A Study of Epileptic Seizure Detection using Machine Learning Algorithms

Rajeev Kamaraju, Nathalia Peixoto

Electrical and Computer Engineering Department, George Mason University
4400 University Dr, Fairfax, VA 22030 United States of America

rkamaraj@gmu.edu

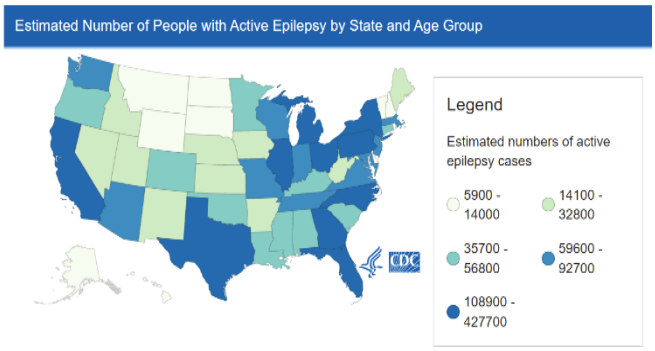
npeixoto@gmu.edu

Abstract— **This paper focuses on studying epileptic seizure detection using machine learning algorithms. Algorithms like Naïve Bayes, Logistic Regression, Stochastic Gradient Descent, K-Nearest Neighbour, Decision trees and random forests have been studied. For each of the classifier, many performance metrics have been computed and Area Under Curve (AUC) has been chosen as our performance metric. The paper also introduces the possibility of detecting epileptic seizures using Neural networks.**

Keywords— **Epilepsy, Naïve Bayes, Logistic Regression, Stochastic Gradient Descent, K-Neasrest Neighbours, Decision trees.**

I. INTRODUCTION

Epilepsy is a disorder in which nerve cell activity in the brain is disturbed, causing seizures. During a seizure, a person experiences abnormal behaviour, symptoms, and sensations, sometimes including loss of consciousness. There are few symptoms between seizures. A seizure is a single occurrence, whereas epilepsy is a neurological condition characterized by two or more unprovoked seizures. Epilepsy is the most common childhood brain disorder in the United States. Nearly 3 million people have been diagnosed with this disease, while 450,000 of them are under the age of 17. Two thirds of the child population will overcome the side effects, including seizures, through treatment during adolescence. Some treatments include surgery, medication and therapy, surgery however is only done if the child has drug resistant epilepsy. In 2015, 1.2% of the US population had active epilepsy (95% CI* = 1.1-1.4). This is about 3.4 million people with epilepsy nationwide: **3 million adults** and **470,000children**. This paper



The above figure shows the estimated number of people with active epilepsy by state and age group in United States of America (*courtesy: cdc.gov website*)

presents a study on predicting epileptic seizures by various machine learning algorithms. The study is conducted on a dataset that is available on UCI's machine learning repository. The dataset includes 4097 electroencephalogram (EEG) readings per patient over 23.5 seconds, with 500 patients in total. The 4097 data points were then divided equally into 23 chunks per patient, each chunk is translated into one row in the dataset. Each and every row contains 178 readings, that are turned into columns; in other words, there are 178 columns that make up one second of EEG readings. All in all, there are 11,500 rows and 179 columns with the last column containing the status of the patient, whether the patient is having a seizure or not. We run various types of machine learning algorithms for predicting epileptic seizures. The paper is divided into three sections, section-I discusses the basic approach involved in various machine learning algorithms, section-II involves the analysis of results obtained in section-I, section-III consists of discussion and conclusions.

II VARIOUS MACHINE LEARNING

Below we will conduct few experiments on our dataset. The experiments include predictions using machine learning algorithms like Naïve Bayes, Logistic regression, Stochastic Gradient Descent, K-Nearest neighbours and Convoluted Neural Networks. Before implementing the learning algorithms in our dataset, we have to first split our data into training, test, validation set respectively. The usual practise is to split the data in a 80:20 ratio, but for this paper we split the data in a 70:15:15 ratio in such a way that we train our model on 70% of data and test it on the 15% data of the tests set. The rest 15% in the validation set is used for hyperparameter tuning and for the selection of best approach for the prediction.

A. NAÏVE BAYES ALGORITHM :-

The naïve Bayes classification method is a supervised learning algorithm applying Bayes' theorem with the 'naïve' assumption of independence between every pair of features. Therefore, the class label of the data can be estimated using a naïve Bayes classifier through the following procedure.

Given a class variable 'y' and a dependent feature vector x_1 through x_n, Bayes' theorem states the following relationship:

$$p(y \mid x_1, \ldots, x_n) = \frac{p(y)p(x_1, \ldots, x_n \mid y)}{p(x_1, \ldots, x_n)}$$

Using the naïve independence assumption:

$$p(y \mid x_1, \ldots, x_n) = \frac{p(y)\prod_{i=1}^{n} p(x_i \mid y)}{p(x_1, \ldots, x_n)}$$

As $p(x_1, \ldots, x_n)$ is constant for any given input, the following classification rule can be deducted:

$$p(y \mid x_1, \ldots, x_n) \propto p(y)\prod_{i=1}^{n} p(x_i \mid y)$$

This results in estimation of a class label as:

$$\hat{y} = arg\max_{y} p(y)\prod_{i=1}^{n} p(x_i \mid y)$$

Naïve Bayes classifiers are simple and work quite well in many real-world situations. They require a small amount of training data to estimate the necessary parameters.

B. LOGISTIC REGRESSION ALGORITHM :-

Logistic regression is generally used to estimate the probability of an observation subject to an attribute. It estimates the observation probability from the given features. A logistic function F(v) is computed from the features as:

$$Logit = log\frac{p(O \mid v)}{p(\overline{O} \mid v)} = F(v)$$

where F(v) depends on the sequence of attributes (i.e. previous experiences):

$$F(v) = a_0 + \sum_{i=1}^{D} a_i \cdot v(i)$$

And for testing,

$$p(O \mid v) = \frac{1}{1 + e^{-F(v)}}$$

C. STOCHASTIC GRADIENT DESCENT ALGORITHM :-

Stochastic Gradient Descent picks a random instance in the training set at every step and computes the gradients based only on that single instance. Hence working on a single instance at a time makes the algorithm much faster because it has very little data to manipulate at every iteration. It also makes it possible to train on huge training sets, since only one instance needs to be in memory at each iteration.

If we have finite data stream, the following is an algorithm which simplifies how a stochastic gradient descent work,

$$f(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{\theta}, \mathbf{z}_i)$$

where $z_i = (x_i, y_i)$ in the supervised case, or just x_i in the unsupervised case, and $f(\theta, z_i)$ is some kind of loss function.

Since we want a single parameter estimate, we can use a running average,

$$\overline{\boldsymbol{\theta}}_k = \frac{1}{k}\sum_{t=1}^{k}\boldsymbol{\theta}_t$$

This is called Polyak-Ruppert averaging, and can be implemented recursively as follows:

$$\overline{\boldsymbol{\theta}}_k = \overline{\boldsymbol{\theta}}_{k-1} - \frac{1}{k}(\overline{\boldsymbol{\theta}}_{k-1} - \boldsymbol{\theta}_k)$$

Initialize $\boldsymbol{\theta}$, η;
repeat
  Randomly permute data;
  for $i = 1 : N$ **do**
    $\mathbf{g} = \nabla f(\boldsymbol{\theta}, \mathbf{z}_i)$;
    $\boldsymbol{\theta} \leftarrow \text{proj}_{\Theta}(\boldsymbol{\theta} - \eta\mathbf{g})$;
    Update η;
until *converged*;

D. K- NEAREST NEIGHBOURS ALGORITHM :-

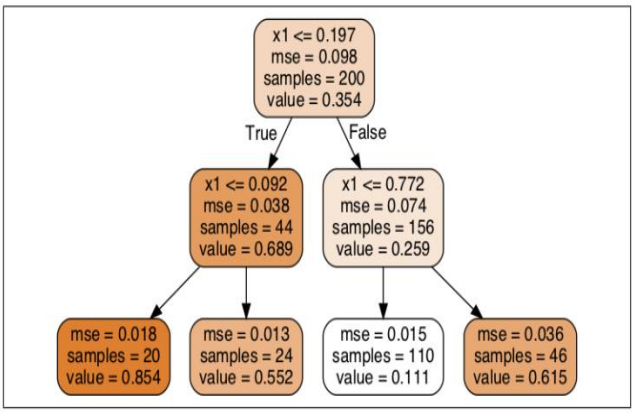
k-Nearest Neighbours (kNN) is a non-parametric learning algorithm. Contrary to other learning algorithms that allow discarding the training data after the model is built, kNN keeps all training examples in memory. Once a new, previously unseen example x comes in, the kNN algorithm finds k training examples closest to x and returns the majority label, in case of classification, or the average label, in case of regression. The closeness of two examples is given by a distance function. For example, Euclidean distance seen above is frequently used in practice. Another popular choice of the distance function is the negative cosine similarity. Cosine similarity defined as

$$s(\mathbf{x}_i, \mathbf{x}_k) \stackrel{\text{def}}{=} \cos(\angle(\mathbf{x}_i, \mathbf{x}_k)) = \frac{\sum_{j=1}^{D} x_i^{(j)} x_k^{(j)}}{\sqrt{\sum_{j=1}^{D}\left(x_i^{(j)}\right)^2}\sqrt{\sum_{j=1}^{D}\left(x_k^{(j)}\right)^2}},$$

is a measure of similarity of the directions of two vectors. If the angle between two vectors is 0 degrees, then two vectors point to the same direction, and cosine similarity is equal to 1. If the vectors are orthogonal, the cosine similarity is 0. For vectors pointing in opposite directions, the cosine similarity is −1. If we want to use cosine similarity as a distance metric, we need to multiply it by −1. Other popular distance metrics include Chebychev distance, Mahalanobis distance, and Hamming distance. The choice of the distance metric, as well as the value for k, are the choices the analyst makes before running the algorithm.

E. DECISION TREES ALGORITHM :-

Decision Trees are versatile Machine Learning algorithms that can per- form both classification and regression tasks and even multioutput tasks. They are powerful algorithms, capable of fitting complex datasets. Decision Trees are also the fundamental components of Random Forests, which are among the most powerful Machine Learning algorithms available today. The following is an example for visualizing a decision tree.



F. RANDOM FORESTS ALGORITHM :-

Random forest classifier creates a set of DTs from randomly selected subset of the training set. It then aggregates the votes from different DTs to decide the final class of the test data. The term random forest comes from random decision forests initially proposed by Tin Kam Ho of Bell Labs in 1995. This method combines Breiman's 'bagging' idea and random feature selection. Bagging or bootstrap aggregation is a technique for reducing the variance of an estimated prediction function. A random forest classifier is in fact an extension to bagging which uses de-correlated trees.

The main advantages of random forest classifiers are that there is no need for pruning trees, accuracy and variable importance are generated automatically,

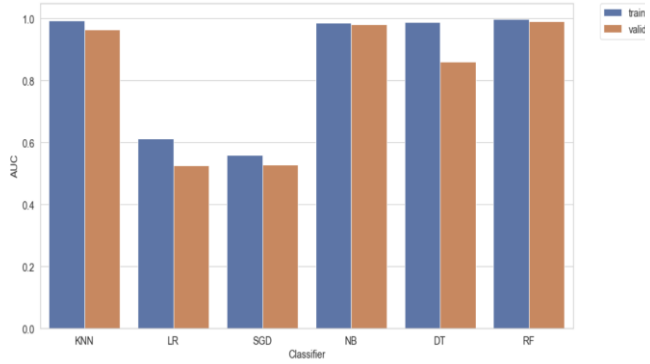
overfitting is not a problem, they are not very sensitive to outliers in training data, and setting their parameters is easy.

One of the limitation of Random forest is that, the regression process cannot predict beyond the available range in the training data and in regression the extreme values are often not predicted accurately. This causes underestimation of highs and overestimation of lows

III RESULTS

The above discussed learning models have successfully implemented and performance metrics like AUC, accuracy, recall, precision, specificity and prevalence have been computed in each case. Among these, the most common type of performance metrics is Area under Curve (AUC). Therefore, we will like to compare all the learning algorithm's performance based on AUC score. The reason behind choosing AUC as a performance metric is that it indicates the degree of separability.

The following is a histogram showing the AUC score of Training and Validation sets for all the algorithms.



A. Naïve Bayes Classifier

From the above figure we can say that the AUC score of Naïve Bayes classifier for Training and Validation set is 0.986 and 0.982 respectively. The reason behind such a high AUC score is that the naïve bayes model assumes that features are independent of each other.

B. Logistic Regression

From the above figure we can say that the AUC score of Logistic regression classifier for Training and Validation set is 0.612 and 0.527 respectively. This indicates that Logistic regression classifier is not at all working well on our dataset.

C. Stochastic Gradient Descent

From the above figure we can say that the AUC score of SGD classifier for Training and Validation set is 0.559 and 0.528 respectively. Among all the classifiers we studied in this paper SGD performance is the worst

D. K-Nearest Neighbours

From the above figure we can say that the AUC score of KNN classifier for Training and Validation set is 0.994 and 0.964 respectively. The reason behind such a good AUC score is the KNN algorithm keeps the training algorithms in memory. Since our validation set is 15% of our entire dataset, we get a high AUC score.

E. Decision Trees

From the above figure we can say that the AUC score of Decision tree classifier for Training and Validation set is 0.989 and 0.860 respectively. We can see that there is a significant drop in the value of AUC score, one of the main reasons for this can be the tendency of decision trees to overfit the data. If we see the training accuracy and validation accuracy they are 98.9% and 90.1% respectively.

F. Random Forests

From the above figure we can say that the AUC score of Random forests classifier for Training and Validation set is 0.998 and 0.990 respectively. The accuracies are also good. Since they are not much affected by outliers and there is no problem of overfitting of data. Random forests are among the best learning models.

III Conclusions

After deploying the above studied classifiers on our dataset, we can see that Random forests does the best classification among all other classifiers with a validation AUC of 0.990. The paper discusses introductory level ML algorithms for classification. In recent times, the study has been extended to ANN, CNN and RNN. Detecting the brain signals using Neural networks is being widely incorporated as the are robust, biologically inspired and give very good results.

References

1) Abu-Mostafa, Yaser S., Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning from data*. Vol. 4. New York: AMLBook, 2012.
2) Geron, A. "Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, ISBN: 978-1492032649." (2019).
3) Sanei, Saeid, and Jonathon A. Chambers. *EEG signal processing and machine learning*. John Wiley & Sons, 2021.
4) Murphy, Kevin P. *Machine learning: a probabilistic perspective*. MIT press, 2012.
5) Albon, Chris. *Machine learning with python cookbook: Practical solutions from preprocessing to deep learning*. " O'Reilly Media, Inc.", 2018
6) Strang, Gilbert, et al. *Introduction to linear algebra*. Vol. 3. Wellesley, MA: Wellesley-Cambridge Press, 1993.
7) Z. Zhang, G. Chen and S. Yang, "Ensemble Support Vector Recurrent Neural Network for Brain Signal Detection," in IEEE Transactions on Neural Networks and Learning Systems, doi: 10.1109/TNNLS.2021.3083710.
8) P. S. Madhukar and S. Madhukar, "Kalman Filters in different biomedical signals-An Overview," 2020 International Conference on Smart Electronics and Communication (ICOSEC), 2020, pp. 1268-1272, doi: 10.1109/ICOSEC49089.2020.9215335.
9) A. Nandy, M. A. Alahe, S. M. Nasim Uddin, S. Alam, A. -A. Nahid and M. A. Awal, "Feature Extraction and Classification of EEG Signals for Seizure Detection," 2019 International Conference on Robotics,Electrical and Signal Processing Techniques (ICREST), 2019, pp. 480-485, doi: 10.1109/ICREST.2019.8644337.
10) E. Santamaría-Vázquez, V. Martínez-Cagigal, F. Vaquerizo-Villar and R. Hornero, "EEG-Inception: A Novel Deep Convolutional Neural Network for Assistive ERP-Based Brain-Computer Interfaces," in IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 28, no. 12, pp. 27732782,Dec.2020,doi:10.1109/TNSRE.2020.3048106.
11) Burkov, Andriy. *The hundred-page machine learning book*. Vol. 1. Quebec City, QC, Canada: Andriy Burkov, 2019.
12) He, Bin, ed. *Neural engineering*. Kluwer Academic/Plenum, 2005.
13) Eliasmith, Chris, and Charles H. Anderson. *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT press, 2003.
14) Kingsford, Carl, and Steven L. Salzberg. "What are decision trees?." *Nature biotechnology* 26.9 (2008): 1011-1013.
15) Bronzino, Joseph D., and Donald R. Peterson. *Biomedical engineering fundamentals*. CRC press, 2014.
16) Foo, Jong Yong Abdiel, et al. *Ethics for biomedical engineers*. Springer, 2013.