Machine learning based EEG classification

Christian Miranda

***Abstract*—The focus of this paper is looking into the use of machine learning to perform classification tasks involving EEG signals. The data being used contains EEG recordings along with their corresponding label. So different classifiers were used to to classify said data, to its correct label. Various tools and different classifier parameters are used to seek out and compare which ones perform the best.**

1. Introduction

Using machine learning to make predictions and classify data is not a new concept, but is one that warrants further exploration. A few examples of where machine learning is used in is with self-driving cars, facial recognition, and apps that recognize your taste in music or movies. A very interesting use in the medical field is creating an artificial pancreas that uses machine learning [1] to make predictions in how much insulin to release into the wearer. The University of California Irvine had conducted an experiment with EEG signals [2], having subjects stay in a state of rest followed by them planning to do a voluntary movement with their right hand. UCI used various methods to interpret the EEG signals and in the end, classify them into one of the two labels, asleep or planning. The methods used in this work were classifiers from Scikit Learn to work on UCI's dataset to attempt to achive similar or better results. Multiple classifiers were used in order to observe which one performed the best in regards to accuracy. Some of the classifiers used here are similar to ones used by UCI and others that have used the same dataset [3] which, the results to said tests, will be compared to the results of this report.

2. Machine Learning Basics

The algorithms find trends or patterns in the data and use them to make its decision-making but have to be trained first before it's able to output any results. The patterns that it looks for is all based on the preliminary data that it is exposed to, to train itself. These algorithms are powerful tools due to the fact that the program doesn't work just by storing and retrieving data, it has a "learning" nature and can "think" to predict future results with the data that it is given.

One task of these machine learning algorithms is classification, using them to classify the giving data based on the class's or "labels" from the training data.

3. EEG Basics

The EEG signals are expressd in time and frequency, gathered from electrodes connected to a subjects head. The recording capture the waves, their amplitudes being the frequency, emitted from the subjects brain. When it comes to EEG test in sleep, classifing the sleep stages of a subject is common goal in many experiments. There are five stages of sleep, stage 1, 2, 3, 4, and REM (rapid eye movement). When classifing the sleep stages, indicators have to be present in the data, which indicate what sleep state in the data [4]. These indicators for sleep stage classification are sets of waves at certain frequencies. A graphical representation of various sleep stages was done in an experiment, resulting in a base to compare to the results of automated programs that seek to perform the same results. Alpha waves of bandwidth 8 to 12 Hz, relate to state of being awake. Theta waves of range to 4 to 8 Hz are of stage 1, 2 and, REM sleep. Delta waves of 0.5 to 4 Hz are of stages 3 and 4. Lastly, the K complex indicator are slow transient waves that are related to stage 2.

The waves of interest in this work are Mu waves that are located in the band $7-13$Hz and Beta waves are at 13 Hz and above [2]. Studies have shown that MU waves were related to planning body movement [2]. The UCI tests dealt with EEG signals recorded from the subjects participating in the experiment. UCI's dataset, which consists of 182 subsets and 13 attributes [2], came from the EEG signals. 12 of those attributes are the features and the last attribute is the label of the subset [5]. The subjects were asked to sleep for five minutes, then wake them up, asking them to plan a movement for the following five seconds [2]. This was repeated over the course of about 30 minutes. Each subset is a five second fragment of the tests.

4. Preprocessing

The EEG signals had to go through a transformation process before being used. The features of the subsets were extracted from the EEG signals using a wavelet packet transform [5]. Coefficients are produced by the transform, called wavelets [6]. These wavelet coefficients are treated as "features" that describe the original signal. The wavelet transform is similar to the Fourier Transform, so the features contain similar time-frequency characteristics as the FT [7]. The features ultimately would describe where the subsets fall in the $7-13$Hz band that gives insight into whether the subject is relaxed or planning a movement.

5. Materials and Methods

We used the following processing pipeline for obtaining the results:

1) NumPy
2) SciPy
3) Scikit-learn

Scikit-learn is a Python-based machine learning library [8] that contains many tools and algorithms that are used to make predictions when introduced to new data.

The raw dataset was extracted from UCI's repository as a text file, and by using a Python script, the data was able to be expressed as numerical arrays, ready for use by the Scikit Learn library.

5.1. K-Nearest Neighbor

The first classification method used was the k-Nearest Neighbor algorithm. This algorithm works by, in newly-introduced data, it recognizes patterns according to data around certain data points [9]. The k-NN models use a user-defined parameter, n, that defines the number of closest training examples. The neighbors weigh in, in the decision of which class that the data points will fall under.

5.2. Support Vector Machines

The second classifier that is being used is the Support Vector Machine. SVM models represent the data that was used to train itself, as a hyperplane [2]. The points of a specific class are mapped out as a group with gaps dividing all groups. So when the SVM is ran with the testing data, the data will fall under one of the groups.

5.3. Random Forest

The last model used for classification is the Random Forest classifier. Random forest method is a type of Ensemble learning that uses a combination of several methods that make predictions and at the end, all the methods are used to make one final major prediction [9]. When training the random forest model, it randomly creates "trees" using the training data. These trees are called decision trees due to the fact that they are made up of multiple decisions that are used to classify the data each tree is presented with. This is one of the user-defined parameters , n estimators, that tells the classifier how many trees to make [8]. Another parameter is "max depth," which defines how many nodes are in each trees [8]. Apparently, many of these decision-making trees are not very helpful on their own. So the way that the classifier works is that among the many bad trees there are still few that are very useful. When the classifer is making a prediction, the date gets passed through all the trees, where each tree assigns a label on the data [9]. In the end, all the values given by the trees are summed up to make the final descision. The decisions from the non-helpful trees sometimes cancel out with the others, which makes the decisions from the good models to be more prominent.

5.4. Model Selection

The data that is used for the classifiers has to be properly prepared for use. Out of the 182 samples from the dataset, a certain percentage has to be set aside to train the algorithms in the classifiers, leaving the rest of the samples to be used as testing data. The reason for this is the fact that it is not practical to just use all of the data to train the classifier and then go back with the same data to test it. In that situation, you're leaving the classifier to predict something it already knows. So three functions in the Scikit Learn library were used to automatically split-up what in the dataset is used for training, and what is used for testing.

K-fold is a cross-validator that splits up the data into training/test sets. A value of k is given to the k-fold function which is a cross-validation parameter. The way it works is, k determines the number of times the entire dataset is "folded," where each fold is validated once and the remainder are used to make the training data. The second cross-validator is Leave One Out. It runs through the dataset and puts one subset aside to use to test. The last cross-validator is the Cross-Validation Score function, which is fairly identical to k-fold. This was used to double check on consistency.

For k-fold and Cross-Val Score, each validation, which is again determined by the CV parameter, outputs a percentage-accuracy score. So all scores in each simulation were appended into an empty array and averaged at the end of the simulations to determine the overall accuracy of each run.

6. Results

Different classifiers were used to see how each one performed in terms of accuracy, since each one works differently. The reason why different data selection models were used was to,

- have some data to train with and data that the classifier has not seen to use and test it with,
- observe that there is at least some consistency in how each classifier performs,
- and finally, to see how much the amount of training data used to train with, affects the accuracy.

With all that said, each classifer was ran numerous times at different cross validations for the k-fold and Cross-Val Score data selectors. Leave One Out leaves out specifically, one, subset everytime, meaning it uses the most training data.

In addtion to using these methods to split the data and for cross-validation, quick measurements were made using all the data to calculate the training accuracy. The three classifiers were simpily trained using all of the subsets, and made to predict the labels using the exact same data. Table 1 shows the training accuracies.

Starting with the k-NN classifier, since each subset falls under the class of "asleep" or "planning," the nearest neighbor parameter was set to be "2." The idea is that every

	K-NN	SVM	SVM	Random Forest
Parameter	$k = 2$	$\sigma = 1.2$	$\sigma = 5$	n estimators = 30
Accuracy	$80.77\,\%$	$80.23\,\%$	$99.45\,\%$	$73.62\,\%$

TABLE 1. TRAINING ACCURACIES

subset will look into the nearest two points and make a split decision between the two labels. Table 2 shows the following results.

Next is the SVM classifier, with Table 3 showing the results. The SVM has a parameter called a kernel. Kernels for the SVM include Linear, Polynomial, and Radial Basis Function. For the RBF, it also has a input variable, σ, and different values were experiemented with to find the best parameter. The kernel RBF with its variable σ = 5, was found to be the best choice.

Table 4 contains the results of the Random Forest. The n estimators parameter was chosen to be 30 and the depth of each tree to be 3. Due to the nature of the Random Forest, each simulation was ran 15 times, accuracies collected, and averaged.

Table 5 shows the Confusion Matrix of the SVM classifier. In this case, 36 subsets were used for testing.

UCI's methodology consisted of also using SVMs [2]. The used half of the dataset to train with, and the other half to test it. They used a RBF kernel with a parameter, $\sigma = 1.2$. They simulated the SVM ten times, averaged the accuracies, and achieved a score of $71.43\,\%$ [2]. The results in experiment [3], using simulations with classifiers similar to Random Forest, resulted in an accuracy of $71.42\,\%$. They used half of UCIs dataset for training and the other for testing. After that, a two-fold cross-validation was perfromed six times. All scores were averaged as well. For comparison, we used a train/test split function in Scikit Learn that simply splits the data as was done in the papers previously mentioned. The SVM algorithm we used was changed to use a parameter of $\sigma = 1.2$, simulated ten times, and averaged. The Random Forest classifier used the k-fold again, except for now, $k = 2$ and will be ran 6 times and averaged with the score it achieves from the train/test split. Through these simulations, our SVM achived a $71.42\,\%$ score and our Random Forest achieved a score of $70.20\,\%$.

7. Conclusion

From the resulting accuracies, one can observe that there is at least some consistency within the classifers. When compared to the results achived by UCI and the Decision Tree test [3], the results in this report are not off to be considered a failure. In fact, they are very similar results, some even being better. The highest accuracy in this test was at $73.87\,\%$ and the second best at $73.63\,\%$, both using the SVM. This classifier performed better than the remaining two, but I think it was not all on its own. From the results, it seems that the models perform better when more data is used on the classifer to train with. The Leave One Out cross-validation used all of the data to train expect for one subset, outputting the second best accuracy and with 20 cross-validations, the best accuracy is achived. The more cross-validations, the more folds are made in the dataset, meaning all the folds but one, are used to train with. In the end, it seems that more data from the same experiment would benefit the models used here. It seems 182 samples might not be enough for them and more training data, would bring better results. Another possible approach for better results would be exploring other methods but with these tests and the ones conducted by others, it seems the issues might come from the data and not so much the methodology.

References

[1] Melanie K Bothe, Luke Dickens, Katrin Reichel, Arn Tellmann, Björn Ellger, Martin Westphal, and Ahmed A Faisal. The use of reinforcement learning algorithms to meet the challenges of an artificial pancreas. *Expert review of medical devices*, 10(5):661–673, 2013.

[2] K Sercan Bayram, M Ayyüce Kızrak, and Bülent Bolat. Classification of eeg signals by using support vector machines. In *Innovations in Intelligent Systems and Applications (INISTA), 2013 IEEE International Symposium on*, pages 1–3. IEEE, 2013.

[3] Rajen B Bhatt and M Gopal. Frct: fuzzy-rough classification trees. *Pattern analysis and applications*, 11(1):73–88, 2008.

[4] Nizar Kerkeni, Frédéric Alexandre, Mohamed Hédi Bedoui, Laurent Bougrain, and Mohamed Dogui. Automatic classification of sleep stages on a eeg signal by artificial neural networks. In *5th WSEAS International Conference on SIGNAL, SPEECH and IMAGE PROCESSING-WSEAS SSIP'05*, 2005.

[5] Uci machine learning repository: Planning relax data set. https://archive.ics.uci.edu/ml/datasets/Planning+Relax. (Accessed on 06/02/2017).

[6] Neep Hazarika, Jean Zhu Chen, Ah Chung Tsoi, and Alex Sergejew. Classification of eeg signals using the wavelet transform. *Signal processing*, 59(1):61–72, 1997.

[7] Haibo He, Shijie Cheng, Youbing Zhang, and J Nguimbis. Home network power-line communication signal processing based on wavelet packet analysis. *IEEE Transactions on Power Delivery*, 20(3):1879–1885, 2005.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[9] yhat — random forests in python. http://blog.yhat.com/posts/random-forests-in-python.html. (Accessed on 06/17/2017).

	Leave One Out	K-Fold				Cross-Val Score			
Cross Validation	—	CV = 5	CV = 10	CV = 15	CV = 20	CV = 5	CV = 10	CV = 15	CV = 20
Accuracy	70.33 %	66.46 %	69.21 %	70.34 %	70.38 %	69.28 %	70.4 %	69.04 %	70.79 %

TABLE 2. ACCURACIES FOR K-NN CLASSIFIER

	Leave One Out	K-Fold				Cross-Val Score			
Cross Validation	—	CV = 5	CV = 10	CV = 15	CV = 20	CV = 5	CV = 10	CV = 15	CV = 20
Accuracy	73.63 %	71.38 %	72.49 %	71.37 %	73.61 %	72.55 %	72.57 %	73.87 %	72.92 %

TABLE 3. ACCURACIES FOR SVM CLASSIFIER

	Leave One Out	K-Fold				Cross-Val Score			
Cross Validation	—	CV = 5	CV = 10	CV = 15	CV = 20	CV = 5	CV = 10	CV = 15	CV = 20
Accuracy	71.28 %	70.5 %	70.26 %	71.37 %	71.5 %	71.15 %	71.5 %	71.55 %	71.5 %

TABLE 4. ACCURACIES FOR RANDOM FOREST CLASSIFIER

	Relaxed	Planning
Relaxed	20 (True Pos.)	0 (False Neg.)
Planning	16 (False Pos.)	0 (True Neg.)

TABLE 5. SVM CONFUSION MATRIX