Analysis of Multilabel Classification Machine Learning Algorithms on BCI Data

Miguel Garcia
University of California, San Diego
mig053@ucsd.edu

Abstract

Here in this review paper, I compare the performance of multilabel classification algorithms on BCI data and check the accuracies and anomalies when testing with these algorithms and how the data interacts with each other within and across subjects. I am also testing how you need to read the data and make sure it is properly cleaned and labeled before running the algorithms. It is important to note that it is not possible to test raw data itself without doing any changes to it. The first set of data sets which has labels that describe daily activities based on the recordings can be tested using multilabel classification machine learning algorithms. The second data set cannot be ran using multilabel classification algorithms (which predicts epilepsy based on EEG data); you can convert it to normal label but note that you cannot do that for any multilabel dataset.

1 Introduction

There have been quite a lot of problems when it comes to finding the relationship between machine learning algorithms and between data that got extracted using Brain Computer interfaces. BCI and Machine Learning are both relatively new fields that have been arising in the 21st century. Machine learning is a great tool when used for BCI data because it helps you determine use EEG data and any other data to develop useful BCI applications that anyone could use; note that it is important to make sure you can make generalizations across subjects to make sure the developed BCI can be universal. In this paper I analyze how that can be made and what algorithms perform better for this task. Machine learning interaction with BCI data can also help us find an effective way to predict diseases by using data from previous patients that were diagnosed with a specific condition or disease (epilepsy predicted from EEG data in this case) and also for the prediction of daily activities. I want to

see the relationship between multilabel machine learning algorithms on BCI data and also analyze what can be done to improve or to interact with the data. The motivation for this is to make BCI's universally available to multiple groups of people, but for that, you need to use efficient algorithms that could make the BCI universal like a lot of researchers intend to.

2 Method

2.1 Data Sets

In this section, you can see the summary of the classifiers and parameters used for each data set. You can also see how these data sets were cleaned and modify so that they can fit the classifiers. The data sets used can be found at the UCI Machine Learning Repository and all the details about each of them and which features they these have.

The first dataset [1] contains data that determines what daily activity is being performed based on measurements on the mounted accelerometer. The seven different activities are: working at computer, standing up walking and going up/down stairs, standing, walking, going up/down stairs, walking and talking with someone, and talking while standing. There is a relevant paper [2] that uses this data to develop a wearable computing platform to detect and monitor daily activities.

The second dataset [3] detects if there is epilepsy based on EEG recordings and my purpose is to test which algorithms perform better for prediction of epilepsy given a dataset taken with EEG BCI measurements.

The data sets were extracted using pandas data frames then convert them to list so that the classifier algorithms can handle them. The activity recognition from single chest mounted accelerometer data set contains 15 different data sets from 15 different subjects with about 100,000+ data points each with 4 attributes and 7 labels each. The epileptic seizure recognition data set [3] contains 11500 data points with 178 attributes with 5 labels.

I used only 6 out of the 15 subjects form the accelerometer data. I only ran the multilabel algorithms for 3 out these 6 and then I concatenated all 6 together and also ran the algorithm, then I checked if the algorithm with the 6 subjects was effective when testing each subject individually.

Before applying the algorithms to each data set, they were shuffled then divided into training and test samples for ‘x’ and ‘y’. The split is done as 80% of the data points for training and 20% for testing.

2.2 Classifiers Used

One vs rest classifier algorithm using linear support vector machine as given by scikit-learn [4].

Multioutput regressor with gradient boosting regressor as given by scikit-learn. [5]

Multioutput classifier with random forest classification as given by scikit-learn. [6]

Decision trees classifier algorithm as given by scikit-learn [7] with a ‘depth list’ that includes [1,2,3,4,5] using an entropy criterion. After that grid search with ten folds cross-validation.

Random forest classifier algorithm as given by scikit-learn [8] with a ‘depth list’ that includes [1,2,3,4,5] and has a maximum depth of five. Then grid search with ten-fold cross-validation is performed.

3 Experiment

In this section, I will run several experiments on the data sets with several classifiers. First, I will apply three multilabel classification algorithms per subject from the daily activity dataset. Then I will apply it to all subjects and compare the scores and performance across all subjects

Cleaning the data

1 Activity Recognition from Single Chest-Mounted Accelerometer Data Set, subjects 1 and 2

```
In [2]: acc_1 = pd.read_csv('1.csv')
        acc_1 = acc_1.drop(acc_1[acc_1['1']==0].index)
        accdummies_1 = pd.get_dummies(acc_1['1'], prefix = 'label')
        acc_1 = pd.concat([acc_1, accdummies_1], axis=1)
        acc_1 = acc_1.drop(['1'], axis=1)

In [3]: acc_2 = pd.read_csv('2.csv')
        acc_2 = acc_2.drop(acc_2[acc_2['1']==0].index)
        accdummies_2 = pd.get_dummies(acc_2['1'], prefix = 'label')
        acc_2 = pd.concat([acc_2, accdummies_2], axis=1)
        acc_2 = acc_2.drop(['1'], axis=1)
```

```
In [4]: #all subjects from 3-6
        #subject 3
        acc_3 = pd.read_csv('3.csv')
        acc_3 = acc_3.drop(acc_3[acc_3['1']==0].index)
        accdummies_3 = pd.get_dummies(acc_3['1'], prefix = 'label')
        acc_3 = pd.concat([acc_3, accdummies_3], axis=1)
        acc_3 = acc_3.drop(['1'], axis=1)

        #subject 4
        acc_4 = pd.read_csv('4.csv')
        acc_4 = acc_4.drop(acc_4[acc_4['1']==0].index)
        accdummies_4 = pd.get_dummies(acc_4['1'], prefix = 'label')
        acc_4 = pd.concat([acc_4, accdummies_4], axis=1)
        acc_4 = acc_4.drop(['1'], axis=1)

        #subject 5
        acc_5 = pd.read_csv('5.csv')
        acc_5 = acc_5.drop(acc_5[acc_5['1']==0].index)
        accdummies_5 = pd.get_dummies(acc_5['1'], prefix = 'label')
        acc_5 = pd.concat([acc_5, accdummies_5], axis=1)
        acc_5 = acc_5.drop(['1'], axis=1)

        #subject 6
        acc_6 = pd.read_csv('6.csv')
        acc_6 = acc_6.drop(acc_6[acc_6['1']==0].index)
        accdummies_6 = pd.get_dummies(acc_6['1'], prefix = 'label')
        acc_6 = pd.concat([acc_6, accdummies_6], axis=1)
        acc_6 = acc_6.drop(['1'], axis=1)
```

2 Clean and Split Subject #1:

```
In [5]: acc_1_XY = acc_1.iloc[:,0:11].values
        np.random.shuffle(acc_1_XY)

        acc_1_X = acc_1_XY[:,0:4]
        acc_1_Y = acc_1_XY[:,4:11]

        num_training = int(0.8*acc_1_X.shape[0])
        num_testing = int(0.2*acc_1_X.shape[0])

        acc_1_X_train = acc_1_X[:num_training]
        acc_1_Y_train = acc_1_Y[:num_training]
        acc_1_X_test = acc_1_X[num_training:]
        acc_1_Y_test = acc_1_Y[num_training:]

        print(acc_1_X_train.shape)
        print(acc_1_Y_train.shape)


         print(acc_1_X_test.shape)
         print(acc_1_Y_test.shape)
```

```
(129999, 4)
(129999, 7)
(32500, 4)
(32500, 7)
```

3 Clean and Split Subject #2:

```
In [6]: acc_2_XY = acc_2.iloc[:,0:11].values
        np.random.shuffle(acc_2_XY)

        acc_2_X = acc_2_XY[:,0:4]
        acc_2_Y = acc_2_XY[:,4:11]

        num_training = int(0.8*acc_2_X.shape[0])
        num_testing = int(0.2*acc_2_X.shape[0])

        acc_2_X_train = acc_2_X[:num_training]
        acc_2_Y_train = acc_2_Y[:num_training]
        acc_2_X_test = acc_2_X[num_training:]
        acc_2_Y_test = acc_2_Y[num_training:]

        print(acc_2_X_train.shape)
        print(acc_2_Y_train.shape)
        print(acc_2_X_test.shape)
        print(acc_2_Y_test.shape)
```

```
(110184, 4)
(110184, 7)
(27546, 4)
(27546, 7)
```

4 Clean and Split Subject 1+2:

```
In [7]: acc_12_XY = np.concatenate((acc_1_XY, acc_2_XY))
        np.random.shuffle(acc_12_XY)

        acc_12_X = acc_12_XY[:,0:4]
        acc_12_Y = acc_12_XY[:,4:11]

        num_training = int(0.8*acc_12_X.shape[0])
        num_testing = int(0.2*acc_12_X.shape[0])
```

```
        acc_12_X_train = acc_12_X[:num_training]
        acc_12_Y_train = acc_12_Y[:num_training]
        acc_12_X_test = acc_12_X[num_training:]
        acc_12_Y_test = acc_12_Y[num_training:]

        print(acc_12_X_train.shape)
        print(acc_12_Y_train.shape)
        print(acc_12_X_test.shape)
        print(acc_12_Y_test.shape)
```

```
(240183, 4)
(240183, 7)
(60046, 4)
(60046, 7)
```

5 Clean and Split Subject 1-6:

```
In [8]: acc_3_XY = acc_3.iloc[:,0:11].values
        acc_4_XY = acc_4.iloc[:,0:11].values
        acc_5_XY = acc_5.iloc[:,0:11].values
        acc_6_XY = acc_6.iloc[:,0:11].values
        acc_all_XY = np.concatenate((acc_1_XY, acc_2_XY,acc_3_XY, acc_4_XY,acc_5_XY, acc_6_XY)
        np.random.shuffle(acc_all_XY)

        acc_all_X = acc_all_XY[:,0:4]
        acc_all_Y = acc_all_XY[:,4:11]

        num_training = int(0.8*acc_all_X.shape[0])
        num_testing = int(0.2*acc_all_X.shape[0])

        acc_all_X_train = acc_all_X[:num_training]
        acc_all_Y_train = acc_all_Y[:num_training]
        acc_all_X_test = acc_all_X[num_training:]
        acc_all_Y_test = acc_all_Y[num_training:]

        print(acc_all_X_train.shape)
        print(acc_all_Y_train.shape)
        print(acc_all_X_test.shape)
        print(acc_all_Y_test.shape)
```

```
(660348, 4)
(660348, 7)
(165087, 4)
(165087, 7)
```

3.1 One vs All Classifier

6 One vs all subject 1

```
In [9]: clf_onevsrest_1 = OneVsRestClassifier(LinearSVC(random_state=0,max_iter=100))

In [10]: clf_onevsrest_1.fit(acc_1_X_train, acc_1_Y_train)

Out[10]: OneVsRestClassifier(estimator=LinearSVC(C=1.0, class_weight=None, dual=True, fit_inte
              intercept_scaling=1, loss='squared_hinge', max_iter=100,
              multi_class='ovr', penalty='l2', random_state=0, tol=0.0001,
              verbose=0),
                   n_jobs=None)

In [11]: clf_onevsrest_1.score(acc_1_X_test,acc_1_Y_test)

Out[11]: 0.72283076923076928
```

7 One vs all subject 2

```
In [12]: clf_onevsrest_2 = OneVsRestClassifier(LinearSVC(random_state=0,max_iter=100))

In [13]: clf_onevsrest_2.fit(acc_2_X_train, acc_2_Y_train)

Out[13]: OneVsRestClassifier(estimator=LinearSVC(C=1.0, class_weight=None, dual=True, fit_inte
              intercept_scaling=1, loss='squared_hinge', max_iter=100,
              multi_class='ovr', penalty='l2', random_state=0, tol=0.0001,
              verbose=0),
                   n_jobs=None)

In [14]: clf_onevsrest_2.score(acc_2_X_test,acc_2_Y_test)

Out[14]: 0.5634574892906411
```

8 One vs all subject 1+2

```
In [15]: clf_onevsrest_12 = OneVsRestClassifier(LinearSVC(random_state=0,max_iter=100))

In [16]: clf_onevsrest_12.fit(acc_12_X_train, acc_12_Y_train)

Out[16]: OneVsRestClassifier(estimator=LinearSVC(C=1.0, class_weight=None, dual=True, fit_inte
              intercept_scaling=1, loss='squared_hinge', max_iter=100,
              multi_class='ovr', penalty='l2', random_state=0, tol=0.0001,
              verbose=0),
                   n_jobs=None)

In [17]: clf_onevsrest_12.score(acc_12_X_test,acc_12_Y_test)

Out[17]: 0.61241381607434298
```

9 One vs all subjects 1-6

```
In [18]: clf_onevsrest_all = OneVsRestClassifier(LinearSVC(random_state=0,max_iter=100))

In [19]: clf_onevsrest_all.fit(acc_all_X_train, acc_all_Y_train)

Out[19]: OneVsRestClassifier(estimator=LinearSVC(C=1.0, class_weight=None, dual=True, fit_inte
             intercept_scaling=1, loss='squared_hinge', max_iter=100,
             multi_class='ovr', penalty='l2', random_state=0, tol=0.0001,
             verbose=0),
                  n_jobs=None)

In [20]: clf_onevsrest_all.score(acc_all_X_test,acc_all_Y_test)

Out[20]: 0.16590646144154295
```

10 One vs all across subjects test

```
In [21]: clf_onevsrest_1.score(acc_2_X_test,acc_2_Y_test) #subject 1's classifier

Out[21]: 0.46801713497422492

In [22]: clf_onevsrest_2.score(acc_1_X_test,acc_1_Y_test) #subject 2's classifier

Out[22]: 0.59747692307692313

In [23]: clf_onevsrest_12.score(acc_1_X_test,acc_1_Y_test) #subject 1+2's classifier test for

Out[23]: 0.69012307692307695

In [24]: clf_onevsrest_12.score(acc_2_X_test,acc_2_Y_test) #subject 1+2's classifier test for

Out[24]: 0.52541203804545122

In [25]: clf_onevsrest_all.score(acc_1_X_test,acc_1_Y_test) #subject 1-6's classifier test for

Out[25]: 0.1451076923076923

In [26]: clf_onevsrest_all.score(acc_2_X_test,acc_2_Y_test) #subject 1-6's classifier test for

Out[26]: 0.16172947070355043
```

3.1.1 One vs All Classifier Performance Summary

In here you can see how one vs all test accuracy for each subject individually it varied, then when we put the subjects 1 and 2 together it did not show an increase in accuracy. To verify that data size is not a factor to take into consideration for the effectiveness of this algorithm, I trained it with 6 subjects which includes a greater amount of data points and still, the testing accuracy did not seem to increase at all, which shows how ineffective one vs all was for this data.

I then calculated the accuracy using subject 1's classifier testing with subject 2's data and similarly the other way around; accuracies turned out bad which shows how it is not accurate to predict/monitor daily activities across different subjects using this specific algorithm. The same results came when I trained the data from subject 1 and 2 concatenated and predicted the labels individually for each subject; accuracy was low. I then added all the data from subjects 1-6 and tried seeing if the data size would help with predicting any subject's labels but it turned out even worse. Increasing the data size would just make the algorithm more and more ineffective.

I wanted to make sure that it is not the data that itself causing this, so for that I trained more algorithms shown below. The accuracies reported by one vs all are really low and would not be good at all if someone were to make a BCI in charge of monitoring daily activity with just the use of this dataset.

3.2 Multioutput Regressor Classifier

11 Multioutput Regressor subject 1

```
In [27]: clf_multir_1 = MultiOutputRegressor(GradientBoostingRegressor(random_state=0))

In [28]: clf_multir_1.fit(acc_1_X_train, acc_1_Y_train)
```

```
Out[28]: MultiOutputRegressor(estimator=GradientBoostingRegressor(alpha=0.9, criterion='friedm
                     learning_rate=0.1, loss='ls', max_depth=3, max_features=None,
                     max_leaf_nodes=None, min_impurity_decrease=0.0,
                     min_impurity_split=None, min_samples_leaf=1,
                     min_sampl...ate=0, subsample=1.0, tol=0.0001,
                     validation_fraction=0.1, verbose=0, warm_start=False),
                    n_jobs=None)

In [29]: clf_multir_1.score(acc_1_X_test,acc_1_Y_test)

Out[29]: 0.99880946821959904
```

12 Multioutput Regressor subject 2

```
In [30]: clf_multir_2 = MultiOutputRegressor(GradientBoostingRegressor(random_state=0))

In [31]: clf_multir_2.fit(acc_2_X_train, acc_2_Y_train)

Out[31]: MultiOutputRegressor(estimator=GradientBoostingRegressor(alpha=0.9, criterion='friedm
                      learning_rate=0.1, loss='ls', max_depth=3, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_sampl...ate=0, subsample=1.0, tol=0.0001,
                      validation_fraction=0.1, verbose=0, warm_start=False),
                     n_jobs=None)

In [32]: clf_multir_2.score(acc_2_X_test,acc_2_Y_test)

Out[32]: 0.99985221837346305
```

13 Multioutput Regressor subject 1+2

```
In [33]: clf_multir_12 = MultiOutputRegressor(GradientBoostingRegressor(random_state=0))

In [34]: clf_multir_12.fit(acc_12_X_train, acc_12_Y_train)

Out[34]: MultiOutputRegressor(estimator=GradientBoostingRegressor(alpha=0.9, criterion='friedm
                      learning_rate=0.1, loss='ls', max_depth=3, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_sampl...ate=0, subsample=1.0, tol=0.0001,
                      validation_fraction=0.1, verbose=0, warm_start=False),
                     n_jobs=None)

In [35]: clf_multir_12.score(acc_12_X_test,acc_12_Y_test)

Out[35]: 0.96952963475614329
```

14 Multioutput Regressor subjects 1-6

```
In [36]: clf_multir_all = MultiOutputRegressor(GradientBoostingRegressor(random_state=0))

In [37]: clf_multir_all.fit(acc_all_X_train, acc_all_Y_train)

Out[37]: MultiOutputRegressor(estimator=GradientBoostingRegressor(alpha=0.9, criterion='friedm
                       learning_rate=0.1, loss='ls', max_depth=3, max_features=None,
                       max_leaf_nodes=None, min_impurity_decrease=0.0,
                       min_impurity_split=None, min_samples_leaf=1,
                       min_sampl...ate=0, subsample=1.0, tol=0.0001,
                       validation_fraction=0.1, verbose=0, warm_start=False),
                      n_jobs=None)

In [38]: clf_multir_all.score(acc_all_X_test,acc_all_Y_test)

Out[38]: 0.79816242496197176
```

15 Multioutput Regressor across subjects

```
In [39]: clf_multir_1.score(acc_2_X_test,acc_2_Y_test) #subject 1's classifier

Out[39]: -0.3683884356620219

In [40]: clf_multir_2.score(acc_1_X_test,acc_1_Y_test) #subject 2's classifier

Out[40]: -1.3919221771553325

In [41]: clf_multir_12.score(acc_1_X_test,acc_1_Y_test) #subject 1+2's classifier on subject 1

Out[41]: 0.96783703488383743

In [42]: clf_multir_12.score(acc_2_X_test,acc_2_Y_test) #subject 1+2's classifier on subject 2

Out[42]: 0.97000736457243775

In [43]: clf_multir_all.score(acc_1_X_test,acc_1_Y_test) #subject 1-6's classifier on subject

Out[43]: 0.56883928752442858

In [44]: clf_multir_all.score(acc_2_X_test,acc_2_Y_test) #subject 1-6's classifier on subject

Out[44]: 0.89462606713411119
```

3.2.1 Multioutput Regressor Performance Summary

In here you can see how the accuracy for the multioutput regressor started decreasing as the data size got larger. Note that for subject 2 the accuracy was higher as the data set size was smaller. When we combined the first 2 subjects the accuracy decreased, and similarly when combining subjects 1-6. Using multioutput regressor would obviously not be good for generalizing a BCI application that uses this data since it significantly decreases the more subjects and data you add to the algorithm. A BCI like the one developed here: https://www.researchgate.net/publication/257132489_BeaStreamer-v01_a_new_platform_for_Multi-Sensors_Data_Acquisition_in_Wearable_Computing_Applications would be a complete failure assuming they were to use multilabel algorithms. But there are still so many algorithms for multilabel classification available at our disposal that we could still try.

When you combine the data from subjects one and two and try to test the accuracy for each subject individually it actually gives a good percentage but when you combine all 6 subjects it gives an incredibly low percentage. When using subject 1's classifier to test for subject 2's data, and similarly the other way around, it does not give a logical value when calculating the accuracy.

To make a BCI application off this data, we want to make sure that we can generalize the accuracies for all subjects and also be sure that adding more data yields higher accuracies instead of the other way around, otherwise, any attempt to create an application to monitor or detect daily activities would be obsolete and would not generally work for any subject that decides to try this BCI on.

3.3 Multioutput Classifier with Random Forest

16 Multioutput classifier with forest subject 1

```
In [45]: forest_1 = RandomForestClassifier(n_estimators=100, random_state=1)

In [46]: clf_mtf_1 = MultiOutputClassifier(forest_1, n_jobs=-1)

In [47]: clf_mtf_1.fit(acc_1_X_train, acc_1_Y_train)

Out[47]: MultiOutputClassifier(estimator=RandomForestClassifier(bootstrap=True, class_weight=N
                     max_depth=None, max_features='auto', max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
                     oob_score=False, random_state=1, verbose=0, warm_start=False),
                    n_jobs=-1)

In [48]: clf_mtf_1.score(acc_1_X_test,acc_1_Y_test)

Out[48]: 0.99993846153846155
```

17 Multioutput classifier with forest subject 2

```
In [49]: forest_2 = RandomForestClassifier(n_estimators=100, random_state=1)

In [50]: clf_mtf_2 = MultiOutputClassifier(forest_2, n_jobs=-1)

In [51]: clf_mtf_2.fit(acc_2_X_train, acc_2_Y_train)

Out[51]: MultiOutputClassifier(estimator=RandomForestClassifier(bootstrap=True, class_weight=N
                     max_depth=None, max_features='auto', max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
                     oob_score=False, random_state=1, verbose=0, warm_start=False),
                    n_jobs=-1)

In [52]: clf_mtf_2.score(acc_2_X_test,acc_2_Y_test)

Out[52]: 0.99996369708850652
```

18 Multioutput classifier with forest subject 1+2

```
In [53]: forest_12 = RandomForestClassifier(n_estimators=100, random_state=1)

In [54]: clf_mtf_12 = MultiOutputClassifier(forest_12, n_jobs=-1)

In [55]: clf_mtf_12.fit(acc_12_X_train, acc_12_Y_train)
```

```
Out[55]: MultiOutputClassifier(estimator=RandomForestClassifier(bootstrap=True, class_weight=N
                    max_depth=None, max_features='auto', max_leaf_nodes=None,
                    min_impurity_decrease=0.0, min_impurity_split=None,
                    min_samples_leaf=1, min_samples_split=2,
                    min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
                    oob_score=False, random_state=1, verbose=0, warm_start=False),
                   n_jobs=-1)

In [56]: clf_mtf_12.score(acc_12_X_test,acc_12_Y_test)

Out[56]: 0.99880091929520698
```

19 Multioutput classifier with forest subjects 1-6

```
In [57]: forest_all = RandomForestClassifier(n_estimators=100, random_state=1)

In [58]: clf_mtf_all = MultiOutputClassifier(forest_all, n_jobs=-1)

In [59]: clf_mtf_all.fit(acc_all_X_train, acc_all_Y_train)

Out[59]: MultiOutputClassifier(estimator=RandomForestClassifier(bootstrap=True, class_weight=N
                    max_depth=None, max_features='auto', max_leaf_nodes=None,
                    min_impurity_decrease=0.0, min_impurity_split=None,
                    min_samples_leaf=1, min_samples_split=2,
                    min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
                    oob_score=False, random_state=1, verbose=0, warm_start=False),
                   n_jobs=-1)

In [60]: clf_mtf_all.score(acc_all_X_test,acc_all_Y_test)

Out[60]: 0.98362681495211612
```

20 Multioutput classifier with forest acros subjects

```
In [61]: clf_mtf_1.score(acc_2_X_test,acc_2_Y_test) #subject 1's classifier

Out[61]: 0.5371741813693458

In [62]: clf_mtf_2.score(acc_1_X_test,acc_1_Y_test) #subject 2's classifier

Out[62]: 0.58999999999999997

In [63]: clf_mtf_12.score(acc_1_X_test,acc_1_Y_test) #subject 1+2's classifier on subject 1

Out[63]: 0.99978461538461538

In [64]: clf_mtf_12.score(acc_2_X_test,acc_2_Y_test) #subject 1+2's classifier on subject 2

Out[64]: 0.99974587961954553

In [65]: clf_mtf_all.score(acc_1_X_test,acc_1_Y_test) #subject 1-6's classifier on subject 1

Out[65]: 0.99824615384615389

In [66]: clf_mtf_all.score(acc_2_X_test,acc_2_Y_test) #subject 1-6's classifier on subject 2

Out[66]: 0.99829376315980545
```

3.3.1 Multioutput Classifier with Random Forest Performance Summary

When using this classifier, we can definitely note the change when we increase the size of the data size or when we add data from more subjects into the classifier the accuracy stays the same/increases and you can use the concatenation of all the subjects to effectively calculate the test accuracy for other individual subjects. Note that in order to effectively adapt a subject to a potential BCI designed based on that data, we need to record some data in order to add it to the algorithm so the effectiveness increases, otherwise the accuracy when predicting a new subject using an algorithm from data of previous subjects the accuracy will be an estimate of 75%-80% which is not fully effective. So, in order to design a BCI that predicts and monitors daily activities, it is necessary to make any new subject that wants to wear it go through a process where we collect the data and add it to the classifier. Once we add the data to the classifier the effectiveness increases over the long run, as seen by the accuracies when adding the data from multiple subjects into the classifier.

3.4 Epileptic Seizure Recognition Data Set from EEG Recordings

Now I will apply multilabel classification algorithms on the data here which will not be as effective since the labels are ambiguous and seem to have nonlinear patterns and the matrices are not properly organized to fit in the algorithm, but I will still try running the multilabel algorithms.

A full description of the data set can be found on the website provided on the reference section including the description of the labels.

3.4.1 Cleaning the data

21 Epileptic Seizure Recognition Data Set from EEG Recordings

```
In [67]: eeg = pd.read_csv('data.csv')
         eeg = eeg.drop(['Unnamed: 0'], axis=1)

In [68]: eeg_dummies = pd.get_dummies(eeg['y'], prefix = 'label')
         eeg = pd.concat([eeg, eeg_dummies], axis=1)
         eeg = eeg.drop(['y'], axis=1)

In [69]: eeg_XY = eeg.iloc[:,0:183].values
         np.random.shuffle(eeg_XY)

         eeg_X = eeg_XY[:,0:178]
         eeg_Y = eeg_XY[:,178:183]

         num_training = int(0.8*eeg_X.shape[0])
         num_testing = int(0.2*eeg_X.shape[0])

         eeg_X_train = eeg_X[:num_training]
         eeg_Y_train = eeg_Y[:num_training]
         eeg_X_test = eeg_X[num_training:]
         eeg_Y_test = eeg_Y[num_training:]

         print(eeg_X_train.shape)
         print(eeg_Y_train.shape)
         print(eeg_X_test.shape)
         print(eeg_Y_test.shape)

(9200, 178)
(9200, 5)
(2300, 178)
(2300, 5)
```

3.4.2 One vs All classifier

22 One vs All on EEG data

```
In [70]: clf_onevsrest_eeg = OneVsRestClassifier(LinearSVC(random_state=0,max_iter=100))

In [71]: clf_onevsrest_eeg.fit(eeg_X_train, eeg_Y_train)

Out[71]: OneVsRestClassifier(estimator=LinearSVC(C=1.0, class_weight=None, dual=True, fit_inte
              intercept_scaling=1, loss='squared_hinge', max_iter=100,
              multi_class='ovr', penalty='l2', random_state=0, tol=0.0001,
              verbose=0),
                   n_jobs=None)

In [72]: clf_onevsrest_eeg.score(eeg_X_test,eeg_Y_test)

Out[72]: 0.048695652173913043
```

3.4.3 Multioutput Regressor classifier

23 Multioutput Regressor on EEG data

```
In [73]: clf_multir_eeg = MultiOutputRegressor(GradientBoostingRegressor(random_state=0))

In [74]: clf_multir_eeg.fit(eeg_X_train, eeg_Y_train)

Out[74]: MultiOutputRegressor(estimator=GradientBoostingRegressor(alpha=0.9, criterion='friedm
                      learning_rate=0.1, loss='ls', max_depth=3, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_sampl...ate=0, subsample=1.0, tol=0.0001,
                      validation_fraction=0.1, verbose=0, warm_start=False),
                     n_jobs=None)

In [75]: clf_multir_eeg.score(eeg_X_train,eeg_Y_train)

Out[75]: 0.39039304545943432
```

3.4.4 Multioutput Classifier with Random Forest

24 Multioutput classifier

```
In [76]: forest_eeg = RandomForestClassifier(n_estimators=100, random_state=1)

In [77]: clf_mtf_eeg = MultiOutputClassifier(forest_eeg, n_jobs=-1)

In [78]: clf_mtf_eeg.fit(eeg_X_train,eeg_Y_train)

Out[78]: MultiOutputClassifier(estimator=RandomForestClassifier(bootstrap=True, class_weight=N
                     max_depth=None, max_features='auto', max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
                     oob_score=False, random_state=1, verbose=0, warm_start=False),
                    n_jobs=-1)

In [79]: clf_mtf_eeg.score(eeg_X_test,eeg_Y_test)

Out[79]: 0.30478260869565216
```

3.4.5 Multilabel Algorithm Classifiers Performance Summary

The reason multilabel classification algorithms did not work at all is because the ambiguity within the labels with the raw data. Here is a description so that it makes sense why the algorithms would not work:

The response variable is y in column 179, the Explanatory variables X1, X2, ..., X178

y contains the category of the 178-dimensional input vector. Specifically y in {1, 2, 3, 4, 5}:

5 - eyes open, means when they were recording the EEG signal of the brain the patient had their eyes open

4 - eyes closed, means when they were recording the EEG signal the patient had their eyes closed

3 - Yes they identify where the region of the tumor was in the brain and recording the EEG activity from the healthy brain area

2 - They recorder the EEG from the area where the tumor was located

1 - Recording of seizure activity

But all subjects within classes 2,3,4,5 do not have an epileptic seizure, so it is definitely possible to convert the labels to single label (0 and 1) and to use this classifier to predict if any given patient has an epileptic seizure based on the raw EEG data.

3.4.5 Reducing Dimensionality of the Labels

25 Convert the data to single label

```
In [80]: eeg = pd.read_csv('data.csv')
         eeg = eeg.drop(['Unnamed: 0'], axis=1)
         eeg['y'] = eeg['y'].replace([2,3,4,5], 0)

In [81]: eeg_XY = eeg.iloc[:,0:179].values
         np.random.shuffle(eeg_XY)

         eeg_X = eeg_XY[:,0:178]
         eeg_Y = eeg_XY[:,178:179]



        num_training = int(0.8*eeg_X.shape[0])
        num_testing = int(0.2*eeg_X.shape[0])

        eeg_X_train = eeg_X[:num_training]
        eeg_Y_train = eeg_Y[:num_training]
        eeg_X_test = eeg_X[num_training:]
        eeg_Y_test = eeg_Y[num_training:]

        print(eeg_X_train.shape)
        print(eeg_Y_train.shape)
        print(eeg_X_test.shape)
        print(eeg_Y_test.shape)
```

```
(9200, 178)
(9200, 1)
(2300, 178)
(2300, 1)
```

3.4.6 Random Forest with 10-fold cross validation

26 Apply Random Forest with cross validation

```
In [82]: depth_list = [1, 2, 3, 4, 5]
         params = {"max_depth": depth_list}
         classifier = RandomForestClassifier(max_depth=5, random_state=0)
         grid_rf_eeg = GridSearchCV(classifier, params,return_train_score = True, cv = 10)
         grid_rf_eeg.fit(eeg_X_train, eeg_Y_train)

Out[82]: GridSearchCV(cv=10, error_score='raise-deprecating',
                estimator=RandomForestClassifier(bootstrap=True, class_weight=None, criterion=
                     max_depth=5, max_features='auto', max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators='warn', n_jobs=None,
                     oob_score=False, random_state=0, verbose=0, warm_start=False),
                fit_params=None, iid='warn', n_jobs=None,
                param_grid={'max_depth': [1, 2, 3, 4, 5]}, pre_dispatch='2*n_jobs',
                refit=True, return_train_score=True, scoring=None, verbose=0)

In [83]: grid_rf_eeg.score(eeg_X_test,eeg_Y_test)

Out[83]: 0.94478260869565223
```

3.4.7 Decision Tree with 10-fold cross validation

27 Apply Decision tree with cross validation

```
In [84]: classifier = tree.DecisionTreeClassifier(criterion='entropy')
         depth_list = [1, 2, 3, 4, 5]


             params = {"max_depth": depth_list}
             grid_dt_eeg = GridSearchCV(classifier, params,return_train_score = True, cv = 10)
             grid_dt_eeg.fit(eeg_X_train, eeg_Y_train)

Out[84]: GridSearchCV(cv=10, error_score='raise-deprecating',
                estimator=DecisionTreeClassifier(class_weight=None, criterion='entropy', max_d
                     max_features=None, max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                     splitter='best'),
                fit_params=None, iid='warn', n_jobs=None,
                param_grid={'max_depth': [1, 2, 3, 4, 5]}, pre_dispatch='2*n_jobs',
                refit=True, return_train_score=True, scoring=None, verbose=0)

In [85]: grid_dt_eeg.score(eeg_X_test,eeg_Y_test)

Out[85]: 0.91217391304347828
```

3.4.7 Single Label Algorithm Classifiers Performance Summary

As you can see, it was indeed possible to convert the data to single label because it is specifically specified which labels are the ones that belong to an epileptic seizure diagnostic; it would not make sense to know each label individually as they are theoretically irrelevant and it is also not the goal to know those specific labels individually when conducting an EEG on a patient, what truly matters is to know if the patient has an epileptic seizure.
Random forest gave the highest performance, so it would probably be a good choice to use this algorithm when trying to predict if a patient has an epileptic seizure based on an EEG raw recording. It is important to perform recordings of multiple patients and add the data to the classifier because it improves accuracy over time; as I mentioned before.

4 Conclusion/Discussion

Comparing two different kinds of data sets gave a good observation on the most effective and recommended way to analyze data that can potentially be used to create BCI's using these. Essentially the goal would definitely be to increase accuracy and performance in the long run (as we add more data) but in order to accomplish that we need to do an analysis on which algorithms work better with each data set; we cannot just say that one algorithm should always be used in all cases; we have to make sure we test each and every single one but not just for one scenario; we need to make sure the matrices follow a linearity and don't have any anomalies that could make it a challenge or even impossible, to generalize the data for multiple and new subjects that start trying the BCI. It was interesting seeing how it would be possible to monitor daily activities just with the use of recordings from a chest mounted accelerometer; as I mentioned before, there is a paper that talks about more in detail about a BCI that was developed with the use of this data. Then I came across a dataset which contained EEG recordings with 5 different labels that had an ambiguous and nonsense description; even though leaving it just as it was definitely not possible, changing it to single label actually gave really good results as expected. After training with single label algorithms, it was definitely possible to have a classifier that can effectively predict if a patient had an epileptic seizure just with the use of BCI recordings from EEG data. Both these fields (BCI and ML) have been arising and getting improved this 21st century and it is important to know how to combine them together to achieve a higher goal and more effectiveness for BCI use, which was the main motivation of this paper. In order to make this even more efficient, we need to train and add more data over time; we need to take way more measurements and add them to the classifier over time. Another way to improve this would be by running more analysis of different kinds that analyze specific patterns and anomalities in the data using data science techniques so that it is easier to understand how the data functions in order to come up with more efficient approaches to making a dataset more efficient when used in a BCI system.

5 References/Related Work from Class

[1] Casale, P. Pujol, O. and Radeva, P (2012). UCI Machine Learning Repository – Activity Recognition from Single Chest-Mounted Accelerometer Data Set [https://archive.ics.uci.edu/ml/datasets/Activity+Recognition+from+Single+Chest-Mounted+Accelerometer]. Personal and Ubiquitous Computing. 16(5), 563-580, 2012

[2] Casale, Pierluigi & Pujol, Oriol & Radeva, Petia. (2009). BeaStreamer-v0.1: a new platform for Multi-Sensors Data Acquisition in Wearable Computing Applications. [https://www.researchgate.net/publication/257132489_BeaStreamer-v01_a_new_platform_for_Multi-Sensors_Data_Acquisition_in_Wearable_Computing_Applications]

[3] Andrzejak RG, Lehnertz K, Rieke C, Mormann F, David P, Elger CE (2001). UCI Machine Learning Repository - Epileptic Seizure Recognition Data Set, [https://archive.ics.uci.edu/ml/datasets/Epileptic+Seizure+Recognition]. Phys. Rev. E, 64, 061907

[4] Scikit-learn: OneVsRest Classifier. [https://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html].

[5] Scikit-learn: Multioutput Regressor. [https://scikit-learn.org/stable/modules/generated/sklearn.multioutput.MultiOutputRegressor.html].

[6] Scikit-learn: Multioutput Classifier. [https://scikit-learn.org/stable/modules/generated/sklearn.multioutput.MultiOutputRegressor.html].

[7] Scikit-learn: DecisionTree Classifier. [https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html].

[8] Scikit-learn: Random Forest Classifier (2018). [https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html].

[9] Scikit-learn: GridSearchCV (2018). [https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html].

[10] Vernon J. Lawhern, Amelia J. Solon, Nicholas R. Waytowich, Stephen M. Gordon, Chou P. Hung, Brent J. Lance (2016). EEGNet: A Compact Convolutional Network for EEG-based Brain-Computer Interfaces. arXiv:1611.08024 (CLASS READING

[11] Mahta Mousavi, Adam S. Koerner, Qiong Zhang, Eunho Noh & Virginia R. de Sa (2017): Improving motor imagery BCI with user response to feedback, Brain-Computer Interfaces, DOI: 10.1080/2326263X.2017.1303253