micro best : {'classifier': LogisticRegression(max_iter=5000),'classifier__C': 10.0, 'classifier__penalty': 'l2', 'classifier__solver': 'lbfgs'}



Roc Auc ovr Weighted: {'classifier': LogisticRegression(max_iter=5000), 'classifier__C': 100.0, 'classifier__penalty': 'l2', 'classifier__solver': 'lbfgs'}


Accuracy best: {'classifier': LogisticRegression(max_iter=5000),'classifier__C': 10.0, 'classifier__penalty': 'l2', 'classifier__solver': 'lbfgs'}

Spoken by: Ryan Cai



Results - Random Forest

params best for accuracy metric: {'criterion': 'entropy', 'max_features': 20, 'min_samples_leaf': 1, 'n_estimators': 1024}



Params best for roc auc ovr weighted: {'criterion': 'entropy', 'max_features': 20, 'min_samples_leaf': 1, 'n_estimators': 1024}

Params best for f1 micro : {'criterion': 'entropy', 'max_features': 20, 'min_samples_leaf': 1, 'n_estimators': 1024}

Spoken by: Ryan Cai



Results - Dataset 2a

From the plots, we can see that there are the same massive outliers throughout the experiment as the EEG data. We assume such outliers are caused by the breaks taken by subjects as mentioned in the description. For EOG data, there is significantly larger fluctuate between the second and third outlier than the rest part of time. From the plot of time over EEG-Fz of both training set and evaluation set, we can see that the EEG-Fz values for these two sets are complying with each other. It looks like the values are opposite against each other over time. However, for the logistic regression, the code did not work as expected, otherwise we can get better and more comprehensive results for the dataset 2a. Right now, we can only make the assumption that imaging and executing motor functions would result in activities in opposite cortex in brains.

Written by: Bingzhe Wang

Discussion





Discussion Classification



For our classification algorithms, we noticed that Decision Trees performed the best with the max depth being the max value over the list of possible parameters that we offered. We predict that this is due to the tendency for Decision Trees to overfit as well as by the noise in the EEG data. Besides that, we noticed that Logistic Regression was most successful with a L2 norm and regularization term of 10. We predict that this is due to the noisiness of the data having many outliers, an L2 norm limits the effects of the outliers. For Random Forest, having a forest of 1024 trees vs. 100 and 10 that was very deep (max depth of 21) performed much better likely due to the fitting to the noisiness of the data. Our mistake was instead of going from Data->Feature Extraction->Classification we did Data->Feature Extraction and Data->Classification.


Spoken by: Ryan Cai

Conclusion





Conclusion Classification



Under all metrics, we noticed that the scores were the best for the same or vary similar algorithm parameters which is surprising since that opposes the free lunch theorem. We believe this is again caused by the noisiness of the data, and the cleaning that we performed was insufficient.

In future projects, we will dedicate more time towards cleaning the data. We only transformed and scaled the data and also removed NAN but we found afterwards that it didn't clean it enough. We should also mix feature extraction and classification in a more orderly manner to achieve better results.

Written by: Ryan Cai



Conclusion Classification

The articles that inspired this project



“Hand Movement Direction Decoded from MEG and EEG” and “EEG-based discrimination between imagination of right and left hand movement” both used EEG data to predict hand movement, while “Hand Movement Direction Decoded from MEG and EEG” predicted 67% of small hand movements and could discriminate the cortical areas with at least 4 distinct neural activation patterns within them. “EEG-based discrimination between imagination of right and left hand movement” on the other hand used EEG data to see the difference in imagining hand movement and experiencing hand movement. They discovered that when you imagine hand movement both sides of the cortex are activated but when you experience hand movement then only one side of the cortex reacts to it.



“How capable is non-invasive EEG data of predicting the next movement? A mini review” and “Performance analysis of left/right hand movement classification from EEG signal by intelligent algorithms” both explored the ways in how the classification of EEG data could be improved. By developing BCIs around the before-movement, the resulting EEG data would allow for better prediction of the intended movement and thus make the movement seem more natural. In “Performance analysis of left/right hand movement classification from EEG signal by intelligent algorithms” we see that RBF kernelized SVM classifier indicated the highest performance accuracy of 82.14% with both original and reduced feature set. This project was done with the intent of seeing how EEG data would play a role in BCIs and how they could be improved for future use.



References

“Waldert, S., Preissl, H., Demandt, E., Braun, C., Birbaumer, N., Aertsen, A., & Mehring, C. (2008). Hand movement DIRECTION decoded from Meg and EEG. *Journal of Neuroscience*, 28(4), 1000-1008. doi:10.1523/jneurosci.5171-07.2008

Pfurtscheller, G., Neuper, C., Flotzinger, D., & Pregenzer, M. (1997). EEG-based discrimination between imagination of right and left hand movement. *Electroencephalography and Clinical Neurophysiology*, 103(6), 642-651. doi:10.1016/s0013-4694(97)00080-1

Ahmadian Pouya, Cagnoni Stefano, Ascari Luca, How capable is non-invasive EEG data of predicting the next movement? A mini review. *Frontiers in Human Neuroscience*, 7, 2013, pp. 1-124, doi: 10.3389/fnhum.2013.00124

S. Bhattacharyya, A. Khasnobish, A. Konar, D. N. Tibarewala and A. K. Nagar, "Performance analysis of left/right hand movement classification from EEG signal by intelligent algorithms," 2011 IEEE Symposium on Computational Intelligence, Cognitive Algorithms, Mind, and Brain (CCMB), Paris, France, 2011, pp. 1-8, doi: 10.1109/CCMB.2011.5952111.

Li, Zhen & Xu, Jianjun & Zhu, Tingshao. (2015). Recognition of Brain Waves of Left and Right Hand Movement Imagery with Portable Electroencephalographs.

Lee, W.H., Kim, E., Seo, H.G. et al. Target-oriented motor imagery for grasping action: different characteristics of brain activation between kinesthetic and visual imagery. *Sci Rep* 9, 12770 (2019). <https://doi.org/10.1038/s41598-019-49254-2>

C. Brunner , R. Leeb , G. R. Muller-Putz , A. Schlogl , and G. Pfurtscheller. “BCI Competition 2008 – Graz data set A”. Institute for Knowledge Discovery, Graz University of Technology, Austria and Institute for Human-Computer Interfaces, Graz University of Technology, Austria. http://www.bbci.de/competition/iv/desc_2a.pdf

The End

