MIT Conference/LSAMP Report 5

Christian Miranda

June 26, 2017

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.

I. Introduction

Using machine learning to make predictions and classify data is not a new concept, but is one that warrants further exploration. The University of California Irvine had conducted an experiment with EEG signals [1], 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, a Python-based machine learning library, 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 [2] which, the results to said tests, will be compared to the results of this report.

II. UCI EEG Dataset

The dataset that UCI aquired from the EEG signals from their experiment, consists of 182 subsets and 13 attributes [1]. 12 of those attributes are the features and the last attribute is the label of the subset [3]. The subjects were asked to sleep for five minutes, then wake them up, asking them to plan a movement for the following five seconds [1]. This was repeated over the course of about 30 minutes. Each subset is a five second fragment of the tests. Mu waves are located in the band 7 to 13 Hz and Beta waves are at 13 Hz and above [1] which, studies have showen that these waves were related to planning body movement. The features of the samples were extracted from the EEG signals using a wavelet packet transform [3]. Coeffcients are produced by the transform, called wavelets [4]. These wavelet coeffcients 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 [5]. The features ultimately would describe where the subsets fall in the 7 to 13 Hz band that gives insight into whether the subject is relaxed or planning a movement.

III. 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 situtation, 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.

IV. Classifiers

IV.I K-Nearest Neighbor

The first model used for classification was the k-Nearest Neighbor in the Scikit Learn library. This model works by, in newly-introduced data, it recognizes patterns according to data around certain data points [6]. The k-NN models use a user-defined parameter, n, that defines the number of points that the model looks out for around the tested data point. If the parameter is defined as "1," the model will search for the one point nearest to the test data and will make its assumptions to what classification it falls under. If the parameter is set to "2," the model with search for the nearest two points, if "3," so on and so forth.

IV.II Support Vector Machines

The second classifier that is being used is the Support Vector Machine. For a two label classification, SVMs try to find a hyperplane which separates the input space with a max margin [1]. The following are the equations to find the optimal hyperlane:

$$w.x_i + b \geq +1, \text{if } y_i = +1 \tag{1}$$

$$w.x_i + b \leq +1, \text{if } y_i = -1 \tag{2}$$

where x_i is the i^{th} input vector, y_i is the classification label of the i^{th} input, w is the weight of the vector which is normal to the hyperplane, and b is the bias [1]. The hyperlane is in between two margins that are parallel to the hyperplane. The margins are found with the following equation:

$$w.x_i + b \leq \pm 1 \tag{3}$$

The input vectors that determine the margins are called the support vectors. If they are not linearly separable, a kernal function should be applied to the support vectors to become linearly separate in a transform space:

$$K(x_i, x_j) = \varphi(x)\varphi(x_j) \tag{4}$$

To the linearly separate them:

$$f(x) = sign(\Sigma_i \alpha_i y_j \varphi(x)\varphi(x_j) + b) \tag{5}$$

IV.III 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 [6]. 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 [7]. Another parameter is "max depth," which defines how many nodes are in each trees [7]. Apparently, many of these decision-making trees aren't very helpful on their own. So the way that the classifer 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 [6]. In the end, all the values given by the trees are summed up to make the final discision. 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.

V. 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.

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 subset will look into the nearest two points and make a split decision between the two labels.

	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 1: Accuracies for k-NN Classifier

Next is the SVM classifier. 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 $\sigma = 5$, was found to be the best choice. UCI also used SVMs with the best accuracy at 71.43 % [1]. The following table is the accuracies with these parameters,

	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 2: Accuracies for SVM Classifier

Lastly, the Random Forest. The n estimators parameter was chosen to be 30 and the depth of each tree to be 3. The results in experiment [2], using simulations with classifiers similar to Random Forest, resulted in accuracies of 68.13 % and 71.42 %. The following table has the results for this present report,

	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 3: Accuracies for Random Forest Classifier

Due to the nature of the Random Forest, each simulation was ran 15 times, accuracies collected, and averaged.

VI. 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 [2], 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.

Bibliography

[1] 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.

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

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

[4] 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.

[5] 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.

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

[7] 3.2.4.3.1. sklearn.ensemble.randomforestclassifier scikit-learn 0.18.2 documentation. http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. (Accessed on 06/22/2017).